Semantic parsing using Lojban – On the middle ground between semantic ontology and language

Master's thesis by Gerold Hintz
August 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Language Technology Group

Semantic parsing using Lojban –
On the middle ground between semantic ontology and language

Submitted Master's thesis by Gerold Hintz

1st certificate: Prof. Dr. Chris Biemann
2nd certificate: Eugen Ruppert

Day of submission:

Warrant

I hereby warrant that the content of this thesis is the direct result of my own work and that any use made in it of published or unpublished material is fully and correctly referenced. I also warrant that the presented work has not been used in any other form (even a similar one) in any other examination.

Darmstadt, August 14th, 2014

(G. Hintz)

Abstract

One of the most highly aspired goals of NLP is the semantic understanding of natural language text. An intermediate step to such an understanding consists of various layers of semantic annotations. Frame semantic parsing (FSP) applies the theory of frame semantics, a framework based on units of knowledge called frames. FSP identifies the frames invoked by a sentence and assigns their semantic arguments. FrameNet is a semantic ontology providing such a database of frames for the English language.

Lojban, a constructed language based on predicate logic is presented as an alternative resource for this purpose. Its well-defined, unambiguous and minimal construction suggests its use as a knowledge base similar to FrameNet. At the same time, Lojban is a language actively used by human speakers, thus enabling the use of statistical methods based on corpora.

In this thesis, Lojban is first introduced as a language and then analyzed as a semantic resource, showcasing properties common to frame semantic ontologies. Accordingly, a statistical alignment of Lojban predicate relations to FrameNet frames is performed and evaluated. It is further demonstrated how parallel Lojban-English corpora can be exploited to automatically extract semantic annotations marking predicate-argument structure within the English text.

Ultimately, a semantic parser is implemented which annotates English text based on the set of predicate relations defined by the Lojban dictionary. A purely knowledge-based, statistical approach is motivated which uses only the textual definitions of Lojban predicates. The resulting annotations of English sentences identify the invoked predicates and their respective arguments. The system is evaluated with gold data obtained from parallel corpora. Finally, the utility of the system is illustrated by providing a semantic dictionary search for Lojban words, which is provided to the Lojban community.

Acknowledgment

I want to express my sincere gratitude to Chris Biemann for giving me the opportunity to pursue this thesis of my choice. His supervision gave me the freedom that was necessary to carry out this project and provided me with excellent guidance where it was needed.

Acknowledgment is also due to the Lojban community for their continuous effort and undying passion for the language.

I would like to offer my special thanks to Anqi Wang for her ongoing support and encouragement.

Contents

1 Foundations

1.1 Introduction

The progress of machine text comprehension has given rise to advanced systems appearing to *understand* the meaning of natural language text. Examples of such systems are question answering engines ("How long is the Golden Gate Bridge?"), semantic search ("wife of the current president"), information extraction, summarization and machine translation. Arguably, most systems dealing with natural language as input are interested not just in the surface form of the input text, but in its underlying meaning. The high-level motivation for the work done in this thesis is grounded in obtaining such a representation of meaning.

To clarify this intricate notion of "meaning", it is necessary to first define a set of elementary terms. In linguistics, the branch concerned with the study of meaning is *semantics*. In terms of De Saussure (1916) it can be defined as the relation of linguistic elements (signifiers) to the entities they stand for in the world (signified). In the context of NLP, we are generally interested in capturing the meaning of a given natural language text in some form of *formal structure*.

One formalism used to capture the semantics of natural language is a *meaning representation language*. An example of a basic kind of meaning representation is First-Order-Logic (FOL), coercing statements of natural language into quantified boolean expressions. More sophisticated formalisms have been defined to capture different aspects of meaning in natural language, such as events, time, states, etc. The common ground of all such models is generally a predicate-argument model, which is refined in *semantic role* theory and *frame semantics*. The field of automated creation of such representations is *computational semantics*. *Shallow semantic parsing* or *frame semantic parsing* (FSP) systems automatically identify such predicate-argument structures with respect to a given model, called a *semantic ontology*. In short, FSP can be described as bridging the gap between the surface structure of the text, and the underlying semantic information.

Lojban can be seen as a different approach directed at the same issue. It is a constructed language, originating from the ambition to have *unambiguous* and *precise* representations of semantic content. However, it is an attempt at directly integrating this level of accuracy into a language. At its core, Lojban is based on a fixed set of logical predicates with well-defined arguments. This essentially eliminates a great deal of the transfer effort FSP systems attempt to accomplish. Therefore, semantic parsing is a trivial matter for Lojban text. Proceedingly, it can be argued that Lojban is very similar in design to semantic inventories such as FrameNet. Thus, the objective of this work is to showcase Lojban as a semantic ontology rather than just a language, and ultimately demonstrate a Lojban semantic parser.

This thesis is structured as follows. In the remainder of this chapter, some theoretical foundations will be briefly covered in Section 1.2, while giving an overview of related work in semantic parsing in Section 1.3. In Chapter 2, Lojban will first be introduced as a language and then further analyzed in the context of being a semantic ontology. In Chapter 3 this analogy is substantiated by performing an automated alignment to FrameNet. Based on this, in Chapter 4 several alignment-based parsers are implemented – both for obtaining FrameNet annotations for Lojban, as well as using FrameNet parsers for *Lojban semantic parsing* (LSP) – a task analogous to FSP, which exploits the inventory of Lojban predicates as an annotation language for English text. Subsequently, a full LSP system is defined and implemented. Finally, the work is concluded in Chapter 5, and closes with an outlook of future work. Figure 1.1 shows an overview of the structuring of this thesis.

1.2 Semantic representations

In the following, various models for representing the "meaning" of natural language text are introduced, which can be summarized as *semantic representations* of language. After briefly discussing *meaning representation languages* in Section 1.2.1, the theory of *semantic roles* is introduced in Section 1.2.2. Subsequently, the theory of *frame semantics*, giving rise to FrameNet is covered in Section 1.2.3, whereas the computational tasks originating from these models are discussed in Section 1.2.4.

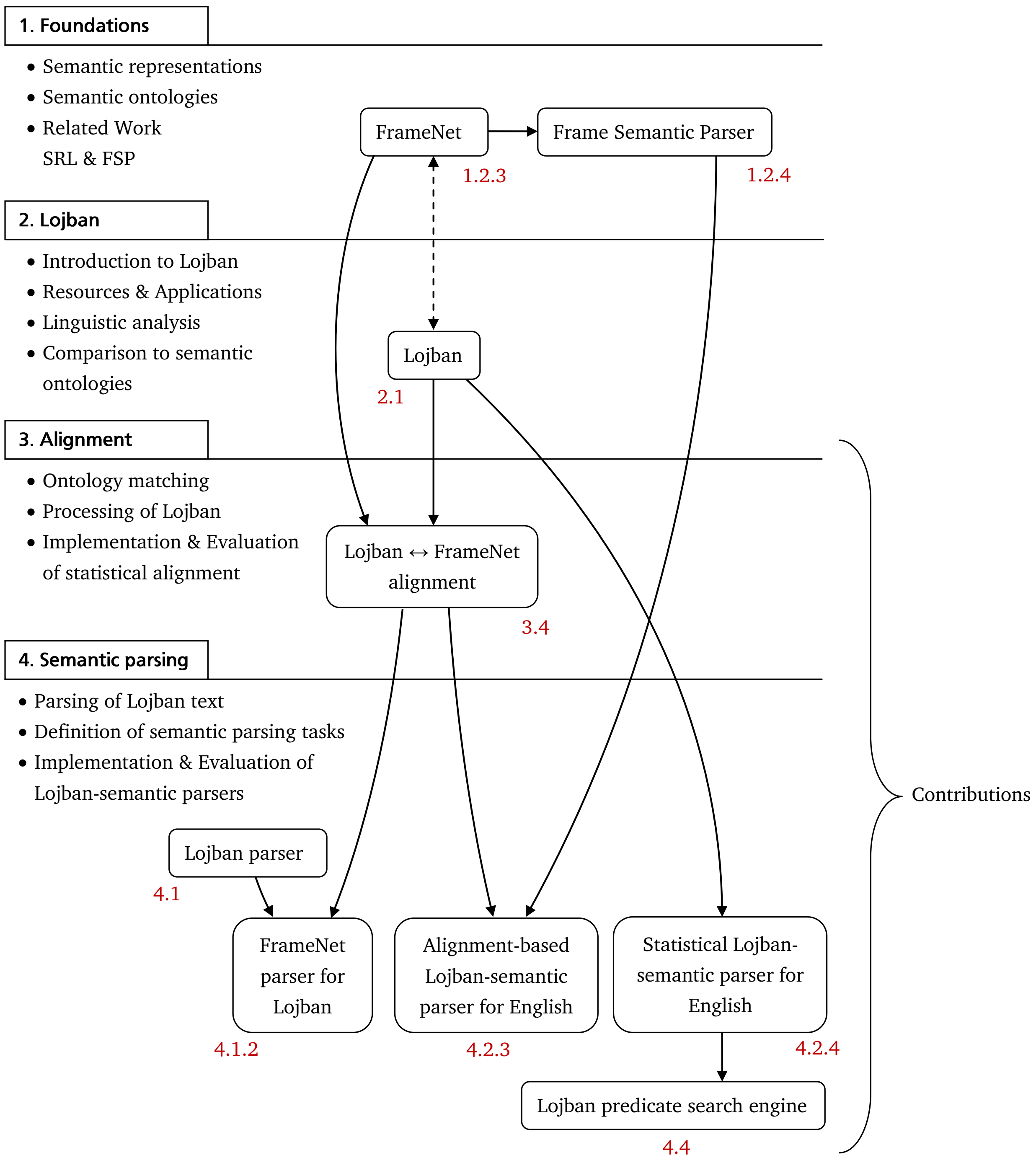

1. Foundations
• Semantic representations
• Semantic ontologies
• Related Work
SRL & FSP
FrameNet
Frame Semantic Parser
1.2.3
1.2.4
2. Lojban
• Introduction to Lojban
• Resources & Applications
• Linguistic analysis
• Comparison to semantic ontologies
Lojban
2.1
3. Alignment
• Ontology matching
• Processing of Lojban
• Implementation & Evaluation of statistical alignment
Lojban ↔ FrameNet alignment
3.4
4. Semantic parsing
• Parsing of Lojban text
• Definition of semantic parsing tasks
• Implementation & Evaluation of Lojban-semantic parsers
Lojban parser
4.1
FrameNet parser for Lojban
4.1.2
Alignment-based Lojban-semantic parser for English
4.2.3
Statistical Lojban-semantic parser for English
4.2.4
Lojban predicate search engine
4.4
Contributions

Figure 1.1: Overview of this thesis

(1) $GO_1\,(goer, destination)$

(2) $GO_2\,(goer, destination, origin)$

(3) $GO_3\,(goer, destination, means)$

(4) $GO_4\,(goer, destination, origin, means)$

(5) $GO_5\,(goer, destination, origin, means, time, cause)$

Figure 1.2: Various predicate-argument representations of "go"

1.2.1 Meaning representation languages

The most basic form of capturing the meaning of language is its formalization into First-Order-Logic expressions. Such an approach postulates that all meaning can be expressed as logical *predicates*, which have a fixed number of *arguments* (including nullary predicates). This assumption is not limited to FOL, but is a general foundation for a wide range of formalizations of semantics[1]. By enhancing this basic predicate-argument model with logical connectives ($\wedge, \vee, ..$), variables and quantifiers ($\forall, \exists$), a formal language is obtained, which can already express a large subset of meaning-related natural language content. As an example, consider the sentence *"Alice went to the door and opened it"*. One possible formalization of this statement in FOL would be:

$$GO\,(Alice, door) \wedge OPEN\,(Alice, door)$$

Here, the symbols GO and $OPEN$ are used to represent two predicates which each take two arguments. It is obvious that such a model does not inherently dictate what part of a linguistic expression actually is the predicate and what its arguments are. Also inherent to such formalizations is a certain amount of loss in meaning. In this example, the conjunction *and* is understood to be a sequential one, meaning that first the GO event took place, immediately followed by $OPEN$. Another loss in meaning is the past tense, which was not modeled in this particular FOL representation.

A major issue regarding the modeling of a fixed set of FOL predicates is their fixed *arity* (In linguistics this is referred to as *verb valency)*. For a given predicate like GO it is unclear what kind of information can or must be regarded as an argument to fully specify its meaning. Figure 1.2 shows some possible representations of a GO-predicate with different valencies. Whereas only the "goer" and a "destination" were needed for the example sentence (1), it would make just as much sense to include the "origin" (2), "means" of going (3), or both of these (4) as possible arguments. In (5), we could even theorize over arguments such as the "time" and the "cause" of the event. While these arguments clearly relate to the event itself, they are by no means essential to the meaning of "going", and are generic to nearly all such predicates. In fact, they form a category of so called *extra-thematic* or *peripheral* arguments, as opposed to the so called *core* arguments (the notion of core arguments will be elaborated in Section 1.2.2 and 1.2.3). One solution to these difficulties is the possibility to provide multiple versions of the predicate, and specify how they relate to each other in so called *meaning postulates*. Another solution to overcome this modeling problem is referred to as *Davidsonian reification*. A quantified variable is introduced, which is substituted into multiple predicates within a logical conjunction. In an extreme version called *Neo-Davidsonian*, a single unary predicate is used to identify the event, then as many arguments as needed are appended as additional predications. The Neo-Davidsonian representation of the example sentence could be given as follows.

$$\exists e_1 . Going(e_1) \wedge Agent(e_1, Alice) \wedge Destination(e_1, door)$$
$$\wedge \exists e_2 . Opening(e_2) \wedge Agent(e_2, Alice) \wedge Theme(e_2, door)$$

Whereas arguments such as *Destination* may be closely related to a particular event such as *Going*, in Neo-Davidsonian representations they are neither required nor associated with that event. Arguments such as *Agent* or *Theme* in fact mean to abstract over highly general *roles* and originate from the linguistic theory of *Thematic roles*.

[1] In fact, we will see that this is also the basis of the *Lojban language*, the introduction of which is deferred to Section 2.1.

```
(:LOGICALFORM (L / PROPERTY-ASCRIPTION )
 :DOMAIN (M / PERSON :LEX man :number plural )
 :RANGE (C / QUALITY :LEX happy )
 :TENSE present
 :POLARITY negative
 :LEX be )
:SET-NAME ADJECTIVAL-GROUP )
```

Listing 1.1: SPL representation producing the sentence "*Men are not happy.*"

```
(s / hum-02
  :arg0 (s2 / soldier)
  :beneficiary (g / girl)
  :time (w / walk-01
    :arg0 g
    :destination (t / town)))
```

Listing 1.2: AMR representation of the sentence *"The soldier hummed to the girl as she walked to town."*

Abstract Meaning Representation Language

More advanced meaning representation languages become increasingly complex as they cover the whole scope of expressiveness of natural language. These efforts partially originate from the field of natural language generation (NLG), whereas others originate from the complementary field of natural language understanding (NLU). In case of NLG, the motivation is to use a single, abstract representation of meaning to produce natural language sentences, possibly in multiple languages. An early example of this effort is the *Sentence Planning Language* (SPL), which is part of the KPML framework (Bateman, 1997). Listing 1.1 shows a formalization of a statement in SPL. It specifies the *process type* of the sentence to be produced, in this case a PROPERTY-ASCRIPTION, and the arguments of the statement, in this case a DOMAIN and a RANGE. It further formalizes the *tense* and *polarity*, so that a unique sentence can be constructed from this specification.

Meaning representation languages geared towards NLU work the other way around. A source sentence in a natural language is transformed into a formal representation. These projects are sometimes called *semantic treebanks*. A recent effort is the *Abstract Meaning Representation Language* (Banarescu et al., 2013). The AMR language is used to annotate an inventory of English sentences, with the primary motivation to encourage statistical approaches to language understanding. The objective is a representation of only concepts and relations between them, by abstracting over various language-particular realization of the same fact. Listing 1.2 shows an AMR formalization for a given sentence. Similar to quantified statements in FOL, variables are used to refer to entities within a sentence. In the example, `s` is used to refer to the main event expressed in the sentence. It has arguments, such as `arg0`, `beneficiary` and `time` associated with the semantic predicate `hum-02`, which are in turn assigned to variables. Note that a reference to the same variable `g` is used to state that "*she*" refers to "*the girl*" in the English sentence. The predicate `hum-02` and its respective arguments used in the AMR Treebank are adopted from PropBank, which is discussed in the following (Section 1.2.2.1).

1.2.2 Semantic and thematic roles

Building upon the notion of predicate-argument models, one of the oldest[2] linguistic theories for representing semantics are *semantic roles* (Potts, 2007; Jurafsky and Martin, 2009). The core idea is essentially applying a *label* to each noun phrase (NP). This label describes the role that the NP plays with respect to the predicate described by the head (generally the verb) of the sentence[3]. The term *"semantic roles"* denotes the class of all such role-labeling structures, without specifying a certain set of roles to be used.

At one end of the spectrum of semantic role systems, these labels could be specific to a certain event, so called *deep* roles. In the context of a *buy* event there would be, among others, a "*buyer*" and a "*seller*" role. At the other end of the spectrum, these labels attempt to generalize roles across all possible predicates; so called *shallow* roles.

[2] Semantic roles were first proposed sometime between 700 B.C. and 400 B.C. known as the *karaka* theory.

[3] Although a bold simplification, in semantic role theory it is often assumed that only verbs or verb phrases are predicates, and other constituents to be their respective arguments.

(a)	*John*	**broke**	*the window*	
	AGENT		THEME	
(b)	*The window*	*was* **broken** *by*	*John*	
	THEME		AGENT	
(c)	*The window*	*was* **broken** *with*	*a rock*	
	THEME		INSTRUMENT	

Figure 1.3: Different surface realizations of a *break* event. (Source: Jurafsky and Martin, 2009)

AGENT: Subject	THEME: Object	
AGENT: Subject	THEME: Object	INSTRUMENT: PP_{with}
INSTRUMENT: Subject	THEME: Object	
THEME: Subject		

Figure 1.4: Thematic grid for the verb *break.* (Source: Jurafsky and Martin, 2009)

In between these two there is a wide spectrum of theories and role sets, ranging from specific to highly general. In some cases, role labels have only been defined for a closed domain[4].

One theory located closer at the generic end of the spectrum are *thematic roles*, using only as few as nine different labels. Some of the most prevalent thematic roles are *AGENT* (an entity which deliberately performs an action), and *THEME* (an entity which undergoes the action). In many cases, these labels apply directly to the subjects and direct objects of a verb, although this might change in more intricate cases (an example being passive voice). Many different sets of thematic roles have been defined, but some elementary roles are common to most such sets (*Agent, Patient, Theme, Force, Experiencer, Result, Content, Instrument, Goal, Location, Direction, Cause, Manner, Purpose*), which is sometimes referred to as "Fillmore's list of nine" (Potts, 2007).

Thematic relations therefore act as a *shallow* meaning representation, which aim to generalize over all possible surface realizations. Consider for instances the sentences illustrated in Figure 1.3. These are all distinct surface representations of the event described by the word *break*. However in (a), the subject is *John* whereas *the window* is the direct object. Due to a passive construct in (b) their grammatical function is inverted, and *The window* becomes the subject of the sentence. A representation on thematic role level however, would abstract over these surface forms and consistently label John as the AGENT and the window as the THEME of *break*. In (c) we can observe that a different role INSTRUMENT is assigned, but the AGENT is being omitted. This again demonstrates that natural language does not directly map to predicates with a fixed set of arguments, a property which is a core issue in formalizing semantics.

Theta theory

An important theory related to thematic roles is concerned with the set of arguments which may or must be assigned for a particular relation. The *Theta theory* states that each verb has a unique *theta grid*, which dictates the number and type of noun phrases that are syntactically required by various realizations. As an example, a possible theta grid for the verb *break* is shown in Figure 1.4. It can be seen that various sets of arguments can be realized in various positions.

The *theta criterion* states, that each theta-role assigned by a verb must by realized by some argument, and each argument is assigned to one and only one theta-role.

1.2.2.1 PropBank

PropBank (Palmer et al., 2005) is an extension of the Penn TreeBank corpus, which provides annotations of semantic roles. Because of the difficulties of defining a universal set of thematic roles, PropBank resorts to using semantic roles with respect to an individual verb sense. To further simplify this definition effort, the semantic role labels are index numbers rather than descriptive names. In the general case, the semantic roles of a verb sense are thus uniformly called Arg_0, Arg_1, Arg_2, and so on, depending on the valency of the verb. Instead of formally defining a fixed set of argument labels, for each verb a short explanatory note is given for each role, along with a set of annotated examples. Despite this generality, Arg_0 is very consistently assigned an *Agent*-like meaning, while Arg_1

4 For example, Stallard (2000) have defined a domain-specific role inventory for booking airplane information.

```
<frameset>
<predicate lemma="walk">
  <roleset id="walk.01" name="walk">
    <roles>
      <role descr="causative agent" n="A"/>
      <role descr="walker" n="0"/>
      <role descr="path" n="1"/>
    </roles>
    <example name="with argA">
      <text>John walked his dog.</text>
      <arg n="A">John</arg>
      <rel>walked</rel>
      <arg n="0">his dog</arg>
    </example>
    ...
  </roleset>

  <roleset id="walk.02" name="baseball" vncls="-">
...
```

Listing 1.3: The PropBank frameset for the lemma "walk", including one roleset.

consistently has a *Patient* or *Theme* meaning. PropBank disambiguates each lemma into multiple senses, called *rolesets*; the collection of all rolesets for a common lemma is called a *frameset*. Sense disambiguation of words is done exclusively with respect to the argument structure. Thus, the discriminating feature between PropBank senses is the number and type of arguments rather than a fine-grained sense distinction (a roleset corresponds to a row in the thematic grid). As a consequence, PropBank has notably less senses than a lexicon (Kingsbury and Palmer, 2003). An exception to the regular indexed argument roles are a few number of *modifier roles* $Arg_{\text{M-*}}$, allowing for extra-thematic roles. Examples of these roles are $Arg_{\text{M-TMP}}$ for *temporal* arguments, or $Arg_{\text{M-DIR}}$ for *directional* arguments. These roles are not bound to any specific predicate and can always be applied.

In its current version, the PropBank contains over 4575 framed verbs and over 113000 annotations[5]. Listing 1.3 shows an exemplary PropBank frameset. Note that the roles of the given roleset (Arg_A, Arg_0, and Arg_1) are completely defined by a short definition string and a set of example sentences (more than shown). The fulltext annotations are given in a format similar to the example shown here. In addition to marking the frame arguments, they also contain annotations for coreference resolution. Thus, a token may be annotated with a reference annotation R_n to mark that it references the argument Arg_n.

1.2.2.2 NomBank

The NomBank project (Meyers et al., 2004) extends the PropBank annotation style to *non-verbal* predicates. Sets of arguments which co-occur with common nouns are annotated with rolesets in analogy to verbs. NomBank further categorizes them into noun-classes and gives abstract labels to the roles. As an example in the expression "*her husband*", NomBank defines *husband* to be an instance of a noun class, *DEFREL* (relational noun for personal relationships). The lexical items in the expression are annotated as roles, in this case "*her*" as Arg_0, and husband as Arg_1. NomBank is also a reasonable approach to deal with multi-word verbs, such as the expression "*make a promise*". It would be unfeasible to make the PropBank rolesets for "*make*" conform to all such cases, or to extend the PropBank dictionary with all possible multi-word expressions. The practical alternative offered by NomBank is to annotate the noun "*promise*" instead.

[5] Source: `https://catalog.ldc.upenn.edu/LDC2004T14`, accessed June 2014. Note: PropBank is non-public and only the lexicon can be obtained freely.

Subcat frame	**Verbs**	**Example**
Ø	eat, sleep	I ate
NP	prefer, find, leave	Find [$_{NP}$ the flight from Pittsburgh to Boston]
NP NP	show, give	Show [$_{NP}$ me] [$_{NP}$ airlines with flight from Pittsburgh]
PP_{from} PP_{to}	fly, travel	I would like to fly [$_{PP}$ from Pittsburgh] [$_{PP}$ to Phiadelphia]
NP PP_{with}	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston]

Figure 1.5: Some subcat frames for sample verbs. (Source: Jurafsky and Martin, 2009)

1.2.2.3 VerbNet

A project which incorporates PropBank through the use of explicit mappings is VerbNet (Schuler, 2005). It is a domain-independent, broad-coverage verb lexicon that groups verbs based on a class hierarchy. It leverages the linguistic theory of *Levin classes*, which classifies verbs according to their linking behavior to achieve syntactic and semantic coherence among members of a class. A primary feature of VerbNet are explicit mappings to other existing resources. In addition to mappings to PropBank senses, it includes mappings to WordNet, Xtag and FrameNet. WordNet (Miller, 1995) is another well-established hierarchical lexicon describing semantic relationships between individual words. Xtag is a grammar development tool including a hand-crafted syntactic grammar for English. FrameNet is introduced in the following.

1.2.3 Frame semantics

Frame semantics are a linguistic theory defined by Fillmore (1976). This theory provides a framework for modeling the cognitive process of understanding language. The core concept is the notion of *framing*, which can be explained as a (hypothetical) process of sentient beings to create, in memory, an inventory of *prototypes*. The act of understanding language is then regarded as perceiving the ways in which an object relates to these prototypes. Using such prototypes to structure, classify and interpret experiences is not only the basis of language understanding, but is also argued to be the root of human thinking. It is important to note that there is no universal *inventory* of such prototypes, but it is in fact constructed by each speaker individually[6].

Fillmore defines such prototypes pertaining to language as *frames*. A frame can be interactional (such as a frame for greeting) or conceptual, and often invokes a specific scenario. A common example of a conceptual frame is the “commercial event” scenario. This frame describes the class of events of commercial transactions, which describes the relationship between a *buyer*, a *seller*, an *object* being sold and possibly a *cost*. Such a frame is said to be *evoked* in the mind of anybody understanding words such as “buy”, “sell”, “pay”, “charge”, etc. As opposed to semantic roles, a frame is not necessarily tied to a specific word in the lexicon of a language, but is rooted at a deeper level of cognition. Therefore, different words may invoke the same semantic frame, and a particular frame might be evoked by a multitude of words.

Subcategorization frames

A closely related theory which deserves mention are so-called *subcategorization frames* (or short *subcat frames*). Whereas frames defined by Fillmore (1976) are a semantic concept, *subcategorization* is a syntactic concept which defines the presence and types of arguments with regard to a certain lexical item, usually a verb[7]. The main idea is that verbs are compatible with different kinds of *complements*, each giving rise to a certain subcat frame. Two highly simplified categories would be *transitive* and *intransitive* verbs; the former requires a direct object, whereas the latter lacks one. In practice however, there are not only two, but hundreds of such categories. A single verb can have multiple subcategorization frames, which are generally defined in terms of the constituents of a formal generative grammar. Figure 1.5 shows some examples of these. It is important to note that subcat frames are defined on the syntactic level, each being distinguished only by their syntactic arguments.

6 This also implies that while the concept of frames is independent of languages, a specific language may influence the construction of the inventory of frames.

7 Subcategorization frames are conceptually overlapping with theta grid theory described in 1.2.2 (Crocker, 1996).

1.2.3.1 FrameNet

FrameNet (Baker et al., 1998) is a realization of frame semantic theory as a lexicon. It models the frame semantic theory as a set of general purpose hierarchical frames, which are constructed through manual work of linguists. In FrameNet, a frame is comprised of

- A frame name (e.g. *Self_Movement*)
- A textual definition of the frame
- A set of *lexical units* (LU) said to evoke the frame (e.g. *advance.v, crawl.v, fly.v, way.n, ...*)
  LUs are tagged with WordNet-style POS tags and can also consist of multi-word expressions (such as *take_a_walk.v*). For each LU, a brief refinement of the definition is given. In the context of an evocation, these items are called *target units* (TU).
- A set of *Frame Elements* (FE). These in turn consist of
  - an FE name (e.g. *Self_Mover*)
  - a textual definition of the frame element
  - a semantic type (e.g. *Sentient*), describing the kind of entities being substituted as an argument
  - semantic relations to other FEs (e.g. specialization-relations to FEs of upper frames)
  - a *Core Type*, describing the saliency of the element in respect to the given frame (*Core*, *Peripheral*, or *Extra-Thematic*)
    The subset of all FEs of type "*Core*" are further defined as its *Core Elements.* These core elements "instantiate a conceptually necessary component of a frame, while making the frame unique and different from other frames" (Ruppenhofer et al., 2006).
- A set of Frame-to-Frame relations (e.g. *Inherits from*, *Is Subframe of*, *Is Causative of*, etc.)
- Annotated example sentences

In FrameNet terminology, a *lexical unit* further refers to the tuple of a lemma paired with a corresponding frame, e.g. (*walk.v, Self_Motion*). In the current version (1.5, as of May 2014) the FrameNet lexicon consists of 1019 frames, 11'829 lexical units, an accumulated total of 8884 frame elements and 1507 frame-to-frame relations. It is noteworthy that FrameNet uses a *top-down* approach for its construction, meaning that linguists start by hypothesizing a novel frame and then collect evidence for this frame in corpora.

FrameNet annotations
In addition to defining an inventory of frames, the FrameNet database also provides a collection of manual frame semantic annotations. These consist of example sentences illustrating different scenarios of a given lexical unit, as well as annotated full text corpora. Whereas the example sentences are tied to a specific frame (chosen subjectively by the annotators) and only annotate the frame in question, the corpora are continuous texts of various domains with complete annotations of all currently defined frames being evoked. These annotations mark the target unit with a corresponding frame they disambiguate to, as well as the spans of the arguments filling the respective frame elements (from now on synonymously called semantic roles). Figure 1.6 shows an exemplary sentence from the full-text corpora of FrameNet (1.5), annotating three distinct frames. The FrameNet corpora also include annotations of lower levels, such as POS, grammatical function, and phrase type.

Discontinuous arguments
FE annotations are also allowed to be *discontinuous*, meaning that in some cases the same FE label appears multiple times relative to a given target. An example of such discontinuous elements can be seen in the following sentence.

What	*was*	*Bill*	***angry***	*about*	*?*
Topic		Experiencer		Topic	

Here, the *Topic* is a discontinuous argument, as a fronting construction moved a preposition to the beginning of a sentence. On a technical level, frame elements are annotated as a set of token spans.

Null instantiations
FrameNet further allows for so called *null instantiations*. These occur if frame elements are "conceptually salient, but do not show up as lexical or phrasal material" (Ruppenhofer et al., 2006). A frame element is marked as present

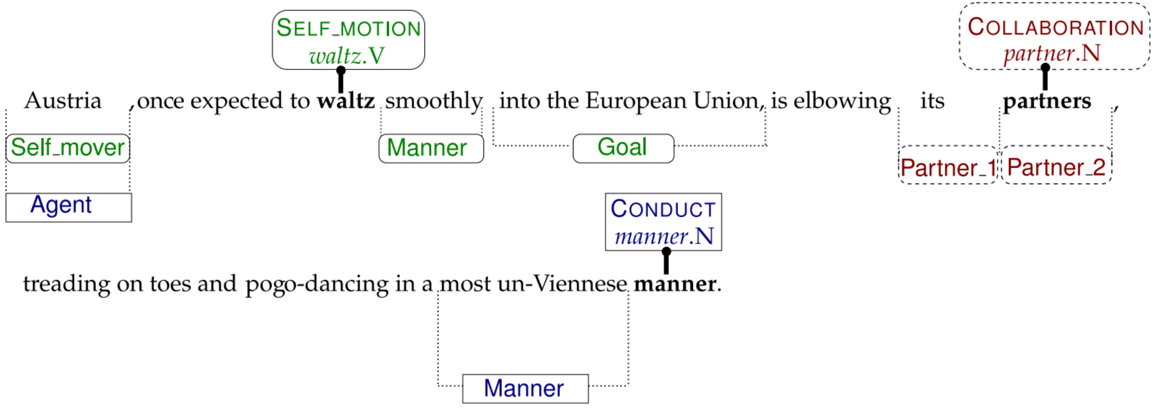

SELF_MOTION
waltz.V
COLLABORATION
partner.N
Austria, once expected to waltz smoothly into the European Union, is elbowing its partners,
Self_mover
Manner
Goal
Partner_1
Partner_2
Agent
CONDUCT
manner.N
treading on toes and pogo-dancing in a most un-Viennese manner.
Manner

Figure 1.6: An example FRAMENET annotation (1.5 release). For a given sentence it annotates *target units* (bold), evoked *semantic frames* (marked above) and the *semantic roles* (marked below). Source: Das et al., 2012

even without providing any lexical items to fill it. Whereas explicit roles are said to be *covert*, such null instantiated roles are said to be *overt*. There are multiple variants of overt roles, such as *definite null instantiations* (DNI), which are *anaphoric* elements known from context. Other types of null instantiations are *indefinite null instantiations* (INI), or *constructional null instantiations* (CNI), which instantiate frame elements which are generally understood even without context. An example of such a case can be observed in the imperative sentence "*Cook on low heat until done*". The verb *cook* evokes the *Apply_heat* frame with a null instantiation of the semantic role *Food*. Although a *Food* entity is not mentioned lexically, it is obvious that some food participates in the frame. All FRAMENET corpus annotations include null-instantiations like these, which account for roughly 5% of all evoked frame elements in the available data (Chen et al., 2010).

1.2.4 Computational tasks

Now that we introduced some essential semantic resources, the computational tasks associated these will be briefly explained. The automatic creation of semantic annotations can be described with the umbrella term of *semantic analysis*. This however, includes some tasks of related areas, which here are not the main focus. As an example, *relationship extraction* deals with the classification of a fixed set of artifacts (such as named entities), with a fixed inventory of relationship labels.

Here, we will elaborate on two closely related tasks which are both considered to be instances of *semantic parsing*. These two tasks respectively deal with the creation of PROPBANK-style and FRAMENET-style annotations of natural text. A related approach, *deep parsing*, is only introduced for completeness.

1.2.4.1 Semantic role labeling

The automatic labeling of semantic roles as described in Section 1.2.2 is called *shallow semantic parsing,* or more commonly *semantic role labeling* (SRL). The first automated systems in this field have been pioneered by Gildea and Jurafsky (2002), which have initiated a substantial amount of research in computational systems using predicate-argument structures for semantics (using both the PROPBANK role inventory as well as FRAMENET frame elements). The task of SRL is defined as recognizing the correct constituents filling a semantic role. It does not include the identification of targets or the disambiguation of word senses.

In 2004 and 2005, the CoNLL shared tasks were dedicated to SRL in this sense (Carreras and Màrquez, 2005). PROPBANK-annotated corpora were provided as training data, including some underlying layers of annotations (e.g. *named entities*). As a baseline for the task, a simple heuristic (shown in Figure 1.7) was defined, which achieved an F_1-measure of 37.1%. In the 2005 CoNLL task over 20 systems participated with leading systems reaching an F_1 of 79.5% for exact constituent matches. In the 2008 CoNLL shared task, the focus shifted to "joint parsing of syntactic and semantic dependencies" (Surdeanu et al., 2008), which proposed a formalism modeling both syntactic dependencies and semantic roles. The FRAMENET inventory was first used as an inventory of semantic role labels

1. Tag *not* and *n't* in target verb chunk as $Arg_{\text{A-NEG}}$
2. Tag modal verbs in target verb chunk as $Arg_{\text{A-MOD}}$
3. Tag first *NP* before target verb as Arg_0
4. Tag first *NP* after target verb as Arg_1
5. Tag *that*, *which* and *who* before target verb as R_0
6. Switch Arg_0 and Arg_1 , and R_0 and R_1 if the target verb is part of a passive *VP* chunk.

Figure 1.7: CoNLL-2005 baseline-system for semantic role labeling

```
talk < v subj (obj n (p about)) (comp (p to with))
     < v subj obj1 (comp1 (p into))
     < n nsubj (nobj n (p about)) (ncomp (p to with))
```

Listing 1.4: A sample lexicon entry for a node in the English Slot Grammar (ESG)

in the 2004 shared task Senseval-3 (Litkowski, 2004). Being only an SRL subtask, the evaluation was restricted to identifying role labels, namely the *frame elements,* for an already given target unit and pre-annotated frame within a sentence. In the 2010 Semeval task (Ruppenhofer et al., 2010), participants could choose between FrameNet and PropBank-style annotations, which were evaluated separately. In addition to the syntactic role labeling task, a novel subtask for *NI linking* was introduced. Here, the challenge is to find links between null instantiations and the wider discourse context, rather than just assigning role labels within a sentence.

1.2.4.2 Frame semantic parsing

The more advanced task of full *frame semantic parsing* (FSP) was first introduced in the 2007 shared SemEval task for "frame semantic structure extraction" (Baker et al., 2007), and was based on FrameNet. An FSP system has to provide full frame semantic annotations, conceptually requiring a number of steps:

1. identify a set of target units within a sentence which invoke a frame
2. disambiguate between the set of frames each target invokes (which relates closely to the word sense)
3. recognize the frame elements filled by lexical items within the sentence, and those which are null instantiated

Thus, FSP intuitively blends the tasks from traditional *word sense disambiguation* and *semantic role labeling*, although this partitioning is not formally defined. The evaluation of such a system is intrinsically more complex than an SRL evaluation, and a scoring metric is provided that gives partial credit for each subtask.

As an additional challenge, competing systems must also account for *unknown* frames (which have been added for the SemEval task, but not disclosed in the training data) and use unseen lexical targets for frame evocation.

1.2.4.3 Deep Parsing

A slightly related approach to obtaining semantic structure of natural language text is *deep parsing*. In the IBM Watson[8] system, deep parsing is performed using an *English Slot Grammar*, and a *predicate-argument structure builder* (McCord et al., 2012). Deep parsing effectively unifies the task of syntactic parsing (surface structure), with logical analysis (deep structure).

The output of the English Slot Grammar (ESG) is a hierarchical tree of predefined nodes. Each node is centered on a *headword* in the text, and has a fixed set of arguments, or complement slots which may be filled with other nodes from the parse tree. Complement slots describe the grammatical role their fillers have with respect to the headword, and are – similar to PropBank roles – idiosyncratic to the sense of their headwords. Listing 1.4 shows a sample entry for such nodes. For the lexical item "talk", three different *sense frames* are listed, which each define the POS of the item, as well as complement slots, and an arbitrary number of semantic and syntactic features for these slots. For the first sense frame, there is first a *subj* slot, and second an *obj* slot with the features *n* and *(p about)*. This means that the second argument allows noun phrases as well as "about"-prepositional phrases. On

[8] *Watson* is an AI system capable of answering natural language questions. It has won against human competitors in the gameshow "Jeopardy!" in 2011: `http://www.nytimes.com/2011/02/09/arts/television/09nova.html`, accessed June 2014

top of such syntactic slot requirements, there can also be semantic type checks which can require the filler to have a semantic type, e.g. “human” (there is a hierarchy of 160 semantic types). This inventory of slot grammar entries is partially hand-built, and partially extended automatically using lexical resources such as WordNet.

The predicate-argument structure builder (PAS) builds on top of the ESG parse tree and provides simplifications over the surface structure of the text. Subtle details are removed from the ESG parse tree, such as the distinction between active and passive constructs, auxiliary verbs, determiners, or certain occurrences of the word “be”. Thus, the output of the PAS component is a simplified directed graph, which abstracts over multiple distinct sentence realizations expressing the same content. In Watson, deep parsing is used as a complementary level of analysis in conjunction with conventional semantic parsing based on thematic roles, which is introduced in the following.

1.3 Related work

The steady increase in labeled training data, as well as the shared CoNLL and SemEval tasks has lead to a boost in SRL and FSP research, in particular for statistical and ML-based approaches. Although SRL and FSP are technically two distinct tasks, research in these areas will be introduced conjointly. In the early era of semantic parsing, systems focused mainly on role-labeling, whereas full frame semantic parsing, being an enclosing and more advanced task, emerged a few years later. This work will largely be introduced historically, leading up to the current state of the art in semantic parsing.

The first domain-independent statistical system pioneered by Gildea and Jurafsky (2002) has provided a great deal of groundwork for forthcoming research. A statistical system was trained for finding and labeling the correct semantic role labels for a given target unit and frame. They proposed a *discriminative model* for estimating the probability for assigning a *constituent* (obtained from the phrase-structure tree output of a statistical parser) to a semantic role label of a given FrameNet frame. Thus, for a given frame f, a target unit tu and features obtained from a constituent c they estimate the probability of assigning a role r

$$P\,(r \mid f,\, tu, \mathit{features}(c))$$

Hence, roles are assigned to constituents individually and independently. Due to the high sparsity of SRL training data, this probability distribution is approximated with a set of local classifiers using an interpolated fallback model. The features proposed in their work represent the syntactic structure obtained from a statistical parser as well as lexical information obtained from the words within a constituent. In fact the relationship between syntactic surface features and semantic roles is grounded in *linking theory*. This theory argues that “the syntactic realization of arguments of a predicate is predictable from the semantics”, suggesting that semantic relationships can in fact be learned only from syntactic cues (Gildea and Jurafsky, 2002). They defined a collection of syntactic features which was extended and adapted by most SRL systems. Figure 1.8 shows an overview of the most relevant of these.

Shortly thereafter, it was demonstrated that this baseline approach could be improved through the use of more sophisticated classifiers, such as *Support Vector Machines* (SVM) or *Maximum Entropy* (ME) classifiers. Fleischman et al. (2003) first applied a ME classifier to FrameNet SRL, which allowed the use of a much larger set of features. This lead to a significant improvement of 6% over previous F_1-scores. SVMs were also successfully applied to SRL with both FrameNet as well as PropBank roles (Pradhan et al., 2004). Their approach was to first identifying a set of constituents which could serve as arguments. Subsequently, role labels were assigned through a collection of binary SVMs. Pradhan et al. (2005) could further show that using multiple syntactic representations (obtained from multiple parser outputs) improved overall performance, finally ranking as one of the leading systems in the 2005 task. The best system in the 2005 task (Koomen et al., 2005) applied a boosting approach, combining the output of multiple classifiers with a set of hand-crafted constraints.

Thompson et al. (2003) first suggested a *generative* approach, which aims to fix a number of shortcomings of the discriminative model. First and foremost, the discriminative approach does not allow for joint inference over the choice of a frame and the respective semantic roles. Further, it cannot handle unexpressed arguments (null instantiations, see Section 1.2.3.1) without retrospective adjustments to the model. Thus, a joint probability distribution over the target unit, frame and constituents filling the corresponding semantic roles was proposed. Thus they estimate the probability

$$P\,(tu, f, c_1, c_2, ..c_n, r_1, r_2, \dots, r_n)$$

where tu is the target unit, f is the evoked frame, $c_1, c_2, .., c_n$ are the constituents serving as arguments and $r_1, r_2, \dots, r_n$ are the corresponding semantic roles. In this notation, a constituent c_i is assigned to the role label r_i.

Feature Name	Description
Predicate	Lemmatization of the predicate word
Path	Syntactic path linking the predicate and an argument, e. g., NN↑NP↑VP↓VBX
Partial path	*Path* feature limited to the branching of the argument
No-direction path	Like *Path*, but without traversal directions
Phrase type	Syntactic type of the argument node
Position	Relative position of the argument with respect to the predicate
Voice	Voice of the predicate, i. e., active or passive
Head word	Syntactic head of the argument phrase
Verb subcategorization	Production rule expanding the predicate parent node
Named entities	Classes of named entities that appear in the argument node
Head word POS	POS tag of the argument node head word (less sparse than Head word)
Verb clustering	Type of verb → direct object relation
Governing Category	Whether the candidate argument is the verb subject or object
Syntactic Frame	Position of the NPs surrounding the predicate
Verb sense	Sense information for polysemous verbs
Head word of PP	Enriched POS of prepositional argument nodes (e. g., PP-for, PP-in)
First and last word/POS	First and last words and POS tags of candidate argument phrases
Ordinal position	Absolute offset of a candidate argument within a proposition
Constituent tree distance	Distance from the predicate with respect to the parse tree
Constituent features	Description of the constituents surrounding the argument node
Temporal Cue Words	Temporal markers which are very distinctive of some roles

Figure 1.8: SRL features used in most systems. (Source: Moschitti et al., 2008)

Therefore, in this approach arguments and semantic roles are *linearized,* which inherently takes into account their ordering. Thompson et al. (2003) approximate this probability distribution with a *Hidden Markov Model*. When conditioned on a particular frame, the linearized semantic roles correspond to the hidden states of the model, whereas the constituents correspond to observations. A major improvement based on such joint models are so called *selection restrictions*, which essentially incorporate the linguistic knowledge from theta theory (Section 1.2.2) or subcategorization frames (Section 1.2.3). For this purpose, global constraints are defined to prevent overlapping or incompatible arguments. These constrained models are solved using integer linear programming (Punyakanok et al., 2004), or through re-ranking strategies based on dynamic programming (Toutanova et al., 2005).

As an exception to the emerging consensus on using statistical approaches, Shi and Mihalcea (2004) stand out for developing a rule-based system. They used hand-crafted pattern-matching rules to encode syntax-semantic mappings.

The first system performing full frame structure extraction has been the *Shalmaneser* toolchain introduced by Erk and Padó (2006). Their approach was chaining Naive Bayes classifiers, first for frame sense disambiguation, followed by semantic role assignment. This decomposition has been adapted by Johansson and Nugues (2007), who instead use SVMs and further incorporated WordNet synsets to extend the vocabulary of predicates to cover unknown words. Their system scored best in the 2007 task.

Another novel approach was introduced by Moschitti et al. (2008) who use *tree kernels* to improve role labeling performance of SVMs. Tree kernels can be used with SVMs using the well-known *kernel trick*, and are used to compute similarity scores between trees. They are applied directly to phrase-structure trees of a syntactic parser and can therefore autonomously discover salient syntactic features.

A semi-supervised strategy to FSP was first demonstrated by Fürstenau and Lapata (2009), who project frame semantic labels from a set of seed examples to unlabeled sentences through the use of a similarity graph.

The current state of the art in FSP is the SEMAFOR system, introduced by Das et al. (2010b). Their system combines the bulk of insights obtained from FSP research into a FrameNet-based semantic parser. At its core, SEMAFOR uses a joint inference model, treating frame disambiguation and role labeling in one stage, and also incorporating selection restriction constraints. In addition, *latent variables* are used for disambiguating predicate words (Das et al., 2010a). They have further extended their system with special treatment of null instantiations (Chen et al., 2010), and adopted semi-supervised techniques for unseen lexical predicates (Das and Smith, 2011). A highly extensive report on the current system, and FSP in general is compiled by Das et al. (2014).

It is also worth mentioning that in a different line of research, an unsupervised approach to frame-semantic representations is pursued (Titov and Klementiev, 2012). Semantic roles are induced from unlabeled corpora by clustering syntactic signatures.

2 Lojban as a semantic resource

Closing in on the core theme of this work, this chapter introduces the artificially constructed language *Lojban*. To begin with, it will present Lojban traditionally – as a language – and subsequently analyze it on a linguistic level. Moving forward, it is then studied as a semantic resource. It is analyzed how Lojban relates to the theories and semantic ontologies described in Chapter 1, and ultimately envisioned as resource closely comparable to PropBank or FrameNet.

Section 2.1 will give a brief introduction of the language and explain the motivations of the project. Section 2.2 then gives a summary of the available resources regarding Lojban, including linguistic resources such as corpora, as well as software such as parsers. Related work dealing with Lojban is subsequently covered in Section 2.3. Section 2.4 conducts some elementary linguistic analysis of the language, and Section 2.5 finally evaluates it as a resource in comparison to semantic ontologies.

2.1 Introduction to Lojban

Lojban is a constructed language based on *predicate logic*. It has been designed for logical, unambiguous human-to-human communication, but also possible human-to-machine communication, covering a middle-ground between natural language and formal meaning representation language. The language is fully specified in "The Lojban Reference Grammar" (Cowan, 1997a), which includes an official context-free grammar defined in Backus-Naur Form (BNF)[1], hence it is trivially parsable.

At the core of the language, basic Lojban sentences express a logical predicate with a fixed set of arguments. Consider for example the sentence "*John is the father of Sam*". In this trivial case, a *is-father-of* predicate is stated between the arguments *John* and *Sam*. Figure 2.1 illustrates this notion in Lojban terminology, which will be adapted from now on.

bridi	a *predicate expression,* the basic sentence construct in Lojban
selbri	the expression (generally a word) acting as the *predicate* of a bridi
sumti	an argument passed to a *selbri*

The above statement is expressed as the Lojban bridi

`la .djon.`	`patfu`	`la .sam.`
sumti	selbri	sumti

The respective Lojban representations of the names John and Sam (`la .djon.` and `la .sam.`)[2] act as the *sumti* (arguments) to the predicate relation expressed by `patfu`, which is the *selbri* (predicate) of the sentence. The word `patfu` is a *content word* expressing this father-of relation as a predicate.

`patfu`: x_1 *is a father of* x_2

The arguments x_1 and x_2 of `patfu` are not identified by their names, which are arbitrary in this definition, but instead by their *place-structure*. The first argument of the `patfu`-relation is the father, whereas the second argument is the son (or daughter). Thus, the set of arguments and the argument structure is directly built into the meaning of a word in the Lojban lexicon[3].

On top of this basic predicate-argument framework, Lojban provides a vast set of features covering the remaining aspects of language; such as connectives, tenses, questions, etc. The design goals behind Lojban are various. Foremost, Lojban is designed for people to communicate with each other clearly by overcoming the inaccuracy of natural languages. This is achieved by eliminating *syntactic ambiguity* and minimizing *semantic ambiguity*.

[1] The original grammar is written in *YACC*, which has recently been superseded by a *PEG* grammar (see Section 2.2.3).

[2] We will adapt the convention that Lojban literals are always given in `typeface`.

[3] The Lojban word "sumti" is often used to refer to both the argument variables, as well as the substituting constituent. To avoid confusion, we will refer to the argument-places x_1, x_2,..., x_n of a selbri definition as "arguments" or "slots", whereas a substitution into a slot is called "sumti".

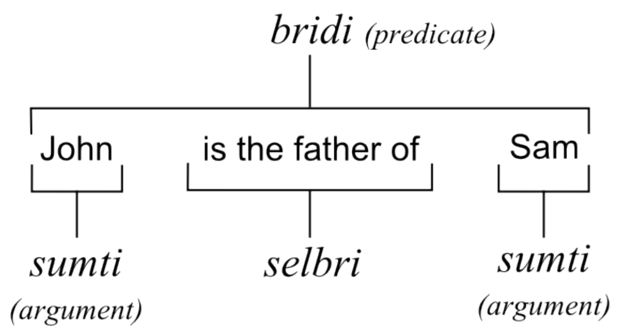


Figure 2.1: The predicate-argument terminology of Lojban. (Source: Cowan, 1997b)

Nevertheless, Lojban acknowledges that ambiguity can and *should not* be completely eliminated (see Section 2.4.2). The language is further designed to be highly regular and easy to learn, and aspires to remove restrictions on creative and clear thought (Cowan, 1997a).

Lojban is a long, ongoing research and design effort. It originated from *Loglan* (Brown, 1989), an earlier version of the language developed from 1955 onward. The language was described as "symbolic logic made speakable", and was designed with the primary goal of testing *linguistic relativity*, also known as the Sapir-Whorf hypothesis (Kay and Kempton, 1984). In its strong version this theory claims that the language people speak shapes their thought and *constrains* their cognitive abilities. Thus, Loglan was an experiment to determine if a perfectly logical language could enable new thought patterns in its speakers[4]. Due to attempts to monetize the language, and subsequent copyright disputes, Loglan was forked to the separate language Lojban in 1987, by the *Logical Language Group* (LLG)[5], a non-profit scientific charity. All work published by the LLG, including the reference grammar and the Lojban dictionary, is placed in the public domain. Between 1988 and 1997, extensive research was conducted during which the language was iteratively refined and revised. After that period, a baseline of the language was compiled, which was followed by a "freezing" period, in which no changes to the language were permitted. This was to ensure that the language remained stable, while people began to adapt and use it. Since the expiration of the "freeze" in 2002, Lojban users are free to make changes to the language and it is since then actively developed by its community[6].

In addition to the official documentation of Cowan (1997a) the LLG also published "What Is Lojban" (Nicholas, 2003), which provides a more concise introduction to the language and briefly examines it from a linguistic perspective. "Lojban For Beginners" (Robin Turner, 2003) takes an approach geared more toward language acquisition.

Lojban is a recognized language (with the ISO language code "jbo"), and is actively used by many speakers throughout the world. The exact number of Lojban speakers is unknown; at its current size the speaker population is difficult to estimate. According to the LLG, as of March 2010 close to two thousand people have ordered one of their Lojban books or signed up to a mailing list; the official Lojban dictionary *jbovlaste* lists 381 registered users[7]. A more realistic number of proficient speakers can be estimated by the official IRC channel, which has an average of 145 users[8] actively speaking the language.

2.1.1 Language summary

Lojban is designed on the complete scope of linguistics, spanning *orthography*, *phonology, morphology*, *grammar* and *semantics*[9]. As we are only interested in a small subset of the language, only a minimal summary is given for the lower levels.

[4] Loglan was further advocated as an *international auxiliary language* (similar to *Esperanto*), although this objective was not sustained. Nevertheless, the language gained notable attention of a growing language community.

[5] The LLG's official Lojban website can be found at `http://www.lojban.org`, which provides publications, dictionaries, learning material and other information on Lojban.

[6] Users can autonomously add and edit word definitions; although this is not allowed for the 1437 root words (gismu). Changes to the grammar are rare, and subject to public discussion. At the current time, there also is no numeric versioning system in place, which sacrifices control over a more fluid evolution of the language.

[7] As stated on `http://www.lojban.org/tiki/Frequently+Asked+Questions+About+Lojban`, accessed June 2014.

[8] According to statistics by `http://irc.netsplit.de`, accessed June 2014.

[9] Although these fields are grounded in *descriptive linguistics*, they are actually used *prescriptively* by the authors, for the purpose of defining the language.

Orthography

Lojban is written in a subset of the roman alphabet (omitting *h*, *q*, and *w*), with the addition of *'* as an extra letter (denoting an /h/ sound), and . representing a mandatory pause. Lojban is written completely in lower case. Proper names may divert from this rule, as upper case may be used to indicate unusual stress, and a comma may be used to indicate vowel separation[10]. Names must always be "lojbanized", so that they follow Lojban orthographic rules. Thus, Lojban text is written very consistently with certain *escape words* allowing the use non-Lojban strings to be embedded. Lojban words must generally be separated by either the pause character, or – more commonly – a form of whitespace.

Phonology

Lojban pronunciation rules ensure an *audio-visual isomorphism*, meaning that there is a bijective mapping between symbols and spoken sounds[11]. For each Lojban sound, a whole range of possible pronunciations is permissible[12]. Lojban phonemes are optimized for distinctiveness, making the spoken language robust to noisy environments. As a consequence of audio-visual isomorphism, Lojban also does not allow unwritten or unspoken language elements. For example, in natural languages *parenthesis* are only present in textual representation, but do not exist in the spoken language.

Morphology

Lojban morphology is extremely regular, so that streams of letters (or streams of sounds, see above) can be uniquely broken down into morphemes. Lojban can be classified as an *isolating language*, in so far as there is no inflection, there are no allomorphs (classes of related morphs which realize a morpheme) and there is a single bijective mapping of words to morphemes.

Lojban does not have *part of speech* like natural languages, but instead uses a simple concept of *word classes*. Classes are hierarchical and unambiguous, and the identification of a word class is possible by pure syntactic analysis of a lexeme. Therefore, the class of a word can always be uniquely identified without any knowledge of the lexicon. For example, *gismu* always have a length of 5 letters and have the special sequence of consonants and vowels *CVCCV* or *CCVCV.*

Figure 2.2 shows a graphical overview of the word class hierarchy.

- **cmavo** are function words[13]
- **cmene** are proper names[14]
- **brivla** are *content words*, defined as predicate relations. They have between 1 and 5 arguments, and can be further divided into the following three subclasses:
  - **gismu** are basic root words of the language[15]. The design of this set is elaborated in Section 2.4.3.
  - **lujvo** are modifications or combinations of gismu, which have been assigned one concrete meaning. The formation of lujvo is elaborated in Section 2.1.2.
  - **fu'ivla** are loan words used for concepts in a narrow field or jargon words. They are used only as a last resort, and are mostly present for names of species, elements, dishes, etc. (e.g. `spageti` for "x_1 *is Spaghetti made of* x_2")

[10] An example is the French name *Juliette*, which could be represented in Lojban as `DJUli,et`, indicating the stress on the first syllable and a pause between /i/ and /e/, approximating the French pronunciation.

[11] Most of Lojban is pronounced exactly as English (*c* being pronounced as /sh/). The interested reader can find extensive pronunciation guides in (Nicholas, 2003).

[12] The definition of Lojban sounds as *allophones* (a set of possible phones for one phoneme), aims to ease pronunciation for speakers of different languages. For example, many natural languages have an /r/ sound which slightly differs, but they are all valid representations of the character *r* in Lojban.

[13] Although *cmavo* require the most explanation, they are not further discussed in this work. Extensive explanations are found in (Nicholas, 2003) or (Cowan, 1997a).

[14] Here we adapt the naming convention of (Cowan, 1997a), so "cmene" refers to the class of *name words*. By now this definition is refined and "cmevla" (cmene valsi = "name word") is considered the correct term.

[15] The set of *gismu* was created using an algorithm to combine the languages English, Hindi, Chinese, Russian, Spanish and Arabic. The motivation of combining these languages is primarily the cultural neutrality of Lojban.

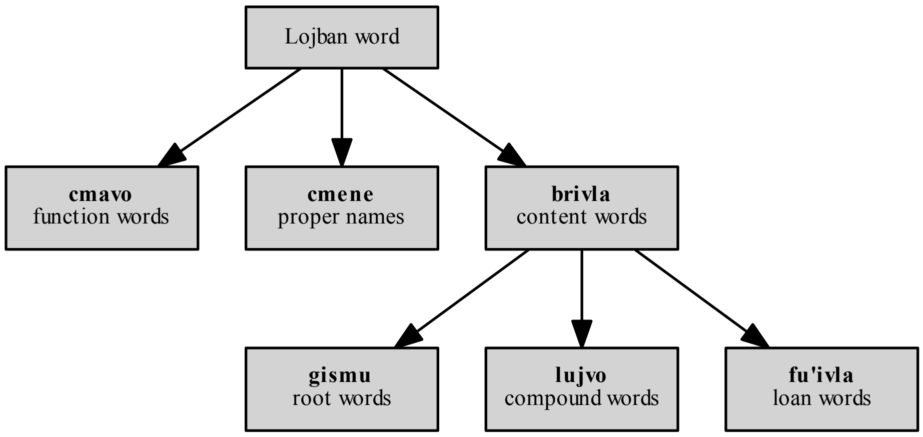


Figure 2.2: Hierarchy of Lojban word classes

The class of words most relevant to this work is that of *brivla*. All brivla are defined as predicates, and they constitute the largest subset of all predicate words[16].

Semantics
Without going into too much detail regarding the Lojban grammar, the basic structure of sentences can be summarized as follows. A single expression within a bridi is marked as the selbri (the main predicate). All other expressions are substituted, in sequence, into the argument slot of that predicate. Thus, the *order* of the arguments in a bridi determines their semantic role. To express the statement "I sell you a car", one has to consider the definition of an appropriate predicate; in this case

`vecnu`: x_1 *sells* x_2 *to* x_3 *for cost* x_4

Then the arguments are put into the right sequence matching the definition, in this case $x_1 = I$, $x_2 = car$, and $x_3 = you$. In Lojban, the complete sentence can be expressed as

```
mi cu vecnu le karce do
```

The words `mi` and `do` refer to "I" and "you", respectively. The word `cu` indicates that the selbri follows, allowing it to be placed anywhere in the sentence. However, in order to make Lojban resemble English sentences, the preferred order generally is

$[\text{arg}_1]$ selbri $[\text{arg}_2][\text{arg}_3]$..

The argument "a car" is `le karce` . The word `karce` is itself a gismu, illustrating an important property of Lojban. All concepts are described as predicates, including those which correspond to simple nouns in natural languages. It is defined as

`karce`: x_1 *is a car or other vehicle for carrying* x_2 *propelled by* x_3

The word `le` essentially converts this predicate into an argument, by referring to something which can be substituted into the x_1 argument of `karce`; thus "a car". By referring to other argument slots of the same predicate

[16] There are also certain *cmavo* (function words) which can act as predicates, but these will be disregarded for most purposes of this work.

different concepts can be expressed, such as *passenger [of a car]* for x_2 and *engine [of a car]* for x_3. In this way, Lojban attempts to remove *redundancy* in word definitions[17].

In Lojban sentences, it is possible to *elide* certain arguments. However, this is semantically equivalent to substituting the word *something*[18] into to respective slot. Lojban brivla are always assumed to fill each of their arguments; they are either specified (explicitly) or assumed to be known from context (implicitly). In the example sentence, this applies to the x_4 argument of `vecnu`. Although we did not specify it within the sentence, there is assumed to be some argument for *cost*; in this case we are just not interested in expressing it.

2.1.2 Word formation

An aspect of Lojban which is particularly relevant to this work is its means of word formation. The two primary ways to obtain a new predicate are

1. ***tanru***. These are sequences of Lojban words which create a *binary metaphor* that is created on-the-fly instead of being formally defined. Any two Lojban expressions which can act as a selbri can be combined. The first predicate is used to modify the second one in an arbitrarily generic or specific way. This *intentional ambiguity* is often described as "the heart of Lojban semantics". As an example, in the tanru `nixli ckule`, the predicate `nixli` (x_1 *is a girl*) modifies the predicate `ckule` (x_1 *is a school*) to obtain a meaning similar to "*girl's school*"[19]. This tanru does not specify in what way the head predicate is modified, so it could be a school for girls, a school run by girls, or even a school which is a girl. The assumption is that the listener should be able to deduce the intended meaning.
2. ***lujvo*** are combinations of Lojban words which have been fused together by a set of morphological rules. Their meaning is fixed and a concrete definition is added to a dictionary. There are a number subcategories of lujvo which can be described.

   a) multiple brivla are *combined*. This is generally done when a tanru would be too ambiguous, or because a tanru is used so frequently that it is desirable to define a fixed meaning. As an example, the lujvo `sorpre` (*crowd*) and the gismu `karce` (*vehicle*), were used to define the word `sorprekarce` to mean *bus*. Its full definition is

      x_1 *is a bus for carrying passengers* x_2 *propelled by* x_3

   b) the argument structure of a brivla can be *permuted*. As an example, by swapping the first and third argument of `vecnu` (the seller and the buyer), one obtains the predicate `terve'u`, which corresponds to the English word *buy* rather than *sell*. Unsurprisingly, the definition for `terve'u` is

      x_1 *is a buyer purchasing goods* x_2 *from seller* x_3 *for cost* x_4

   c) certain cmavo are used to *modify* the meaning of a relation. As an example, the *"property abstractor"* `ka` "x_1 *is the property exhibited by [relation]*" turns the predicate `jikca` (x_1 *socializes with* x_2) into `kamjikca` (x_1 *is social*).

These features of the language have a range of useful properties. Consider the example of the Lojban predicate for *bus*, depicted in Figure 2.3. It can be seen that lujvo are created hierarchically from other brivla (not restricted to binary combinations). The resulting compound word was created by a language user following a set of syntactic rules. The depicted tree can be regarded as the *etymology* of a Lojban word. It is not explicitly documented in the data, but the lujvo creation algorithm is designed in such a way, that it can be easily *reverted*. Thus, given the lujvo `sorprekarce` and a dictionary, the tree can be easily deduced. An important observation is that this structure creates a *well-defined hierarchy* between Lojban words. It can generally be inferred for any tanru t being composed of the source brivla $b_1, b_2, \ldots, b_n$, that t *is a kind of* b_n. In the given example, one can infer that `sorprekarce` is a kind of `karce`; meaning that a *bus* is a kind of *vehicle*. Consequently, the Lojban lexicon contains a natural tree hierarchy with *kind-of* relations, directly modeling *hyponymy* relationships within the vocabulary. Furthermore, the

[17] Lojban predicates are used to define both the *relations* and their *roles*. In English, for each concept, there exists not only a lexical item for the relation, but also distinct lexical items for each of the roles participating in the relation. Consider the word "hunt", which in Lojban is defined as `kalte`: x_1 *hunts* x_2. This brivla not only defines the relationship between x_1 and x_2, it also functions as a word for these roles. In English, there exist the additional lexical items "hunter" for x_1 and "prey" for x_2.

[18] The cmavo `zo'e` is used to fill an argument slot with the value *unspecified*. For convenience the cmavo `fa`, `fe`, `fi`, `fo`, `fu` can be prefixed to skip to the 1st, 2nd, 3rd, 4th or 5th slot respectively.

[19] tanru can be used in a recursive way, and they are by default always *left-grouping*. `cmalu nixli ckule` refers uniquely to the grouping *((little girl's) school)*.

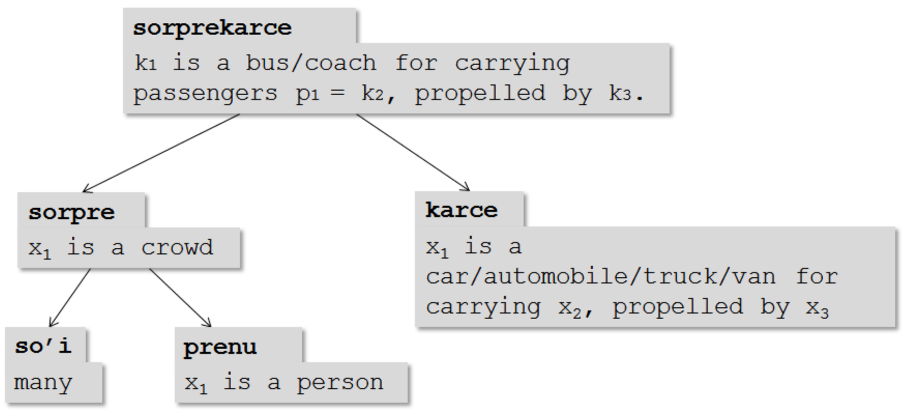

sorprekarce
k1 is a bus/coach for carrying passengers p1 = k2, propelled by k3.
sorpre
x1 is a crowd
karce
x1 is a car/automobile/truck/van for carrying x2, propelled by x3
so'i
many
prenu
x1 is a person

Figure 2.3: The hierarchical etymology of the Lojban word for *bus*

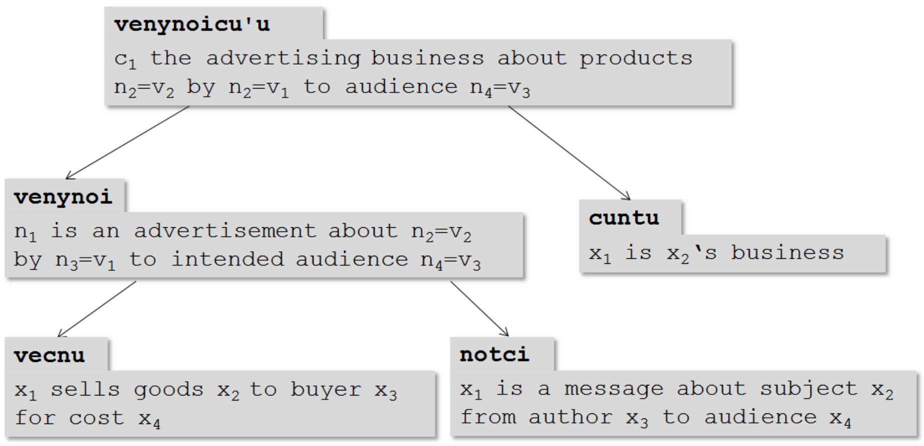

venynoicu'u
c1 the advertising business about products n2=v2 by n2=v1 to audience n4=v3
venynoi
n1 is an advertisement about n2=v2 by n3=v1 to intended audience n4=v3
cuntu
x1 is x2's business
vecnu
x1 sells goods x2 to buyer x3 for cost x4
notci
x1 is a message about subject x2 from author x3 to audience x4

Figure 2.4: The hierarchical etymology of the Lojban word for *advertising business*

argument names indicate the semantic relationship of each argument slot with respect to the root predicates. This *argument etymology* is indicated by the letter of each variable within a definition, which refers to the initial of the respective root predicate. Consider for example the Lojban predicate for "advertising business" visualized in Figure 2.4. This predicate includes the semantic information that the x_1-argument is a kind of *business*, the x_2-argument is both a kind of *message* and a type of *goods* to be sold, the x_3-argument is a kind of *seller*, and the x_4-argument is both a kind of *audience* (of the message), and a kind of *buyer* (of the product). In addition to this, many other *semantic relationships* between lexemes can be inferred.

1. The semantic relationship between the English lexemes *buy* and *sell* is described as *relational antonymy*. In Lojban, this relationship (between the brivla `vecnu` and `terve'u`) is perfectly explicit; it merely flips the perspective between buyer and seller on a syntactic level.
2. The semantic relationship between lexemes such as *hot* and *cold* is described as *scalar antonymy*. Lojban has a whole set for the creation of such pairs, such as `na'e` (other than), `to'e` (absolute opposite), or `no'e` (neutral opposite; "not really"). However, just as in natural language, these words are sometimes present as syntactically unrelated primitives, e.g. *good* = `xamgu` and *bad* = `xlali`, and sometimes morphologically constructed, e.g. *young* = `citno` and *old* = (`to'e` + `citno`) = `tolci'o`. This means that in some instances antonymy is apparent on the syntactic level, but in others it is not.
3. From the use of event abstractors, as defined in (2c), many more semantic relationships arise. For example, the relationship between words such as "marry", "wedding", "married" can be described in Lojban with certain abstractors, for *processes*, *events* and *states*.

2.1.3 Advanced language features

So far, we have seen how Lojban expresses simple predicate-argument relations. However the true richness of the language manifests in the gap that still exists to achieve the expressiveness of natural language. These important features are merely listed in the following. Despite their relevance to the language, this work does not attempt to explain all of them. Instead, we give a coarse summary, without delving too much in detail.

Firstly, Lojban contains all features to fulfill the requirements of natural communication. This includes

- different types of discourse elements, such as *statements*, *observatives*, *requests,* or *questions* (binary, interrogative, etc.)
- *negations* and various constructs for opposites
- *vocatives* such as greetings and other discourse markers
- *attitudinals* for marking attitude or emotion
- *coreference* expressions, such as pronouns, cataphora, etc.
- *tense markers* for specifying time and space
- *quantifiers* for expressing cardinality and determiners
- *quotations, citations* and other literal embeddings of language

In addition to that, Lojban also has features which are not commonly associated with spoken languages.

- the use of *variables* and *functions* (a *typing* system was also suggested)
- the integration of nearly all mathematical expressions as Lojban words
- the use of logical quantifiers and conjunctions in non-mathematical discourse
- spoken meta-linguistic elements, such as a *backspace* word, *toggles*, or *escape characters*

Concludingly, Lojban is able to express a superset[20] of everything that can be said in a natural language.

2.2 Existing resources

In this section, an overview of available secondary resources on the Lojban language is given, including dictionaries, corpora, parsers, and other community efforts.

2.2.1 Dictionaries and wordlists

The lexicon of Lojban can be considered one of its most important features. However, the nature of the language is an iterative *revision* and *extension* of its vocabulary, allowing growth and flexibility. Nevertheless, too much freedom in word usage would make Lojban become too volatile, sacrificing its consistency. To balance out these requirements, the definition of Lojban words is twofold. Firstly, the LLG has published a set of word lists, which act as an official standard. Secondly, a community-driven dictionary is provided which can be edited by the users of the language. Although the general effort is to unify all Lojban information into a single source, a lot of the data is spread across distinct resources.

The official word lists contain the currently accepted *gismu* and *cmavo*. This set of words is not allowed to be edited. A Lojban word is fully defined by a definition string. A set of English gloss words is optional, and should not be considered for the deduction of the meaning. Listing 2.1 shows an example definition of the word `vecnu`. Note that the definition string generally contains a *description* of the argument in square brackets, and *synonymous terms* separated by slashes. Other official word lists contain thesauri, disambiguation dictionaries, or lists of multiword expressions. The "oblique keyword" list further maps each single argument slot of a given brivla to an English expression. An example is shown in Listing 2.2, which describes each argument slot of each brivla by itself. An overview of the official word lists is given in Table 2.1.

Although these official word lists are important for ensuring the stability of the language, the most comprehensive resource is the community-edited dictionary *jbovlaste*. It is a set of distinct *unidirectional* dictionaries. Thus, entries *eng→jbo* and *jbo→eng* are independent of each other. In addition to English, *jbovlaste* contains entries for 62 other languages (with various degrees of completion). Among these languages, there also is a monolingual

[20] An example of Lojban's expressiveness is the term "being enough-th in line", which cannot be expressed in English.

```
Word: vecnu
Type: gismu
Gloss Word: sell
Rafsi: ven ve'u
Definition: x1 [seller] sells/vends x2 [goods/service/commodity] to buyer x3 for amount/cost/expense x4.
```

Listing 2.1: Example definition of the root word *vecnu*

```
vecnu1: sell; seller
vecnu2: goods; sold
vecnu3: buyer; sold to
vecnu4: sale cost; price
```

Listing 2.2: Example entry of the "oblique keywords" dictionary

jbo-jbo dictionary, containing definitions for 1645 words. The English version of *jbovlaste* contains 8486 *jbo-eng* entries and 11,495 *eng-jbo* entries (as of June 2014). The database is provided in XML format. Listing 2.3 shows an exemplary entry for the Lojban word `barkla`. The entry lists the word and its type (*lujvo*), along with a definition string. It should first be noted that *jbovlaste* includes no "gloss words", as it is a unidirectional dictionary. It also omits the gismu-etymology of the word (`bartu + klama`), which is only annotated in form of a note. It can be seen that the definition is a latex string, which would render as

$$x_1 = k_1 \textit{ exists/goes out/outside from } x_2 = k_2 = b_1 \textit{ with route } x_3 = k_3 \textit{ with transportation method } x_4.$$

The semantics of this naming convention was explained in Section 2.1.2. In this case, the letters refer to the scope of the lujvo (x), and its respective argument places of the parent words `bartu` (b) and `klama` (k). Correctly parsing these definition strings is thus an important implementational detail.

2.2.2 Corpora

The gathering of Lojban text is not trivial. Search engines do not recognize Lojban as a language, making it impossible to restrict a search to Lojban (searching for Lojban strings generally yields discussions about Lojban, as opposed to actual corpora). To the knowledge of the author, no attempt at automatically *crawling* the web for Lojban was made so far. However, the LLG's official website contains a collection of Lojban texts[21]. From these collections, a subset was selected and processed for the purpose of this work.

Monolingual
The largest monolingual corpus is a public chatlog of the official Lojban channel (`irc://irc.freenode.net/#lojban`), pre-filtered to contain only Lojban text. This corpus dates back more than a decade and contains over 6

List	Description	Entries
gismu lists	textual definition + gloss word of all Lojban gismu in 63 languages	1437*
simplified	a simplified English definition string for better readability	1342
noralujv	early list of some lujvo, partially superseded by newer words	5272
cmavo list	official definition of cmavo (functional Lojban words)	1091
disambiguation list	a list disambiguating multiple senses of English words to brivla	4143
oblique keywords	English labels for argument slots of each brivla	3542
functional list	inventory of multiword Lojban expressions for common English terms	2909
thesaurus	early attempt at a thesaurus, grouping gismu by semantic fields	1342
antonyms	gismu grouped as opposites and contraries	1342

Table 2.1.: Official Lojban wordlists
*not all of the 1437 gismu are defined for each of the 63 languages.

[21] Lojban corpus collection: `http://www.lojban.org/corpus/`, accessed June 2014.

```
<valsi word="barkla" type="lujvo">
  <definitionid>13091</definitionid>
  <definition>
    $x_1=k_1$ exits/goes out/outside from $x_2=k_2=b_1$
    with route $x_3=k_3$ with transportation method $x_4$.
  </definition>
  <notes>
    Omitted: x5 = klama2 (destination) = bartu1 (something external).
    Cf. {bartu}, {klama}, {zevykla}.
  </notes>
</valsi>
```

Listing 2.3: The XML entry for the Lojban word *barkla* (exit) as given by *jbovlaste*

Corpus name	**Description**	**sentences (words)**
chatlogs	public chatlogs of the official channel	680,556 (6,415,243)
jbowiki	Lojban Wikipedia	9,028 (175,839)

Table 2.2.: Summary of Lojban corpora

million words (37 MB). There is also a Lojban edition of Wikipedia[22], containing roughly 1000 articles. However, most of these articles are comprised of just one sentence, which is why this corpus is not suitable as parallel text. Table 2.2 lists monolingual corpora, including size and the name by which it will be referred to.

Although the monolingual corpus is already of high quality, some preprocessing was performed in order to clean out non-Lojban text. This preprocessing is documented in Appendix B.1.

Bilingual

Although not nearly comparable in size, there is some bilingual text available. In order to create a parallel corpus from this, a set of documents was selected which is reasonably alignable to an existing English text. Most of these corpora are translations of fiction, and the original English version could be easily obtained[23]. Furthermore, a community created collection of example sentence exists, which lists a whole collection of possible English translations for a given Lojban expression. Table 2.3 gives an overview of the parallel corpora and their sizes[24].

These available parallel corpora have been aligned. Sentence level alignment was performed using *hunalign* (Varga et al., 2007), whereas a word-level alignment was obtained using *GIZA++* (Och and Ney, 2003). The exact procedure for alignment is documented in Appendix B.2.

Corpus name	**Description**	**sentences (words)**
wizard_oz	Translation of L. Frank Baum's "The Wizard of Oz"	3213 (53,353)
alice	Translation of Lewis Carroll's "Alice in Wonderland"	2464 (33,756)
die_verwandlung	Translation of Franz Kafka's "Die Verwandlung"	2030 (26,563)
little_prince	Translation of de Saint-Exupéry's "Le Petit Prince"	1634 (21,174)
princess_and_pea	Translation of H. C. Andersen's "The Princess and the Pea"	481 (2216)
snowwhite	Translation of the German fairy tale "Snow White"	252 (3833)
books	Concatenation of the book corpora above	7666 (108,733)
tatoeba	Collection of example sentences from tatoeba.org	4639 (31,022)

Table 2.3.: Summary of parallel corpora

[22] The Lojban Wikipedia: `http://jbo.wikipedia.org`, accessed June 2014.
[23] All English originals are in the public domain and can be obtained from Project Gutenberg: `http://www.gutenberg.org/`
[24] Some of these works are even available as audio files and could thus function as speech corpora. In this work however, this will be of no further interest.

2.2.3 Software

There is a vast collection of Lojban software available[25]. In particular, there are many projects implementing Lojban parsers, most of which operate on the syntactic level. Some of the more peculiar features of the language (an example being spoken backspace characters) are difficult to realize in formal grammars. As a result, these projects deal with grammar issues in various ways; some are only able to handle a subset of the complete language. Here, only a short overview of the most relevant software is given.

- Parsers and grammars

  CLL formal grammars: a set of official grammars defined by Cowan (1997a)[26]. A *YACC* grammar acts as the de-facto definition of the language, while the equivalent *EBNF* grammar provides a more human-readable form. The LLG also provides an official parser.

  jbofi'e: a command-line tool suite containing a parser which implements the official grammar. As opposed to a simple YACC-generated parser, *jbofi'e* also transforms the syntactic parse tree to a representation containing semantic information. A similar processing step was also done for this work and is elaborated in Section 4.1.1.

  camxes: a *PEG* grammar and a *Rats!*-based[27] parser for the JVM[28]. The hand-modified grammar of *camxes* is one of the most complete for Lojban and was used for parts of this work. As opposed to other grammars it parses Lojban on a purely syntactic level. The grammar processes complete documents (including chapters and paragraphs) and ranges down to a fine-grained level identifying the morphemes in a single parse tree. Sample output of this parser can be seen in Listing 4.2 in Section 4.1.1.

- A simple translator tool, *jboski*[29] transforms Lojban sentences into an English representation. For this, it substitutes the definition strings of the Lojban dictionary in a way which is intelligible to an English speaker; it does not however produce a natural English utterance.
- A corpus search system, *korpora zei sisku*[30] allows the user to browse corpora and view collocations and statistics of Lojban words.
- Software that is more commonly associated with programming languages includes syntactic correctness checkers, syntactic simplifiers, and syntax highlighted editors with tab-completion.

2.3 Applications of Lojban

Besides its use as a spoken language for human communication, there exists a body of research suggesting a whole range of further use cases of Lojban. Most of the attention in academia spiked shortly after the completion of the baseline (2003 - 2005), and declined after that. Some of the work introduced in this section has thus never made beyond the stage of a research proposal; it is nevertheless mentioned here for the sake of completeness.

Lojban as an interlingua

One ambitious line of argumentation suggests that Lojban could function as an *interlingua* for machine translation – as such serving as a semantic meaning representation when translating between a pair of natural languages (Nicholas, 1996). The main idea of interlingual machine translation is a two-step process, in which the source language is first transformed into a language-independent abstract representation, which is then rendered into the target language. It is observed that a primary desideratum of such a representation is the ability to represent all the shades of meaning which can be realized in *any* of its target languages. Therefore, an interlingua should not only be unambiguous, but also *maximally explicit*. It is argued that Lojban is a prime candidate for this intermediate representation, due to its foundation in predicate logic, syntactic unambiguity, and the ability to be highly explicit. Another important requirement for an interlingua is a reasonable coverage of language. As a public community

[25] Most of Lojban-related software can be found at `https://github.com/lojban/`

[26] The CLL grammars can be obtained at `http://lojban.github.io/cll/21/1/`, accessed June 2014.

[27] *Rats!* is a parser generator building on *Parsing Expression Grammar* (Ford, 2004): `http://cs.nyu.edu/rgrimm/xtc/rats-intro.html`, accessed June 2014.

[28] Camxes is documented and available for download at `http://users.digitalkingdom.org/~rlpowell/hobbies/lojban/grammar/index.html`, accessed June 2014.

[29] An online demo of the translation tool *jboski*: `http://www.lojban.org/jboski/`, accessed June 2014

[30] The Lojban corpus search tool is available at `http://www.alexburka.com/~danr/`, accessed June 2014

```
coi mi'e jimpe                         | Jimpe:  Hello, I'm jimpe
> ganai la bab crino gi la erik blanu  | User:   If Bob is green, then Eric is blue.
je'e                                   | Jimpe:  Okay.
> la erik na blanu                     | User:   Eric is not blue.
i ua la bab na crino                   | Jimpe:  Aha! Bob is not green.
> lo cukta na viska                    | User:   Books don't see.
je'e                                   | Jimpe:  Okay.
> xu lo cukta cu viska mi              | User:   Does a book see me?
na go'i                                | Jimpe:  No.
```

Listing 2.4: The output of JIMPE, illustrating the inference engine

effort, Lojban is deemed more likely to cover language breath than academic research projects. Ultimately, it is acknowledged, that albeit Lojban may not be utilizable in its current shape, it would at least provide valuable pointers in the development of an interlingua-based MT system. It is claimed that "[any interlingua based on predicate logic] would resemble Lojban greatly, and designers of such interlinguas would profit from drawing on the results of Lojban design" (Nicholas, 1996).

Lojban for AI research

Another line of argumentation demonstrates Lojban's usability as a human-machine interface, which is employed by a human user to handle complex input and output, which would be unfeasible to achieve with natural language. In early experiments, a semantic analyzer was implemented which interfaced Lojban with a *Prolog* question answering system (Nicholas, 1993). The system accepts a subset of Lojban sentences as input to be stored in a knowledge database. This database can then be queried with simple binary questions, stated in Lojban. The Lojban sentences are translated to quantified statements, which are then stored as axioms, such as $\exists x.\mathsf{gerku}(x) \rightarrow \mathsf{prami}(x, \mathsf{mi})$ ("there exists a dog that I love"). Prolog is then able to do logical inference based on these axioms, and yield a ternary answer (yes, no, unknown).

A sophisticated application motivated by this central idea was eventually realized by Speer and Havasi (2004). They propose a system that communicates with the user in Lojban, as a means of prototyping *conversational NLP*. They argue that understanding natural language on a semantic level is not currently possible – "meeting the computer halfway" by using Lojban would help to make progress in these areas. A further motivation of their system is studying the way people communicate with a computer system. For this, a conversation-driven knowledge engine is implemented. JIMPE (Lojban for "understand"), is a modular system performing all necessary steps to simulate a complete conversational agent. Given a Lojban input text, it performs a series of processing steps to yield a Lojban reply. A syntactic parse of the input is first transformed into a semantic representation. This representation is further reduced into logical expressions, which are then stored into a database as a network of objects. As it is possible to say things that are semantically vague, these logical expressions have probabilistic values, so that the system can make "guesses" on the semantic value of expressions which are unclear. On top of this database, JIMPE uses a simple inference engine, which is used to draw logical conclusions. The conversation system actively outputs inferred knowledge and answers questions of the user. Listing 2.4 shows a dialog with the system and illustrates the inference engine. Interestingly, JIMPE does not use any lexicon or inventory of predicates and relies solely on Lojban's fixed set of syntactic rules.

A similar system has been demonstrated by Meyer et al. (2005). They also implement a system which performs logical reasoning given Lojban prose as input data. As a novel feature they also realize a *speech synthesizer* for Lojban, enabling the reasoning engine to reply *verbally* to the user query.

Another closely related utilization of Lojban has been proposed in the LojLink project (Goertzel, 2005a). The research proposal suggests a knowledge management system using Lojban as an input language. The main interest in this case is entering large amounts of unstructured knowledge into a computer database, which is then stored as semantic relationships. They propose to train knowledge encoders in Lojban, which they then use instead of a formal language. This is justified by an estimated 10-fold speedup over conventional input methods. This encoded information would then be stored in a semantic database, which can be queried. Similar to earlier projects, they suggest an automated probabilistic inference system operating on these semantic representations. The remaining ambiguity in Lojban is handled by a system called *probabilistic term knowledge* (PTL), which propagates uncertainties through the notion of *probability* and *evidence weight*. Ultimately, a user queries the database by asking the system a question in Lojban, which yields a statistical answer with a numeric uncertainty. As a subsequent step,

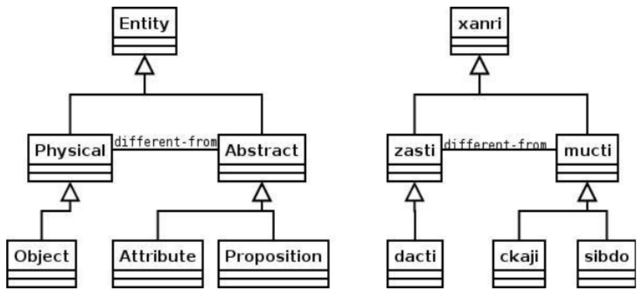


Figure 2.5: A semantic web ontology based on Lojban brivla. (Source: Wirick, 2005)

it was suggested to use machine translation to translate those Lojban representations back to English, effectively building a knowledge engine with Lojban input and English output. Ultimately, the author envisions that the system could "automatically ingest information from free text in an unsupervised way", in an attempt to build an AI reasoning agent.

Extension of Lojban

Lojban++ is a subsequent proposal by Goertzel (2006), addressing a major obstacle related to the language. It is observed that, although the formalism of the language is not difficult to teach, the necessity to acquire Lojban vocabulary may be a major hindrance for the adaption of the language. *Lojban++* attempts to overcome this vocabulary gap by extending Lojban with English vocabulary. The primary modification is the possibility to embed English lexical items within Lojban text, assuming the role of a predicate. As these English words do not have the fixed argument structure essential to Lojban, an argument-naming mechanism is proposed. In case of an English predicate the argument is prefixed with the Lojban cmavo `fi'o`, followed by a description word for the argument. As an example, the sentence "go from Atlanta to Boston by car", would be expressed in Lojban as

```
klama fi la .atlantas.  fe la bastn.  fu le karce
```

In case the speaker is not familiar with the Lojban word `klama`, they can instead substitute it with the English word *go*. In this case, instead of employing the fixed argument structure, the arguments *Atlanta*, *Boston*, and *car* must instead be specified literally.

```
go fi'o source Atlanta fi'o destination Boston fi'o vehicle car
```

Concludingly, *Lojban++* is a superset of Lojban, which is a compromise aiming to balance simplicity with familiarity.

Lojban as an encoding language

Another more practical utilization of Lojban is its use as an encoding language for the semantic web (Wirick, 2005). In this work it is first argued that the ontologies for encoding relationships in the semantic web, such as SUMO (Pease et al., 2002), use classes which have more or less arbitrary English names. The authors thus suggest creating a similar semantic ontology using Lojban brivla as names for these classes. The rationale behind this is that both the concepts in this hierarchy and their respective terms would be unambiguous. Figure 2.5 shows how the classes in SUMO could be renamed with Lojban brivla. In addition to the advantage in labeling, the authors observe that Lojban syntax is well-suited for semantic annotations, such as describing cardinalities of objects with Lojban quantifiers.

Further it is demonstrated that Lojban prose can be easily parsed to reference the defined ontology. A parser is implemented which yields an XML annotation of the predicate-argument structure, given plain text of Lojban prose.

2.4 Linguistic analysis

Before regarding Lojban as a semantic ontology it is briefly analyzed from a linguistic perspective. First, a quantitative analysis will be performed by using the `chatlogs` corpus for historical data. On the theoretical level, we then proceed by analyzing the intricate notion of *ambiguity* with respect to both Lojban and natural languages. We

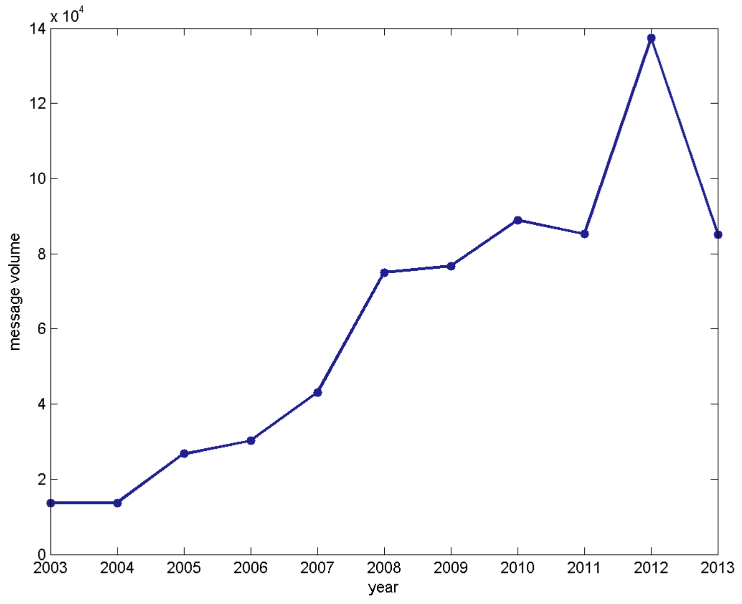


Figure 2.6: Lojban usage over time, measured with the IRC corpus.

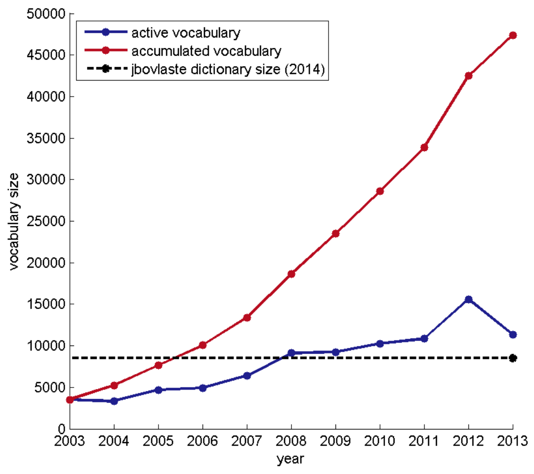
conclude this analysis by discussing the rationale behind the creation of gismu (the root words), and see how they relate to *semantic field* theory.

2.4.1 Quantitative corpus analysis

Through the scattering of Lojban speakers throughout the world nearly all communication in the language takes place online. The advantage of this is that the data obtained from the `chatlogs` corpus gives an almost complete picture of the language. By measuring the *message volume* of Lojban utterances over time, it is possible to infer a steady usage growth as illustrated in Figure 2.6. The spike in 2012 remains to be explained (popularity in the language may has been sparked due to a mentioning in public media).

Furthermore, it is possible to measure the growth in vocabulary. Figure 2.7 shows Lojban vocabulary size, measured by word occurrences in public chat. It can be seen that the growth is approximately linear. We can infer that many words are coined, which are never used again. It could be the case that these words are never accepted into the vocabulary, or that they are simply too specific to be ever used again. Therefore, the *accumulated vocabulary* should be regarded as an upper bound of Lojban words in existence (nearly 50,000), whereas the *actively used vocabulary* serves as a more accurate assessment of the true size of the language (about 11,000 words). By regarding only new words which re-occur in the following year, about 637 words are created per year. Thus, a new Lojban word is created every 14 hours. Concludingly, the official dictionary (8486 entries as of May 2014) cannot catch up with the true amount of Lojban words. Devising a sophisticated model on the formation and decay of vocabulary (especially in context of an emerging, and well-documented language such as Lojban) is an interesting area of research, which is however not further elaborated in the scope of this work.

To examine if Lojban vocabulary behaves similar to that of natural language, we can further look at its frequency distribution. For natural languages, *Zipf's law* states that when ordering words by their frequency, the rank r of a word is inversely proportional to its frequency and can be described with the power law $f(r) \sim r^{-z}$. Interestingly, Lojban vocabulary does in fact follow such a distribution. Figure 2.8 shows its frequency distribution on a logarithmic scale, in comparison to two natural languages (German and English). Towards the 10^3-th rank, the Lojban curve slightly deviates from the power law. This may be explained by newly coined words which will eventually



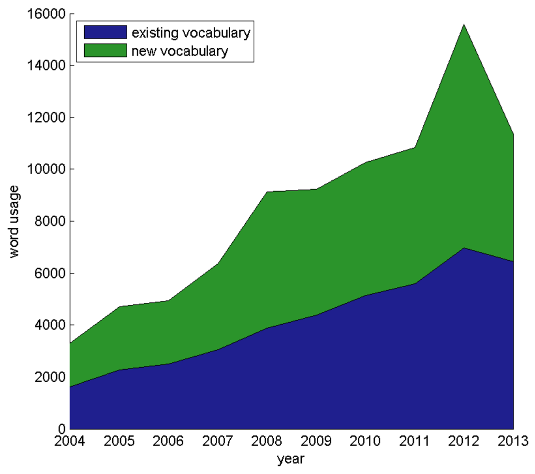


Figure 2.7: Lojban vocabulary growth, measured with the IRC corpus. On the left chart, *active vocabulary* refers to the set of lexical items occurring at least once within a given time range (one year), whereas accumulated vocabulary refers to the size of the union of these sets. Therefore, *accumulated vocabulary* includes words which may have been coined on-the-fly but which not have been formally adopted into the language, or which have died out over time. As a reference value, the size of the *jbovlaste* dictionary is shown as of 2014 (no historic data is available)

On the right chart, *new vocabulary* refers to lexical items which occurred first within a given year, whereas *existing vocabulary* refers to previously seen, reoccuring items.

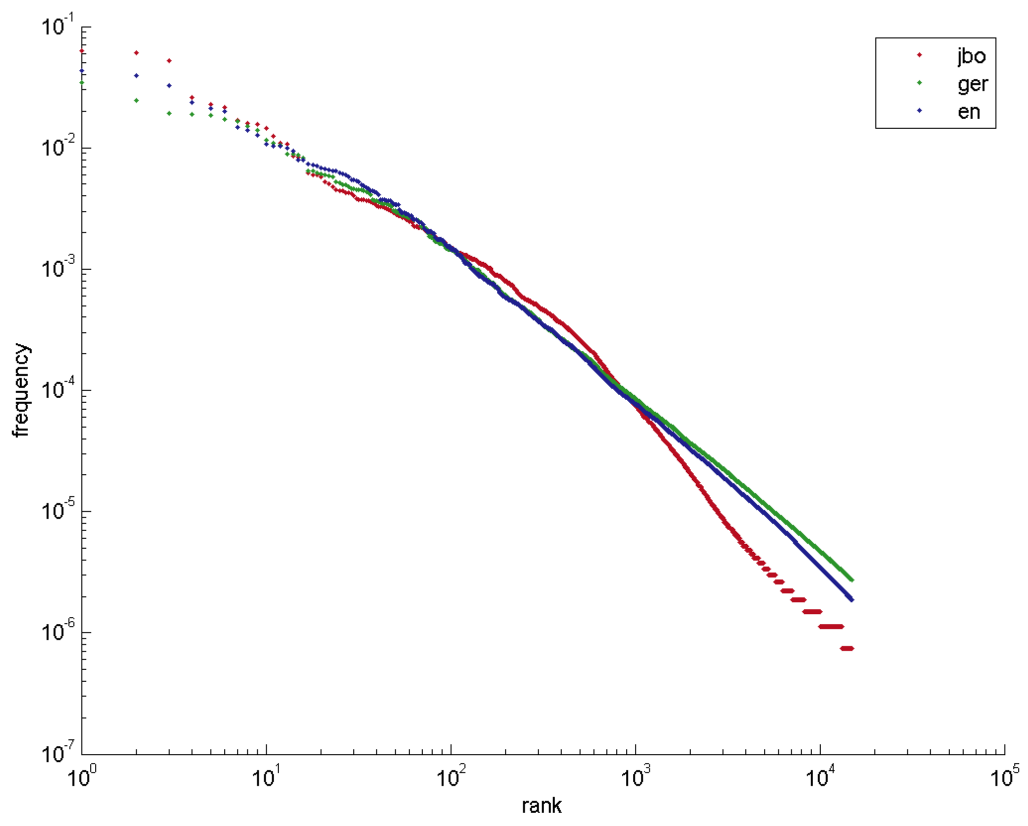


Figure 2.8: Frequency distribution of Lojban vocabulary, compared to natural languages. Note that all languages are cut off at $r = 15000$.

die out during the evolution of the language[31]. Nevertheless, it can be concluded that Lojban does not behave fundamentally different from natural languages.

2.4.2 Ambiguity

The claim of non-ambiguity is crucial to Lojban, so it deserves some clarifying remarks. In fact, in natural languages ambiguity occurs on all levels of linguistic analysis, and is a central challenge to all classical tasks of NLP. We will now briefly summarize these types of ambiguity (at the example of the English language) and explain whether and how Lojban distinguishes itself in this regard.

ambiguity is the presence of two or more distinct meanings or interpretations within a single sentence or sequence of words

2.4.2.1 Lexical ambiguity

Lexical ambiguity arises when words have more than one meaning. This can manifest on the orthographic, phonetic, morphological or semantic level.

Homonymy and polysemy

Homonymy (in here loosely used to mean both *homographs* and *homophones*) describes groups of words with the same spelling or pronunciation which have different, distinct meanings. Lojban does not have this kind of ambiguity, because such words are not present. On the orthographic and phonetic level, any ambiguity is already ruled out by audio-visual isomorphism, resolving such issues as *two* and *too*; but also true homonyms such as *can* (be able to) and *can* (container).

Polysemy refers to a similar set of words which have different, but closely *related* concepts. An English example would be the word *bank*, which is polysemous in so far that it can refer either to the financial institution or the building where that institution offers its services. Lojban gismu are generally defined with such a broad sense that they allow the same polysemous use. The Lojban word for bank, banxa is defined as "*x_1 is a bank owned by x_2 for banking functions x_3*". Although Lojban always provides the tools to be sufficiently precise, in this case specifying a "*bank-building*", it is also acceptable to use the gismu to mean *bank* in its sense of a building. As Lojban is spoken between thinking human beings, the listener is usually assumed to be capable of such basic inferences based on context. Concludingly, it is important to note that Lojban sometimes permits polysemous ambiguity, but it strictly avoids any homonymy; in this example, *bank* could never be used in the homonymous sense of "*the land alongside a river or lake*". This property of the language is elaborated in Section 2.4.3.

Morphological ambiguity

Another type of lexical ambiguity arises for words which could have been morphologically composed in more than one way. Consider for instance the word "unlockable". There are two possible morpheme constituent structures, one being (un-(lock-able)) which means "*that cannot be locked*", the other being ((un-lock)-able) meaning "*that can be unlocked*".

This kind of ambiguity also does not exist in Lojban. Modifying cmavo which serve the role of functional morphemes such as the prefix *(un-)* always have a well-defined grouping rule (by default, Lojban is always left-grouping).

2.4.2.2 Structural ambiguity

Structural ambiguity occurs when a whole sentence exhibits a constituent structure which can be interpreted in more than one way. This is generally the consequence of at least one lexically ambiguous word. There are many famous examples of structurally ambiguous sentences, such as

[31] In fact, Dorogovtsev and Mendes (2001) have theorized that the true frequency distribution of a language can be described more accurately with two power law functions, in which one covers the region of core vocabulary (called the *kernel lexicon*), and the other covering the evolving parts of a language. It is quite plausible that due to the small size and rapid evolution of Lojban, this effect is much more pronounced. The fact that the frequency distribution seems to curve at roughly rank $\sim 10^3$ coincides with the size of the artificially frozen set of 1437 root predicates, which would support this theory.

Flying aircraft may be hazardous.

which can be be interpreted as $[_S [_{VP}$ Flying$] [_{NP}$ aircraft$]][_{VP}$ may be$][_{ADJP}$hazardous$]$ – the act of flying an aircraft is hazardous, or $[_{NP}$ Flying aircraft$][_{VP}$ may be$][_{ADJP}$ hazardous$]$ – the aircrafts themselves are hazardous when flying.

A great effort was put into the design of Lojban to eliminate this kind of ambiguity completely. The grammar of Lojban is formally proven to be unambiguous, and the *function* of each word in an utterance is always uniquely deducible.

2.4.2.3 Semantic ambiguity

Semantic ambiguity may be a disputable term, as all ambiguity is related to semantics. It does however describe a class of ambiguity which (although it may arise from the previously discussed problems) can only be resolved on the semantic level.

Contextual ambiguity

One instance of this can be described as *contextual ambiguity*. Consider for instance the English sentence

The chicken is ready to eat.

This can obviously mean that a meal, consisting of chicken, has been prepared and is ready for consumption. It could however also carry the meaning that a hungry chicken is waiting for food – picture a cartoon character holding cutlery. In Lojban, it is *possible* to express such an ambiguity if desired.

`le jipci cu bredi lo nu citka`
(the chicken) is ready for (the event of- (x_1 eats x_2))
(the chicken) is ready for (eating)

Obviously, Lojban also provides more precise versions of this statement; in this case the equivalent of (the chicken, it_1) *is ready for* (x_1 *eats* it_1) or, more correctly, by specifying the sumti in the expression as `le rectu be jipci` (*chicken-meat*). In practical use however, speakers may not feel the need to perfectly specify their utterances on such a detailed level. As a consequence, Lojban often contains such *underspecifications*, which are assumed to be correctly understood by simple reasoning and contextual knowledge.

Metaphorical ambiguity

Another case of semantic ambiguity does not even require the knowledge of a specific context, so called *metaphorical ambiguity*. As an example, consider the English sentence

I did not sleep for 10 days.

Although this sentence appears to state that the speaker remained awake for a continuous span of 240 hours, anybody hearing such an utterance would identify it as *figurative speech*, simply because such a thing is not physically possible[32]. Human beings can generally deduce figurative speech by *world knowledge*. The philosophy of Lojban discourages from using the language in such a metaphorical way. The reason for this is that metaphors are usually grounded in the culture of the speaker, and thus may not be understood by speakers of different cultures. Nevertheless, it provides a function word serving as a marker for metaphoric expressions, altering the listener to a culturally dependent construction. Arguably, such a specification may not be enforceable on a whole language community, and as a consequence the existence of figurative speech in plain Lojban text cannot be completely ruled out. Variants of metaphorical speech are idioms, sarcasm and irony, which in fact pose difficulties even for human listeners.

Modifier vagueness

As a last example, consider the expression "*religious leader*". Here, it is unclear in what manner the adjective *religious* modifies the meaning of the head word *leader*. It could be "*a leader who happens to be religious*", or a "*leader of some religion*". Lojban supports this exact type of ambiguity in tanru constructs (see Section 2.1.2). In

32 Unless, perhaps, when writing a thesis.

NLP task	Solution in Lojban
word segmentation	algorithm based on consonant-vowel pattern, or on whitespace*
sentence segmentation	split at *new-sentence* separator `i`, or *new-paragraph* separator `ni'o`
POS tagging	algorithm based on consonant-vowel patterns
syntactic parsing	based on formal grammar
word sense disambiguation	not needed
named entity recognition	identified by name descriptor `la`
coreference resolution	entities in a text are assigned fixed variables with `goi`

Table 2.4.: Trivial solutions for classical NLP tasks in Lojban
*whitespace is often not formally required, but practically all Lojban is written with whitespace

fact the English example can be directly translated to the Lojban expression "`lijda jatna`", to obtain the same facet of meaning.

Concludingly, Lojban does not attempt to eliminate *semantic* ambiguity completely. In fact, its creators acknowledge the fact a certain level of ambiguity is crucial to human communication:

> *Lojban [...] must support the same breadth of human thought as natural languages. Every human being has different meanings attached to the words they use [...] so it is impossible to eliminate semantic ambiguity [...] completely. Rather, Lojban attempts to minimize [it] by removing the clutter and confusion caused by other forms of ambiguity. (Nicholas, 2003).*

This removal of "low-level" ambiguity already eliminates a large portion of problems which have to be addressed for natural languages. It should be obvious that most NLP tasks can be solved extremely trivially for Lojban. Table 2.4 gives an overview of these tasks and shows how they can be accomplished for Lojban text.

2.4.3 Lojban root words and semantic spaces

The 1437 gismu are called the *root words* of Lojban and all other meaning is derived from them. Therefore, the gismu set must be a sufficient base to express all possible meaning. In fact, there have been numerous attempts at creating such a *sense inventory*. At one end of this spectrum, these inventories try to identify a set of *primitives*; whereas on the other end of the spectrum, the attempt is to distill a set of words precisely referring to one concept, which cannot be further refined.

The gismu set is certainly not minimal. The linguistic theory of semantic primitives postulated by the *Natural semantic metalanguage* (NSM) suggests a much smaller number of primitives (Goddard, 2008). It states the existence of a small set of roughly 70 language-independent atomic concepts called *semantic primitives*. These atomic units of meaning are defined to be *irreducible* and can be used to express all other concepts.

On the other end of the spectrum, the vocabulary of meaning is not constructed bottom-up, but top-down. Accordingly, a tree hierarchy is constructed in which the leaves correspond to non-refinable concepts. Attempts at creating such a "hierarchy of the universe" dates back hundreds of years[33] and ultimately provided the foundation for the creation of thesauri and hierarchical databases such as WordNet.

Within this spectrum of sense inventories, Lojban can be described as a compromise solution. Gismu are neither atomic nor especially precise. Instead they were selected according to the *frequency* of keywords in a set of natural languages and therefore rather arbitrary. However, they have been constructed through an iterative process of refinement while being actively used within the language. In Section 2.4.2.1 we have already observed that Lojban words sometimes cover a very broad range of meaning. To better explain this design choice we will adopt a visualization commonly used for depicting *lexical overlap*. Figure 2.9 shows an illustration of this concept in form of a *semantic space* containing a number of exemplary lexical items. The 2D-area of this chart is meant to illustrate the space of all meaning. Thus, each point within this space denotes an exact concept. The distance between two points accordingly represents their semantic relatedness. The English words occupy a subset of this space, illustrating that they do not represent a definite concept, but rather a *range* of meaning (in linguistics this is

[33] John Wilkins "*Essay towards a Real Character and a Philosophical Language*" (1668) was the first attempt at a universal language in which words uniquely identify nodes in a concept tree. Based on a proposal from a letter by Descartes, he undertook the task of dividing the universe into thousands of hierarchical categories, which each were assigned two-letter monosyllables. Each word would then describe a unique path in this concept tree.

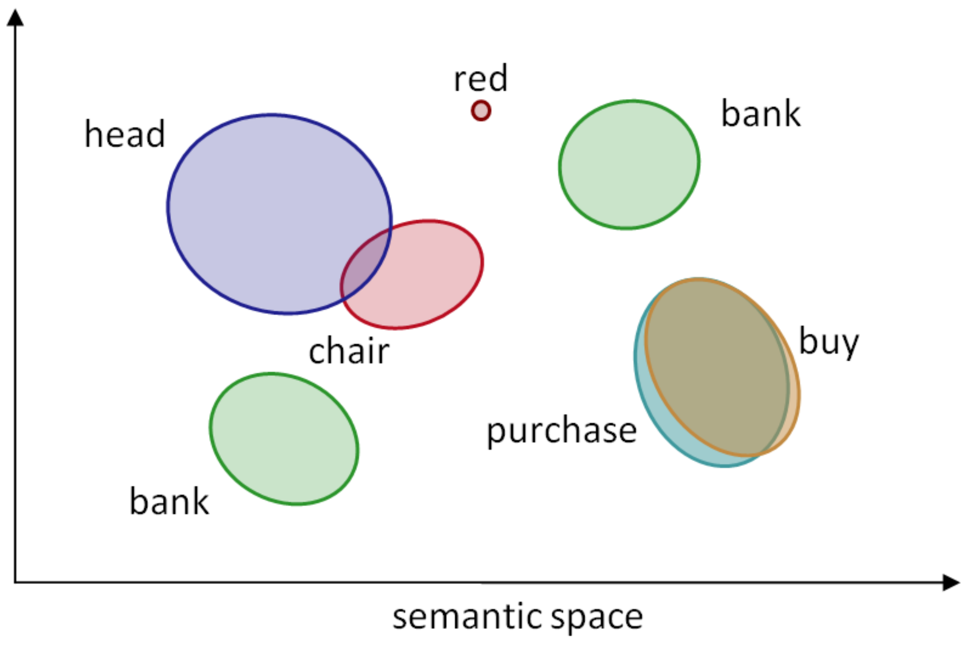


Figure 2.9: Illustration of the concept of a "semantic space".

called the *semantic range* of a word). In this model, different phenomena of natural language can be explained in terms of sets of meaning. Words with near-overlapping meaning sets – such as *buy* and *purchase* – are considered *synonymous*. On the other hand, two words occupying multiple distinct subsets within the semantic space – such as *bank* (one meaning being the financial institution, the other a strip of land alongside a body of water) – are *homonyms*. In this model, *polysemy* can be conveniently explained through the size of a meaning set. The word *red* occupies a much smaller semantic range than the word *chair*. This illustrates the high range of *polysemous* meanings of the latter that exist in addition to its canonical meaning (such as the chair of a meeting).

It can be observed that most endeavors at creating an inventory of senses have either attempted to find a minimal amount of concepts which still covers the complete semantic space (semantic primitives of NSM), or have minimized the semantic range of words, so as to identify fixed points within this space (e.g. WordNet). In stark contrast to that, Lojban does not incur this undertaking. As stated earlier, a certain level of *semantic range* is considered necessary for practical human communication: "*The emphasis in the design [of gismu] has been on comprehensiveness and practicality, rather than minimalism.*" (Nicholas, 1996).

To summarize, Lojban strictly avoids homonymy, but it does allow words to have all kinds of semantic overlap, including polysemy, synonymy, hyponymy and other more subtle notions of shared meaning[34]. The designers of Lojban deemed it more important to allow for some semantic space, rather than enforcing a fixed, fine-grained meaning. Thus, Lojban words "*are explicitly defined, with a unique, though at times broad, sense.*" (Cowan, 1997a). To give an example of the potential breadth in gismu meaning, consider the predicate `tcana`, which can be used to refer to any kind of *node* in a graph. In context, it could be used to mean either a *railway station* or a *router* in an internet network. Although this property may be an impediment at proper formalization, it is often held in high regard and can be considered the necessary compromise between a formal and a spoken language.

> *The construction of ambiguous expressions, together with the implicit assumption that the listeners will be able to disambiguate based on shared contextual knowledge, is essential to the nature of language itself. (Goertzel, 2005b).*

[34] This overlap also manifests in definitions which could be considered *redundant*: the gismu for *mother* and *father* could be easily constructed by combining the gismu for *female* and *male* with the gismu for *parent*.

Nevertheless, it is assumed that the set of gismu "constitutes a complete vector space for the expression of all meaning", which is known as the *Gismu Deep Structure hypothesis*[35].

2.5 Comparison to semantic ontologies

In Chapter 1 we have already given an overview of *semantic resources*, whereas in this chapter we introduced Lojban under the aspect of being an instance of such. We now summarize these observations and assess to what extend Lojban is comparable to a resource such as FrameNet or PropBank. To briefly recapitulate those resources, they consist of an inventory (which may be hierarchical) of concrete concepts which are defined through arguments (which may be labeled) denoting their semantic relation.

Firstly, the most obvious similarity to Lojban is the use of a predicate-argument structure for modeling semantic relations. Lojban also defines an inventory of predicates, which have a (fixed) set of arguments. The major difference is how this inventory is intended to be used. Disregarding their theoretical underpinnings, the practical application of both FrameNet and PropBank is the *annotation* of natural language text. This means that the annotations attached to a sentence are serving as *pointers* to the meaning of the sentence – the annotations by themselves however cannot capture the meaning completely. This contrasts the purpose of Lojban, which is a self-contained lexicon used to express all meaning. As an example, consider the FrameNet annotation of the sentence "*What kind of work does Goodwill do?*"[36]. It evokes the following three frames with their respective frame elements.

What ***kind***$_{\text{Type}}$ of ***work***$_{\text{Working\_on}}$ does Goodwill ***do***$_{\text{Intentionally\_act}}$?

- ***Type***: *nouns denoting* Subtype *of a more general* Category

Subtype	"kind"
Category	"of work"

- ***Working_on***: *An* Agent *expends effort towards achieving a* Goal

Agent	*"Goodwill"*
Goal	*definite null instantiation*

- ***Intentionally act***: *A sentient* Agent *performs an intentional act*

Agent	*"Goodwill"*
Act	*"What kind of work"*

In this example, a number of observations are relevant to the comparison at hand. Foremost, from only the annotations, the content of this sentence cannot be recovered. The frame annotations by themselves do not formalize that the sentence is a *question*, and what is being asked. We also observe that all lexical items within the sentence are annotated if they evoke a frame, even if they are redundant in meaning. In this case, both the lexical item *work* and *do* evoke frames, each containing *Goodwill* as an Agent and having some overlap in meaning.

Although Lojban is not intended for *annotation*, we can however translate the English sentence. A possible Lojban sentence expressing the same meaning could be

`la .gudvil.  gunka ma`
(Goodwill) works on/at *(what?)*

Obviously, this version of the sentence is extremely simplistic and it could be expressed in countless variations (which may preserve more of the meaning and connotation of the original). Nevertheless, content-wise it can be regarded as equivalent to the English version. A key observation is that at the same time it also serves the purpose of semantic annotations given by FrameNet. A relation is specified, in this case x_1 *works on/at* x_2 *with objective* x_3, in which arguments are (partially) substituted. The named entity `.gudvil.` (*Goodwill*) is the agent who is working; furthermore, a function word, the cmavo `ma`, is used to indicate that the listener is inquired to fill in a gap in the second argument of the selbri – which is *the work which Goodwill does*. In summary, the Lojban sentence is not just a realization of the same utterance in a different language, it precisely specifies a concept and its arguments. This property already suggests the use of Lojban predicates for annotation purposes (in this case annotating *work*

[35] Discussions on the GDS hypothesis: `http://www.lojban.org/tiki/Gismu+Deep+Structure`, accessed June 2014.
One can certainly deduce, that if the NSM theory is correct, the GDS hypothesis is also true, because all NSM semantic primitives are already contained within the set of gismu.

[36] Taken from the fulltext FrameNet annotation of the American National Corpus (ANC).

property	FrameNet	PropBank	*Lojban*
argument style	role labels	indexed	indexed (+ hierarchy labels)
hierarchical	✓	✗	✓ (implicit)
non-verbal predicates	✓	in NomBank	✓
overt arguments	explicit	✗	implicit
non-instantiated arguments	✓	✓	✗
extra-thematic arguments	fixed set of frame elements	fixed set of modifier roles	arbitrary

Table 2.5.: Properties of semantic ontologies

to evoke gunka, and mapping its respective arguments to sub-expressions within the English sentence). This is, however, postponed to Chapter 4.

A feature which on a first glance may appear to be lacking in Lojban, are *overt arguments*. In FrameNet null instantiations are used to mark the existence of a frame element which is not lexically present. In the example sentence, the *Goal* of the *Working_on* frame is annotated as a definite null instantiation, meaning that there exists some *Goal*, which is however not explicitly mentioned. In Lojban, this notion is a little more subtle, because *all* arguments defined in the brivla definition are assumed to be substituted with something. In case there is no syntactic element present in the sentence, it is semantically equivalent to a zo'e (=undefined) value. Therefore, all arguments are evoked, and those which are not explicitly filled can be considered implicit overt arguments. In this example, the x_3 argument of gunka, also referring to the *objective* of working, can be considered overt. In particular, we can analyze Lojban with respect to *theta-grid theory* (see Section 1.2.2). Here we can observe that any subset (including the empty set) of arguments may be evoked for a given predicate. Thus, the theta grid contains 2^n rows, for each possible subset of n arguments. Furthermore, the *theta criterion*, stating that each argument fills only one theta-role, and each theta-role must be filled by exactly one argument, is trivially true for Lojban, as this property is built-in into the syntax of the language.

The absence of role labels is the next obvious property of Lojban. In this regard, it closely resembles PropBank, which also resorts to the more simplistic solution of *indexed arguments*. The PropBank labels Arg_0, Arg_1, Arg_2, etc. are used in the same way for describing roles as the Lojban placeholders x_1,x_2,x_3, etc. We can observe that Arg_0 almost always corresponds to x_1 for the same concept, because both resources reserve this slot for *agent-like* roles.

Semantic ontologies also deal with *extra-thematic* roles. In FrameNet, this is realized by defining a large set of possible arguments per frame; only a subset of them is considered to be *Core* elements (see Section 1.2.3.1). This means that each frame has its own set of possible extra-thematic roles; a feature which is obviously not present in Lojban. PropBank has a slightly different approach, by defining a fixed set of modifier roles which can be applied to any predicate (see Section 1.2.2.1). Lojban has a number of function words which serve exactly this purpose. However, it also has a language feature, which allows *any* brivla to be converted into a descriptor for an additional argument slot brivla. This means that Lojban predicates can, in theory, have completely arbitrary additional arguments.

Lastly, we can consider a *hierarchy* to be a possible property of a semantic resource. Whereas FrameNet frames are defined in a hierarchical tree, the PropBank inventory does (by itself) not have any such tree structure. Lojban inherently also has no hierarchy, but in Section 2.1.2 we have demonstrated how a taxonomy is implicitly defined by Lojban's rules of word formation; therefore making it also a hierarchical resource.

This hierarchy also constitutes a compromise between a fixed set of role labels and predicate-specific roles. In Section 1.2.2 we have discussed the difficulty of choosing a universally accepted set of semantic role labels. PropBank avoids this issue by using theory-agnostic labels $Arg_0 \ldots Arg_n$, but at the same time it sacrifices any information of the relationship between semantic roles across different frames. In Lojban, this information is partially available for compound predicates. In Section 2.1.2 it was demonstrated how the compound predicates specify their argument etymology (see Figure 2.4). For this reason, Lojban's argument style can be described as indexed, but extended with *hierarchy labels*.

Table 2.5 summarizes the properties of Lojban in comparison to FrameNet and PropBank. It can be seen that it shares some features with each of these resources. Concludingly, although Lojban was not intended for the same purpose as semantic ontologies, its *structure* would allow for it to fulfill the same function. This structural similarity motivates first an alignment of Lojban to other resources (Chapter 3) and further suggests its use as an inventory of predicates for semantic parsing (Chapter 4).

3 Aligning Lojban to semantic ontologies

We have now established that Lojban shares many properties with resources intended for semantic annotation, such as FrameNet. In this chapter it is demonstrated that Lojban can in fact be *aligned* to such resources, a common processing step which finds corresponding elements between multiple ontologies. In Section 3.1 we first motivate the use of Lojban as a semantic ontology and then discuss why it is beneficial to produce an alignment. In Section 3.2 we then introduce some commonly applied techniques for the automatic alignment of ontologies, called *ontology matching*. Subsequently, in Section 3.3 we analyze Lojban in regard to an alignment with FrameNet, and define an alignment task for which some gold data is produced. Finally, in Section 3.4, a statistical alignment system is devised which is applied to this alignment task and evaluated.

3.1 Motivation

To begin with, we might be interested in the benefits of using Lojban as a semantic ontology with so many well-established resources already in existence. The argument which is brought forward multiple times, is that of coverage. Although it does not offer detailed in-depth distinctions of concepts, Lojban is argued to have good breadth coverage.

> *Lojban, with the avowed intention of producing a speakable language, is much likelier to cover language breadth well, even if does not explore semantic depth as soundly as a professional linguistic research project. Linguistic research, by contrast, concentrates on specific problems of meaning and form; as a result, they do not typically attempt to cover the whole of language, but only those parts of language relevant to the particular research topic. (Nicholas, 1996)*

It should be considered that in the last two decades since this argument was formed, a lot of effort has been put into the extension of semantic resources such as FrameNet which have increasingly good coverage. Nevertheless, even FrameNet is not a truly "general purpose" resource, as it focuses on frames evoking actions, events or relations. It is acknowledged that other resources, such as WordNet, are more suited as an inventory of "artifacts" or "natural kinds" and that "the FrameNet database is not readily usable as an ontology of things" (Ruppenhofer et al., 2006).

Lojban on the other hand inherently covers all concepts. Aside of actions, states or events, Lojban brivla also describe "things" and also include descriptive predicates which would be associated with adjectives in natural language. Arguably, the sufficient coverage of Lojban is demonstrated in the existence of translated text. It is reasonable to assume that the lack of a specific concept becomes much more apparent during the act of *translation*, rather than in the process of annotating an existing text.

Independent of the current state of Lojban coverage, another benefit is the simplicity of the resource. Whereas FrameNet frames require a huge amount of formalization and the attention of professional linguists, Lojban brivla are minimal definitions. This enables the inventory of concepts to be very quickly extended or edited. It makes a steady growth of the dictionary possible (which is community-driven and thus has no associated costs).

Lastly, and perhaps most importantly, in Chapter 2 we have demonstrated that Lojban goes beyond being an inventory of predicates. It fills the gap between predicate-argument formalisms, and the expressiveness of natural language. This suggests that Lojban could be used as a much richer meaning representation than simple annotations (this idea is briefly elaborated as future work in Section 5.2.2).

As a meaningful next step, in this chapter we set out to align Lojban to semantic ontologies. It is briefly justified, why such an alignment might be desirable. We have already seen that semantic ontologies – although each having a different concept and focus – essentially serve the same purpose. By obtaining a mapping which describes the relationship between entities of distinct ontologies (for example equivalent FrameNet frames and PropBank rolesets), we essentially obtain a unified resource. Through this, additional semantic information can be obtained. For example unlabeled PropBank arguments Arg_n were annotated with a thematic role label from VerbNet. A further benefit is that of increased coverage, by identifying concepts which are disjunct between two ontologies. In matters of semantic parsing, there is a potential for using an alignment to obtain additional training data. In fact, it has been shown by Shi and Mihalcea (2005) that by integrating FrameNet, VerbNet and WordNet into a unified resource, semantic parsing results could be improved. In many cases, the alignment to a richer ontology

```
kill
catra: x1 (agent) |-s/slaughters/murders x2 by action/method x3
TransitiveAction
arg1: Agent
arg2: Patient
arg3: Process OR Action OR Object
```

Listing 3.1: Example entry of the "Lojban FrameNet" suggested by Goertzel (2005b)

can mitigate the sparsity of another. Yi et al. (2007) have reported an increased performance in the CoNLL task for semantic role labeling, when first applying a mapping from PropBank role labels to VerbNet thematic roles.

As a result of these obvious benefits of a unified semantic resource, there exist a large number of alignment projects. Some alignments were obtained automatically, whereas others are manual efforts. A notable example is SemLink (Palmer, 2009), an alignment project producing the *Unified Verb Index*. It is a manually curated resource providing a set of mappings between PropBank, VerbNet, FrameNet and WordNet. It defines redundant mappings between each of these resources, and aligns all structural levels. Therefore, it includes alignments of frames, senses and rolesets but also *role mappings* between semantic roles and frame elements. The project is motivated by the complementary nature of these resources. It is argued that VerbNet provides the "clearest links between syntax and semantics", FrameNet provides the "richest semantics", and PropBank provides the "best training data". It is therefore hoped that a mutual alignment enhances the usefulness over its parts.

The idea of aligning Lojban to other resources, has been suggested before. It was proposed to map the Lojban dictionary to WordNet.

> *The main thing that's needed, it seems, is the construction of a systematic mapping from Lojban vocabulary into a standard English dictionary such as* WordNet. *In almost all cases, each Lojban word corresponds naturally to a single WordNet sense. (Goertzel, 2005b)*

The primary motivation for doing so are practical advantages for human translators, who could look up unknown Lojban predicates through the WordNet hierarchy, identify the correct sense, and find the brivla most closely matching that given sense. No details on an actual alignment process are discussed, but it is assumed that such a mapping "will have to be checked out by hand, one by one" (Goertzel, 2005b). In terms of aligning Lojban to a frame semantic ontology, no prospect is given by the author. Instead, it is suggested that a novel semantic resource, resembling FrameNet, should be created which is titled "*Lojban FrameNet*". This database would provide additional information for each brivla, such as an "indication" of the semantic relationship of each argument slot, and a classification of brivla based on their argument structure semantics. Listing 3.1 shows an exemplary entry for this proposed resource. For the lexical item "*kill*", it lists the brivla `catra`, alongside its definition. It is assigned a class, in this case *TransitiveAction*, and further labels each argument with a semantic role, in this case *Agent*, *Patient*, and {*Process*, *Action*, *Object*}. This semantic formalism is argued to make Lojban a valuable semantic resource on par with FrameNet.

The work conducted in this chapter is based on the same motivation. However, instead of building such a resource from scratch, we investigate the feasibility of directly aligning Lojban to existing semantic ontologies, at the example of FrameNet. In fact, from obtaining this alignment a number of immediate use cases arise, which will be discussed in detail in Chapter 4. First of all, we will be able to trivially obtain semantic annotations of Lojban prose with respect to the aligned ontology. In case of an alignment to FrameNet, we can generate complete FSP annotations for Lojban prose without much effort (by virtue of its well-defined formal grammar). This is done in Section 4.1.2. Another use case emerges when using Lojban itself as a semantic ontology, with the aim to annotate natural language text with Lojban brivla. Here we can profit from the existence of highly optimized semantic parsers operating on FrameNet. The output of a FrameNet-based parser can then simply be *converted*, using the alignment as a static mapping. This use case will be discussed in Section 4.2.3.

Consequently, in this chapter it is analyzed how, and to what degree Lojban can be aligned to FrameNet (for PropBank, only some prospects are given). This analysis begins on a structural level. Then, an alignment task is formally defined, and some gold annotation data is created with a graphical annotation tool implemented for this purpose. Lastly, an automated *statistical alignment* is computed and evaluated with the annotated gold data.

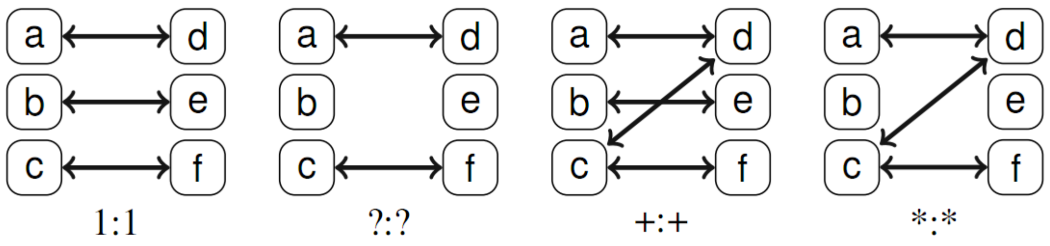


Figure 3.1: Four exemplary multiplicity classes of alignments (Source: Euzenat and Shvaiko 2007)

3.2 Introduction to ontology alignment

To clarify what is meant by an "alignment", we first define some elementary terminology of *ontology matching*. Then, some related work in statistical alignment will be introduced.

3.2.1 Formal definition

Alignment An alignment is formally defined as a set of *correspondences* between two or more ontologies. We will adapt a simplistic view of an ontology as a *set* of entities, and further confine the definition to the binary alignment of entities between the ontologies O_1 and O_2. Then, an alignment can be modeled as a set of N binary relations $R_n \subseteq O_1 \times O_2$ with $n \in 1 \dots N$. These relations can denote different types of correspondence such as *equivalence* ($\equiv$), *containment* ($\geqq$), or *approximated equality* ($\simeq$).

An alignment can be classified to be either *directed* or *undirected*. In terms of a correspondence relation R, an undirected alignment necessitates that R is symmetric, so that $\forall x.y.(x,y) \in R \Rightarrow (y,x) \in R$; for a directed alignment, R is not necessarily symmetric. Note that these terms refer to the underlying correspondence, rather than the matching task (which can be unidirectional or bidirectional).

Mapping A special kind of oriented alignment is a mapping. This maps each entity from one ontology O_1 to at most one entity in O_2. Thus, it complies with the mathematical definition of a mapping and we can define a function $f : O_1 \rightarrow O_2$, which in ontology alignment terminology is called a *mapping rule*. If f is surjective, the alignment $O_2 \mapsto O_1$ is said to be a *total alignment* (each element in O_2 is aligned to at least entity in O_1). Likewise, if f is injective, the alignment $O_1 \mapsto O_2$ is called an *injective alignment* (no entity in O_2 is aligned to more than one entity in O_1). If f is bijective, this implies the relation R to be an *equivalence relation*. In ontology alignment terminology, this is called a *one-to-one* alignment.

Multiplicity For alignment correspondences that are not *one-to-one* alignments, we can further define multiplicity terms. Common cardinality notations include $1:1$ (one-to-one), $1:m$ (one-to-many), $n:1$ (many-to-one), or $n:m$ (many-to-many). Euzenat and Shvaiko (2007) further define an adapted cardinality notation, in which 1 stands for an injective and total mapping, ? for injective, + for total, and $*$ denoting none of these properties. Using this classification, they obtain 16 different multiplicity classes for (undirected) alignments. Figure 3.1 shows an illustration of these classes.

Matching The matching problem is the process of finding a set of alignment relations A_n approximating the true underlying correspondences R_n. One possible simplification of the matching problem is its subdivision into two mapping tasks, each given the ontologies O_1 and O_2 and yielding the respective mappings $f_1 : O_1 \rightarrow O_2$ and $f_2 : O_2 \rightarrow O_1$. An alignment process which yields both of these mappings and aggregates them into an alignment is defined to be a *bidirectional* matcher. Accordingly, a process computing an alignment from only f_1 (w.l.o.g.) is called a *unidirectional* matcher. Note that for an undirected alignment relation R, the resulting mapping function f_1 can be *inverted*.

Note that in this simplification we omit a formal definition of an ontology. Furthermore, any alignments between more than two ontologies, or alignments using non-binary relations between entities are disregarded. It is also possible to define an alignment as a *fuzzy* relation through the use of *confidence structures*, which associates each alignment pair with a probability. The *evaluation* of ontology matching is deferred to Section 3.4.6.

3.2.2 Overview of matching techniques

The correspondence relations between entities of multiple ontologies are generally *semantically* motivated. For example an equivalence relation $e_1 \equiv e_2$ assumes that e_1 and e_2 are used to refer to the *same concept*[1]. Ontology matching is thus based on the assumption that statistical evidence for the alignment is present in the ontology data. For classifying different matching techniques, we can first observe different levels on which they operate and give an example in case of FrameNet as an ontology.

element-level Techniques which compute correspondences between entities by analyzing them in isolation. In case of FrameNet, this would employ the definition of a single frame, frame element or lexical unit.

instance-level A refinement of element-level techniques, which further incorporate *data instances* of entities. In case of FrameNet, this would employ annotated text which evokes a respective frame, frame element or lexical unit.

structure-level Techniques which compute correspondences by analyzing the ontology structure. In case of FrameNet, this would employ the frame-to-frame relations, or FE-to-FE relations.

Element-level techniques rely on the properties of entities within the definition of the ontology. For linguistically defined resources, properties such as the *name*, or other strings (such as a definition) are considered. Other properties could also be numeric, in case of FrameNet for instance the number of frame elements. *Instance-level* approaches can, for example, consider the frequency of an entity within a dataset or be based on co-occurrence. This assumes a single dataset to be associated with two different ontologies, and relies on co-occurring instantiations of entities within both ontologies. These occurrence counts can be used to compute a joint distribution, and were shown to perform very well in a supervised machine learning approach (Doan et al., 2004).

An orthogonal taxonomy of matching techniques classifies them according to the kind of resources that are used. Pure *syntactic* techniques operate solely on the input data, whereas *external* techniques employ some external auxiliary resource. An ontology matching system generally belongs to more than one class, and may combine different matching approaches.

Similarity based approaches

A very general element-level approach is based on the notion of *similarity measures* between entities. Thus, a set of k similarity measures $sim_i(e_1, e_2) \in \mathbb{R}$ for $e_1 \in O_1$, $e_2 \in O_2$ and $i \in 1, \ldots, k$ are computed. Similarity measures are generally required to satisfy

$$\begin{aligned} \forall x.y.sim(x,y) &\geqslant 0 && \text{(positiveness)} \\ \forall x.y.sim(x,y) &= sim(y,x) && \text{(symmetry)} \\ \forall x.y.z.sim(x,x) &\geqslant sim(y,z) && \text{(maximality)} \end{aligned}$$

Similarity computation can be classified as either *syntactic*, or *external*. An example of a syntactic similarity measure is purely string-based. This measure generally refers to the textual comparison of an entity string with that of a corresponding entity. For string comparison a wide range of measures is used, including *n-gram overlap, Levenshtein distance, DTW, SMOA, Dice coefficient, substring similarity* (some relevant measures are elaborated in Section 3.4).

However, syntactic similarity measures cannot capture the semantic similarity between lexical items, such as “car” or “automobile”, despite them referring to the same concept. To overcome this shortcoming, the use of external resources in similarity measures incorporates this linguistic knowledge. As an example, WordNet is used to compute a similarity score between two lexical items, which is for instance based on the path between them in its hierarchy (Pedersen et al., 2004). Once a set of similarity measures is defined, one traditional approach is to *aggregate* the different scores. There exist a number of techniques to compute an optimal aggregation including *weighted product, weighted sum*, and more sophisticated strategies. The final aggregated similarity matrix is then converted to an alignment relation using an optimal *threshold*. However, choosing the optimum parameters for weights, thresholds and other constraints is a difficult optimization problem. In the SemEval-2012 task for *semantic textual similarity*, the best performing system applied a log-linear regression model to aggregate different similarity scores (Bär et al., 2012).

[1] A theoretical principle underlying ontology matching suggests that any two ontologies approximate each other, and their alignment is an approximation of a common *ideal ontology* (Euzenat and Shvaiko, 2007).

In case of ontology alignment, an alternative machine learning-based approach solves this aggregation as a supervised binary classification task. Pairs of entities (e_1, e_2) are labeled as *true* if they are in a gold alignment, and *false* otherwise (Nezhadi et al., 2011). In this model, the set of k similarity measures can be used as features. A wide range of classifiers has been evaluated for ontology matching, including SVMs, Decision Tree (DT), K-Nearest-Neighbor (KNN) classifiers, as well as *boosting* and *ensemble* meta-learners (these models are covered conjointly for instance in Bishop, 2006 and Rokach, 2010).

3.3 Lojban alignment

It is not obvious what actually constitutes an alignment of Lojban to another given ontology. It has to be considered what *source entities* should be aligned to which *target entities*, and what kind of *relation* we aim to identify. In the following, an analysis is done to compare Lojban conceptually to FrameNet. The purpose of this is to firstly illustrate the *alignability* of Lojban to these resources, and secondly facilitate the definition of an alignment task.

When regarding Lojban as an ontology, all dictionary entries could be considered for alignment. One criterion is the possibility of a word to act as a predicate. This would include certain cmavo which can be used as selbri. For example `moi` is used to create any numeric predicate for a number n, stating that

> n-`moi`: x_1 *is n-th member of set* x_2 *ordered by* x_3 e.g.
> `remoi`: x_1 *is second among* x_2 *ordered by* x_3

However, these predicates are mostly used to cover exceptional cases (in case of `moi`, an infinite number of predicates is defined). For practical reasons, we will not consider these cmavo and instead focus only on brivla. Within this set, which per definition only contains predicates, we may be interested in regarding only a subset, such as gismu. As target entities in FrameNet, possible choices are either the set of *frames* or the set of *lexical units*. On the argument-level, it seems plausible to find an alignment between Lojban sumti and FrameNet frame elements[2]. In the following, first the predicate level is analyzed, followed by an analysis of the argument level.

Predicates

As a first indicator, the absolute entity counts of different ontologies should be considered. The rationale here is that assuming all ontologies to sufficiently cover the breadth of all concepts in language, the same number of entities could suggest a similar level of abstraction. For FrameNet we also take into consideration the number of LUs.

FrameNet frames (LUs)	**PropBank rolesets**	**Lojban brivla (gismu)**
1019 (11,942)	6204	8486 (1342)

Judging only by these statistics, it appears that Lojban brivla are too fine-grained to align to FrameNet frames. A more promising alignment would be to align brivla to the FrameNet lexical unit level, or in case of PropBank, aligning brivla to rolesets. Another alignment which could be suggested by these statistics is the alignment of only the set of gismu to FrameNet frames. If we manually inspect some sample entities of both ontologies, we can see that Lojban brivla are usually more specific than a FrameNet frame, but less specific than a lexical unit. We will continue with some examples of Lojban brivla to illustrate why that is the case. To first give an example of a fairly generic brivla, which corresponds to at least one FrameNet frame consider the predicate for *speaking*:

> `tavla`: x_1 *talks/speaks to* x_2 *about subject* x_3 *in language* x_4

This brivla corresponds closely to the frame *Statement*, defined as:

> The act of a *Speaker* to address a *Message* to some *Addressee* using language.

When regarding the lexical units of this frame, one can observe that most of them could be adequately translated with `tavla`, including *address.v, describe.v, talk.v, tell.v, say.v, speak.v* and *state.v*. It does however also contain some lexical units which are not adequately covered by `tavla`, such as *acknowledgment.n, admit.v, claim.n, explain.v,*

[2] Analogously, in case of PropBank we could either consider framesets or rolesets as targets on the frame-level, and their respective semantic roles on the argument level.

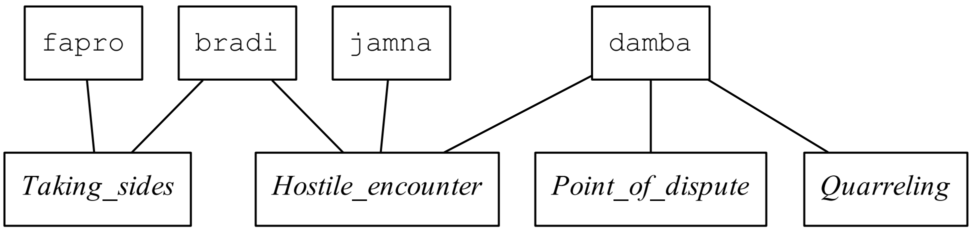


Figure 3.2: Alignment of gismu to frames, showing an *n:m* alignment multiplicity

mention.v or *write.v*. We can further argue that `tavla` does not align to those lexical units, because other brivla exist, which precisely cover the meanings of *admit*, *claim*, *explain*, etc. The matter is further complicated by the fact that some lexical units which match the meaning of `tavla` are contained within a different frame, e.g. (*chat.v*, *Chatting*). There also exists a multiplicity in backward alignments, because (*chat.v*, *Chatting*) would also overlap with `fazyta'a` (x_1 *chatters / gossips*).

In conclusion, we can observe that there is an arbitrary *n*-to-*m* mapping between brivla and frames, so that each brivla aligns to multiple frames, and each frame aligns to a different subset of brivla. This *n*:*m*-relation is illustrated in Figure 3.2, which shows an alignment between multiple gismu and frames in the field of "hostile encounters".

On the level of lexical items, these alignments only holds for a subset of LUs within each of its alignable frames. The reason for this intricate mapping is that both for FrameNet and PropBank the primary distinguishing feature of frames (or rolesets respectively) is a shared set of arguments, which must be identical in number and types. Frame definitions are kept as abstract as possible (while retaining a shared argument structure), and the concrete meaning of a frame is only resolved in conjunction with a lexical unit. This means that words with different meanings can still belong to the same frame; such as the grouping of antonyms into a single unpolarized frame. As an example, the *Temperature* frame contains both the lexical items *hot.a* and *cold.a*. In Lojban, this can obviously not be the case, because a single brivla uniquely describes a single concept. Hot and cold are modeled as the two separate gismu.

> `lenku`: x_1 *is cold by standard* x_2
> `glare`: x_1 *is hot by standard* x_2

Therefore, both of these brivla should align to the same frame, but each to a distinct subset of lexical units.

An idiosyncrasy of FrameNet arises from its hierarchical structure. Many frames which are far up in the tree hierarchy serve mostly the purpose of building a taxonomy and are not intended for instantiation. An example is the *Use_vehicle* frame, which by itself is not evoked by any lexical unit. It does however have multiple specializations, such as *Operate_vehicle* which can be invoked by LUs such as *drive.v*. As Lojban lacks any hierarchy on this abstract level, there exists only one predicate describing the operation of a vehicle (`ma'ersazri`: x_1 *drives a vehicle* x_2 *with goal* x_3). It is debatable if this brivla should align to only the *Operate_vehicle* frame, or if it should also align to frames which are entailed by it, including the *Use_vehicle* frame.

Although Lojban brivla are not hierarchical, they still contain abstract concepts comparable to "non-instantiated" frames. As such, some abstract gismu are rarely used by themselves; for example the gismu describing an intentional act, or an agentive cause, which is similar to the English word *do*.

> `gasnu`: x_1 *[person/agent] is an agentive cause of event* x_2

The predicate `gasnu` still serves a very important function. It often acts as a prefix to distinguish non-agentive predicates from their agentive version. For example, it transforms the state-descriptive `cikna` (x_1 *is awake*) into a transitive action, `cikna` + `gasnu` = `cikygau` (x_1 *wakes up* x_2). This gismu would most closely correspond to the FrameNet frame *Intentionally_act*, even though that frame serves mostly the purpose of inheritance, similar to the *Use_Vehicle* frame. As a consequence, we can observe that in some cases, it should be permissible to align a gismu to an abstract or non-instantiated FrameNet frame, and even permitting the subset of overlapping LUs to be empty.

On the other end of the spectrum, one can observe brivla with extremely specific, fine-grained definitions. These are sometimes domain-dependent, and in some cases exceptionally complex. Consider for instance the Lojban brivla for the mathematical concept of an eigenvector.

`aigne`: x_1 *is an eigenvalue (or zero) of linear transformation/square matrix* x_2*, associated with/owning all vectors in generalized eigenspace* x_3 *(implies neither nondegeneracy nor degeneracy; default includes the zero vector) with 'eigenspace-generalization' power/exponent* x_4 *(typically and probably by cultural default will be 1), with algebraic multiplicity (of eigenvalue)* x_5

It is unclear if such a word is seriously used in discourse, or if it was added for humorous reasons. Certainly, no corresponding frame can be found in any other ontology. Nevertheless, for Lojban is has to be considered that some predicates exist which have to be regarded as *spam*. For example, the dictionary contains some entries intended as jokes, such as

`zu'erxiolo`: x_1 *does* x_2 *because YOLO*[3].

Although such entries are rare, and they often contain a note stating them to be jokes, it would be plausible to filter them out for the purpose of alignment. Another class of brivla which needs to be filtered out are gismu which act as assignable variables, namely `broda`, `brode`, `brodi`, `brodo` and `brodu` which by themselves have no meaning, but instead are referring to another brivla assigned in the previous discourse.

Lastly, there is also a set of *missing* concepts on both sides. This is mostly the case for frames for *natural kinds* ("things"). In many instances, these frames are currently missing or incomplete in FrameNet. For example, although there exists a *Food* frame, mere 70 lexical units have been assigned to it. In other cases, artifact frames are entirely missing; this variation in completion seems arbitrary. For example frames for animals, clothing, or furniture are lacking even if there exist frames for *Vehicle*, *Buildings* or *Accoutrements*. This poses a notable problem for alignment, as the Lojban vocabulary to a large extend consists of predicates for "things".

Arguments

The major difference in the modeling of arguments of Lojban brivla is already apparent by their limitation to a maximum of 5 argument slots. The number of FrameNet frame elements is much larger, although one could assume that Lojban arguments correspond much rather to just the *Core* frame elements. To support this presumption, we first can compare the average number of arguments.

FrameNet Core FE count (all FEs)	**PropBank arg count**	**Lojban argument count**
2.94 (9.45)	2.57	2.70

The rough equivalence of the number of arguments in Lojban brivla, ProbBank arguments, and *Core* frame elements suggests that arguments are modeled in a similar way. In fact we can find some instances, in which there exists a perfect correspondence between *Core* FEs and Lojban sumti. Consider the following gismu-frame pair

`bolci`: x_1 *is a ball/sphere/orb/globe of material* x_2

Shapes: words which describe the dimensional extent and *Shape* of a *Substance*

Here the arguments x_1 and x_2 align exactly to the two core FEs *Shape* and *Substance*. Although in most cases Lojban arguments can be assigned uniquely to one frame element, this is not always the case. Upon closer inspection, there are some inherent discrepancies between the Lojban argument model, and that of FrameNet (and PropBank alike). The main reason for this is that Lojban arguments serve multiple purposes. In stark contrast to conventional ontologies, arguments in Lojban are not always meant to be filled with a syntactic element. As described in Section 2.1.2, some arguments in Lojban exist in order to be instantiated as noun-like concepts. In these cases, English nouns are attached as argument slots rather than being predicates of their own right. For instance, the definition for the brivla for "bottle" appears quite idiosyncratic.

`botpi`: x_1 *is a bottle/jar/closable container for* x_2*, made of material* x_3 *with lid* x_4

3 YOLO is an acronym from popular culture for "you only live once".

Here the existence of the argument x_4 seems rather arbitrary. However, when considering that arguments are also defining innate concepts, it makes sense to attach the English word *lid* to the predicate for *bottle* rather than creating a new distinct predicate. The x_4 argument is intended to be referenced (e.g. `lo velbo'i` = "the lid"), but it is unlikely that any syntactic construct will be substituted into this slot. This is a striking difference to the way semantic roles are defined in linguistic theories. There are further types of argument slots which appear peculiar if compared to conventional ontologies. These arguments are sometimes added to ensure "scientific accuracy", or "political correctness". Consider for instance the predicates for "up", or "beautiful" which are defined as

`gapru`: x_1 *is directly/vertically above/upwards from* x_2 *in gravity/frame of reference* x_3
`melbi`: x_1 *is beautiful to* x_2 *in aspect* x_3 *by aesthetic standard* x_4

Although it can be argued, that "up" can only be defined in relation to some frame of reference, the x_3 argument denoting the *gravity* of "up" is not familiar from discourse in natural languages, and thus not formalized in semantic ontologies. In a similar manner, it can be seen that the x_4 argument of "beautiful" formalizes the *subjectivity* of the predicate. Concludingly, Lojban predicates strive for a level of scientific accuracy and political correctness not commonly expressed in semantic ontologies. In FrameNet, frame elements corresponding to these kinds of arguments are largely not present. Thus, some Lojban arguments are unalignable to FrameNet frame elements.

There also exist some cases in which a single Lojban argument must be assigned to multiple frame elements. This occurs for instance when FrameNet defines alternative roles. Consider for example the Lojban predicate for "at the same time", aligned to the FrameNet frame for *Temporal_collocation*

`cabna`: x_1 *is during/concurrent/simultaneous with* x_2 *in time*

Here, the FrameNet frame defines the (mutually exclusive) set { *Trajector_entity*, *Trajector_event*, *Trajector_period* } which would correspond to x_1, and the (mutually exclusive) set { *Landmark_entity*, *Landmark_event*, *Landmark_period* }.

In summary, FrameNet has a more fine-grained notion of arguments than Lojban, so there exists a 1:*n* multiplicity for argument alignments between Lojban slots and frame elements (with some $n = 0$, meaning that a Lojban argument is *unalignable* to FrameNet). In most cases however, Lojban arguments can be assigned *uniquely* to one frame element. In Section 3.3.2, this statement will be supported by empirical evidence from the manual annotations.

3.3.1 Task definition

Concluding from the analysis, various kinds of alignments are possible. Obviously, it is necessary to make decisions to constrain the alignment tasks in the scope of this work. Therefore a limited task is defined which formulates unambiguous guidelines for manual annotation.

The primary task is based on the observation that there is a high overlap between the set of gismu and FrameNet frames, although this certainly is an *n*:*m* alignment. We thus define an alignment task for finding a relation between gismu and frames, indicating that they have some *semantic overlap*. Based on this task, subtasks are defined for the *lexical unit* level and *argument* level.

Task GF Align gismu to FrameNet frames

For any gismu g identify a set F of FrameNet frames so that for each $f \in F$ the *textual definition* of g could be *annotated* with f (disregarding the lexical unit). F may be empty, in which case g is said to be *unalignable*. For simplification reasons we further require $|F| < F_{max}$.
The following subtasks can be defined for a given gismu-frame alignment pair:

Task GF-LU Determine a subset of lexical units adhering to the alignment.

For a given alignment pair (g, f), determine a subset $L \subseteq LU(f)$, where $LU(f)$ is the set of lexical units evoking f.

Task GF-Args Align sumti to frame elements.

For a given alignment pair (g, f), find an argument alignment $A \subseteq SUMTI(g) \times FE(f)$ where $SUMTI(g)$ is the set of sumti slots of g, and $FE(f)$ are the frame elements of f. The guideline here is that the sumti slot within the definition of g could be annotated with the respective frame element.

This guideline only makes use of ontology data for both Lojban and FrameNet and no instance data is used (in case of Lojban this would refer to Lojban prose, whereas for FrameNet this would refer to annotated text). The main

alignment multiplicity $(1:n)$	count (% of all aligned)
0 (unalignable)	75 (29%)
1	126 (48%)
2	39 (15%)
3	18 (7%)

Table 3.1.: Frame-level alignability stats, regarding the direction of aligning Lojban gismu to FrameNet frames

sumti slots	number	uniquely aligned $(1:1)$	ambiguously aligned $(1:n)$	unalignable $(1:0)$
1	12 (1.5%)	12 (100%)	0 (0%)	0 (0%)
2	110 (14%)	91 (83%)	7 (6%)	12 (11%)
3	251 (32%)	206 (82%)	8 (3%)	37 (15%)
4	248 (31.6%)	195 (79%)	5 (2%)	48 (19%)
5	102 (13%)	81 (79%)	1 (1%)	20 (20%)
6	36 (4.6%)	28 (78%)	0 (0%)	8 (22%)
7	23 (2.9%)	20 (87%)	1 (4%)	2 (9%)
any	783	633 (80%)	22 (2.8%)	127 (16.2%)

Table 3.2.: Argument-level alignability stats, regarding the direction of aligning Lojban sumti to FrameNet frame elements

reason for this is a simplification of the manual alignment task. By simply requiring the textual definition of the Lojban gismu to be annotated with a frame, most of the intricacies of annotation are effectively "outsourced" to the guidelines included with FrameNet.

The overlap relation described here is *symmetric* so that for any $g \rightleftharpoons f$ also $f \rightleftharpoons g$. However, the task is defined *unidirectional*. The reason for this is that the number of gismu which align to any frame is expected to be much higher than the number of frames any given gismu aligns to. Finding all possible gismu for a FrameNet frame is therefore much more difficult than the opposite direction. If all 1437 gismu were to be annotated with gold data, this would be irrelevant. However, as only a subset of all gismu are annotated, this has some consequences which have to be considered for evaluation. Namely, for any gismu in the gold alignment, we can be certain that the alignment relation is *exhaustive*, but for a given FrameNet frame this is not the case.

3.3.2 Annotation of gold alignments

For the defined tasks, manual annotations have been created. No statistics on rater-agreement can be provided, as the author is the sole annotator for all data. To improve the annotation process, a graphical annotation tool was implemented which integrates with the automatic system developed in Section 3.4. For selecting the gismu for annotation, alphabetical order was chosen. This can be considered sufficiently arbitrary in respect to the kind and occurrence frequency of words[4]. For each gismu, at most $F_{max} = 3$ frames were annotated as alignable. This artificial limit is based on the general observation that only a few gismu correspond to more than 3 frames – and if they do this results from a very loose interpretation of their meaning.

In total 258 gismu have been aligned to a total of 177 frames, resulting in 258 alignment pairs. Of these 258 gismu, 75 were unalignable (29%). On the LU level, this alignment is refined to 541 pairs. On the argument level, 681 items have been aligned. Regarding the multiplicity, the theory that gismu can generally be uniquely assigned one frame is supported by the statistics of the alignment. Table 3.1 shows the multiplicity counts of frames, which has a clear majority on $1:1$ alignments, and much fewer $1:n$ alignments for $n > 1$. Likewise, Table 3.2 shows these statistics on the argument level. it should be noted that *ambiguous* argument-level alignments ($1:n$, for $n > 1$) constitute only 2.8% of all alignments and can thus considered not to be a significant issue. A large fraction of all Lojban arguments (80%) could be uniquely assigned to a frame element, whereas 16.2% were unalignable. This supports the theory that FrameNet frame elements are sufficiently fine-grained and comprehensive to account for most Lojban definitions.

[4] The examples used in this chapter are also included in the alignment and are exception from this order.

Brivla class	**definition**	**example**
Frame	stating some action, state, event, etc.	`bajra`: x_1 *runs on surface* x_2 ..
Entity	denoting physical or conceptual entities (FrameNet "artifacts")	`karce`: x_1 *is a car for carrying* x_2 ..
Function	functional purpose, such as specifying time or occurrence	`purci`: x_1 *is earlier than* x_2 ..
Descriptive	describing the static property of an entity	`barda`: x_1 *is big in dimension* x_2 ..
MetaLojban	words for Lojban concepts	`gismu`: x_1 *is a root word expressing* x_2 ..

Table 3.3.: Definition of the "Brivla class" labels

Brivla class	**total (% of gismu)**	**alignment completion (% of class)**	**alignable (% of aligned)**
Frame	245 (18.3%)	70 (28.6%)	66 (**94.3%**)
Entity	569 (42.4%)	118 (20.7%)	68 (57.6%)
Function	18 (1.3%)	6 (33.3%)	6 (100.0%)
Descriptive	134 (10.0%)	28 (20.9%)	24 (**85.7%**)
MetaLojban	10 (0.7%)	7 (70.0%)	0 (0.0%)

Table 3.4.: Distribution of brivla-classes and their alignability

A further annotation step was performed on all gismu to categorize them into certain classes, such as *Frame*, *Entity* or *Descriptor*[5]. These classes correspond roughly to verbs, nouns, or adjectives, but are named differently to indicate that their definition is independent of these language-specific POS. Table 3.3 shows how these classes were defined, whereas Table 3.4 shows the distribution of these classes. More than 42% of all gismu belong to the class *Entity*, and 18% were classified as *Frame* (this is an interesting contrast to the POS distribution in the English lexicon, of 76% nouns and only 7.5% verbs, according to WordNet). An important fact to note is that nearly all gismu of the *Frame*-class are alignable to FrameNet frames. Interestingly, also gismu of class *Descriptive* pose no problem for alignment. Within the *Entity*-class, only slightly more than half of the gismu are alignable, which already anticipates this class as a major source of errors for statistical alignment.

3.4 Statistical alignment

We can now devise an ontology matcher performing the alignment automatically. In the following, we only consider the primary alignment level (gismu to frames), although the alignment strategies introduced in the following can be applied to all levels.

In contrast to most traditional ontology alignment tasks, the available data is highly limited. For Lojban, we do not have any "annotated" corpora, so no *instance-based* approaches are possible, including basic co-occurrence-based strategies. One possible instance-based technique would be to assume a correlation between frequency counts of Lojban brivla (obtained from the `chatlog` corpus) and the frequency of frame annotations (obtained from the fulltext annotations). Although this assumption may appear plausible, a visualization of gold alignment pairs (shown in Figure 3.3) suggests that there is nearly no correlation (Pearson's $r = 0.144$). We further cannot use any *taxonomy-based* information for the GF task, because no hierarchical structure is present between gismu[6]. This effectively rules out any *instance-based* and *structure-based* approaches, and confines the remaining options to *element-based* alignment.

The available data we can employ on the Lojban side consists primarily of the dictionary entries. Therefore, a statistical aligner could make use of this entity-level syntactic information.

One straight-forward approach is the use of an FSP system. The annotation guideline defines that a gismu aligns to a frame exactly if its definition could be annotated with that frame, which strongly suggests that the alignment task can be restated as an FSP task. A FrameNet-based FSP system is applied to the (preprocessed) definitions of gismu, and the annotations are interpreted as alignment results. This method was tested by using SEMAFOR. However, the peculiarity of gismu definitions[7] throws off the system, resulting in many incorrectly assigned frames (precision is less than 10%; for the complete evaluation see Section 3.4.6).

[5] The remaining brivla were annotated with such a class label using a classifier trained on the annotated set of gismu (using only syntactic features, an SVM was evaluated to have an F_1 of 76%). These classes were intended to be used for the subsequent ML components described in this thesis, but were not found to be useful.

[6] For a possible alignment task between all brivla and FrameNet, the hierarchy between brivla could however prove useful.

[7] The use of placeholders "*x1*" to "*x5*" throws of the statistical system, even when replacing them with a generic word such as "this".

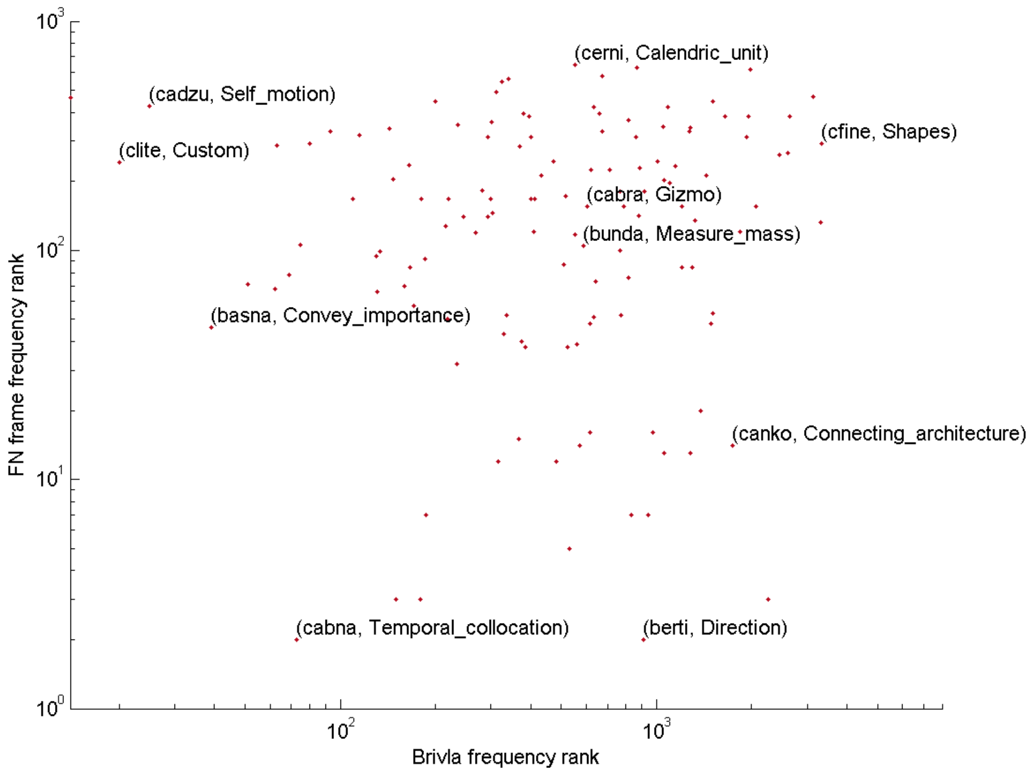


Figure 3.3: Frequency correlation plot of (gismu, frame) alignment pairs

A more traditional method would use the syntactic information of gismu definitions to compare them to FrameNet frames. In Section 3.2.2 we have introduced a generic alignment strategy based on *similarity measures*, which will be adapted in the following. For a statistical alignment system we define a set of features to be extracted from Lojban definitions, on which a number of similarity measures are defined. Some basic means to aggregate these different similarity measures are introduced, as well as a brief evaluation of ML-based alignment using these similarities as features. However, the focus is primarily on the feature engineering step, and the evaluation of similarity scores, as the introduction of Lojban as a semantic resource is a novel component not previously explored. The application of machine learners is only done coarsely for the sake of argument, but no optimization is conducted. As an outlook, we devise a system aligning the ontologies O_1 and O_2 by performing the following steps:

1. extract keywords $k \in K$ from entities $e_1 \in O_1$ and $e_2 \in O_2$
2. compute a *significance score* to obtain a weighting for a keyword k with respect to an entity e_1 or e_2
3. based on these keyword features, define N similarity measures $sim_1, \dots, sim_N$
4. use these N similarity measures as feature space for an ML-classifier

```
$x_{1}$ utters verbally/says/phonates/speaks [vocally makes sound] $x_{2}$.
```

$\downarrow$cleanup

```
x1 utters verbally / says / phonates / speaks x2
```

$\downarrow$ expand

```
x1 utters verbally x2
x1 says x2
x1 phonates x2
x1 speaks x2
```

$\downarrow$ process

Tokenization, POS-tagging, syntactic parsing

Figure 3.4: Preprocessing steps of dictionary entries

3.4.1 Processing and feature extraction

Using only the dictionary definition string of a gismu g, and the FrameNet data of a frame f, a similarity $sim(g,f) \in \mathbb{R}$ should be obtained. For this, it is necessary to define features on which similarity measures can be computed. As the data is purely textual, a simplification is made regarding the kind of features. A straight-forward baseline approach for obtaining a similarity score would be a simple *overlap* score of a *bag-of-words* feature. The method defined in the following is based on this idea, but is defined more abstractly to allow for more flexibility.

We can first observe that there are not any *numeric* properties of interest, so a *nominal* feature space is sufficient. It can further be observed that Lojban and FrameNet – or in general the two ontologies O_1 and O_2 – are in vastly different, so it will not be possible to define a single feature extraction function on the domain of $O_1 \cup O_2$. Instead we define separate *keyword extraction functions* $k_1 : O_1 \to K$ and $k_2 : O_2 \to K$. The co-domain K is a shared *keyword space*, which could for instance be *words*, *lemmas*, *POS tags*, or *numbers*. Assuming any such tuple of arbitrary extractors (k_1, k_2) for the respective ontologies, we can define a unified extraction function $k : O_1 \cup O_2 \to K$ for any $e \in O_1 \cup O_2$ as

$$k(e) = \begin{cases} k_1(e) & \text{for } e \in O_1 \\ k_2(e) & \text{for } e \in O_2 \end{cases}$$

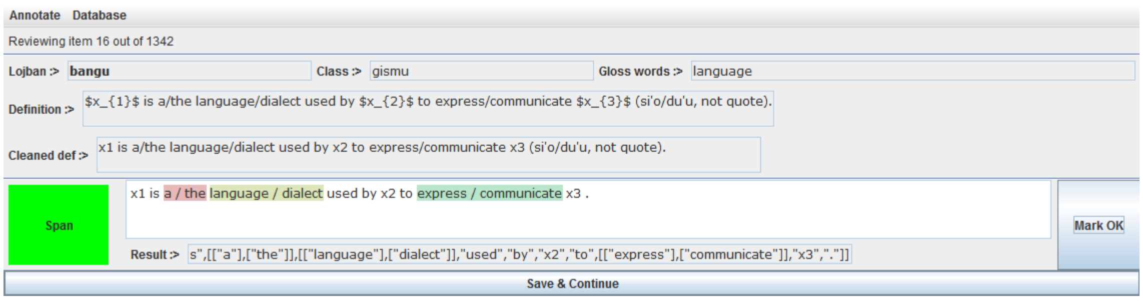
For each feature, we then obtain a document-term matrix for each ontology. We will further see that it is essential to introduce a *weighting* function for the relevance of keywords within either ontology. In the next steps, we first consider how to extract useful keywords (with a focus on Lojban), and then discuss how they are weighted and used for similarity measures.

Preprocessing

The definition of Lojban words is by itself not well suited for linguistic processing. It contains arbitrarily named placeholders, LaTeX markup, brackets, and a slash notation to indicate multiple lexical alternatives. To enable a common linguistic processing pipeline, a number of preprocessing steps is performed first.

1. **Cleanup**. In this step any LaTeX code is removed, and arguments are uniformly renamed `x1`, `x2`, etc. (recall from Section 2.2.1 that argument names use different letters for indicating Lojban etymology). Optionally, any bracketed text is also discarded[8].
2. **Slash expansion**. The "/" tokens, and the phrasing resulting from the notation of lexical alternatives would throw off further NLP processing (e.g. syntactic parsing). Therefore, the indicated alternatives are expanded into the set of possible sentences. This also counteracts the lexical sparsity of Lojban definitions.
3. **Common NLP pipeline**. On the set of expanded sentences, the Stanford Parser pipeline is run, performing *tokenization*, *POS tagging*, and *syntactic parsing*.

[8] Lojban uses round and square brackets for various meta-linguistic notes, e.g. "(agent)". In the implemented framework, this bracketed information is in retained for other processing steps.



Reviewing item 16 out of 1342

Lojban :> bangu Class :> gismu Gloss words :> language

Definition :> x_{1} is a/the language/dialect used by x_{2} to express/communicate x_{3} (si'o/du'u, not quote).

Cleaned def :> x1 is a/the language/dialect used by x2 to express/communicate x3 (si'o/du'u, not quote).

x1 is a / the language / dialect used by x2 to express / communicate x3 .

Result :> s",[["a"],["the"]],[["language"],["dialect"]],"used","by","x2","to",[["express"],["communicate"]],"x3","."]]

Figure 3.5: The *span* module, illustrating the manual annotation of "slash spans" for expanding brivla definitions

Figure 3.4 illustrates the preprocessing steps for one example entry. The cleanup step (1) consists of simple replacements. The expansion step (2) however is not trivial. It is ambiguous because the *range* of the slash-operator is unknown. In the example, it is correctly assumed to begin at the token "*utters*", but it could also begin at the token "*verbally*", resulting in a different expansion. To resolve this ambiguity, we model it as a set of spans, each covering a substring of the definition string and containing at least one "/" character[9]. These "slash-spans" can, to begin with, be obtained from manual annotation. For this purpose a span-module was added to the annotation tool to annotate the gismu with their correct spans, as shown in Figure 3.5. For the set of 1437 gismu, all slash spans have been annotated manually.

An approach to obtain these spans automatically was also devised. The automatic span computing is done relatively costly through the use of a *language model*. However, this can be disregarded, as it is a one-time processing step. The procedure works by permuting all possible variants of slash-spans, generating all possible expansions, and aggregating their perplexity scores according to a language model. The rationale of this approach is that incorrect span ranges result in ungrammatical expansions which have a high perplexity. Evaluating this method of estimating spans shows that it exactly matches the gold annotation in 61% of all cases and still provides a reasonable approximation in the other cases. These automatic spans were used for the remaining 7144 brivla not manually annotated. The complete algorithm is documented in Appendix B.4.

Keyword extraction

When extracting keywords, we can first consider meaningful keyword spaces – regardless of *how* (from which *part* of the entity) they are extracted. Extracting only the *tokens* within a text would result in the keyword set of all *word forms*. In the earlier example, this would lead to keywords such as *"says", "speaks", "utters*", etc. to be extracted. These keywords would be distinct from the same word assuming a different word form, such as "*say*", "*spoke*", or "*uttering*". To address this shortcoming, we can use *lemmatized* words. Another problem we can address at this processing stage is the presence of *stopwords* ("*is*", "*the*", "*a*", etc.) which can be eliminated in the extraction through a simple *stopword filter*.

The converse issue are keywords which are incorrectly shared by distinct words (e.g. homonyms, and other ambiguity issues. See Section 2.4.2). These issues cannot be resolved fully – however, the use of *POS-tagged lemmas* is a simple way to mitigate this problem. Because the data is very sparse, the Penn tagset was reduced in a post-processing step. For a given source token w and Penn tag t a simple transformation is applied.

$$simplify((w,t)) = \begin{cases} \text{"ARG"} & \text{if } w \text{ is an argument "} x_n \text{"} \\ \text{"NN"} & \text{if } t \text{ starts with "N"} \\ \text{"JJ"} & \text{if } t \text{ starts with "J"} \\ \text{"VB"} & \text{if } t \text{ starts with "V"} \\ \text{"RB"} & \text{if } t \text{ starts with "R"} \\ \text{"OT"} & \text{otherwise} \end{cases}$$

This transformation also handles the tagging of argument slot tokens "x_n" in Lojban definition strings. In order to limit the error of statistical tagging, these tokens are first converted to a generic English word (out of "*something*", "*that*" and "*this*", the latter appeared to work best), and then converted back, resulting in POS-tagged sentences such as

[9] This model for resolving slash-expansion does not cover all possibilities, because slash-operations are sometimes *nested*. However, these rare incidences of inaccuracy will be condoned for the sake of simplicity.

resource	name	type	description
Lojban	gismuWords	word forms	tokens from the definition
	gismuGloss	lemmas	gloss words from the dictionary entry
	gismuPosGloss	POS-lemmas	tagged gloss[10]
	gismuKW	lemmas	lemmatized keywords from definition
	gismuPosKW	POS-lemmas	POS-tagged keywords from definition
FrameNet	frameWords	word forms	tokens from the frame definition
	frameLU	lemmas	lexical units without POS
	framePosLU	POS-lemmas	lexical units
	frameKW	lemmas	lemmatized keywords from definition
	framePosKW	POS-lemmas	POS-tagged keywords from definition

Table 3.5.: Overview of keyword extractors for Lojban and FrameNet

extractor	keywords
gismuWords	{ *speaks, utters, phonates, says, verbally* }
gismuKW	{ *speak, utter, phonate, say, verbally* }
gismuPosKW	{ *speak#VB, utter#VB, phonate#VB, say#VB, verbally#RB* }
gismuGloss	{ *utter* }
gismuPosGloss	{ *utter#VB* }

Table 3.6.: Keyword extraction results for the gismu "bacru"

x1#ARG utters#VB verbally#RB x2#ARG

A similar mapping function resulting in the same tagset was defined for the WordNet tagset $\{n, v, a, j\}$, which is used by FrameNet and PropBank.

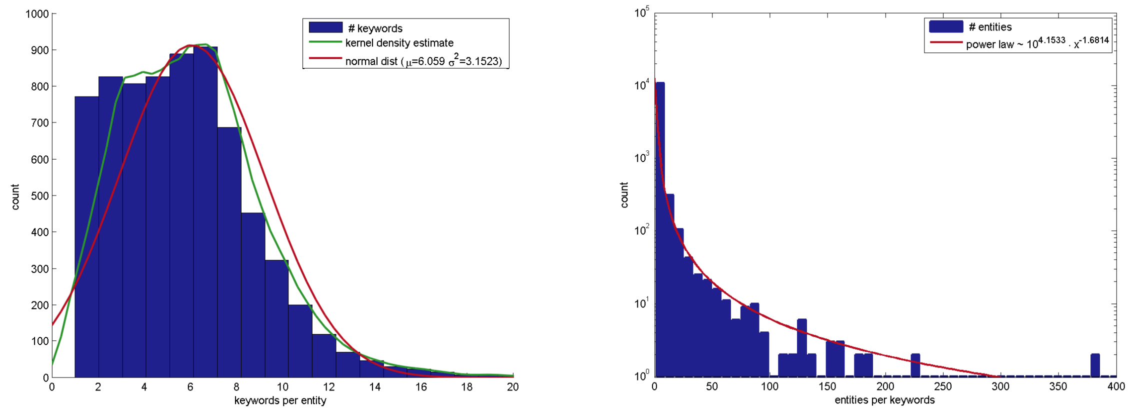
Instead of choosing one variant (word forms, lemmas, POS-tagged words, POS-tagged lemmas), we retain all these alternatives as different keyword spaces, as they may contain complementary information. For each of these, we can define a separate feature extractor, operating on a textual property of an entity. For Lojban dictionary entries, we can regard two primary text elements. Firstly, the definition string, and secondly the set of English gloss words associated with it. For FrameNet, we consider two properties of a frame that can be seen as equivalent, its set of lexical units, and its definition text. Given these properties and extraction variants, we define 5 keyword extractors for both Lojban and FrameNet which are listed in Table 3.5. Table 3.6 shows an example of the results of these feature extraction functions for the gismu `bacru`.

Keyword weighting

We have already given the prospect of *weighting* the extracted keywords. Such weights should be an indicator of the "relevance" of a keyword in respect to a given entity. In case of Lojban, this step is especially important, as some words occur very often throughout the dictionary. For example, hundreds of definitions contain the word "*material*", such as

`batke`: x_1 *is a button or knob on* x_2 *with purpose* x_3 *of material* x_4
`blaci`: x_1 *is a quantity of glass of material* x_2
`creka`: x_1 *is a shirt of material* x_2
`tanxe`: x_1 *is a box or crate for contents* x_2 *made of material* x_3
...

The keywords which are truly relevant to these gismu are {*button, knob*}, {*glass*}, {*shirt*} and {*box, crate*} for the respective entries. Yet an unweighted extraction would rate these on par with {*material*}, which is not beneficial for alignment purposes. When regarding the distribution of keywords across an ontology, it is obvious that they are not evenly distributed. Figure 3.6 shows the distribution of gismu-keyword mappings across the complete Lojban dictionary. Subfigure 3.6a depicts that the number of keywords for a given entity can be reasonably approximated with a normal distribution. In contrast, Subfigure 3.6b shows that the inverse distribution, the number of entities for a given keyword, can be much closer approximated with a *power law* – likely originating from the distribution of natural language words. Clearly, it is necessary to downweight this "long tail" of unimportant keywords (those which occur very often), and to upweight *salient* keywords (those which are rare).



(a) number of keywords per entity **(b)** number of entities per keyword

Figure 3.6: Distribution of keyword/entity associations across Lojban dictionary entries

item	weighted POS-keywords (tf-idf)
`batke`	*button#NN* → 2.94, *knob#NN* → 3.24, *item#NN* → 2.89, *purpose#NN* → 1.76, *material#NN* → 1.43, *make#VB* → 1.30, *be#VB* → 0.10
`blaci`	*glass#NN* → 3.00, *composition#NN* → 1.99, *include#VB* → 1.83, *contain#VB* → 1.50,*quantity#NN* → 1.32,*make#VB* → 1.30, *be#VB* → 0.10
`creka`	*shirt#NN* → 3.37, *blouse#NN* → 3.37, *top#NN* → 3.07, *material#NN* → 1.43, *be#VB* → 0.10
`tanxe`	*carton#NN* → 3.85, *crate#NN* → 3.85, *box#NN* → 3.54, *trunk#NN* → 3.07, *contents#NN* → 2.54, *material#NN* → 1.43, *make#VB* → 1.30, *be#VB* → 0.10

Table 3.7.: Tf-idf weighted POS keywords extracted for some example items

A common numeric statistic for obtaining the saliency of a keyword with respect to a corpus is *tf-idf*. For a corpus consisting of documents D, each consisting of a set (or multiset) of terms, the tf-idf score is defined for a term $t \in d \in D$ as the product of the *term frequency* tf and the inverse document frequency idf

$$\text{tf-idf}(t,d,D) = \text{tf}(t,d) \times \text{idf}(t,D)$$

The inverse document frequency denotes how common a given term is across all documents. It is defined as

$$\text{idf}(t,D) = log\frac{|D|}{|\{d \in D \,:\, t \in d\}|}$$

where the denominator indicates the number of documents in which t appears. The term frequency can be defined in various ways, the simplest choice being a simple *count* of term occurrences (in case a document is modeled as a set $\text{tf}(t,d) \in \{0,1\}$)

$$\text{tf}(t,d) = |\{\tau \in d \,:\, \tau = t\}|$$

Tf-idf is therefore used to obtain weights for each pair of entity and keyword. This weighting also handles *stopwords* in a meaningful way, because they obtain almost irrelevant weights (the stopword filter is retained for keyword extraction nonetheless). To illustrate the effect of *tf-idf*, the resulting weighted POS-keywords are given for the earlier examples in Table 3.7. Here, the keyword {*material#NN*} gets much less weight than those we consider significant.

3.4.2 Similarity measures

We have motivated the use of keyword extraction as an abstraction over a simple overlap count. In terms of a pair of keyword extraction functions $k_1 : O_1 \to K$ and $k_2 : O_2 \to K$, we can therefore define an overlap count similarity based on the extractors (k_1, k_2) for $k_1 \in O_1$ and $k_2 \in O_2$.

$$\text{Overlap count for } (k_1, k_2): \qquad sim(e_1, e_2) = |\, k_1(e_1) \cap k_2(e_2)\, |$$

This simple approach, although it fulfills all properties of a similarity function, does not take into account that some entities may have a larger set of keywords than others, so its value can be arbitrarily large. Other coefficients amend this issue by normalization. The *Jaccard index* normalizes with the union between both sets, whereas *Dice's coefficient* divides by the sum of keyword set cardinalities.

$$\text{Jaccard index for } (k_1, k_2): \qquad sim(e_1, e_2) = \frac{|\, k_1(e_1) \cap k_2(e_2)\, |}{|\, k_1(e_1) \cup k_2(e_2)\, |}$$

$$\text{Dice's coefficient for } (k_1, k_2): \qquad sim(e_1, e_2) = \frac{2\,|\, k_1(e_1) \cap k_2(e_2)\, |}{|k(e_1)| + |k_2(e_2)|}$$

These measures are defined on sets, whereas we have already motivated a weighted approach. To abstract over *tf-idf*, we define the notion of a weighting function. For an ontology O and keyword set K, a function $weighting(k, e, O) \in \mathbb{R}$, where $k \in K$ and $e \in O$ is used to define a weighted version of any extractor function $O \to K$. Thus, given a pair of extractors (k_1, k_2) we define a function $weight : O_1 \cup O_2 \times K \to \mathbb{R}$, which obtains the weight of a keyword k, with respect to any entity e of either ontology:

$$weight(e, k) = \begin{cases} weighting(k_1(e), e, O_1) & \text{for } e \in O_1 \\ weighting(k_2(e), e, O_2) & \text{for } e \in O_2 \end{cases}$$

This means that a weighting function, such as *tf-idf,* is applied in the scope of one ontology (it should yield the relevance of an item with respect to the ontology, not the global set of keywords). Note that for any combination of keyword extractors, and an optional weighting function, we can compute a unique similarity measure. The only requirement for this is that the extractor functions map into the same keyword space K. Figure 3.7 illustrates this fact. $k_{\{1,2,3\}}$ are extractors for the ontology O_1, whereas $k_{\{4,5,6\}}$ are extractors for O_2. As all extractors $k_{\{1,2,4,5\}}$ map to the same keyword space, 4 different similarity measures can be constructed from them.

For illustration, mappings between extractor functions from Table 3.5 have been visualized[11] in Figure 3.8 & 3.9. In these representations the elements O_1 (gismus) are shown on the left, whereas elements O_2 (frames) are on the right. Equal colors between items illustrates alignment pairs from the gold data. The nodes in the middle display the set of keyword items; overlapping keywords are connected. The thickness of the lines corresponds to the *tf-idf* weights of the keyword associations. Figure 3.8 shows the mapping “gismuPosGloss ⟷ framePosLU”. It is apparent that these extractor functions only yield a very sparse collection of keywords, and therefore only very few overlaps. Figure 3.9 shows the mapping “gismuPosKW ⟷ framePosKW”, which yields a much larger set of keywords resulting on more overlaps. Based on such weighted keyword mappings, we can now devise appropriate similarity measures.

Each mapping of extractor functions gives rise to a unique similarity measure, fully defined by the *weight* function. We can now apply a *vector space model*, and regard the weights as the non-zero entries in a $|K|$-dimensional vector space. Thus, for any entity e we can define a sparse vector $\boldsymbol{w}_e \in \mathbb{R}^{|K|}$ with non-negative elements. We can now define a similarity measure from the *dot product* of the weight vectors:

$$sim(e_1, e_2) = \boldsymbol{w}_{e_1} \cdot \boldsymbol{w}_{e_2} = \sum_{k \in K} (weight(e_1, k) \cdot weight(e_2, k))$$

This again raises the issue of normalization. We could normalize the sum of weights for each e, and thus divide by $|\boldsymbol{w}_e|_1$. A more common way of normalization however, is the use of the *cosine similarity*. This measure determines

[11] The implemented alignment framework can automatically plot these visualizations as part of the evaluation process.

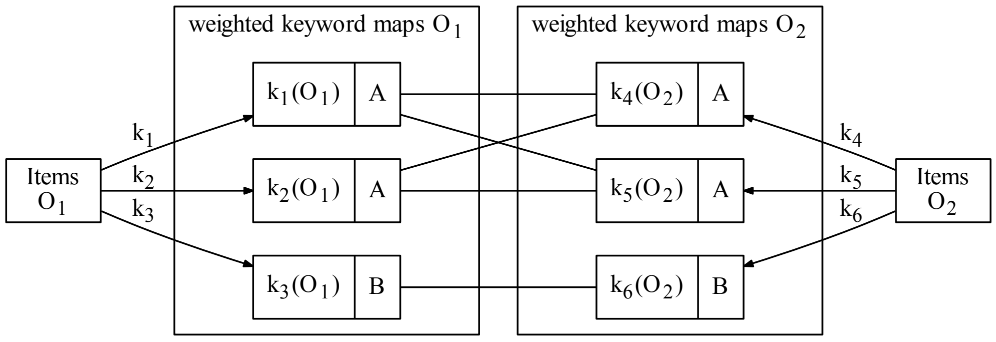


Figure 3.7: Illustration of weight mappings for different extractor functions $k_{\{1..6\}}$ and two different keyword spaces *A*, *B*

the angle between two vectors $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{R}^n$ by applying the Euclidean cosine rule $\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\|\|\boldsymbol{b}\| \cos(\theta)$. Therefore we define the cosine similarity as

$$sim(e_1, e_2) = \frac{\boldsymbol{w}_{e_1} \cdot \boldsymbol{w}_{e_2}}{\|\boldsymbol{w}_{e_1}\| \cdot \|\boldsymbol{w}_{e_2}\|} = \frac{\sum_{k \in K} (weight(e_1, k) \cdot weight(e_2, k))}{\sqrt{\sum\limits_{k \in K} weight(e_1, k)^2} \cdot \sqrt{\sum\limits_{k \in K} weight(e_2, k)^2}}$$

More abstractly, as each similarity measure spans an inner product space, cosine similarity can be used as a "wrapper" for any *inner* similarity measure. Cosine similarity is a well established similarity measure for text matching and is commonly applied to term frequency vectors and *tf-idf* weights. As these weights are always non-negative, it yields a value from 0 to 1 and is guaranteed to adhere to *positiveness*, *symmetry* and *maximality*.

Pruned lookup

On an implementational level, it is obviously not feasible to compute the similarity score between the complete set $O_1 \times O_2$. We can use *sparse matrix* representations for representing keyword mappings and only compute the similarity measure for pairs (e_1, e_2) with a non-empty set of overlapping keywords $|k_1(e_1) \cap k_2(e_2)|$. As this would still result in many cells of the final similarity matrix to be computed, we can further reduce the computational effort by performing a *pruned lookup* of possible matching candidates. To find a set of candidates from O_2 for a given element $e_1 \in O_1$, we first compute the set of relevant keywords $K' = \{k \in K \mid weight(e_1, k) > 0\}$. Based on this keyword set a pruned lookup is performed at a given threshold t.

$$candidates\left(K', t\right) = \left\{ e_2 \in O_2 \mid \sum_{k' \in K'} score(e_2, k') > t \right\}$$

$$\text{where} \qquad score(e, k') = \frac{weight(e, k')}{\sum_{k \in K} weight(e, k)}$$

For $t = 0$ we would obtain the full similarity matrix, whereas increasing this threshold would eliminate bad alignment candidates at an early stage, before computing the full similarity score. The framework thus unifies the traditional alignment steps of *search selection* and *similarity computation*.

3.4.3 Semantic similarity

An issue that remains is that keywords are only compared by equality. However, two keyword elements which are not equal may still be closely related – in case of lexical items we have already mentioned word pairs such as

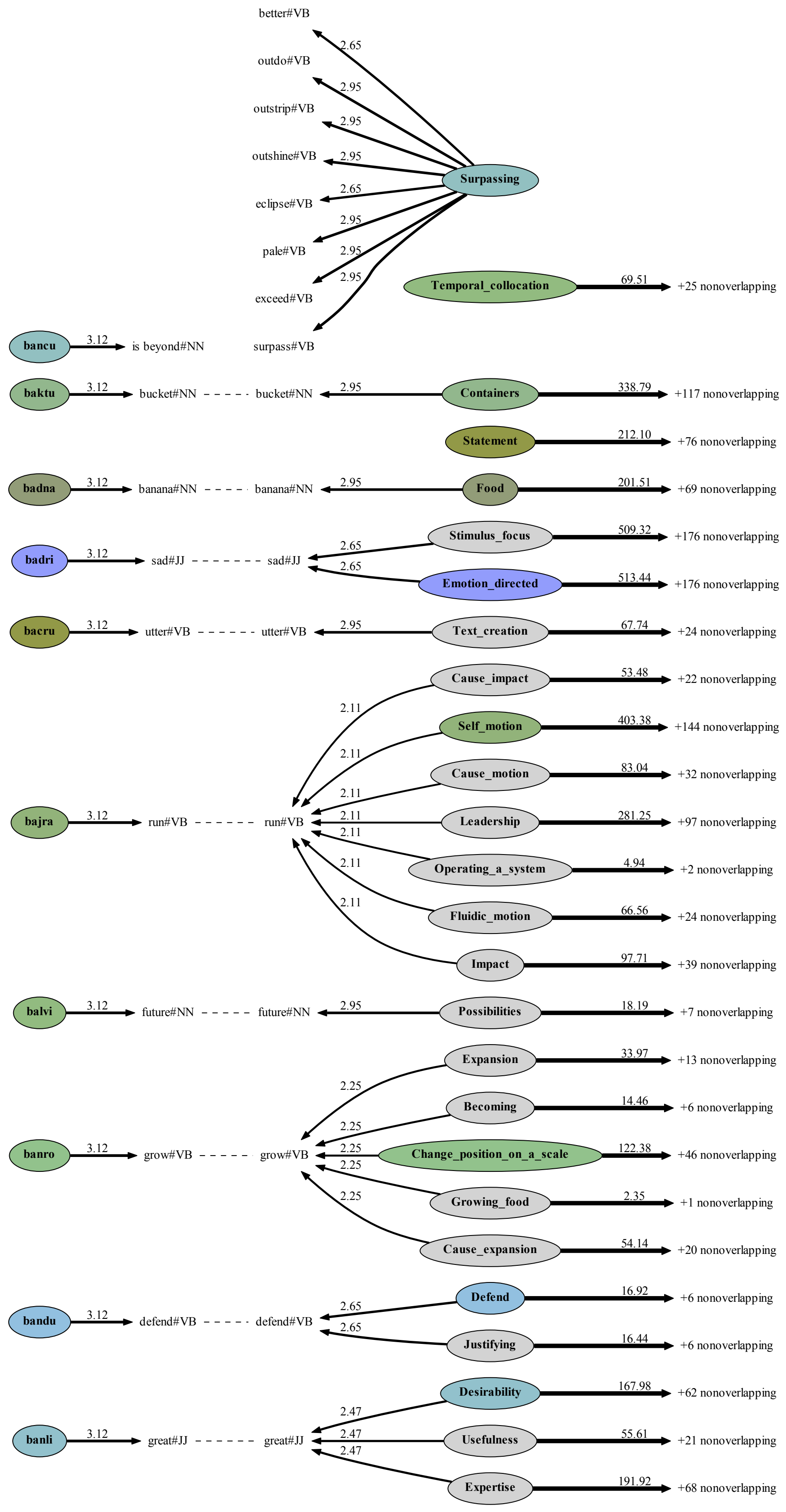


Figure 3.8: Weighted keyword visualization of the mapping gismuPosGloss ⟷ framePosLU

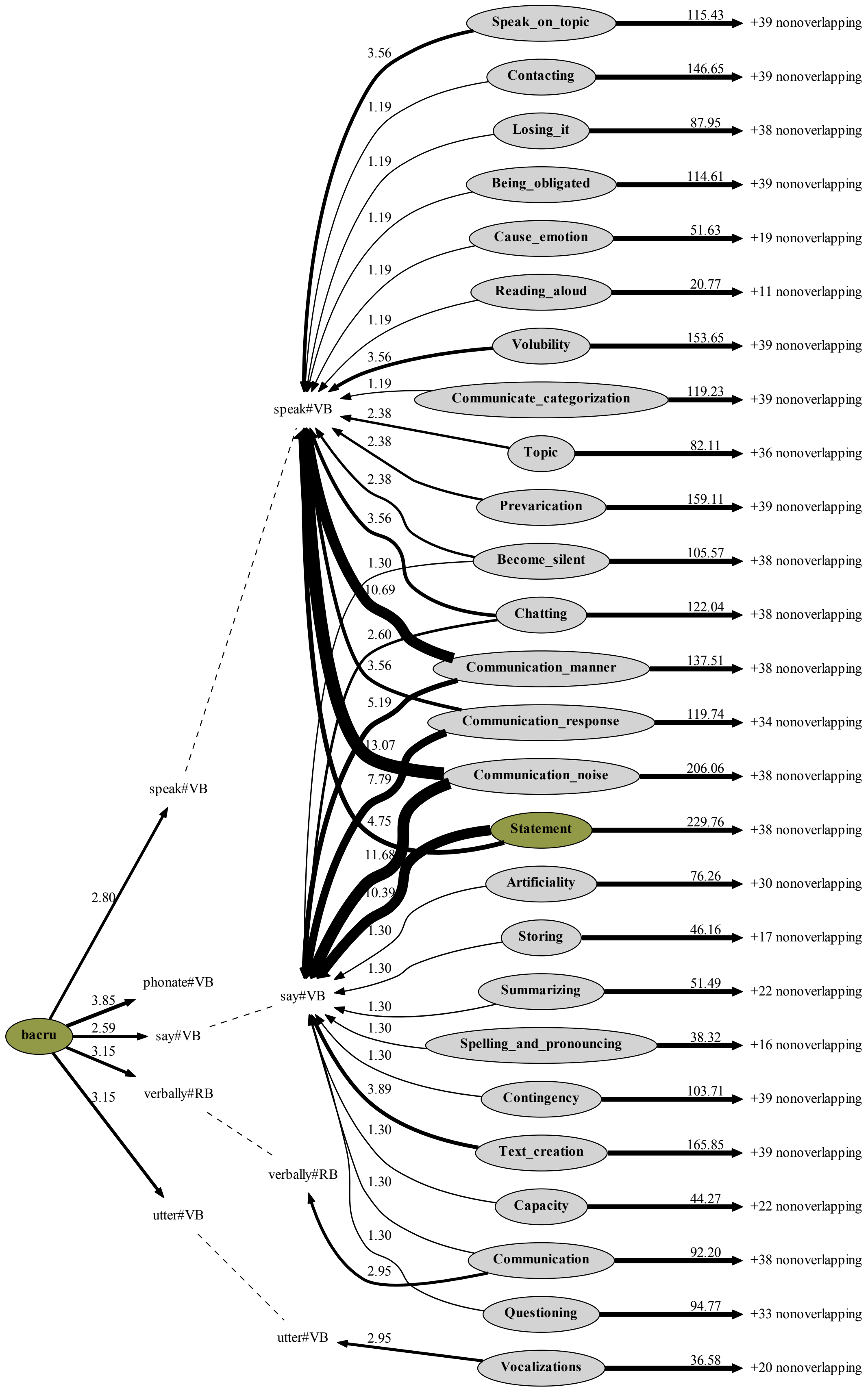


Figure 3.9: Weighted keyword visualization of the mapping gismuPosKW ⟷ framePosKW

(*“car”*, *“automobile”*). We are thus interested in the *similarity* of keywords. An elementary observation is that this is just another “layer” of the problem at hand. Conceptually, we can thus address this issue by adding another level of similarity computation within the keyword space K

$$sim_K : K \times K \rightarrow \mathbb{R}$$

In case of lexical items K, this is the well-studied problem of *semantic similarity* or *semantic relatedness*. We have previously motivated the use of external resources such as WordNet to compute similarity scores of words. These similarity measures include *LCH*, *Path*, *RES*, *Lin*, and *Lesk* (described in detail in Pedersen et al., 2004). For a pair of *words* (existing in the WN dictionary) these metrics each yield a real number indicating their semantic relatedness. Thus, given one such WN similarity sim_{WN} and a pair of entities (e_1, e_2), we can compute the complete similarity matrix between each permuted pair of keywords yielded by the extractors k_1 and k_2 and aggregate it in some way. Such an aggregation of *tf-idf-weighted* pairwise word similarities, is discussed in abundance – for instance in the context of *text similarity*. Mihalcea et al. (2006) suggest to to compute the row-wise and column-wise maxima of similarity scores multiplied with their weights, and average these values. This however is motivated by finding one-to-one correspondences between words in paraphrased sentences. For the use case at hand, we define a much simpler aggregation strategy and simple produce the *mean* of all weighted similarity scores in the matrix. Thus for a WordNet similarity measure sim_{WN} we can define

$$\text{WordNet similarity}: \qquad sim(e_1, e_2) = \frac{1}{|k_1(e_1)| \cdot |k_2(e_2)|} \sum_{\substack{a \in k_1(e_1) \\ b \in k_2(e_2)}} weight(e_1, a) \cdot weight(e_2, b) \cdot sim_{\mathrm{WN}}(a, b)$$

An alternative to such a resource-based approach is the automatic acquisition of semantic similarity scores using a *distributional thesaurus* (Lin, 1998). Here, similarity is defined in terms of a *shared context*. Biemann and Riedl (2013) describe how *distributional similarity* can be computed from large corpora in an unsupervised way. The similarity between language elements can be computed through an overlap of context features (such as syntactic features obtained from a parser), and in fact can be defined not globally but within the context of a sentence. Although in this work the use of *contextualized similarity* was not explored, the computed distributional similarity scores were used for a variety of advantages they have over the use of a lexical resource. Most importantly, they can be used much more efficiently. The unsupervised scores can be stored in a static resource, as the framework prunes the “long tail” of insignificant entries and can be parametrized to generate arbitrarily small databases. Generating such lookup tables for WordNet similarity measures is unfeasible; instead the similarity scores have to be computed at runtime. Yet another advantage of the DT framework is the fact that the language elements are not dictionary entries, but tokens processed by the same pipeline as the one that is applied to generate the keywords (in this case the Stanford POS-tagger). For these reasons, static similarity score tables are used as an efficient and high quality alternative to WordNet-based similarity measures.

Thus, we use precomputed similarity scores based on common context features. For computing the similarity between two words t_1, t_2, this score is obtained as

$$sim_{\mathrm{LMI}}(t_1, t_2) = \sum_{\substack{f \in rankedfeatures(t_1,p) \cap rankedfeatures(t_2,p) \\ f(t_1) > t \wedge f(t_2) > t \wedge score(f) > s}} 1$$

where t and s are thresholds for insignificant terms and features, and *rankedfeatures* is a function, returning only the p significant features, based on some frequency significance measure, such as Lexicographer’s Mutual Information (Kilgarriff et al., 2004). *LMI* is defined as

$$LMI(t, f) = \mathrm{freq}(t, f) \log_2 \left(\frac{\mathrm{freq}(t, f)}{\mathrm{freq}(t) \cdot \mathrm{freq}(f)} \right)$$

The similarity scores used in this work are based on a precomputed result obtained from a 120-million sentence corpus[12], using LMI and pruning to $p = 1000$ significant features. Thus, the resulting similarity scores sim_{LMI} are

[12] For computing the DT, a 120-million sentence corpus was compiled from newspaper corpora from the *Leipzig Corpora Collection* (`http://corpora.uni-leipzig.de`), and from the English *Gigaword* corpus (`http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2011T07`)

values between 1 and 1000, which can be easily normalized with *cosine similarity*. Based on these scores, a basic similarity measure can be defined analogously to the WordNet-based similarity.

$$\text{LMI similarity:} \qquad sim(e_1, e_2) = \frac{1}{|k_1(e_1)| \cdot |k_2(e_2)|} \sum_{\substack{a \in k_1(e_1) \\ b \in k_2(e_2)}} weight(e_1, a) \cdot weight(e_2, b) \cdot sim_{\text{LMI}}(a, b)$$

Likewise, we can define a different *aggregator* function aggregating the cells of the resulting matrix, which is omitted here.

3.4.4 Lexical expansion

The use of a semantic similarity measure addresses the issue of items with different but semantically related words as keywords. A problem that remains is the vast search space of $O_1 \times O_2$ alignment pairs, which were previously preselected by an overlap in keywords. Within the developed alignment framework, the semantic similarity measures as defined above work by computing a score only for those cells in the similarity matrix with at least one overlapping keyword pair. This, however, ignores all alignment pairs which do not have any keyword overlap, but would nonetheless obtain a high semantic similarity score. To address this shortcoming, *lexical expansion* can be used instead of the strategies defined above. A lexical expansion is simply a list of similar terms pruned at some significance threshold.

Conceptually, we can define a lexical expansion similarity measure which "wraps" an *inner* measure by first performing an expansion of all keywords. Then the *inner* measure is computed on this expanded set.

$$\text{LMI lexical expansion for } (inner, t, p): \qquad sim(e_1, e_2) = inner(expand(k_1(e_1), t, p), expand(k_2(e_2), t, p))$$

where p is a threshold for a maximum number of expanded terms and t is a threshold for the *LMI* similarity scores. The *expand* function simply obtains the weighted keywords from an extractor, expands them, and multiplies the weight of an original keyword with the similarity score of the expanded word. On an implementational level we can perform this lexical expansion step at an early stage. All weighted keyword mappings can first be expanded to obtain static mappings containing all expansions. As a result, this measure is able to retrieve such items with an originally non-overlapping keyword set, and also has the advantage of being much faster.

3.4.5 Alignment strategies

For a given tuple (k_1, k_2, sim) where k_1,k_2 are extractors and sim a measure as defined above, we obtain a similarity measure $M : O_1 \times O_2 \to \mathbb{R}$ – which, on an implementational level, is a *sparse matrix*. As we have multiple such matrices, we can define some elementary aggregation techniques. In Section 3.2, we have already introduced the basic *weighted sum model* or *weighted product model*. For a set of similarity measures $M_1, \ldots, M_n$ the weights $\lambda_1, \ldots, \lambda_n$ are introduced.

$$\text{Weighted sum for } (e_1, e_2) := \sum_{i=1}^{n} \lambda_i \cdot M_i(e_1, e_2)$$

$$\text{Weighted product for } (e_1, e_2) := \prod_{i=1}^{n} \lambda_i \cdot M_i(e_1, e_2)$$

These weights λ_i can be obtained in a supervised setting given some gold alignment.

In order to convert a similarity matrix into an alignment, we can consider different strategies which select a subset of $O_1 \times O_2$ entries from a given matrix M. A straightforward approach to only selecting a subset of entries is a *threshold*-based selection, that applies a threshold t to only select items (e_1, e_2) where $M(e_1, e_2) > t$. Alternatively, a *best-n* selection strategy would return the n pairs (e_1, e_2) with the highest score. We can also consider applying a *probability mass threshold* to this selection. A threshold pmt is introduced so that so that for $pmt = 0$ no pairs are selected and $pmt = 1$ all pairs are selected.

Lastly, a *limited multiplicity alignment* is motivated by the observation that alignments may have a maximum multiplicity $n : m$. Therefore, we add the constraint that an item in O_1 can align to at most n items in O_2 and an item in O_2 can align to at most m items in O_1. This strategy can be expressed as a constrained optimization problem

$$\begin{array}{rl} \text{maximize} & \sum_{(e_1,e_2)\in A} \mathrm{M}(e_1,e_2) \\ \text{subject to} & \forall e \in O_1 . |\{(e_1,e_2) \in A \,|\, e_1 = e\}| < n \\ \text{and} & \forall e \in O_2 . |\{(e_1,e_2) \in A \,|\, e_2 = e\}| < m \end{array}$$

which was approximated with a greedy strategy (shown in the Appendix, Algorithm 1). This strategy greedily selects the largest items, unless the constraints would be violated.

These selection strategies are useful for evaluating the performance of single similarity measures. In a complete alignment system, we can make use of a supervised approach using the manually aligned items as gold data. An important observation is that the combination of the *weighted sum model*, with a *threshold selection*, makes the alignment problem equivalent to that of a linear model

$$y(e_1,e_2) = sign\left(\sum_{i=1}^{n} \lambda_i M_i(e_1,e_2) - t\right)$$

This means that given binary training data $y(e_1,e_2) \in \{-1,1\}$, we can train the threshold t and weights λ_i; for instance by using logistic regression. We have already observed that we can make use of arbitrary ML classifiers by modeling the alignment task as a binary classification problem, where the similarity measures M_i make up the feature space. In the following evaluation, we therefore identify a set of useful similarity measures to be used as features, and then apply some well-proven classifiers to this classification task.

3.4.6 Evaluation

The introduced alignment strategies are evaluated with the given gold annotation data. First, the metrics, permutations of feature mappings, weighting and selection strategies are discussed. Then a set of similarity measures is selected as a feature space, and a final ML-based aligner is trained and evaluated.

Evaluation metrics

In general, an undirected alignment between the ontologies O_1 and O_2 can be evaluated by comparing a gold set $G \subset O_1 \times O_2$ against the alignment $A \subset O_1 \times O_2$. In this case, we can simply define precision P and recall R as

$$P = \frac{|G \cap A|}{|G|} \quad \text{and} \quad R = \frac{|G \cap A|}{|A|}$$

This evaluation however is only sound for a *complete* gold set. Yet, only a subset of the alignment between O_1 and O_2 has been annotated, which leads to difficulties when applying the above metric. If only the annotated subsets of the ontologies were fed into the alignment system, this would skew the results; as the subsets to be aligned get smaller, the prior probability of a correct alignment increases. At the extreme, only having one item from each ontology would make the only possible retrievable alignment the correct one.

For this reason, the annotation process was defined as unidirectional. It was required that for any $(e_1,e_2) \in G$ the alignment for e_1 is *complete*, which means that e_1 aligns to only the elements in the gold alignment $\{e \in O_2 \,|\, (e_1,e) \in G\}$. For e_2 no such guarantee is made, so it is possible that e_2 aligns to other elements in O_1 than those in G. Therefore, we evaluate an alignment strategy as follows

1. The complete ontologies O_1 and O_2 are the input of the alignment process, yielding an output alignment A.
2. We evaluate only the alignment direction $O_1 \rightarrow O_2$. For this we define the set I to be the annotated O_1 items in the gold data. Now we can define a subset of the alignment used for evaluation, and the according metrics for the "forward" precision and "forward" recall.

$$A_{\text{eval}} = \{(e_1,e_2) \in A \,|\, e_1 \in I\}$$

$$P = \frac{|G \cap A_{\text{eval}}|}{|G|} \quad \text{and} \quad R = \frac{|G \cap A_{\text{eval}}|}{|A_{\text{eval}}|}$$

alignment mapping	R	P	F_1
gismuKW ⟷ frameLU	0.436	0.656	0.524
gismuGloss ⟷ frameLU	0.445	0.633	0.522
gismuPosKW ⟷ framePosKW	0.301	0.530	0.384
gismuPosKW ⟷ framePosLU	0.428	0.765	0.549
gismuPosGloss ⟷ framePosKW	0.292	0.413	0.342
gismuPosGloss ⟷ framePosLU	0.479	0.702	0.569

Table 3.8.: Evaluation of mappings

The F_1-measure is the harmonic mean of P and R

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R}$$

Note that in this evaluation, we disregard *unalignable* items.

For the basic alignment strategies discussed in the following, we do not need any training data, so the evaluation is done on the complete set of gold data.

Keyword mapping

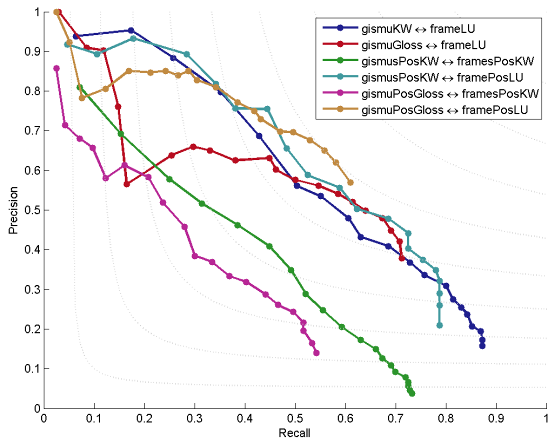
As a first evaluation, the performance of different mappings is evaluated. For the sake of reducing the number of permutations, on the Lojban side the mappings {gismuGloss, gismuKW, gismuPosGloss, gismuPosKW} are regarded, whereas on the FrameNet side only the mappings {frameLU, framePosLU, framePosKW} are selected for evaluation. This results in 6 different (compatible) mapping permutations. Table 3.8 shows the results for a simple aligner using the *dot product* similarity measure, with an *n-best* selection strategy ($n = 1000$ was chosen arbitrarily but loosely based on the absolute size of the ontologies). It is apparent that the mappings behave differently. The mapping from any gismu feature to POS-tagged lexical units on the FrameNet side has a very high precision. On the other side, any mapping using the POS-tagged keywords from the FrameNet definition is very inaccurate, and has a low recall in particular.

We can further evaluate the range of possible P/R tradeoffs. A *probability mass threshold* is used to sample the complete range of alignment selections, which is illustrated in Figure 3.10. Subfigure 3.10a shows the P/R measures for varying thresholds. An important fact to note is that no alignment mapping achieves a perfect recall of 1.0 even without any thresholding (a threshold $pmt = 1$ is equivalent to selecting all entries of the resulting similarity matrix). As expected, a pmt close to 0 achieves a very high precision for all measures. It is also obvious that some measures such as "gismuPosGloss⟷framePosKW" perform much worse than others. The reason for this is probably the sparseness of gloss words for gismu, paired with relatively arbitrary keywords in the frame definitions. The measure "gismuKW⟷frameLU" has the highest recall while still having an even balance to precision. The reason for this is probably that both gismu keywords as well as frame lexical units are definite and reliable. As expected, an overlap here is a good indicator for alignment. A notable peculiarity is a sharp drop in precision of the "gismuGloss ⟷ frameLU" mapping. This is due to most gloss words appearing only once in the gloss dictionary, and therefore obtaining the same *tf-idf* weight. These hapax alignments all have the same score, and are thus included in the alignment at the same threshold. Subfigure 3.10b shows the F_1 score for a varying threshold. Obviously, there is some optimal value at which this metric is maximized. For most values, $pmt = 0.5$ is a reasonable choice, although the mapping "gismuPosKW⟷framePosKW" drops much earlier. This is probably due to relatively arbitrary words being used in the definitions of either ontology – only a large degree of word overlap is a good indicator for alignment in this case.

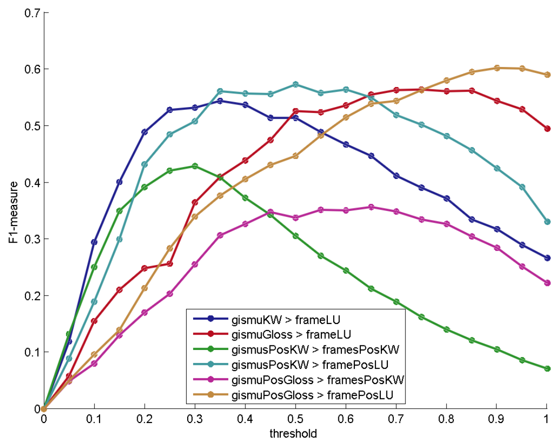
Similarity measure

The next aspect of evaluation is the identification of good similarity measures. As all methods are parameterized, we could permute these possible parameters to obtain all unique similarity measures. This however, would result in a vast number[13] of possible similarity measures which cannot be exhaustively explored. Instead, some subspaces of this search space will be evaluated.

[13] Even when disregarding numeric parameters (such as thresholds), as well as nested similarity measures, the number of possible parameter permutations is nearly 30,000.



(a) P/R plot for varying threshold



(b) F_1 plot for varying threshold

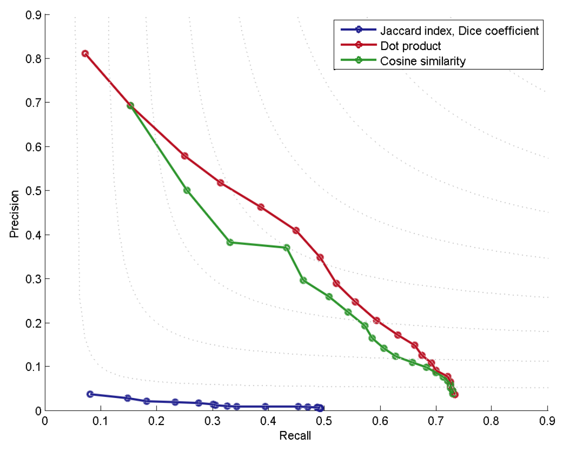
Figure 3.10: Evaluation of alignment mapping with pmt threshold

similarity measure	**weighting**								
	unweighted			*tf-idf*			*row-normalized tf-idf*		
	R	P	F_1	R	P	F_1	R	P	F_1
Jaccard index	0.297	0.016	0.030	0.297	0.016	0.030	0.297	0.016	0.030
Dice Coefficient	0.301	0.015	0.029	0.301	0.015	0.029	0.301	0.015	0.029
dot product	0.487	0.016	0.031	0.449	0.409	**0.428**	0.356	0.251	0.295
cosine similarity	0.390	0.017	0.033	0.508	0.259	0.343	0.508	0.259	0.343

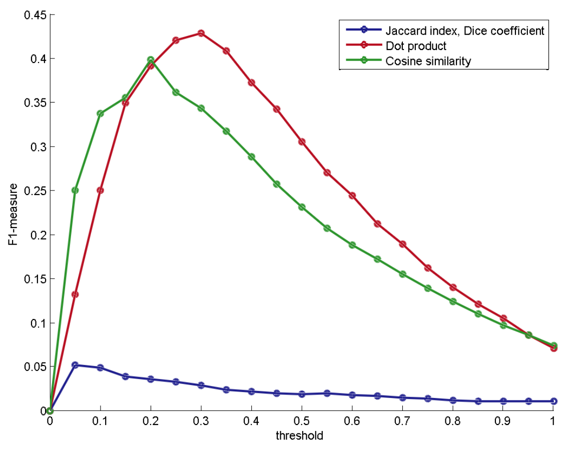
Table 3.9.: Comparison of different similarity measures (without lexical expansion) for a single mapping of POS-tagged keywords

As a first step we can evaluate the performance of basic measures in isolation. For this, we regard the averagely performing mapping "gismuPosKW ⟷ framePosLU", and fix the selection strategy to a probability mass threshold selection ($pmt = 0.3$). Table 3.9 shows the metrics of different weighting schemes and similarity measures. It is apparent that weighting is essential for the performance (Jaccard index and Dice coefficient do not seem to work at all). Normalizing the *tf-idf* scores per row – to account for a different number of keywords per entity – does not have any improving effect. Interestingly, *cosine similarity* normalization also impacts performance negatively (this is further visualized in Figure 3.11). An explanation why an unnormalized *dot product* outperforms the *cosine similarity* may be that – in contrast to document-term models where each document can be adequately described with the *direction* of a term vector – in this case most "documents" consists of only very few keywords. Therefore, strong agreement in *one* dimension may be more important than an agreement in *many* dimensions, and otherwise highly ranking pairs are wrongly penalized by cosine similarity.

Next, *semantic similarity* measures as defined in Section 3.4.3, are evaluated. For this purpose, we regard the "gismuGloss ⟷ frameLU" mapping (chosen for its small number of keywords) and compare the different measures. For all lexical similarities, aggregating the score matrix with a *mean* was most efficient (on par with *max*, whereas *sum* generally scored bad). The WordNet similarity scores did not differ much in this evaluation, so only the *RES* similarity is considered. Figure 3.12 shows the evaluation of some lexical similarity measures, namely WordNet *RES*, *LMI* similarity scores and a lexical expansion measure. Note that for "WordNet *RES*" and "*LMI* similarity" the existing entries of an alignment matrix $O_1 \times O_2$ are "updated" with the score obtained from a semantic similarity measure, and the recall is thus bounded to those pairs entries which have at least one common keyword. As a result, for a threshold $t = 1$ these measures result in the same alignment. Interestingly, the overall performance (F_1) is almost the same for these measures. WordNet *RES* (and for that matter, all WN scores) performed worse than the unexpanded baseline, and can be considered ineffective for this purpose. The LMI similarity, while retaining almost the same F_1 as an unexpanded alignment, achieves a higher precision. Lexical expansion can effectively improve the alignment for high ranking alignment pairs, but drops in performance very fast when not properly pruned.

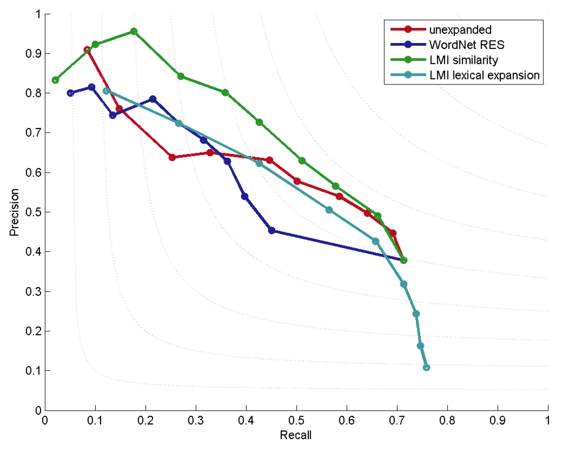


(a) P/R plot for varying threshold

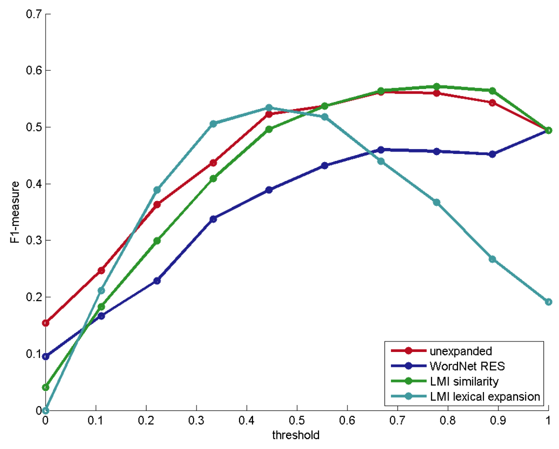


(b) F_1 plot for varying threshold

Figure 3.11: Evaluation of different similarity measures for the mapping "gismuPosKW $\longleftrightarrow$ framePosLU"



(a) P/R plot for varying threshold



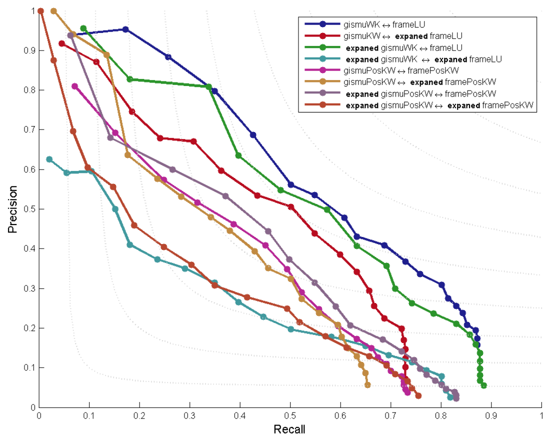
(b) F_1 plot for varying threshold

Figure 3.12: Evaluation of lexical similarity measures (mapping gismuGloss $\longleftrightarrow$ frameLU)

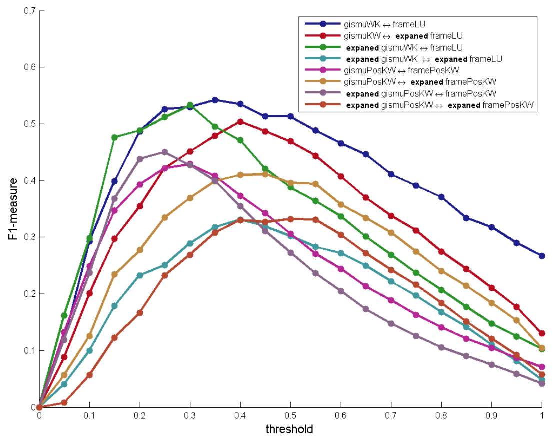
Lexical expansion was further evaluated in isolation, to first determine how it performs for two different mappings and to determine the difference of expanding only one side of the keyword mappings. Figure 3.13 shows this evaluation. Nearly all combinations exhibit the same behavior, although it can be observed that expanding *both sides* of an alignment mapping performs worst. It can also be observed that the permutations in which only the Lojban side was expanded, perform better than others. The Lojban dictionary entries are very sparse, which means that expanding these sparse keywords is generally better than expanding the denser keywords on the FrameNet side.

Machine Learning

We have now established the parameters to be used and identified a set of similarity measures. Although multiple formulations as a classification task are possible, we have already outlined the training of a binary classifier, given this set of similarity measures as features. As the input of this classifier, we chose any alignment pair for which at least one similarity measure is positive, and define its label to be {*true*} if it is in the gold data, and {*false*} otherwise. This means that there is a huge class bias for {*false*} instances which has to be considered for tuning the parameters of the respective models. For evaluation, only the {*true*} class is relevant (in concordance with the definition of P and R in this alignment task). As a feature set the keyword mappings from Table 3.8 is chosen,



(a) P/R plot for varying threshold



(b) F_1 plot for varying threshold

Figure 3.13: Evaluation of lexical expansion strategies for different mappings

classifier-based aligner	**on training set**			**10-fold cross validation**		
	P	R	F_1	P	R	F_1
Logistic regression	0.766	0.587	0.664	0.734	0.563	0.637
SVM (RBF kernel)	1.000	0.967	0.983	0.523	0.802	0.634
SVM (polynomial kernel)	0.821	0.467	0.595	0.696	0.659	0.677
Decision tree (C4.5)	0.934	0.677	0.785	0.729	0.563	0.635
Random forest	0.976	0.972	0.974	0.754	0.668	**0.709**
Rule learner (JRip)	0.820	0.737	0.776	0.697	0.664	0.680

Table 3.10.: Classifier performance

combined with both *dot product* and *cosine similarity*, in both expanded and unexpanded variants. This results in a feature set of 24 different similarity measures for each pair.

Various classifiers[14] have been evaluated, both on the training set and using 10-fold cross validation. The training set performance is given in order to assess how well the learner can be applied, and decide if the task is learnable in the given model. For the SVM models, the parameters have been adapted to account for class bias, but otherwise these learners have not been tuned. Judging only from the on-training metrics, a relatively complex model is necessary to obtain a good alignment. *Decision trees*, and *random forests* (Svetnik et al., 2003), seem to work well for this task. The SVM with RBF kernel performs best on the training set, but does not generalize well. Using a polynomial kernel instead reduces overfitting, but is not sufficient to learn a good model. The best performing aligner is obtained with a *random forest*.

At a maximum $F_1 = 0.709$, the final performance of the statistical aligner is still not outstanding. However, only very few data is available for Lojban words, and thus only basic entity-level syntactic similarity measures could be employed. For example, the minimal definition of Lojban strings makes it unfeasible to employ word n-grams of $n > 1$ as a keyword space. Furthermore, no instance-based training data is available. For the subset of gismu, there is also no graph-based information which could be incorporated into the alignment. Considering these limitations, the quality of the alignment is surprisingly accurate. Possible improvements to the alignment process are outline as future work in Section 5.2.1.

Ultimately, an ontology alignment such as the task at hand is often fully supervised by an expert. The output of the statistical aligner should therefore be regarded as a guideline for the annotator. By feeding a high-recall aligner with a reasonable precision into the graphical alignment tool implemented in this work, the annotation process is sped up drastically.

[14] For SVM classifiers *LibSVM* was used (`www.csie.ntu.edu.tw/~cjlin/libsvm/`).
For the remaining classifiers, the *WEKA 3.6* implementation was used (`http://www.cs.waikato.ac.nz/ml/weka/`).

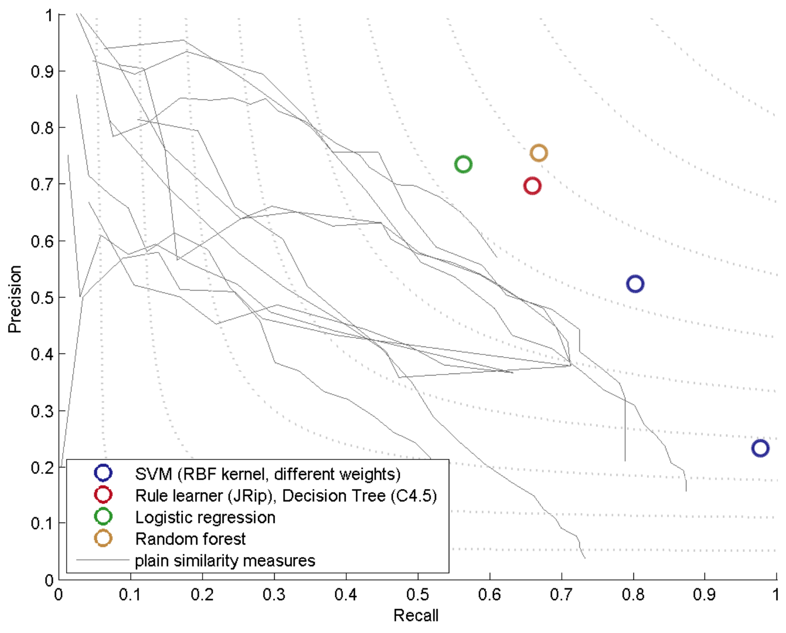


Figure 3.14: P/R plot of various ML-classifiers. The gray lines visualize the single similarity measures in isolation

4 Semantic parsing with Lojban

In Chapter 1, various semantic parsing tasks were introduced. In Chapter 2 Lojban was presented as a semantic resource similar to those commonly used for semantic annotation. Ultimately it was shown in Chapter 3 that Lojban can be reasonably aligned to such ontologies, at the example of FrameNet. Through this alignment we anticipated the use of Lojban as a semantic ontology, and have already laid the groundwork for a semantic parsing system.

The relevance of Lojban in context of semantic parsing is in fact twofold. First, we may consider it as a language and apply existing semantic parsing tasks (such as FrameNet-based semantic annotations) to Lojban prose. As a matter of fact, we can very easily devise an FSP system for this task by combining a Lojban syntactic parser with an ontology alignment obtained through means discussed in Chapter 3. Semantic parsing of Lojban prose is discussed in Section 4.1, and can be regarded as a preface to the second, and main interpretation of Lojban as an ontology.

In this second scenario, the Lojban dictionary is regarded as inventory of semantic predicates that can be used to annotate natural language text. Accordingly, in Section 4.2 a Lojban-based semantic annotation task is defined, and the implementation of the system is outlined. Subsequently, the semantic parser is evaluated on English text, through the use of parallel corpora (Section 4.3). Table 4.1 gives an overview of the resulting tasks and references the relevant sections. As a conclusion, a use case of the semantic parser as a brivla search system is showcased in Section 4.4.

4.1 Semantic parsing of Lojban prose

Semantic parsing tasks, such as FSP and SRL, were outlined in Chapter 1 to be very difficult NLP problems. Most of these difficulties arise from the ambiguity and irregularity of natural language. As Lojban is an attempt to eliminate these linguistic issues, applying these tasks to Lojban prose is much easier. The well-defined syntactic structure of Lojban combined with an unambiguous semantic interpretation based on a finite set of rules makes the implementation of a Lojban semantic parser near-trivial. A non-statistical (and perfectly accurate) semantic parser *for* Lojban was implemented and is described in the following.

4.1.1 Semantic Lojban parser

In the overview of Lojban parser projects (Section 2.2.3) it was observed that most existing parsers struggle with capturing the full set of possible Lojban expressions. Parsers producing a highly abstract ("semantic") representation of Lojban generally handle only a smaller subset of the language than those producing a purely syntactic representation. To illustrate this difference in abstraction, consider the Lojban equivalent of the sentence *"I'm going to university by bus"*

```
mi klama lo balcu'e fu lo sorprekarce
```

A high-level parser would (among many other steps) identify the arguments (namely, `mi`, `lo balcu'e` and `lo sorprekarce`) as the x_1,x_2, and x_5 arguments of the `klama` relation. This is done for example by *jbofi'e*, the output of which is shown in Listing 4.1. In contrast, a syntactic parser like *camxes* does not operate on such an abstract level. Its parse output (shown in Listing 4.2) is a tree representation with relatively arbitrary node names. It is not apparent that `klama` is the main predicate of the sentence, and what constituents are substituted into which argument places. For example, the cmavo `fu` indicates that the following expression is substituted into the 5th argument; yet this information is not processed. Arguable, the first representation is much closer to what we

	Lojban text	Natural language text
FrameNet semantic parsing	Lojban FN annotator (Section 4.1.2)	*FSP* (Section 1.2.4)
Lojban semantic parsing	Lojban parser (Section 4.1.1)	*LSP* (Section 4.2)

Table 4.1.: Overview of Lojban-related semantic parsing tasks

```
[ ( mi            ) << klama >> ( lo     balcu'e    ) ( lo            sorprekarce)]
[ ( I, me         ) << go-ing >> ( the   univeristy(s)) ( the                bus)]
[ ( klama1 (go-er(s))) <<    >> ( klama2 (destination(s))) ( klama5 (transportation means)]
```

Listing 4.1: The output of the *jbofi'e* parser, yielding a "semantic" representation

```
text=(
  sentence=(
    CMAVO=( KOhA=( mi ) )
    bridiTail3=(
      BRIVLA=( gismu=( klama ) )
      terms=(
        sumti6=(
          CMAVO=( LE=( lo ) ) BRIVLA=( lujvo=( balcu'e ) ) )
        term1=( CMAVO=( FA=( fu ) )
          sumti6=( CMAVO=( LE=( lo ) ) BRIVLA=( lujvo=( sorprekarce ) ) ) ) ) ) ) )
```

Listing 4.2: The output of the camxes parser, yielding a pure syntactic representation

have established to be "semantic parsing". However, *jbofi'e* is an unmaintained project[1] which can only handle a small subset of the language. *Camxes* on the other hand is a robust system that supports most Lojban syntax features and is even resilient to malformed sub-expressions. Therefore, a camxes-based semantic parser has been implemented, which transforms the syntactic parse tree into a semantic representation. Such a multi-pass system has been proposed before (Wirick, 2005; Goertzel, 2005a). This task essentially requires the "grammar rules" of Lojban (as defined in the *CLL*, Cowan, 1997a) to be implemented one-by-one. Figure 4.1 shows this transformation for the given example sentence. Whereas the nodes in the syntactic parse tree are simply the rule-names of the PEG-grammar, the nodes in the semantic representation are the formalization of Lojban grammar as a datastructure. This representation captures the *function* of Lojban expressions rather than their syntactic realization and can be considered a form of *linguistic normalization*[2]. This means there exists an $n : 1$ mapping of syntax trees to the semantic format; all syntactic realizations of the same content are mapped to an equivalent semantic tree. In this semantic format, we can easily recover the information of interest – in this case, the predicate-argument relations. By extracting all *Bridi* nodes from the semantic parse tree, we can effectively create annotations for Lojban prose identifying each predicate and its respective arguments, as shown in Figure 4.2.

The resulting annotation system is therefore fundamentally different from those employed for natural languages as it is non-statistical. Sentences of arbitrary complexity can still be parsed, and the result is well-defined. Nevertheless, a note on coverage of the parser has to be made. Even though the rules of Lojban are finite, they are still numerous and complex. The implementation effort for a *complete* system is beyond the scope of this work, so that the subset of acceptable expressions is further reduced. The order of implemented rules was chosen in decreasing size of their respective error classes. The final system has a coverage of 65% of Lojban prose according to the `all_books` corpus. In order to fully implement the system, a "long tail" of grammar rules would have to be formalized[3].

4.1.2 FrameNet-parser for Lojban

At this point, the semantic annotations of Lojban prose do not provide any more information than the textual representation (in fact, they are obtained by *dropping* most information). However, we have already given the prospect of devising a FrameNet-parser for Lojban prose simply by mapping those annotations to the corresponding frames. For this, an ontology alignment from Lojban brivla to FrameNet frames, as discussed in Chapter 3, is used. Although we have defined the alignment task as an $n : m$ mapping, we also observed that if a brivla is alignable, it generally aligns to 1 to 3 frames. In contrast to natural language, in which a target unit also evokes multiple frames, for a brivla the set of alignable frames is not vastly different in meaning (see Section 3.3). It is sufficiently

[1] As stated on the project page. The parser is publicly available at `https://github.com/lojban/jbofihe`, accessed June 2014.
[2] The goal of linguistic normalization is to map different but semantically equivalent phrases onto one canonical representative phrase.
[3] For example, Lojban has a complete sub-language for mathematical expressions, which accounts for a large fraction of grammar rules, but only a minuscule amount of actual text.

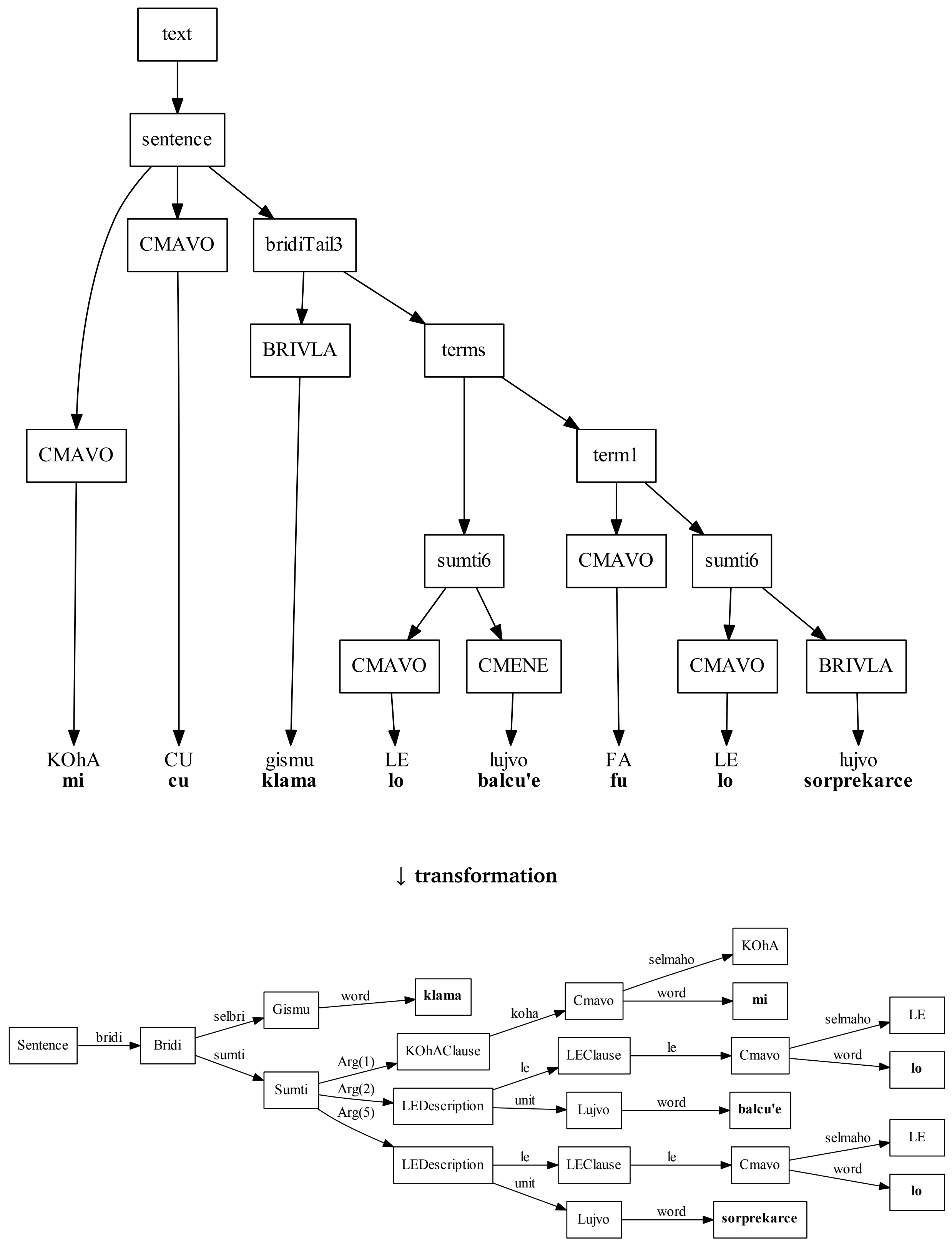


Figure 4.1: Transformation of the *camxes* syntactic parse into a Lojban-semantic representation

```
mi klama lo balcu'e fu lo sorprekarce
- klama
  |- x1="mi"
  |- x2="lo balcu'e"
  |- x5="lo sorprekarce"
```

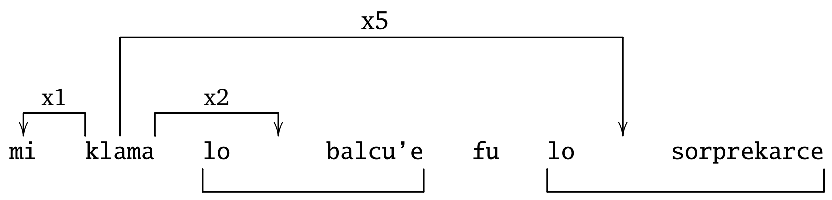


Figure 4.2: "Frame semantic" annotation of Lojban, in textual representation and as dependency edges

accurate to just select the *primary*[4] frame which has been aligned. The complete FrameNet FSP system is thus very simple, the full algorithm is shown in Appendix 4.1.2. Only frames for which a mapping exists are annotated, and unalignable predicates as well as unaligned argument slots are ignored. If we now input an alignment that is incomplete but error-free – such as the gold annotations – we obtain a *partial*, but correct annotation of any Lojban corpus. If we instead choose an automatic alignment, we get an FSP system that can be argued to perform equivalent to the quality of the alignment . Using this FSP system with the existing gold alignment, FrameNet-annotations of all Lojban corpora were produced. The example sentence rendered into FrameNet's fulltext XML format is shown in Listing 4.3. It corresponds to the FrameNet annotation

$$[_{\text{Theme}}\,\texttt{mi}]\ \texttt{klama}^{\text{Motion}}\ [_{\text{Goal}}\,\texttt{lo balcu'e}]\ \texttt{fu}\ [_{\text{Carrier}}\,\texttt{lo sorprekarce}]$$

Obtaining these annotations for Lojban prose may be interesting for building FSP-based systems on top. In the scope of this work, it is merely observed that FrameNet-annotations for Lojban corpora can be acquired easily, as a direct consequence of the properties of the language.

4.2 Lojban semantic parsing

The use of Lojban as a resource for semantic annotation of natural languages has been envisioned various times throughout this work. In the following, we will formally define this semantic annotation task, and ultimately devise a *Lojban Semantic Parser* (LSP). For this, we will annotate English[5] text with predicate-argument structures obtained from the Lojban dictionary. The motivation for such a system are various, but are more or less the same as those for any formal meaning representation (as illustrated in Chapter 1). Obviously an accurate annotation of English sentences with well-defined argument structures would support the semantic understanding of the text – regardless of the sense inventory in use. The argumentation is thus, that Lojban can work just as well for semantic annotation and may even offer advantages such as a larger coverage and language independence. Another outlook which arises is the use of such a system to perform a full translation of natural language text to Lojban. In a formal language such as Lojban, "sentences" can be trivially constructed. Given Lojban semantic annotations, the gap to a complete translation system is therefore much smaller than that for a natural language. This outlook is further envisioned as future work in Section 5.2.2.

In the following, we first formally define the LSP task, sketch the outline of the system, and then describe its components in detail.

[4] For the manual alignment, the corresponding FrameNet frames were annotated as a ranked list. For an automated alignment, the highest ranking alignment pair is regarded as the *primary* frame.

[5] The task could be performed easily for other languages as well, as Lojban definitions are available for many languages. However, the NLP toolchain necessary for the following steps is most advanced for English. For this reason, and to limit the scope of the work, Lojban Semantic Parsing is not performed on other languages. In the following, we use "English" as a placeholder for what in most cases could be substituted by "any natural language".

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="fullText.xsl"?>
<fullTextAnnotation
xsi:schemaLocation="../schema/fullText.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://framenet.icsi.berkeley.edu">
  <header>
    <corpus name="Example Sentences" ID="001">
      <document description="Automatically created annotation of Lojban text" ID="001"/>
    </corpus>
  </header>
  <sentence corpID="001" docID="001" paragNo="0" sentNo="0">
    <text>mi klama lo balcu'e fu lo sorprekarce</text>
    <annotationSet
    luName="go.v" luID="4345" frameName="Motion" frameID="7" ID="1000001" status="MANUAL">
      <layer rank="1" name="Target">
        <label cBy="1000001" start="3" end="7" name="Target"/>
      </layer>
      <layer rank="1" name="FE">
        <label name="Theme" feID="26" start="0" end="1" fgColor="FFFFFF" bgColor="800080"/>
      </layer>
      <layer rank="1" name="FE">
        <label name="Goal" feID="29" start="9" end="18" fgColor="FFFFFF" bgColor="FF0000"/>
      </layer>
      <layer rank="1" name="FE">
        <label name="Carrier" feID="1968" start="23" end="36" fgColor="FFFFFF"
            bgColor="00008B"/>
      </layer>
</annotationSet>
  </sentence>
</fullTextAnnotation>
```

Listing 4.3: Automatically generated FrameNet annotation of the example sentence, exported into FN-fulltext XML format.

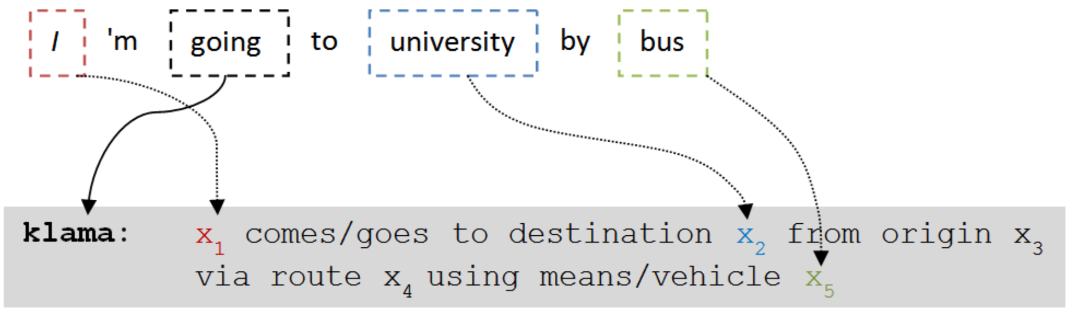

I
'm
going
to
university
by
bus
klama: x1 comes/goes to destination x2 from origin x3
via route x4 using means/vehicle x5

Figure 4.3: Illustration of a Lojban Semantic Parser (LSP) for the English sentence *"I'm going to univeristy by bus"*

4.2.1 Problem definition

Lojban has not been designed to be used as an annotation language like FRAMENET. Yet, it is capable of formalizing the complete scope of meaning in language, which is even a superset of the information captured by FRAMENET annotations. A full "Lojban-formalization" of an English sentence can be regarded as a complete *translation*. This however is a much harder task than the one we define. In LSP, we attempt to perform only the following steps, given an English sentence

1. Identify a set of *brivla* being evoked by a lexical item in the English sentence
2. For each identified brivla, identify its arguments as annotations of spans in the English original

This task, which is clearly inspired by FSP, is illustrated in Figure 4.3 with an example sentence. Here, we are given an English source sentence, "*I'm going to university by bus*". The objective of LSP is to identify that the lexical item "*going*" evokes a Lojban brivla, which in this case is `klama`. The next step is the assignment of arguments in regard to the definition of this brivla, which are the assignments

$x_1 \leftarrow \text{"}I\text{"}$
$x_2 \leftarrow \text{"}university\text{"}$
$x_5 \leftarrow \text{"}bus\text{"}$

Diverging from FRAMENET, we do not assume prepositions ("*to*" and "*by*" in this case) to be included in the argument spans, as this is clearly covered by the definition string – the idea is that we can easily substitute the substrings from the English sentence into the definition and get a reasonable statement.

Next, we should provide a more specific definition as to what exactly constitutes in "evocation" in step (1). The main difference of Lojban to FRAMENET is that it was not designed to be built "on top" of a given natural language sentence and instead spans an independent meaning space. As a consequence, it is not always possible to reproduce the exact predication contained within a sentence, which is mostly the case when *idioms* are used. To illustrate this issue, consider the following simple sentence from the "`IntroOfDublin`" corpus annotated with both FRAMENET and LSP annotations.

This busy, modern European city sits on a thousand years of history [...]

The FRAMENET annotation for the main verb in this sentence can be visualized as

$$[_{\text{Theme}} \textit{This busy, modern European city}]\ \textbf{\textit{SITS}}^{\text{Being\_located}}\ [_{\text{Location}} \textit{on a thousand years of history}].$$

Here, the highly abstract frame *Being_located* is evoked, which is defined as "A *Theme* is in a stable position with respect to a *Location*". However, such an annotation does not at all convey the content of the sentence, which could be paraphrased as "*This* [...] *city has a long history*". In Lojban, such abstract predicates do not exist, and retrieving the gismu `zutse` (x_1 *sits [assumes sitting position] on surface* x_2), which corresponds to the *Change_Posture* frame, would obviously not make much sense. Instead, identifying a brivla such as `clani` (x_1 *is long in dimension* x_2 [..]) is more correct, although disproportionately harder. Nevertheless, it leads to a very simple correctness criterion of brivla invocation.

brivla evocation for an English sentence s, a set of evoked brivla predicates B is correct, if there *exists* a Lojban translation of s in which all $b \in B$ occur
(strong version: all $b \in B$ are used as *selbri*)[6].

In case of the example sentence, we can assert that `clani` is a correct evocation, as the English sentence could be translated as "`lo tcacu be lo ti tolcando citri cu clani`" ((the history of (this modern city)) is long), with `clani` being the only selbri. Here, "a thousand years" was also considered an idiom, which was not translated to Lojban. In this work, we do not attempt to resolve such issues of idiomatic expressions. Still, the above criterion is considered an elementary aspect of the LSP task definition. As Lojban enforces clarity and disallows idiomatic expressions, the translation constraint effectively enforces an annotation task which *isolates* meaning from linguistic mannerisms. This is a stark contrast to frame semantic parsing. Although it addresses ambiguity on the level of senses and semantic roles, it is not free from idiosyncrasies of natural languages – in fact, FrameNet embraces the fact that there exists a frame such as *Being_located*, which accounts for a city to "sit" on its history.

A direct consequence of the definition for *brivla evocation* is that there is not just one correct annotation. As there are hundreds of possible translations for a single given sentence, equally many correct annotations can be produced. It should thus be expected that LSP annotations have a low inter-annotator agreement. Nevertheless it can be argued that the LSP task is a worthwhile effort. The low inter-annotator agreement originates not from an inaccurate model or ambiguously defined annotation guidelines; instead it stems from the inherent challenges of semantics[7].

For the practical implementation of an LSP system we can simply assume that *one possible* annotation should be produced. Evaluating the correctness of such annotations is an essential step that will be addressed in the following. Not incidentally, the correctness criterion for *brivla evocation* conveniently lines up with the existence of parallel corpora, which already contain thousands of English sentences paired with Lojban translations.

4.2.2 Use of parallel corpora

When implementing a Lojban semantic parser one major challenge is the complete lack of annotated data. As opposed to all other semantic parser systems, absolutely no data is available for training, as Lojban has never been used for this purpose. Fortunately, there exist at least some parallel corpora, which can be used to automatically create annotated data. To anticipate overly optimistic presumptions, this data is not sufficient to serve as *training data* for a statistical system (for a lack in quantity). In this work, this data will therefore only be used for evaluation[8]. To illustrate the usefulness of Lojban parallel text, we first recall the result of two elementary processing steps:

1. The text has been automatically aligned on sentence and word level, as described in Appendix B.1
2. Lojban text can be unambiguously parsed, and Lojban semantic annotations can be produced as shown in Section 4.1.1

An interesting synergy emerges from the combination of the output of these steps, which is visualized in Figure 4.4. Consider the following sentence pair from the `alice_in_wonderland` corpus

Alice went timidly up to the door, and knocked

`ni'o la .alis. cu toldarsi klama lo vorme gi'e darxi`

Firstly, we can obtain a word-alignment between the Lojban words and English tokens. Secondly, semantic annotations are obtained from the Lojban semantic parser as shown in Listing 4.4. Note that the argument x_2 for `darxi` ("*knock*") is not present because it is erroneously left unspecified in the Lojban sentence. We now have a set of predicates in the sentence, each with a set of arguments. When now considering the word alignment table, it is possible to find correlating English words in the source sentence, as illustrated in Figure. 4.5 The resulting annotations can thus be listed as

[6] The strong version is a slightly modified task which effectively excludes any brivla which do not act as predicates (in most cases this means that they are arguments).

[7] Diverting not too far off in philosophy, cognitive semantics theorizes that meaning is created when language is perceived by a sentient being. This means that a *different* meaning can exist for each perceiver of an utterance. This difference can be argued to ultimately manifest in different translations. Therefore it can be argued that any semantic annotation task attempting to formalize meaning must acknowledge the existence of multiple interpretations. Constraining a meaning formalization task in such a way that only one realization is possible thus also confines the cognitive process which creates meaning in the first place.

[8] Supposing a much larger set of parallel Lojban-English text, the outlined procedure could be used unaffectedly, thus creating sufficiently large sets of data to be used for training in future approaches.

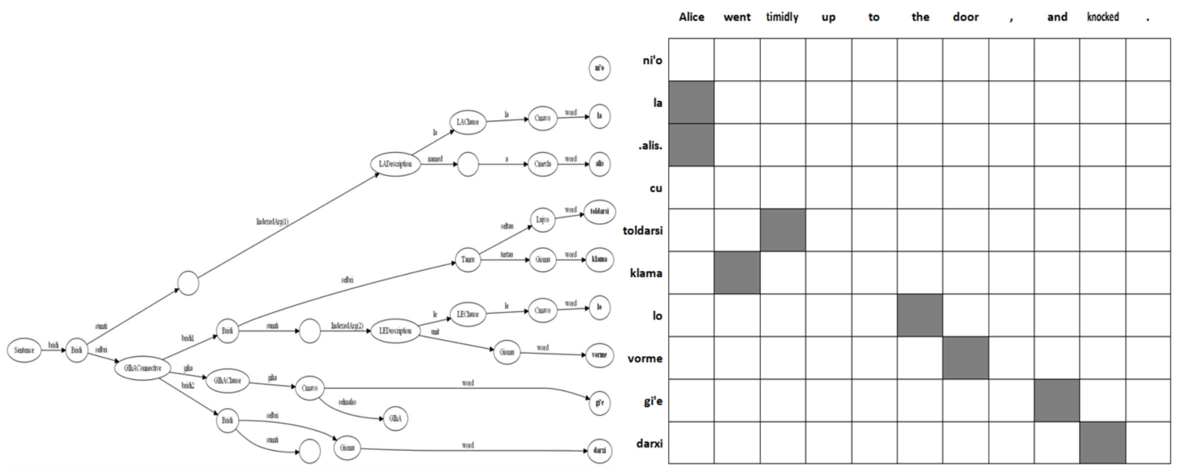


Figure 4.4: The output of a Lojban semantic parser for the Lojban translation of an English source sentence, and the corresponding word-alignment

```
- klama
  | - x1="la .alis."
  | - x2="lo vorme"
- darxi
  | - x1="la .alis"
```

Listing 4.4: Semantic parse result for the Lojban sentence "`ni'o la .alis. cu toldarsi klama lo vorme gi'e darxi`"

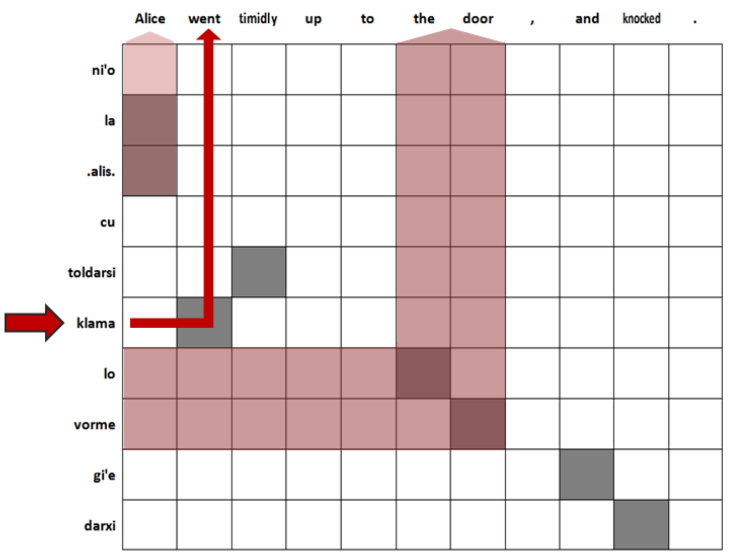


Figure 4.5: Correlation of Lojban predicate annotations with English source tokens

```
# Sent. 172 from alice_in_wonderland
The Mouse gave a sudden leap out of the water, and seemed to quiver all over with fright.
  leap     plipe   x1=Mouse   x2=water
  seemed   simlu   x1=Mouse   x2=quiver
  quiver   desku   x2=fright
  fright   terpa
```

Listing 4.5: Automatic annotation using a customized word-aligner, yielding only one-token arguments

Alice went timidly up to the door, and knocked.

$\left[_{x_1} \textit{Alice}\right]$ *timidly* ***went***$^{\texttt{klama}}$ *up to* $\left[_{x_2} \textit{the door}\right]$ *and knocked.*

$\left[_{x_1} \textit{Alice}\right]$ *timidly went up to the door and* ***knocked***$^{\texttt{darxi}}$.

This means, that we can automatically produce annotations for both evocations of predicates as well as argument assignments. An issue that should be noted however is the bad alignment quality of the word alignment which was obtained. Using the GIZA++ toolkit, IBM models 1-5 and a HMM word alignment model are used to obtain alignment probabilities between tokens in the source text (Och and Ney, 2003). This process relies on co-occurrence counts and generally requires large corpora. At a complete size of less than 1 MB, this is obviously not fulfilled, and as a result the word alignment is incorrect in many cases. As an alternative to the word alignment produced by *GIZA++*, a workaround was implemented only based on the unigram alignment probabilities. All non-cmavo from the Lojban sentence and all non-stopwords for the English sentence were selected, and the best possible alignment between single tokens was produced using a *LimitedSelection*$(n{=}1, m{=}1)$ strategy, as devised in Section 3.4.5. This yields a much more reliable result, although it sacrifices multiword argument spans. As an example, consider Listing 4.5, which does not contain the correct spans, but still lists a correct token covered by an argument.

With this data available, we can now use it as a gold standard for evaluating LSP systems. Due to the non-statistic parsing of Lojban, which either yields a correct result or fails otherwise, this data is highly reliable. However, as explained above, this gold data constitutes only one possible annotation, so the evaluation should be taken with a grain of salt.

4.2.3 Use of frame semantic parsers

Before elaborating on the LSP system implemented in this work, we first comment on an alignment-based approach envisioned in Chapter 3. In Section 4.1.2 we employed Lojban-FrameNet alignments to transform the output of a Lojban-parser to FrameNet annotations. It is plausible to perform this step in the opposite direction. This method was implemented with the SEMAFOR FSP system. Here, the mapping process is not as trivial as the Lojban→FrameNet direction for a number of reasons. First and foremost, in the inverse direction we have to consider lexical units instead of whole frames, as there is a one-to-many multiplicity for a given frame, and the evoked lexical unit is necessary to disambiguate the exact predicate. This means that we need a complete alignment between LUs and brivla, which is a much harder alignment task (not performed in the scope of this work). However, presupposing such an alignment on lexical unit level, we can infer the brivla quite easily. The remaining alternatives for a given LU can usually be selected by regarding the compatibility of arguments. To illustrate this, consider an invocation of the *Statement* frame in the following sentence fragment

$\left[_{\text{Message}} \textit{'I'd rather finish my tea,'}\right]$ ***said***$^{\text{Statement}}$ $\left[_{\text{Speaker}} \textit{the Hatter}\right]$ ...

Here the alignment alternatives narrow down as we incorporate information about the semantic parse

FSP information	**possible brivla**
Statement frame	{ ba'urnoi, skicu, bacru, cusku, ciksi, ba'usku, tavla }
LU ("*say.v*", *Statement*)	{ bacru, cusku, ciksi, ba'usku, tavla }
LU ("*say.v*", *Statement*) with FEs { *Speaker, Message* }	{ cusku, ba'usku }

In this case, `cusku` and `ba'usku` are the only brivla to be used with direct speech, whereas the other ones are used with a *Topic* or *Audience* argument. The difference in meaning of these two remaining brivla is minuscule (`ba'usku` refers to audible talking, whereas `cusku` can mean any form of expression), and we can simply chose any

```
# alice_in_wonderland sent. 730
"I 'd rather finish my tea , " said the Hatter , with an anxious look at the Queen , who was reading
    the list of singers .
  queen     catni x1=Queen
  read      tcidu x2=the list of singers
  tea       tcati x1=tea
  finish    mulno x1=my tea
  anxious   xanka x1=the Hatter
  look      catlu x1=anxious  x2=at the Queen , who was reading the list of singers
  said      cusku x1=the Hatter x2=" I 'd rather finish my tea , "
  list      cfika x1=list
```

Listing 4.6: LSP result obtained from the alignment-based transformation of SEMAFOR output

(or prefer the gismu, as it is more abstract). The complete algorithm is given in Appendix B.5. The important part is a selection among multiple predicates by the following simple heuristic:

$$b \leftarrow \arg\max_{b \in B_c} \left(\frac{2 \cdot |Aligned(b) \cap E|}{|Aligned(b)| + |E|} \right)$$

Here, $b \in B_c$ is a predicate candidate, $Aligned(b)$ is the set of frame elements which are *aligned* for b, and E is the set of *evoked* predicates. The given score is the *Dice coefficient*, measuring the set overlap of *instantiated* and *evoked* elements, which formalizes the selection with respect to frame elements. Listing 4.6 shows the final output for the example sentence, using an automatic frame-level and LU-level alignment with gold-annotated argument-level alignments. A fallback-strategy which first looks for an alignment on LU-level and then falls back to frame level works reasonably. For example, the LU (*tea.n, Food*) existed in the alignment, so this annotation was correctly mapped to the Lojban word for tea (`tcati`). In case of the word "queen", no lexical unit (*queen.n, Leadership*) was present, so the fallback-strategy selected the frame-level alignment for the *Leadership*-frame. Instead of the correct brivla `noltruni'u` (queen), it thus yielded `catni` (authority), but this can be regarded as a reasonable approximation. Surprisingly, the fact that FrameNet has much more frame elements ($\varnothing = 9.5$) than Lojban has arguments ($\varnothing = 2.7$; see Chapter 3) did not occur as an error in any of the manually inspected annotations. This is probably due to the fact that many of the defined frame elements are only rarely evoked. The remaining gaps in argument overlap can generally be resolved by the described selection strategy. We can therefore observe that if a complete alignment on all levels existed (frame-level, LU-level and argument-level), this strategy would yield a very accurate LSP result that is only bounded by the quality of the underlying FSP system. Obviously, the errors from the SEMAFOR system propagate. For example the brivla `catlu` ("x_1 *looks at* x_2"), obtained from the *Perception_active* frame, has incorrect argument assignments. Here, the word "anxious" is erroneously labeled as the frame element *Perceiver_agentive*, and we therefore obtain the wrong assignment

$$x_1 = \text{"anxious"}\ \textit{looks at}\ x_2 = \text{"the Queen, .."}$$

Even when the underlying FSP system makes no errors, the result of this system is not perfect. First, there is a technical difficulty of identifying the lexical unit in case of multiword expressions. For example, the sentence "*I took a picture*" invokes the LU "*take_((picture)).v*" (verbatim). Inferring this LU entry is non-trivial and cannot be resolved through simple lemmatization. Secondly, FSP systems such as SEMAFOR even account for *unseen* lexical items, meaning that the frame-evoking element in the text does not exist in FrameNet (and is therefore not present in the alignment). Lastly, it is not possible to ascertain the accuracy of this system without acquiring a complete ontology alignment on all levels. Since only an incomplete alignment is available (lexical units and arguments have not been automatically aligned in this work) this LSP system cannot be adequately evaluated quantitatively.

Nevertheless, it is an important observation that FrameNet annotations correlate strongly with Lojban annotations, which supports the assumption that the ontologies can be unambiguously aligned. In Section 5.2.1 it is envisioned as future work to incorporate the approach given here with the main system (explained in the following).

4.2.4 System outline

In the following, an autonomous system for Lojban Semantic Parsing will be introduced. Although we have illustrated that the use of an existing FrameNet-based system and an alignment-based translation strategy is feasible, this delegates the task to an underlying FSP system in the first instance, and the obtainment of a full alignment in the second instance. Both of these subtasks are error-prone, and can be regarded as a compromise solution. Therefore a novel semantic parser is implemented tailored to the LSP task at hand. We have already determined that – as opposed to most other semantic parser systems – we cannot rely on training data. Therefore it is not possible to apply a state of the art approach, such as a joint inference model for evocations, brivla selection and argument assignment. For this reason, we will compartmentalize the system into subcomponents, which perform each step in isolation.

For a given English sentence s, the following sub-steps are performed.

1. **Target unit extraction**
   Despite not formally required by LSP, the notion of target units (TU) will be adapted for practical reasons. Accordingly, this component extracts a set TU from s. As target units we consider single tokens[9], each evoking exactly one brivla (frame). The following steps are defined for each $tu \in TU$.
2. **Brivla matching**
   This corresponds to *frame disambiguation* for FSP systems. In contrast to FrameNet, given a target unit tu, we do not have a limited set of frame candidates. Instead, we must select a matching brivla b by considering *all* dictionary definitions. This step makes LSP notably more difficult than FSP, which we will elaborate after the outline. The brivla matching component is modeled as a *ranked retrieval* task, given the sentence and target unit, and a set of *features* computed from them. We can motivate this ranking by the *probability* of a brivla b given the features obtained from s and tu, thus $P(b \mid features(s, tu))$. This enables us to adapt a Bayesian view, incorporating the prior probability of a brivla $P(b)$.
   $$P(b \mid features(s, tu)) \propto P(b) \cdot P(features(s, tu) \mid b)$$
   We can estimate these priors easily based on their frequency counts in corpora.
3. **Argument parsing**
   Given the sentence s, target unit tu, and an evoked brivla b, this step assigns *continuous substrings* of the sentence to the argument slots of b. We subdivide this task further.
   a) **Argument candidate selection**
      This component identifies a set of argument candidates C. As a simplification, this step only considers different subsets of Phrase-structure tree constituents as obtained from a statistical parser.
   b) **Slot assignment**
      This component assigns the argument slots of the brivla S_b to argument candidates C to produce an argument parse $A \subset S_b \times C$. This assignment can be modeled by considering the alignment probability of each $(slot, arg)$ pair given features obtained from s, tu, and b
      $$P((slot, arg) \mid features(s, tu, b))$$
4. **Reranking**
   This optional last step assigns a score to the alignment A obtained in step 3 and uses this score to rerank the retrieved frame b in step 2. However, it was found to hurt overall performance.

In the defined components, we used a *feature* extraction function. In a general ML setting, this feature extraction is performed on instance data. However, no LSP annotations for English text are available, so it is not possible to train a system in this conventional sense.

We briefly compare this to the setting of FrameNet, which makes it plausible to perform the frame disambiguation step as a multiclass classification. Only a small number of frames are possible per target unit (average of 5.9 frames per TU). Adapting this model to Lojban would necessitate the number of classes to be equal to all brivla in the dictionary (over 7000). This is clearly unfeasible, as the data is too sparse to account for most of these instances. In fact, most brivla are not even instantiated once in the training data.

Although *some* data would be available for this purpose, instead an unsupervised, *knowledge-based* approach is motivated. It is based on the observation that Lojban is a self-contained resource which is fully and minimally defined only with a finite set of brivla definition strings. For human speakers, these short definitions suffice for correct usage without any additional instruction. Motivated by this, a LSP system is implemented that only requires the Lojban dictionary as input[10]. There are two main ideas to this process, loosely based on the alignment work done in Chapter 3.

[9] For simplicity, we disallow multiword expressions as target units.
[10] The use of frequency counts as priors are an exception to this, but they can be trivially acquired.

1. A *similarity measure* between a brivla definition string and a brivla evocation (target unit within a sentence) can be computed given a set of features.
2. As brivla definitions themselves are strings in the respective language to be parsed, the *same* feature extraction can be applied to both brivla definitions and text instances.

Thus, the approach can be summarized as using a set of feature extraction functions on both the instance level (target units in English text) and the ontology's entity-level (brivla definitions). From there, we can use saliency-based weighting and obtain a set of similarity measures as a new feature space. This similarity-based feature space is more resistant to sparsity as it abstracts over the individual ontology entities.

In the next sections, the application of this concept to the subtasks of LSP is described.

4.2.5 Features

The features introduced in the following are not typical in FSP and SRL settings (See Figure 1.8 in Section 1.3). Although some of the standard features have been implemented, the focus is on features which can be used to define similarity measures and therefore counteract the extreme sparsity of available data. The features introduced in the following have a different object domain for each subcomponent (1), (2) and (3).

$$\begin{aligned} \text{sentence feature:} \quad & f(s) \subset K \times \mathbb{R} && (1) \\ \text{target unit feature:} \quad & f(s, tu) \subset K \times \mathbb{R} && (2) \\ \text{span feature:} \quad & f(s, tu, (from, to)) \subset K \times \mathbb{R} && (3) \end{aligned}$$

Sentence features $f(s)$ operate on a complete sentence s, target unit features $f(s, tu)$ further require a target unit tu (a token within s), and lastly span features $f(s, tu, (from, to))$ further operate on a complete span in s, identified by the token indices *from* and *to*. Span features are also defined with respect to a target unit, as they can produce contextual information relative to the TU token. As a co-domain for features we consider nominal values from a keyword space K which may be weighted with a real weight.

Chapter 3 already provides a solid set of lexical features which can be easily adapted. Accordingly, we can extract the *words*, *lemmas*, *poswords*, and *poslemmas* of either a complete sentence or a specific token tu. For spans, we consider all *covered* tokens within the span, and thus obtain *coveredWords*, *coveredLemmas*, etc. Lexical expansion can also be integrated into feature extraction, and we can define "expanded" versions of these features, *wordExpanded*, *coveredWordsExpaned*, etc. This lexical expansion is equivalent to the expansion described in Section 3.4.4. For whole ranges, the similarity scores are normalized across all tokens. This means that longer spans generally receive lower weights for all expansions, whereas shorter ranges receive higher weights.

"went to the door" $\xrightarrow{expand}$ *go (0.23), door (0.22), window (0.05), ..*
"the door" $\xrightarrow{expand}$ *door (0.70), window (0.16), gate (0.13), ..*

This effect is intended, as the first expression is neither very similar to "*go*" nor to "*door*".

Analogously we define *ISA-expansion*. On an implementational level this is similar to a lexical expansion but here we replace the lookup table with one obtained from "ISA" patterns. These patterns can be used for automatically acquiring hyponymy relations from large corpora (Hearst, 1992). The name "ISA" is given by the most simple of these patterns.

$$(a)\#NN \textit{ is a } (b)\#NN \implies a \xrightarrow{is\ a} b$$

This expansion can be applied to a single token or a complete span, and yields a weighted list of words indicating possible "classes" or "kinds" a token belongs to. For example the token "*jaguar#NN*" is expanded as

jaguar#NN $\xrightarrow{is\ a}$ *car:0.20, brand:0.17, company:0.13, automaker:0.12, vehicle:0.10, animal:0.09*

Note that we can observe at least two senses of this word; one refers to an animal whereas the other refers to a vehicle. These IS-A expansions could be *contextualized* for the occurrence of the "*jaguar#NN*" token within a sentence – analogously to semantic similarity expansion (Biemann and Riedl, 2013). In this work however, we

sentence feature	description	example
		Alice went to the door and opened it
words (s)	extracts all non-stopword tokens from s	Alice, went, door, opened
head (s)	extracts all head-words from s	went
verbs (s)	extracts all tokens from s, tagged as *#VB*	went, opened

TU feature	description	example
		Alice went to the door and opened it
word (s, tu)	extracts token	went
lemma (s, tu)	extracts lemma	go
pos (s, tu)	extracts POS tag	VB
posword (s, tu)	extracts token with POS tag	went#VB
poslemma (s, tu)	extracts lemma with POS tag	go#VB
isHead (s, tu)	true, if tu is the headword	true
depEdgeLemma (s, tu)	extracts all dependency edges with lemmas	nsubj(*go,Alice*) prep_to(*go,door*) conj_and(*go,open*)
depEdgeHoledLemma (s, tu)	extracts dep-edges with “holed” targets	nsubj(*go,*@) prep_to(*go,*@) conj_and(*go,*@)
depEdgeDirLabels (s, tu)	extracts directed labels from dep-edges	-nsubj, -prep_to, -conj_and, root

Span feature	description	example
		Alice went to the door and opened it
length $(s, tu, (from, to))$	token length of span (*to - from*)	4
relpos $(s, tu, (from, to))$	relative position of span to tu, (negative for left side)	0
coveredWords $(s, tu, (from, to))$	extracts all non-stopword tokens in the span $(from, to)$	went, door
coveredLemmas $(s, tu, (from, to))$	extracts all non-stopword lemmas in the span $(from, to)$	go, door
coveredPosw $(s, tu, (from, to))$	extracts all non-stopword poswords in the span $(from, to)$	went#VB, door#NN
coveredPoslem $(s, tu, (from, to))$	extracts all non-stopword poslemmas in the span $(from, to)$	go#VB, door#NN
lexExpand $(s, tu, (from, to))$	performs ISA-expansion on all lemmas	go (0.23), door (0.22), window (0.05), gate (0.048), ..
isaExpand $(s, tu, (from, to))$	performs ISA-expansion on all tokens	story (0.08), time (0.08), thing (0.07), tool (0.07), ..
topNodes $(s, tu, (from, to))$	extracts all minimal span-covering PST nodes	VP
allNodes $(s, tu, (from, to))$	extracts all PST nodes within the span	VP, PP, TO, DT, NP
depLemmas $(s, tu, (from, to))$	extracts all non-inner dep-edges and holes the span	nsubj(@,Alice), conj_and(@, open)
depInner $(s, tu, (from, to))$	extracts all inner undirected dep-edge labels	det, prep_to

Table 4.2.: Overview of simple syntactic features. Not all possible permutations are shown.

will not perform this disambiguation and simply consider all senses of the word for expansion[11]. For a complete span, the *isaExandedLemmas* feature further regards all lemmas within the span as a bag-of-words, expands each in isolation, and unifies their expansions. Whereas a proper sense disambiguation is an interesting addition for future work, the way in which this feature will be used makes this not strictly necessary (see Section 4.2.5.2).

4.2.5.1 Argument holing

Argument holing is a method inspired by the holing operation devised by Biemann and Riedl (2013). They define the @@-operator (spoken: holing), which defines a “hole” in a structural observation, which is used to split it into two parts. The first is the holed out part, which can be thought of as a *word*[12], whereas the second is the remainder of the observation and can be considered a *context feature*. For example, for dependency edges of the format “$label(from, to)$”, one can define a holing operation $label(@@, to)$. This operation generates pairs such as $go \rightarrow prep\_to(@@, door)$. In a semantic parsing setting we use a similar kind of “hole”, by defining the “@”

[11] The obtained ISA-patterns were pruned at a manually tuned threshold and some erroneous words were removed. For example words like “problem” and “issue” were ISA-expanded for most words, because given a sufficiently large corpus everything “*is an* issue”, and everything “*is a* problem”.

[12] Although the holed-out part in many cases is a *word*, the @@-operation is generic and does not specify the kind of extracted elements. To lift these preconceptions, the two parts were arbitrarily named *Jo* and *Bim*.

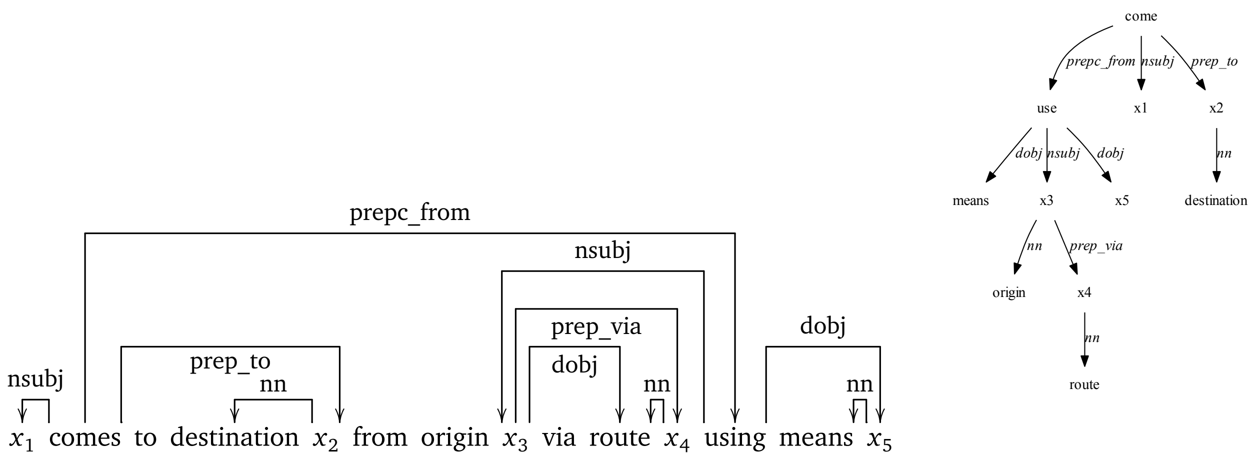


Figure 4.6: Collapsed dependency edges for a dictionary definition

placeholder to always be an *argument*, whereas the context is an indicator of the *predicate*. We first consider how this is applied to Lojban dictionary definitions, and then define general feature extraction functions.

In case of Lojban definitions we can syntactically identify arguments. They are always letters with a numbered index and can be trivially and reliably found. We therefore "hole out" these placeholder markers. Consider for example, the definition for `klama`

x_1 comes/goes to destination x_2 from origin x_3 via route x_4 using means/vehicle x_5

When processing this sentence, we first perform "slash expansion" (See Figure 3.4 in Section 3.4.1), and then apply a default NLP processing pipeline to each of the resulting sentences. The Stanford PCFG parser yields the following *typed dependencies edges* and their respective *collapsed* representation (De Marneffe and Manning, 2008).

typed dependencies	**collapsed typed dependencies**
nsubj(comes, **x1**)	nsubj(comes, **x1**)
prep(comes, to)	prep_to(comes, **x2**)
pobj(to, **x2**)	nsubj(using, **x3**)
prep(comes, from)	nn(**x4**, route)
nsubj(using, **x3**)	prep_via(**x3**, **x4**)
prep(**x3**, via)	prepc_from(comes, using)
nn(**x4**, route)	nn(**x5**, means)
pobj(via, **x4**)	dobj(using, **x5**)
pcomp(from, using)	
nn(**x5**, means)	
dobj(using, **x5**)	

These dependency edges are also visualized in Figure 4.6. It can be observed that the dependency parse is not very reliable for Lojban definition strings as the placeholder structure throws off the statistical parser[13]. However, even if the dependency graph is incorrect on a sentence level, the dependency edges immediately adjacent to the main verb and most of the Lojban arguments are correct.

Accordingly, the holed dependency edges are extracted for each argument slot. For every argument placeholder (shown in bold) we replace this argument with an "@" character and regard the remainder as a feature. The result of this extraction is shown in Table 4.3. In this setting, the *collapsed* dependency edges have proven to contain much more valuable information than the ordinary ones. We will see that this structural feature is useful for both

[13] As opposed to POS tagging, any attempt at substituting other tokens (e.g. "something") for these placeholders has made the results worse.

brivla	short definition	x_1	x_2	x_3	x_4	x_5
`klama`	x_1 *goes to* x_2 *from* x_3 *via* x_4 *to* x_5	nsubj(come,@), nsubj(go,@)	prep_to(come,@), prep_to(go,@)	nn(@,origin), nsubj(use,@)	prep_via(x3,@)	dobj(use,@)
`vecnu`	x_1 *sells* x_2 *to* x_3 *for* x_4	subj(sell,@), nsubj(vend,@)	dobj(sell,@), dobj(vend,@)	prep_to(sell,@), nn(@,buyer), prep_for(@,x4), prep_to(vend,@)	prep_for(x3,@)	-
`cteki`	x_1 *is a tax on* x_2 *levied against* x_3 *by authority* x_4	nsubj(tax,@), nsubj(levy,@), nsubj(duty,@)	nsubj(levy,@), prep_on(tax,@), prep_on(levy,@), prep_on(duty,@)	prep_against(levy,@)	prep_by(levy,@)	-
`cukta`	x_1 *is a book containing* x_2 *from* x_3 *for* x_4 *preserved in* x_5	nsubj(book,@)	dobj(contain,@)	agent(contain,@), prep_for(@,x4)	partmod(@,preserve), prep_for(x3,@)	prep_in(preserve,@)

Table 4.3.: Argument-holed dependency edges

the *brivla matching* step as well as *argument parsing*.

We will now generalize this processing step as a set of feature functions which can be applied to free text. For this, we define slightly different behavior for a *target unit feature* than for a *span feature*. The TU feature $depEdgeHoledLemma(s, tu)$ extracts all dependency edges in which tu is either of the two connected nodes. The other participating token is replaced by a hole "@". For example, given the sentence *"Alice went to the door and opened it"* and a target unit index $tu = 1$ (*went*), this feature would extract the dependency edges

> *nsubj(go,@)*
> *prep_to(go,@)*
> *conj_and(go,@)*

In this simple case it can be observed that this syntactic information already overlaps with the definition to a great degree. In some cases, it may even be desirable to only consider the *(directed) label* of the dependency edge, which effectively discards both lexical items and would only yield the result *-nsubj*, *-prep_to*, and *-conj_and*[14]. For span features an inverse behavior is desired, so that the dependency edges are holed for all tokens within the covered token range. As an example, consider the following underlined parts to be token spans. In *"Alice <u>went to the door</u> and opened it"*, the extracted dependency edges are *conj_and(@, open)*, and *nsubj(@,Alice)* whereas for the span *"Alice went to <u>the door</u> and opened it"*, one would obtain the edge *prep_to(go, @)*. Here, all "inner" dependency edges within the span are discarded, and only incoming edges $depLemmaIn(s, tu, span)$ and outgoing edges $depLemmaOut(s, tu, span)$ are considered. Practically, this distinction of incoming and outgoing edges has no effect on the final system, so we only consider the union of these features, $depLemmas(s, tu, span)$. The feature extraction on brivla can now be restated as using *span features* on all argument tokens within the definition, and *target unit features* on all remaining non-stopword tokens.

4.2.5.2 Noun types

Although we have motivated the same feature extraction function for both brivla definitions and text instances, there are a few cases in which we have to divert from this rule. One reason to do that is because we have additional information about the format of brivla definition strings. A *noun type* feature is motivated by what could be considered typing information for each argument slot. In most cases some information about the *type* of each argument filler is provided in form of an English *noun*. When regarding the definition for `klama` (given above), it can be observed that it provides the information that x_2 is a "destination", x_3 is an "origin", x_4 is a "route", and x_5 is either a "means" or a "vehicle". We can extract these types with very simplistic syntactic processing.

1. **Argument-holed dep-edges**
   We apply argument holing as described above. By considering only *nn*-edges, we can already cover most of the typing information.

[14] To avoid confusion, the *direction* of an edge is always defined in regard to hole, so the "-" character indicates a backward direction from the point of view of the "@" placeholder.

brivla	short definition	x_1	x_2	x_3	x_4	x_5
`klama`	*x_1 goes to x_2 from x_3 via x_4 to x_5*	come	destination, place	origin, issue	route, road, infrastructure	means, vehicle, vehicle, thing, car
`vecnu`	*x_1 sells x_2 to x_3 for x_4*	sell, seller, company, issue	goods, sold	buyer, group, people, company	expense price, amount, cost, factor	-
`cteki`	*x_1 is a tax on x_2 levied against x_3 by authority x_4*	tax, levy, duty, cost	goods, service, event, taxed, taxable, fund	taxed	organization, people, collector, agency, authority	-
`cukta`	*x_1 is a book containing x_2 from x_3 for x_4 preserved in x_5*	book, item, material, document	work, program, project	author, professional, expert, people	audience, people, group, lot	medium, industry, sector

Table 4.4.: Automatically acquired *noun-types* for some dictionary entries (weights omitted)

$x_2 \rightarrow$ nn*(@, destination)*
$x_3 \rightarrow$ nn*(@, origin)*
$x_4 \rightarrow$ nn*(@, route)*
$x_5 \rightarrow$ nn*(@, means),* nn*(@, vehicle)*

2. **Chunking rules**
   For the remaining instances, we can use the following simple chunking rules operating on the modified POS tags (recall that arguments are tagged with *#ARG*).
   a) Type prefix: `(<N*>)+(<ARG>)`
   b) Simple IS-A pattern: `(<ARG>) is (an?|the) (<N*>+)`

Combining these results with the "oblique keywords" dictionary (See Section 2.2.1) containing even more type information, we can acquire noun-types for nearly all argument places. This result can further be expanded with IS-A expansion. Table 4.4 shows the resulting entries for some brivla using conservative thresholding.

4.2.5.3 Lojban-specific features

We can also consider features not suited to construct a similarity score. These features are only relevant for a later stage in which they support a machine learning approach. Such feature functions can either be based only on a target unit, $f(tu)$, or based on a frame – in this case a brivla, $f(b)$. For the first, a lot of features could be defined; they can be adapted from existing FSP systems, outlined in Section 1.3, and will thus not be elaborated. For brivla however, this is a novel area, so some examples are given.

1. $brivlaClass(b) \in \{\text{gismu}, \text{lujvo}, \text{fu'ivla}\}$
   the wordclass of b, based on a dictionary lookup
2. $nSumtiSlots(b) \in \{1,2,3,4,5\}$
   the number of slots defined for the brivla b
3. $nPosTags_T(b) \in \mathbb{N}$
   the number of pos tags T occurring in the definition string of b
4. $hasAgent(b) \in \{\text{true}, \text{false}\}$
   true if the definition string of b contains the hint "(agent)" after a sumti

The list given here is obviously far from exhaustive. In the evaluation (Section 4.3) it could be shown that the incorporation of such features could only minimally improve overall results. Ultimately, the amount of available data at this point is insufficient to motivate a data-driven approach which could make use of such features.

4.2.6 Target unit selection

For the TU selection step, only a set of empirically determined heuristics is used. It was also attempted to train a ML component for this task, however there was no gain over the baseline heuristics. Ultimately, a small number of binary features was most significant in the data, upon which the heuristics are based.

1. $isNotStopword(tu)$
   true for all tokens *not* covered by an English stopword list
2. $isContentWord(tu)$
   true for all tokens having a POS $\in \{NN, VB, RB, JJ\}$ in the *reduced* tagset
3. $isNotAux(tu)$
   true for all tokens which do *not* have an incoming aux(_,_) dependency edge
4. $isNotNPP(tu)$
   true for all tokens not having an "NNP*" POS tag in the *unreduced* tagset
5. $isHead(tu)$
   true for all tokens marked as a head word by the PST parser

These features are formulated in such a way that they yield *true* for tokens which should be selected as target units. The different heuristics we consider are then intersections of the subsets of tokens yielded by these filters. To briefly motivate these properties, the most important filter is the *isNotStopword* feature, as stopwords generally never invoke a brivla (which are content words). This selection is further refined by requiring a POS tag indicating a *content word* (which we simply consider to be verbs, nouns, adjectives and adverbs within the reduced tagset). This filter is to some extent redundant to the stopwords filter, but it does cover cases in which the stopword list is incomplete. Next we want to exclude all words which are used as *auxiliary verbs*. This includes all uses of the verb "*be*", but also verbs such as "*have*" in the construct such as "*I have got a car*". Auxiliary verbs are identified by the dependency parser (as *aux* dependency edges). Lastly, we do not want to extract proper nouns as target units, so we exclude tokens with an "*NPP*" POS tag (within the unreduced tagset). The following combinations of filters are considered as selection strategies, each refining the subset of potential target units further.

$h_{baseline}$: isNotStopword
h_1: isNotStopword $\land$ isNotAux
h_2: isNotStopword $\land$ isNotAux $\land$ isContentWord
h_3: isNotStopword $\land$ isNotAux $\land$ isContentWord $\land$ isNotNPP

Lastly, a useful selection strategy h_{head} simply selecting the *head words* of the sentence. This strategy, although not exhaustive, has a very high precision and often selects "the most important" TU within a sentence. For evaluation purposes, we also consider an *oracle* selection of target units, which is obtained from the word-alignment of parallel corpora (as described in Section 4.2.2).

4.2.7 Brivla matching

The brivla matching component of the system should yield a ranked list of possible brivla b given a sentence s and target unit tu. We have already motivated that we do this by computing a similarity score between (s, tu) and $def(b)$, the definition string of b. Therefore we presuppose

$$P(b \mid s, tu) \approx P(b) \cdot sim(def(b), (s, tu))$$

where *sim* is a similarity score based on lexical and structural features. The similarity measures are equivalent to those used for alignment (introduced in Section 3.4.1), and are merely extended with more feature extraction functions, such as dependency edges as defined above. On an implementational level, each similarity measure thus serves a dual purpose. They are both *retrievers*, yielding a set of possible brivla candidates, as well as rankers, giving each brivla a similarity score. This ranked list is capped at a fixed number.

To combine multiple features, for each feature extraction function f_k, $k \in 1, \ldots, n$ we define a unique similarity measure sim_k and model the final system as a linear combination using weights λ_k

$$P(b \mid s, tu) \approx P(b) \cdot \sum_{k=1}^{n} \lambda_k sim_k(def(b), (s, tu))$$

similarity measure	description
LemmaToGloss	overlap between lemma and gloss words
LemmaToKeywords	overlap between lemma and keywords
LemmaToGiza	unigram alignment probabilities obtained from parallel corpora (held-out)
DepEdges	overlap between dependency edges

Table 4.5.: Basic similarity measure sources (omitting parameters)

In a simple linear model, these weights λ_i can be determined through regression. It is also possible to use each similarity measure sim_k, $k \in 1, \ldots, n$ for a given matching pair $(b, (s, tu))$ as a feature vector of dimension n, for which the gold data provides *binary* feedback. In this setting, any ML model can be applied to this task as long as it yields a probability distribution over the binary class. This probability is then used as the ranking score. Different ML models are elaborated and evaluated for this component in Section 4.3.

Table 4.5 shows the source data from which similarity measures are obtained. When permuting all possible parameters (use of POS tags, lexical expansion, etc.), the effective number of possible similarity measures is in fact much larger than those given here.

4.2.8 Argument parsing

We have outlined that argument parsing is subdivided into two parts, (a) *candidate selection*, and (b) *slot assignment*. Step (a) yields a subset among all constituents of the sentence. Strictly speaking, this step is not necessary as the subsequent component could just as well consider *all* constituents of a sentence for assignment. However, this filter was added for simplification reasons. On a technical level, many constituents of a phrase-structure tree have an equivalent token span. For example the parent node of each *NNP* is a *NP* node. Step (a) only selects the top-level nodes among these ones which are equivalent with respect to the covered tokens, in this case the *NP* node and not the *NNP* node. Further simplifications can be made, such as removing the top-level sentence node *ROOT* and eliminating all spans in which the target unit appears. Thus, the candidate selection can be considered a simple “sanity check” of possible arguments[15].

Slot assignment therefore performs the primary work of argument parsing. When given a brivla b with its respective argument slots S_b and a set of argument candidates C within a sentence, it yields an argument parse $A \subset S_b \times C$. In Lojban, each slot *slot* $\in S_b$ and each argument $arg \in C$ can only be assigned once[16]. Therefore an important constraint to the assignment A can be added.

$$\forall (s_1, a_1) \in A. \forall (s_2, a_2) \in A. a_1 = a_2 \iff s_1 = s_2$$

Although such a constraint would impede a traditional ML-setting, in the similarity-based approach enforcing this constraint is trivial. Using one or more similarity measures $sim : S_b \times C \to \mathbb{R}$ a similarity matrix is computed, containing a score for each assignment pair. Based on this matrix, the *LimitedSelect*$(n{=}1, m{=}1)$ algorithm (shown in Appendix B.3) can be used to obtain an “alignment” between arguments and constituents, in which each element of either side can only be assigned once.

The main idea how these similarity scores are obtained has been almost fully described by the set of features that is used. Firstly, *holed dependency edges* can be used for this purpose. A constituent with an incoming edge *nsubj(go,@)* would thus have an overlap with an argument slot *nsubj(go,*x_1*)*. This overlap can further be computed on just the (directed) labels and (directed) targets of the edge, in this case *-nsubj* and *-go*. Another kind of similarity measure is based on *noun types* described in Section 4.2.5.2. Here, the spanned tokens are either lexically expanded (or ISA-expanded) and a weighted overlap score is computed. This similarity measure is for instance useful in the example sentence *“I go to university by bus”*. Here, “*bus*” is not covered by an appropriate dependency edge in the dictionary, but instead is ISA-expanded to “*vehicle*” and thus correctly assigned to the x_5-argument of the `klama` predicate.

Similar to brivla matching, a linear combination of multiple such measures can be computed. A linear regression model can be trained by computing multiple similarity measures for each cell in the assignment matrix. To illustrate

[15] It was also tried to train a classifier for argument candidate selection, based on their syntactic features. However this did not prove effective with the given training data.

[16] This is a direct application of the *theta criterion*, explained in Section 1.2.2, and applied to Lojban in Section 2.5

	x_1	x_2	x_3
NP: "You"	**1.65**	1.0	0
VP: "can help them to know that feeling"	0	0.06	0.16
MD: "can"	0.5	0	0
VP: "help them to know that feeling"	0	0.06	0.16
S: "them to know that feeling"	0.5	0.06	0
NP: "them"	0.71	**1.0**	0
VP: "to know that feeling"	0	0	0.16
TO: "to"	0	0	0
VP: "know that feeling"	0.5	0.06	0.16
VB: "know"	0.5	0.09	0
NP: "that feeling"	0	0	1.13
DT: "that"	0	0	0
NN: "feeling"	0	0	**1.17**

Output of one similarity measure sim_k

$\xrightarrow{\text{train } \lambda_k}$

	x_1	x_2	x_3
NP: "You"	1.0		
VP: "can help them to know that feeling"			
MD: "can"			
VP: "help them to know that feeling"			
S: "them to know that feeling"			
NP: "them"		1.0	
VP: "to know that feeling"			
TO: "to"			
VP: "know that feeling"			1.0
VB: "know"			
NP: "that feeling"			
DT: "that"			
NN: "feeling"			

Gold data

Figure 4.7: Example of assignment matrices for the sentence "*You can help them to know that feeling*", given the predicate `sidju` (x_1 *helps* x_2 *achieve* x_3). On the left, a single similarity measure is shown. On the right, the gold data is shown, illustrating the correct assignment.

this, Figure 4.7 shows an exemplary assignment matrix yielded by a single similarity measure, as well as a gold matrix containing the correct alignment. In this basic model, we only train a single weight λ_k for a given similarity measure sim_k. In theory, it would be possible to condition these weights, for instance on the argument slot or any feature extracted from the target unit or predicate. However, the training data available is vastly insufficient to justify such an approach. The bold numbers in the left matrix of Figure 4.7 also illustrate the result of the greedy assignment algorithm, which would assign the arguments in the order x_1, x_3, x_2, each time eliminating the respective row and column.

It was also attempted to use argument priors in analogy to priors for predicates. For this, the monolingual Lojban corpus was parsed, and the count of each argument instantiation was obtained. From these counts, probabilities $P(slot \mid b)$ for each predicate b were obtained using basic *add-one* smoothing. However, the use of these priors did not affect overall performance.

4.3 Evaluation

For evaluating the system, two different sources of gold data will be considered

1. English text manually annotated with LSP labels
2. Parallel corpora transformed to gold data as described in 4.2.2

Actual LSP annotations are the preferable alternative, because the evaluation does not get distorted by errors in the alignment process. However, much fewer annotated data is available, so the parallel corpora will serve as the primary gold data. The evaluation dataset consists of 5 books, amounting to over 35000 target units (brivla), and a much smaller set of LSP-labeled data amounting to only 100 annotated sentences[17].

For most configurations of the system we do not have to consider splitting the data into test and training sets, as the similarity scores are obtained only from the ontology itself (the Lojban dictionary). Special care has to be taken however when using similarity measures obtained through the parallel corpora itself; namely the GIZA unigram alignment probabilities. Here, we must not use the same data used for evaluation, as obtaining these probabilities can be considered "training"; thus separate training data is required. Furthermore, we cannot perform a randomized percentage split to separate training data. The reason for this is that within a single book corpus, constructs are often used repeatedly (e.g. in the text `little_prince`, the word "little" occurs 260 times). In order not to skew the results, we therefore only test configurations making use of GIZA-scores with an held-out test set

[17] For the annotated data, text from the fulltext corpus collection of FrameNet was chosen (parts of the *American National Corpus*), to further generate some data for comparison.

corpus	alice_in_wonderland			little_prince			wizard_of_oz			snow_white			die_verwandlung		
heuristic	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1
h_{head}	0.11	0.51	0.18	0.20	0.57	0.30	0.12	0.62	0.20	0.15	0.67	0.25	0.08	0.66	0.15
h_{baseline}	1.00	0.53	0.69	1.00	0.51	0.68	1.00	0.59	0.75	1.00	0.67	0.80	1.00	0.61	0.76
h_1	1.00	0.54	0.70	0.99	0.52	0.68	0.98	0.60	0.75	0.99	0.67	0.80	0.98	0.62	0.76
h_2	1.00	0.60	0.75	0.95	0.60	0.74	0.96	0.64	0.77	0.96	0.71	0.82	0.95	0.64	0.76
h_3	1.00	0.67	0.80	0.95	0.61	0.74	0.96	0.73	0.83	0.96	0.72	0.83	0.95	0.67	0.78

Table 4.6.: Evaluation of different heuristics for TU selection, according to the word-align oracle

of one complete book (`alice_in_wonderland`). This book was completely removed from the input of the GIZA alignment and should contain a realistic portion of unseen frames.

Further complications emerge when attempting to train the weights of feature sets including a corpus-based similarity measure. To make sure the weights are not learned in favor of GIZA-measures, this training has to be performed on yet a separate training set, distinct from both the data used to obtain the unigram table and the final data used for evaluation. As this is no longer feasible with such a small amount of data, for all ML-based settings of the system we refrain from using corpus-based (GIZA) features at all.

In the following, we will evaluate the components of “brivla matching” and “argument parsing” in isolation, as an error in the first component would propagate to the second component and cause very bad performance measures, even if the annotation is still reasonable. This is because there is usually a large number of possible brivla which are equally correct. When performing a full-system evaluation, the gold data does not account for the true performance of the system, as only *exact matches* are rewarded.

Target unit selection

The heuristics for TU selection are briefly evaluated according to the word alignment gold. It should be noted that in the gold data, all pairs aligning to a *stopword* have been removed, which means that h_{baseline} always has a perfect recall. The heuristics h_1, h_2, and h_3 then refine this set by removing unlikely TU candidates. Table 4.6 shows these results. Nearly all recall measures are still very satisfactory ($\geqslant 0.95$ for any corpus), whereas the amount of incorrectly retrieved target units decreases for the stricter heuristics. h_3 performs best among the evaluated options. The resulting average F_1 score of 0.79 still causes a notable drawback in overall performance. However, the quality of TU selection could not be improved even when training a dedicated ML component for this task. The remaining issues are caused mainly for *modal verbs* like *can*, *will*, *have*, *want*, *should*, etc. Some of these can be directly translated as Lojban predicates, such as *want* → `djica` (x_1 *wants (event/state)* x_2), in which case they should be selected as TUs, whereas others such as *will* are expressed by special cmavo and are therefore not part of the LSP task. Lastly, there is a set of words which may or may not be expressed with a predicate in Lojban depending on its sense. An example is the word *can*. In its meaning as *being allowed to*, it has to be translated as a brivla (`selcru`), whereas the meaning of *being able to* is primarily translated as a cmavo and is therefore not necessarily a TU. Solving these special cases has not been attempted. In the following evaluations, the best heuristic h_3 is used unless otherwise stated.

Brivla matching

For evaluating the performance of different similarity measures, we will consider two different interpretations of the result. One is a *binary* interpretation considering only the first frame yielded by the matcher, whereas the other considers the *ranked retrieval*. The reason for this is that in a complete LSP system, only the final frame is important ($P@1$), whereas the use of retrieval scores as a new feature space warrants a more accurate assessment of how well the feature performs (beyond the first result). Thus, the performance measures P (precision) and R (recall) are defined on the *first* frame of the ranked retrieval, with respect to the gold predicate.

In other settings, such as the brivla search engine “*brivlasis*” (see Section 4.4), and for assessing the overall quality of a single similarity measure, it is more adequate to interpret the result as a ranked retrieval. Therefore, we can apply standard information retrieval (IR) performance measures, such as the *mean average precision* (MAP)

$$\text{MAP} = \frac{\sum_{tu \in TU} AveP(tu)}{|TU|}$$

Here, TU is the set of target units, and $AveP$ is the average precision of the retrieval, defined as

$$AveP(tu) = \frac{\sum_{k=1}^{n} (P(k) \cdot rel(k))}{\text{number of correct brivla}}$$

corpus	similarity measure	$R_{\max}$	R	P	F_1	MRR
`alice_in_wonderland`	LemmaToKeywords	0.52	0.32	0.43	0.37	0.44
	LemmaToGiza (train)	0.50	0.34	0.48	0.4	0.44
	LemmaToGiza (full) *	0.80	0.45	0.54	0.49	0.64
	DepEdge	0.17	0.11	0.32	0.17	0.15
`little_prince`	LemmaToKeywords	0.59	0.41	0.5	0.45	0.51
	LemmaToGiza (train) *	0.84	0.56	0.64	0.60	0.71
	LemmaToGiza (full) *	0.84	0.52	0.60	0.55	0.69
	DepEdge	0.20	0.15	0.4	0.22	0.19
`wizard_of_oz`	LemmaToKeywords	0.67	0.43	0.54	0.48	0.56
	LemmaToGiza (train) *	0.88	0.6	0.68	0.64	0.75
	LemmaToGiza (full) *	0.88	0.58	0.67	0.62	0.75
	DepEdge	0.23	0.16	0.39	0.22	0.20
manual annotations	LemmaToKeywords	0.61	0.5	0.52	0.51	0.54
	LemmaToGiza (train)	0.58	0.51	0.63	0.56	0.54
	LemmaToGiza (full)	0.57	0.51	0.57	0.54	0.54
	DepEdge	0.45	0.38	0.51	0.44	0.41

Table 4.7.: Performance of different similarity measures with default parameters (use of frequency priors, no POS, no lexical expansion). The measures marked with * depict similarity measures that were obtained using the evaluation corpus, and thus have to be considered evaluation on the "training" set.

where $P(k)$ is the precision among the first k entries; $rel(k)$ equals 1 if the item at rank k is correct, and 0 otherwise. If we assume that there is only one correct brivla for each target unit (as it is the case in our evaluation setting), the MAP would be equivalent to a similar measure, the *mean reciprocal rank* (MRR)

$$\mathrm{MRR} = \frac{1}{|TU|} \sum_{tu \in TU} \frac{1}{rank(tu)}$$

The function $rank(tu)$ indicates the index of the correct retrieval (and ∞ if none is present). The MRR is a very useful metric for this task, as it measures how "far up" in the ranking the correct brivla appears. Even if the highest-ranking result does not match the gold brivla, a high MRR score still indicates an agreement with the gold data. As a last metric, we may want to disregard the ranking entirely, and only count for how many instances the correct brivla was retrieved at all. For a retriever with a capping c we thus define

$$R_{\max} = \frac{|\{tu \in TU \mid rank(tu) < c\}|}{|TU|}$$

In the following, we will first evaluate the different similarity measures in isolation. Afterward a set of well-performing measures is chosen as a feature space, and weights will be trained to obtain a combined result using various ML models. As the first evaluation step, we can assess the quality of different basic similarity measures across multiple corpora, as shown in Table 4.7. For comparison, also the cases which have to be considered "evaluation on the training data" are included (marked with * and in gray). As to be expected, there is a notable difference in these cases where the unigram alignment table has been trained on the evaluated corpus. On the held-out test corpus (`alice_in_wonderland`) and the held-out manual annotations this similarity measure performs on par with the *LemmaToKeywords* measure, but with generally higher precision. The *DepEdge* measure is much worse than the other measures based on lexical items when used in isolation. Next, we will evaluate some parameters for these measures, by permuting the following alternatives:

source $\in \{keywords, gizatrain, giza\}$
useFrequencyPriors $\in \{true, false\}$
usePOS $\in \{true, false\}$
useExpandedKeywords $\in \{true, false\}$
expandTU $\in \{true, false\}$[18]
simMeasure $\in \{DotProduct, CosineSimilarity\}$
capping $\in \{5, 10, 20\}$

[18] Thresholded to 20 expansions per TU.

These parameters were each evaluated with the following TU selection strategies

$$\text{selection} \in \{h_{baseline}, h_{head}, h_3, \textit{oracle targets}\}$$

resulting in 512 different configurations. These setups have each been run on 500 random different sentences from the `alice_in_wonderland` corpus, and the 100 manual annotation sentences. To assess the performance of each option, the scores have been aggregated over all other setting permutations and a mean of each measure has been produced. Table 4.8 shows these results. It can be observed that some parameters do not influence the results much. For different retrieval cappings, we have to regard $R_{\max}$ to assess if correct retrievals are found at lower ranks. However, only a very weak improvement in recall can be observed, indicating that only the very top retrievals are useful. Using *cosine similarity* over an unnormalized *dot product* seems to have no effect.

Interestingly, the use of POS tags makes the results worse on average. A possible explanation for this is the construction of words in Lojban, which often have a dual purpose and act both as verbs and nouns. For example, the brivla `gunta` can function both as *attack#VB* and *attack#NN*, yet the definition string only accounts for the verb (x_1 *attacks#VB* x_2). Without POS tags, this brivla is still retrieved for a target unit *attack#NN*, whereas the use of POS tags makes this retrieval impossible. Nevertheless, opposed to the statistical results, POS tags are useful in many cases as they are able to disambiguate *some* homonyms. When considering the two simple sentences "*I fell down yesterday*" and "*I'm waiting for fall*", the POS tagged retriever already suffices to distinguish the brivla for *fall#VB* (`farlu`) from the one for *fall#NN* (`critu`), whereas this distinction is impossible for the pure lemma-based retrievers. The intent to include POS-tag-based retrievers in the final system is therefore that an ML-model (such as a decision tree) could learn to prioritize such sparser similarity measures, and fall back to a lemma-based approach otherwise.

A further surprising result is that the use of lexical expansion – both on the TU side and brivla side – also harms performance. Even if $R_{\max}$ is slightly increased with expansion, the final result did not improve. The expansion of the target unit seems to perform better than the static expansion of keywords. This seems plausible for very infrequent target units. The lexical expansion of such infrequent items would still include more frequent words which potentially overlap with keywords, whereas in the other direction these rare words are not covered.

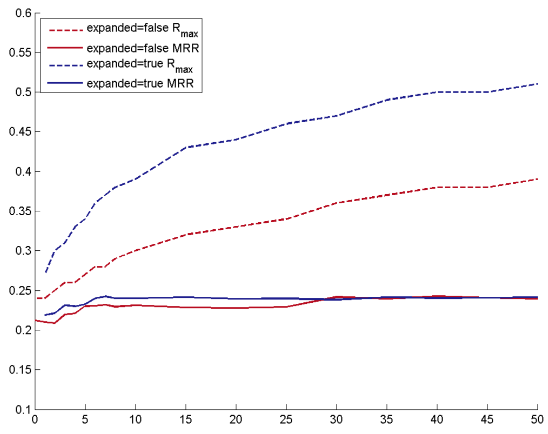
When regarding the performance of different *selection* strategies, it becomes obvious that this step is a crucial part of the system. The use of an *oracle target* yielded almost a 3-fold improvement in overall performance over the baseline approach. This can be considered an upper bound for improvement in extraction strategies. Clearly, many of the target units extracted by the baseline and the best selection heuristic h_3 still contain "difficult" TUs, as they dramatically hurt the precision of the system. By using the h_{head} selection strategy, a lot of this loss can be mitigated, as head words are much "easier" target units. However, it also forfeits all other potential TUs, and has thus a much weaker recall.

When regarding the source parameter, measures using the GIZA alignment table outperform the keyword approach by far. Obviously the unigram probabilities obtained on all corpora perform best, but even the *gizatrain* alignment (which has not been exposed to the corpora in this evaluation) works better than simple keywords. To give an example of the performance of a good configuration, a *gizatrain*-based similarity using frequency priors, no POS tags, and no expansion, already has a performance of $F_1 = 0.56$ ($R = 0.51$, $P = 0.63$) for oracle targets. As an important intermediate result, a vast improvement can be seen when using frequency priors.

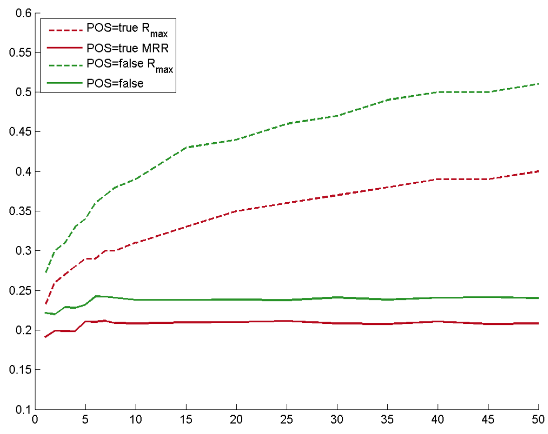
We can further evaluate lexical expansion in isolation. It was established that expanding the target unit (*expandTU*=true) seems to work better than using a static expansion of keywords (*expandedKeywords*=true), so it is worthwhile to see how these two parameters interact. For this, we can consider a parameter n which sets the maximum number of expansions per target unit. Figure 4.8 shows the maximum recall $R_{\max}$ and the resulting mean reciprocal rank *MRR* for various parameters. Note that in this evaluation the number of retrievals is constant, so for each configuration at most *capping*=50 brivla are retrieved. The behavior of $R_{\max}$ shows that lexical expansion clearly increases the number of relevant items that are retrieved. The slope of this increase is largest for a low number of expansions ($n < 10$) and decays for larger n. To evaluate the effect of lexical expansion on the ranking, we have to consider the *MRR*. For nearly all settings, the *MRR* stays the same. This seems reasonable, as the lexically expanded items have weights multiplied with a similarity < 1, so they do not obtain a high rank. Still, the score even increases minimally for small n. An important observation is thus that using lexical expansion for retrieval does not harm overall system performance, and may sometimes yield better results than an unexpanded system. When regarding the effect of using expanded keywords (shown in Subfigure 4.8a) it is obvious that the combination of TU expansion with expanded keywords works best. The use of POS tags however is not beneficial (shown in Subfigure 4.8b). Here, the version not using POS tags clearly works better.

parameter	value	mean R_{max}	mean R	mean P	mean F_1	mean MRR
selection	h_{baseline}	0.21	0.32	0.14	0.18	0.14
	h_{head}	0.38	0.16	0.23	0.18	**0.27**
	h_3	0.23	0.32	0.15	0.18	0.16
	oracle targets	0.56	0.30	0.35	0.32	**0.40**
source	keywords	0.33	0.25	0.18	0.18	0.22
	giza (train)	0.33	0.37	0.34	0.31	**0.29**
	giza (full)	0.44	0.43	0.36	0.35	**0.36**
useFreqPriors	false	0.31	0.19	0.16	0.15	0.19
	true	0.38	0.37	0.28	0.28	**0.29**
usePOS	false	0.35	0.26	0.19	0.19	0.23
	true	0.32	0.23	0.18	0.18	0.21
expandTU	false	0.31	0.26	0.20	0.20	0.22
	true	0.36	0.23	0.17	0.17	0.22
useExpandedKeywords	false	0.33	0.26	0.20	0.20	0.23
	true	0.34	0.23	0.17	0.17	0.21
similarityMeasure	dot product	0.33	0.24	0.18	0.18	0.22
	cosine similarity	0.34	0.25	0.19	0.19	0.22
capping	5	0.31	0.28	0.22	0.21	0.24
	10	0.35	0.28	0.22	0.21	0.24
	20	0.37	0.28	0.22	0.21	0.24
corpus	alice_in_wonderland	0.37	0.20	0.22	0.20	0.25
	ANC annotations	0.32	0.36	0.21	0.23	0.23

Table 4.8.: Evaluation of parameters for lexical similarity features



(a) effect of maximum expansions n for *expandedKeywords* $\in \{\text{true}, \text{false}\}$



(b) effect of maximum expansions n for *usePOS* $\in \{\text{true}, \text{false}\}$

Figure 4.8: Evaluation of lexical expansion for varying maximum expansions

type	feature	domain
similarity measures	*LemmaToGloss*(usePOS=false, expandedKeywords=false, expandTU=false)	$\mathbb{R}$
	LemmaToKeywords(usePOS=false, expandedKeywords=false, expandTU=false)	$\mathbb{R}$
	LemmaToKeywords(usePOS=true, expandedKeywords=false, expandTU=false)	$\mathbb{R}$
	LemmaToKeywords(usePOS=false, expandedKeywords=true expandTU=15)	$\mathbb{R}$
	HoledDepEdges(type=lemma, expandEdge=false, expandTargets=false)	$\mathbb{R}$
TU features	*IsHeadword*	$\mathbb{B}$
	IsNonstopword	$\mathbb{B}$
	IsNotAux	$\mathbb{B}$
	POS	$\mathbb{B}$
	HasForm (true if the lemma is different from the word form)	$\mathbb{B}$
	PST_top (label of the highest PST node covering the TU)	{S, NP, PP,..} (83)
	PST_bottom (label of lowest the PST node covering the TU)	{S, NP, PP,..} (83)
	NumOutEdges	$\mathbb{N}$
Frame features	*BrivlaWordClass*	{gismu, lujvo, fu'ivla}
	BrivlaSlotCount	$\mathbb{N}$
	IsAgentiveBrivla	$\mathbb{B}$

Table 4.9.: Feature-set yielding the best results on the held-out test corpora. Global settings for all similarity measures are (useFrequencyPriors=true, capping=5, similarityMeasure=CosineSimilarity).

(held-out) corpus	manual annotations (100 sent.)				`alice_in_wonderland` (1500 sent.)			
retrieval	$R_{max} = 0.71$				$R_{max} = 0.84$			
ranking	R	P	F_1	MRR	R	P	F_1	MRR
baseline (unweighted sum)	0.36	0.41	0.38	0.47	0.19	0.23	0.21	0.36
linear regression	0.06	0.07	0.07	0.19	0.07	0.09	0.08	0.22
maximum entropy log-linear	0.11	0.12	0.11	0.25	0.09	0.11	0.10	0.25
SVM (RBF kernel)	0.21	0.24	0.23	0.32	0.11	0.13	0.12	0.28
logistic regression	0.38	0.43	**0.40**	**0.49**	0.35	0.42	**0.38**	**0.53**
random forest	0.41	0.47	**0.44**	**0.52**	0.45	0.54	**0.49**	**0.63**

Table 4.10.: Final results for "brivla-matching" on unseen test data. All matchers use the same feature set, and only differ in ranking.

Feature combination

Based on these preliminary results on feature performance, we can now select a subset of similarity measures to be used as a new feature space, analogously to the work done for alignment in Chapter 3. In addition to similarity measures we can also add frame-level and TU-level features. For the sake of brevity, the evaluation of feature subsets is omitted here, and only the best performing feature set is reported. The feature set that was used is shown in Table 4.9.

As the similarity measures within this feature set effectively *retrieve* a set of possible brivla, the *retrieval* and the maximum possible recall R_{max} is the same for all ML strategies, and they only differ in ranking. Again, the measures R, P, and F_1 refer to only the *first* retrieval in the ranked result, and thus evaluate the system in a conventional FSP setting. The *MRR* considers the overall quality of the ranking. As a baseline for the system we can consider an unweighted sum of all similarity measures (a linear model with all weights $\lambda_i = 1$). This baseline already outperforms each of its components used in isolation. Various ML models[19] will now be evaluated in place of this baseline. The results are shown in Table 4.10. Training the weights λ_i with linear regression appears not to work with the given training data and decreases performance notably. The same is true for a maximum entropy log-linear model and a SVM trained on this binary data. The reason for this is most likely the use of negative examples. As only the exact brivla used in a translation is labeled as correct and all others as incorrect – even if they are perfectly valid outputs – the resulting model becomes extremely overfitted to these arbitrary data points.

Logistic regression as well as ensemble learners such as random forests appear to work well for this purpose. These models yield a notable improvement both in the F_1 score for binary evaluation, as well as in *MRR* over the

[19] For maximum entropy log-linear models, the *OpenNLP* MaxEnt implementation was used. For support vector machines, *LibSVM* was used. For the remaining classifiers, the *WEKA* implementation was used.

id	similarity measure	R	P	F_1	*EditDistAcc*
e	full dependency edges	0.27	0.65	0.38	0.29
l_1	dependency label	0.44	0.53	0.48	0.41
l_2	directed dependency label	0.44	0.61	0.51	0.43
l_3	incoming dependency label	0.36	0.52	0.43	0.39
t_1	dependency target	0.10	0.17	0.12	0.11
t_2	directed dependency target	0.09	0.17	0.12	0.11
t_3	directed incoming dependency target	0.09	0.15	0.11	0.10
nt_0	noun types	0.10	0.16	0.12	0.12
nt_1	noun types (prob. mass threshold = 0.8)	0.08	0.21	0.12	0.15

Table 4.11.: Performance of single similarity measures for argument parsing

baseline approach. For the (unseen) `alice_in_wonderland` corpus, the score has improved to $F_1 = 0.49$ over a baseline of $F_1 = 0.21$, which is evidence that the learned model generalizes well. For the manual annotations the improvement is less striking but performance is still improved. Random forests and boosting approaches with decision stumps (which yield near-equivalent results), clearly work best. These models effectively build a *hierarchy* among the similarity measures using various *thresholds*.

Argument parsing

As outlined earlier, argument parsing will be evaluated in isolation. The automatically acquired gold data is of very low quality on the argument level (due to incorrect word-level alignments), and thus only the manual annotations are used for evaluation. To adapt the performance measures for argument parsing, we can define P and R in a very conservative way. An argument is counted as a *true positive* only if it overlaps *exactly* with the gold argument assignment. However, this does not take into consideration the cases where an argument is assigned *nearly* correct. Consider for instance the earlier example illustrated in Figure 4.7. Here, we could make two near-equivalent assignments

(x_3, *VP: "to know that feeling"*)
(x_3, *VP: "know that feeling"*)

However, only the latter one is marked correct in the gold data. To get a more accurate assessment of the alignment quality, a simple measure is defined in addition to the exact precision and recall, which relies on the *edit distance* between the substituted strings. For a given argument assignment $\left(slot_n, span_{\text{output}}\right)$ with the gold assignment $\left(slot_n, span_{\text{gold}}\right)$, we define the *edit distance accuracy* as

$$EditDistAcc := \left(\frac{editDistance\left(span_{\text{output}}, span_{\text{gold}}\right)}{\max\left(length\left(span_{\text{output}}\right), length\left(span_{\text{gold}}\right)\right)} \right)$$

where *editDistance* is the Levenshtein distance. This metric has a number of useful properties. For each argument slot, it yields a value between 0 and 1. Exact assignments ($span_{\text{output}} = span_{\text{gold}}$) are ranked as 1, missing assignments as 0. In addition to this, partial assignments are also given credit with respect to the overlapping substring. For example, the *EditDistAcc* of the example (“*to know that feeling*”, “*know that feeling*”) yields a value of 0.85. This score is then averaged over all argument assignments and is given in addition to the conventional metrics P, R, and F_1.

In Table 4.11 the different similarity measures are defined and evaluated. Complete dependency edges of the form *nsubj(go,@)* have the highest precision, but lack in recall. Considering only the label of the dependency edge (or its target) improves recall at the cost of precision. The similarity measure l_2, directed dependency labels, works best by itself with an F_1 score of 0.51. Table 4.12 shows unweighted combinations of these measures. In general, summing multiple measures achieves a higher performance than each of their components in isolation (with the exception of an unmatched F_1 for the l_2-measure). The best combination ($e + l_1 + l_2 + nt_0$) is then further evaluated in Table 4.13, in which the effect of priors is shown, as well as the effect of training weights via linear regression. In fact, linear regression mildly decreases performance. The reason for this is probably the very small amount of training data on which these weights could be tuned. Priors also have almost no impact on the performance, and yield a near-negligible increase in both precision and recall.

unweighted combination	*R*	*P*	F_1	*EditDistAcc*
$t_1 + t_2 + t_3$	0.09	0.17	0.12	0.11
$e_1 + l_1 + t_1$	0.45	0.45	0.45	0.41
$l_1 + l_2 + l_3$	0.41	0.48	0.45	0.40
$e + nt_0$	0.35	0.37	0.36	0.34
$e + l_1 + l_2$	0.44	0.54	0.48	0.42
$e + l_1 + l_2 + nt_0$	0.46	0.49	**0.48**	**0.43**
all	0.51	0.40	0.45	0.43

Table 4.12.: Performance of unweighted combinations of similarity measures

weights	**usePriors**	*R*	*P*	F_1	*EditDistAcc*
unweighted sum	*false*	0.46	0.49	0.48	0.43
unweighted sum	*true*	0.47	**0.50**	**0.48**	**0.44**
linear regression	*false*	0.47	0.43	0.45	0.42
linear regression	*true*	0.47	0.44	0.45	0.43

Table 4.13.: Performance of argument parsers combining multiple similarity measures

Argument parsing has an uncompetitive performance if compared to traditional SRL systems. However, the system at hand can not truly be compared to these related tasks, as the given approach does not rely on any information other than the dictionary definitions of brivla.

In many cases, the argument parser correctly assigns "simple" one-token arguments, such as subjects and direct objects. It fails mostly when the correct argument of a brivla consists of a longer span, or when the whole sentence is long. In its current state, the system cannot account for arguments which are "far away" from the governing head word, and treat each constituent equally, even if they occur, for example, within an unrelated prepositional phrase. Figure 4.9 shows the performance of the system for a subcorpus containing only sentences to a maximum length n. The system clearly performs much better for shorter sentences (starting from $F_1 \approx 0.75$ for $n = 7$), and drops to the average performance for sentences of length $n = 20$. This is partially attributed to the dependency parser which has an increasing error rate for longer sentences. However, some of this error likely originates from the lack of features considering the global sentence structure, and instead using only a "flat" representation of dependency edges.

Error analysis

It should be noted again that the evaluation performed here can only serve for relative comparisons. It cannot offer an absolute assessment of the system performance. The reason for this has already been discussed: The author of a Lojban translation has made a particular choice of words, which at best constitutes *one possible option* with respect to the LSP task. In the following analysis of error classes, we will therefore make a top-level distinction between errors caused by incorrect evaluation and true errors caused either in the TU selection step or in the retrieval and ranking step. For this analysis, 150 sentences were randomly selected from all corpora, and each incorrect item was assigned one error class (in the priority of the list given here). Errors are only analyzed by their surface realization, and are not broken down to underlying components (such as errors in the POS tagger or dependency parsers).

1. **evaluation errors** (56%)

   a) **incorrect sentence alignment** (5.3%)
   incorrect items resulting from a completely or partially misaligned sentence.

   b) **incorrect word alignment** (24.6%)
   incorrect items resulting from a wrong word alignment. This is the second largest error class, illustrating the impact of bad alignment quality to the evaluation

   c) **choice of words** (26%)
   this class generically refers to any retrieval for which the translated corpus uses a different brivla, but both of them can be considered "correct" with respect to the definition of the LSP task. Example:

   Written by Antoine de Saint-Exupéry.
   `finti fa la'o me.  Antoine de Saint-Exupéry .me`

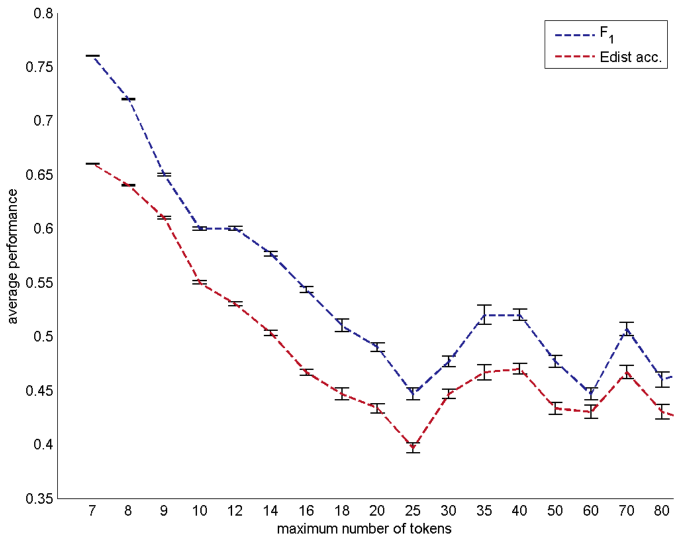


Figure 4.9: Argument parsing performance for limited sentence length. For each maximum number of tokens n, 5 random samples of 50 sentences with less than n tokens were selected from the manual annotation corpus

Here, the item *write#VB* is in fact translated as `finti` (x_1 *invents/creates/composes/authors* x_2), whereas the system yields the actual predicate for *write#VB*, `ciska` (x_1 *writes* x_2). An even more convoluted example is the sentence pair

I said nothing.

`mi smaji`

Here, the translation does not use a predicate for *say#VB*, but `smaji` (x_1 *is quiet / silent*). This freedom in translation accounts for many items counted as incorrect, while they can, however, be considered correct (and are in some cases even "better" than the translation by subjective standards). As expected, this is the largest error class.

2. **TU selection errors** (20 %)

   a) **missing target units in English** (4%)
   in some cases, a target unit is missing which could evoke a predicate. This is the generally the case when a stopword carries meaning. Example:

      On one star [...] there was a little prince to be comforted.

   Here, the word *be#VB* should in fact invoke `zvati` (x_1 *is at present at* x_2)[20].

   b) **multiword expressions** (6.6%)
   the current system cannot account for multiword targets. Example:

      I like my misfortunes to be taken seriously.

   Here, the target unit should be "*take_seriously*", instead of two target units *take#VB*, and *seriously*#RB. In many cases, multiword expressions do not only cause superfluous target units (such as *take#VB* in this case), but are often essential for the meaning. For example *get_in_touch*, has a distinct meaning neither related to *get#VB* nor *touch#NN* and is therefore regarded as a unique lexical item, which the system currently does not incorporate.

   c) **superfluous target units** (9.3%)
   this class covers instances in which the retrieved target unit does not map to a predicate. This could be fixed expressions, such as "*Good morning*", which incorrectly generate target units. Also included are modal verbs, which are not realized as predicates in Lojban, or words such as *never*, *always*, *again*, *enough,* etc. which are translated with function words rather than predicates.

3. **retrieval and ranking errors** (24%)

[20] This special case may be resolvable through the use of the multiword expression "*there_be*"

a) **no lexical connection** (6%)
this class covers all cases in which no lexical connection between the target unit and the correct predicate exists, and the brivla was not retrieved. For example etiquette#NN (despite the use of lexical expansion) could not be mapped to `tarti` (x_1 *behaves/conducts oneself as/in-manner* x_2).

b) **homonymy** (6.6%)
this class covers cases of truly homonymous lexical items. Some of these cases are in fact resolved by the use of POS tags and dependency edges, whereas some remain unresolved. Example:

But it was not because she had a cold
cold#NN → `lenku`: (x_1 *is cold*)

Here, the correct retrieval would be `bilma` (x_1 *is ill / sick/ diseased with symptoms* x_2 *from disease* x_3). In fact, this is already the second retrieval item, but it is not rated sufficiently high because of a missing lexical overlap with the target unit (the definition does not contain "*cold*").

c) **figurative meaning** (2%)
some verbs are used in a non-literal meaning. This figurative speech cannot be adapted to Lojban. Example:

And he sank into a reverie , which lasted a long time.
sink#VB → `derse'a` (s_1 *sinks / embeds* s_2 *in ground* s_3)

Here, the retrieval (and in fact all other possible brivla for *sink#VB*) is incorrect because the verb is used figuratively.

d) **wrong predicate more salient** (7.3%)
this error class covers all remaining cases in which the correct predicate could be retrieved, but another predicate was more salient. This class contains many special cases, such as an (uncommon[21]) distinction between nouns and verbs. In that case *answer#NN* → `danfu` (x_1 *is the answer to* x_2), and *answer#VB* → `spuda` (x_1 *answers* x_2) are incorrectly assigned. In this case, `danfu` always outweighs `spuda` as the first only uses the keyword *answer#NN* and the latter is associated with many more lexical items (*respond*#VB, *reply*#VB, *response*#NN, *object*#NN, *event*#NN, *situation*#NN). Another example can be seen in the following sentence.

And the little prince, completely abashed, went to look for a sprinkling-can of fresh water.

Here the target unit *completely#RB* gets assigned to `culno` (x_1 *is full/completely filled with* x_2) instead of `mulno` (x_1 *is complete/done/finished*) probably because the word *completely#RB* appears in its exact form only in the first predicate.

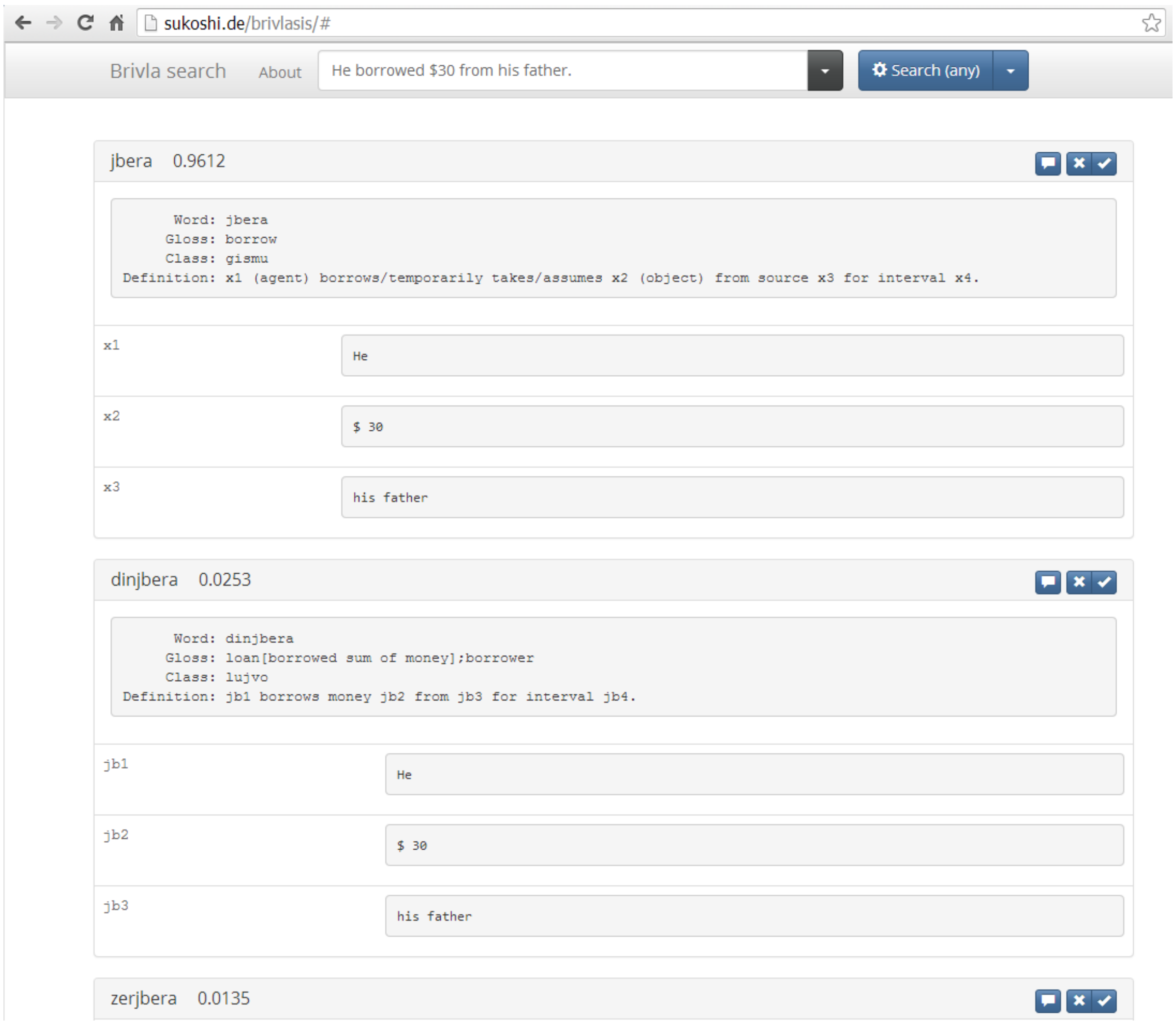
A further subclass is related to inaccurate brivla priors. Currently, only the monolingual chatlog corpus is used for obtaining frequency counts. Although this corpus has the best coverage, it is obviously skewed. Most of the discussion in the Lojban chat is done on the subject of the language itself. For this reason, Lojban-related vocabulary has an unproportionally high probability (`valsi`, "*word*", is actually the most frequent predicate), which is why the LSP results are generally skewed towards such terms.

e) **not in dictionary** (2%)
in some cases, the gold brivla are not present in the dictionary, and are thus impossible to retrieve. As an example, *misfortunes#NN* is translated as `malfunca`, constructed from `mal` (*derogative connotation*) + `funca` (x_1 *is determined by luck of* x_2). Whereas human speakers can understand this word without a dictionary entry, the system obviously cannot account for such cases.

In summary, a great amount of the errors reported by the system (56%) are in fact caused by the approximated evaluation[22]. A lot of actual errors originate from incorrect TU selection, which could be tackled relatively easily. The truly "difficult" error classes, such as homonymy or figurative speech, only account for a small fraction of errors. Reducing these would probably involve unproportionally high effort.

[21] Lojban attempts to use the same predicate for the verb form as well as the noun form of the English equivalent, but in some cases there are actually two distinct brivla.

[22] This does not mean that 56% of the error in the performance measure can be subtracted; the evaluation errors just obscure the true error class distribution.



jbera 0.9612

```
      Word: jbera
     Gloss: borrow
     Class: gismu
Definition: x1 (agent) borrows/temporarily takes/assumes x2 (object) from source x3 for interval x4.
```

```
x1    He
x2    $ 30
x3    his father
```

dinjbera 0.0253

```
      Word: dinjbera
     Gloss: loan[borrowed sum of money];borrower
     Class: lujvo
Definition: jb1 borrows money jb2 from jb3 for interval jb4.
```

```
jb1    He
jb2    $ 30
jb3    his father
```

zerjbera 0.0135

Figure 4.10: The *brivlasis* webinterface. The input of the system is an English sentence, the output is a ranking of different Lojban predicates matching the sentence, including the respective argument assignments.

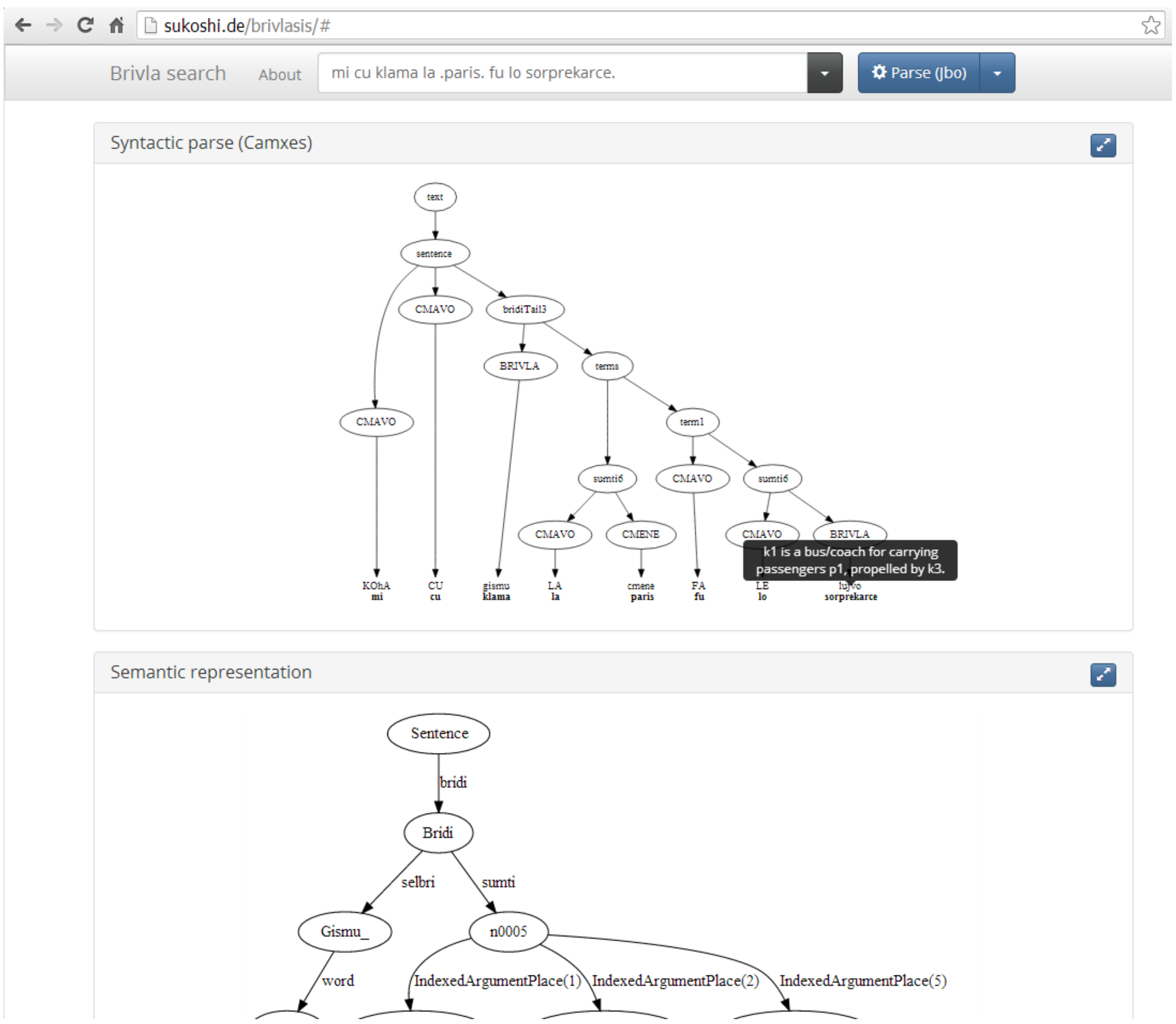
4.4 Use case: Brivlasis

Although the LSP system implemented here does not reach high scores with respect to the gold annotations, it still provides a good output of possible Lojban predicates when given an English source sentence. In fact, when only using the system as a dictionary search, it often provides a much better result than the dictionary search engines currently available for Lojban. This use case has motivated the application of LSP as a *predicate search engine*.

This service was named "*brivlasis*", a newly coined Lojban word from the gismu "`bridi valsi sisku`" – "predicate word search". It is provided in form of a REST[23] API, as well as a web user interface. A screenshot of the website is shown in Figure 4.10. Here, the user can provide either a single word or a complete sentence, and get a ranked list of possible Lojban predicates matching the sentence and its arguments.

As an additional feature, the system also supports looking up Lojban words as well as obtaining the semantic parse representation of a Lojban sentence. In Figure 4.11 it is shown how the website displays the syntactic parse of a Lojban sentence, along with the semantic transformation as described in Section 4.1.1. The website has serviced requests from 43 different hosts over the course of two months. Considering that the Lojban community consists of fewer than 200 members, this is a relatively large fraction.

[23] *Representational State Transfer*, an architecture for stateless web-interfaces.


sukoshi.de/brivlasis/#

mi cu klama la .paris. fu lo sorprekarce.

k1 is a bus/coach for carrying passengers p1, propelled by k3.

Figure 4.11: The brivlasis webinterface used to look up the semantic parse representation of a Lojban sentence.

5 Conclusions and future work

5.1 Conclusions

The main focus of this work lies on the artificially constructed language Lojban. Instead of merely treating it as an amended version of natural language, we have regarded it in the context of semantic ontologies and computational generation of meaning representations. Throughout this thesis we have asked the question if Lojban can be used as a semantic resource, and have substantiated this claim with practical applications comparable to those of Frame Semantic Parsing.

To recapitulate the contributions of this work, each chapter is briefly summarized. In Chapter 2 we have introduced Lojban from an analytic point of view by first performing a linguistic analysis showing its properties as a language and subsequently doing a qualitative comparison to a semantic ontology. The observation of notable similarities has lead to the concrete definition of an alignment task to FrameNet in Chapter 3. From there, a statistical approach for this alignment task has been devised and an automatic alignment of Lojban predicates to FrameNet lexical units has been obtained. A total of 1149 alignment pairs have been generated, which were estimated to have a precision of roughly 75%. Finally, in Chapter 4 we have considered semantic parsing applications pertaining to Lojban. Based on the possibility to unambiguously parse Lojban, we have used the automatic alignment to create FrameNet annotations for Lojban prose. Finally, we have considered Lojban as a semantic parsing resource, assuming the role of FrameNet. Analogously to FSP, the dictionary of Lojban is used as an inventory of senses to create predicate-argument annotations of English text. This task was coined *Lojban Semantic Parsing* (LSP). It was shown how parallel Lojban-English corpora can be exploited to perform a quantitative evaluation of LSP systems in the absence of actual annotated data. Ultimately, a baseline Lojban semantic parser was implemented. As a demonstration of one possible use case, a search engine for Lojban predicates has been provided as a webservice to the Lojban community.

As a conclusion of this work, it was demonstrated that Lojban has a multitude of potential applications in NLP, that go beyond its use as a linguistic toy language. Even if its number of speakers is neglectable in comparison to spoken languages, it can still be considered competitive when regarded as a semantic resource. In fact, it was shown that a translation of natural language text to Lojban implicitly creates semantic annotations. Therefore it must be asked if it has more “speakers” than – for example – FrameNet. This bold statement illustrates a fundamental point; it may be easier for human annotators to use a full-fledged language to formalize semantics, rather than a semantic ontology which has to be learned just as well[1]. Another merit of Lojban is that the dictionary is not based on English, but instead predicates are defined for a whole array of languages (for a subset of the language, more than 60). As a resource, Lojban implicitly encodes a vast amount of intra- and interlinguistic information.

Compared to conventional ontologies Lojban is also much more *minimalistic*. The complete dictionary of core predicates is only 118 KB in size. Yet it has been reused by hundreds of speakers for nearly three decades. A set of root words, combined with a trivial formalism for defining predicates, makes it possible for new predicates to be created in a manner of seconds. This stands in contrast to the highly complicated linguistic task of creating an inventory of semantic frames, which is an effort that is still far from completion. Furthermore, the system implemented in this work is independent of the concrete dictionary in use. It could be replaced with any other set of predicates that is specified in single-sentence definitions, using predefined placeholder-markers as arguments. The implemented semantic parser relies exclusively on such definitions. Therefore, replacing the dictionary or adding new predicates to it works seamlessly, as the parser does not have to rely on instance-based training. It can therefore adapt to the evolution of the Lojban language without having to be adjusted.

Even when disregarding the lack of training data, LSP can be argued to be a much harder task than FrameNet-based FSP. The reason for this is that a semantic inventory such as FrameNet is tailored to natural languages. It captures figures of speech, idioms, and collocations used in a particular language (and therefore has to be constructed for each language anew). Lojban on the other hand attempts to disassociate from any particular natural language. The predicates defined in the Lojban dictionary are not based on the occurrence of predicates in

[1] It has been demonstrated that it is feasible for human speakers to achieve *fluency* in Lojban; it is unclear if the same is true for human FrameNet annotation.

natural language discourse. Instead, they have been defined out of the need to convey meaning while using Lojban itself for communication. Therefore, Lojban attempts to provide a language-independent, idiom-free and culturally neutral set of predicates, which have a single, well-defined meaning.

Lastly, and perhaps most importantly, Lojban can be used to formalize more than just predicate-argument structures. The set of function words defined in the language serve to encode any form of semantic content. To emphasize this point, semantic ontologies such as FrameNet or PropBank focus on capturing *frames* and their *elements* but are not designed to represent meaning on other linguistic levels, such as *sentence type, tense*, *modality*, *negation*, *quantification*, and so on. A different line of research such as AMRL (Banarescu et al., 2013) attempts to bridge this gap by providing a formal language on top of PropBank roles which encodes this semantic information. The design of such a formal language can be seen as a top-down approach, in which language features are added to approximate the full expressiveness of natural language text. In contrast to that, Lojban can be seen as a bottom-up approach. A small set of root predicates, and a minimal grammar were defined to construct a full language, which was actively used and iteratively refined when more expressiveness was needed. As a consequence, Lojban can be argued to exhibit all properties of a complete meaning representation language, although it has never been intended for this purpose. The task of *Lojban Semantic Parsing* is particularly interesting in regard to natural language understanding. Given a sufficiently advanced LSP system, the remaining steps toward a complete semantic annotation would directly integrate with language, and would be equivalent to a *translation* to Lojban. For simple sentences this translation is trivial, but to cover all remaining gaps, a notable effort is still necessary. This use case of LSP is briefly outlined as future work in Section 5.2.2.

5.2 Future work

This work primarily motivates the use of Lojban as a novel kind of semantic resource. Hence, many interesting directions for future work have emerged which could not be pursued in the scope of this thesis. In Section 5.2.1 we will first describe some subsequent steps immediately following up to the work done in this thesis to improve the performance of the developed system. Then, in Section 5.2.2, we will motivate some higher level tasks operating on the basis of a Lojban semantic parser and summarize some related directions that could be studied.

5.2.1 System improvements

Alignment

In order to improve the statistical alignment of Lojban to FrameNet, it is necessary to incorporate instance-based data. For the alignment task, the existence of parallel corpora has not been exploited. One possibility is to apply a Lojban semantic parser to the Lojban text and an FSP system to the English text. From this, co-occurrence counts of FrameNet frames and Lojban brivla could be obtained (these co-occurrences could further be extended to the lexical unit level, and the argument level). A probabilistic alignment of entities can be directly computed from these co-occurrences using an association measure such as *pointwise mutual information* (PMI), and an alignment could be performed based on these association measures (Doan et al., 2004).

The instance-based data could further be used to produce a *contextualized* mapping between Lojban and FrameNet. Given sufficient training data, it would be possible to condition the alignment on certain context features (such as those used by SRL systems). The resulting conditional mapping could potentially eliminate the remaining problems of ambiguity of the alignment-based semantic parser (see Section 4.2.3).

A further next step would be to extend the alignment process to the set of *all* Lojban brivla – and not just the core set of gismu. Here, it would be possible to make use of the *hierarchy* data in both ontologies. As each non-primitive brivla is created from one or more gismu, we can make use of the hyponomy relations which are explained in Section 2.1.2. In many cases, the alignment of a *parent* predicate also holds for specializations. For example the gismu `karce` ("*vehicle*") aligns to the *Vehicle* frame, which is also true for its specialization `sorprekarce` ("*bus*"). In other cases, the same hierarchical structure is mirrored in both ontologies. For example `sfacatra` ("*execute*") is a type-of `catra` ("*kill*"). The same hierarchy can be observed in FrameNet, in which the *Execution* frame inherits the *Killing* frame. This suggests that the use of graph-based methods (which were briefly covered in Section 3.2) may be useful for the full alignment task. However, the presence of such *graph isomorphisms* is very rare between Lojban and FrameNet, so it remains to be evaluated if this approach is beneficial to the alignment performance.

Lojban semantic parser

Similar to the alignment task, the LSP system implemented in this work does not rely on instance-based training. Training is only used to obtain weights or thresholds for the combination of different similarity measures. These trained weights are to a large extent independent of the data itself, so the resulting system is neither tailored to the effective lexical items in the source text, nor the predicates in its inventory. While this independence can be seen as the greatest merit of the system, it is probably also the main performance bottleneck. Probably, a major performance increase could be achieved by taking the information of instance-based data into consideration. For example, it would be statistically evident that predicates such as `cusku` ("say", in the sense of quoting direct speech), have an argument containing a quoted string (or a phrase structure tree node of a certain type). Most state of the art FSP systems rely primarily on such syntactic properties of the assigned arguments, so for an LSP system of the same quality such data can be considered obligatory. If sufficient instance-based data was available, the LSP parser could resort to the bulk of research in FSP feature engineering. Nevertheless, for the amount of available annotated training data, implementing such strategies would unlikely be worthwhile, so these improvements could only be considered in the presence of a parallel corpora much larger than the currently existing ones.

Assuming that only the currently used data is available, there is less room for system improvements. A notable improvement could probably be achieved by accounting for *multiword expressions*. In fact, the Lojban dictionary already contains entries such as "to get back" → `xrucpa`. However, for reasons of implementational effort, identifying these target units has not been performed. It would also be plausible to obtain undocumented multiword expressions from parallel corpora.

Another improvement could be achieved by computing more accurate predicate priors. As explained in Section 4.3, the monolingual Lojban corpus has the best coverage, but a slightly skewed word distribution. Other corpora, while having a more natural word distribution, are however much smaller and thus lack in coverage. A plausible step is applying a smoothing model which combines the frequency counts of multiple sources to obtain better frequency estimates.

To address errors related to *homonymous* words, it could be attempted to perform a more sophisticated type of lexical expansion. By using the lexical items within a sentence as the context, Biemann and Riedl (2013) show how the expansion step can be *contextualized*. In a sentence such as "*I caught a nasty cold*", the target unit "*cold*" would then be expanded only with illness-related words. This would lead to a higher overlap score for the illness-related brivla than the temperature-related brivla, and could reduce homonymy-based errors in the brivla matching step.

Improvement of the *argument parsing* component clearly has to rely on instance-based training data. Using only the definition strings from the dictionary is clearly insufficient to account for all possible instantiations of a predicate. On the argument-level, data sparsity is even more profound. One solution to overcome this extreme sparsity of training data is to learn a generalized model across all predicates, assigning the argument labels $x_1, \dots, x_n$. However, these argument labels are idiosyncratic to each predicate, so they do not generalize well. The same issue is faced by SRL systems for PropBank labels. Yi et al. (2007) suggest to overcome this problem by using a mapping to VerbNet thematic role labels. Instead of training a system on the generic labels $Arg_0 \dots Arg_n$, which could be shown to be very incoherent across frames, they assign a thematic role to each label for each verb. Their approach overcomes the sparsity in training data and was able to generalize better to unseen training data. It seems reasonable to apply the same strategy for Lojban, as the challenge is nearly equivalent.

Integration of FSP systems

In this work, an existing FSP system was employed to obtain LSP annotations through an alignment-based mapping algorithm (see Section 4.2.3). The resulting LSP output was only considered as an *alternative* to the parser implemented in this work. In fact, it would be plausible to integrate the two systems. This can be achieved by feeding the output of the distinct systems as features into a meta-system. This approach would very likely outperform the current baseline. However, this direction was not pursued for two reasons. Firstly, the performance of such a system would be attributed mostly to the underlying FSP system. Secondly, there is an enormous computational overhead caused by the use of state of the art FSP systems. The SEMAFOR pipeline requires 8 GB of system memory and is optimized for throughput rather than response time for single sentences. Thus, it would be unfeasible to employ such a component in the "*brivlasis*" webservice, which was designed to have fast response times.

5.2.2 Next steps and related directions

Translation to Lojban

One of the primary motivations for *Lojban Semantic Parsing* is the fact that the gap towards a complete translation to Lojban text is relatively small. When given the correct predicate and arguments, constructing a Lojban sentence is often trivial. In case of "simple" noun arguments, this translation is nearly complete. Consider again the example "*I'm going to university by bus*", with the LSP result

`klama` ("go")
$x_1 \leftarrow$ "*I*"
$x_2 \leftarrow$ "*university*"
$x_5 \leftarrow$ "*bus*"

From there, we can first substitute the original argument span into a valid Lojban sentence, and then translate each argument in isolation

("I") `klama` ("university") `fu` ("bus")

`(mi) klama (lo balcu'e) fu (lo sorprekarce)`

This results in a valid translation of the sentence. In case of complex arguments such as relative sentences it may be feasible to recursively translate these constituents by invoking the system only on the subexpression. Consider for instance the more complex sentence "*I want to go to university by bus*" with the LSP result

`djica` ("*want*")
$x_1 \leftarrow$ "*I*"
$x_2 \leftarrow$ "*go to university by bus*"

Here, it is often possible to just call the system recursively to translate complex constituents such as x_2

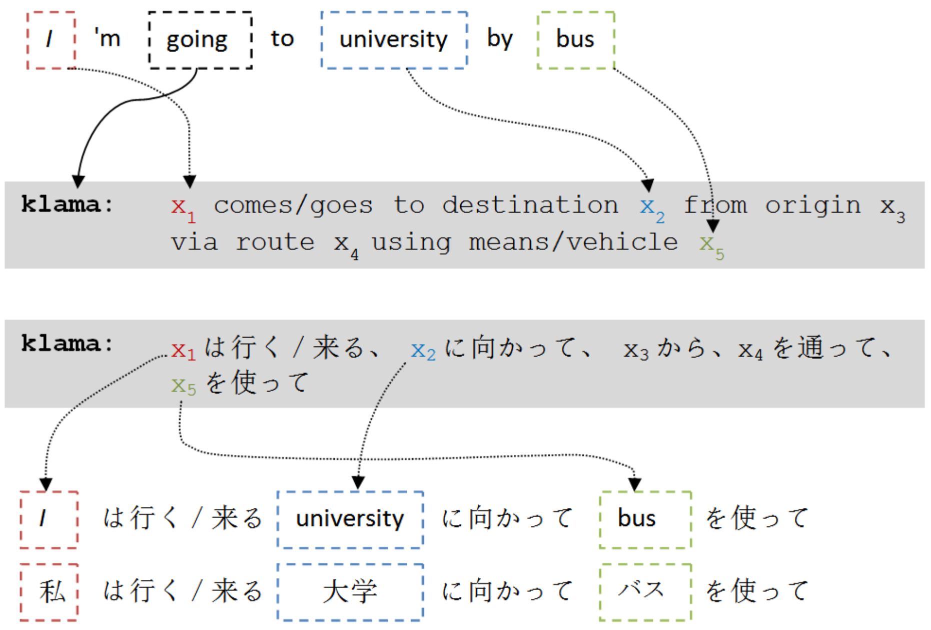

I
'm
going
to
university
by
bus
klama: x1 comes/goes to destination x2 from origin x3 via route x4 using means/vehicle x5
klama: x1 は行く / 来る、 x2 に向かって、 x3 から、x4 を通って、x5 を使って
I
は行く / 来る
university
に向かって
bus
を使って
私
は行く / 来る
大学
に向かって
バス
を使って

Figure 5.1: Illustration of a basic translation system, based on syntactic replacement. An approximated Japanese sentence is obtained

(“I”) `djica` (“go to university by bus”)

`(mi) djica (klama` (“university”) `fu` (“bus”))

`(mi) djica (klama (lo balcu'e) fu (lo sorprekarce))`

`mi djica lo nu klama lo balcu'e fu lo sorprekarce`

which again, is a valid translation to Lojban.

A further issue not addressed by this prototype is the dictionary lookup of primitive expressions. For simple atomic expressions, such as “I”, there exists only one possible translation. For others however, it may be necessary to *disambiguate* the source words in case multiple translations are possible.

Lastly, such a system could be extended to handle not only primitive statements, but also sentences types such as *yes/no-questions*, *investigative questions*, *requests* and so on. The finite and well-defined grammar of Lojban makes it possible to treat each linguistic concern in isolation and gradually refine this “translation” process. For example, at a later stage, the *tense* of the sentence could be identified, and this (optional) information can be added to the Lojban expression with a simple keyword.

Translation of natural languages

Lojban has been envisioned as an interlingua for machine translation. A very simplistic prototype for this was implemented, which should not be considered a serious attempt at MT, but instead can be considered a “toy translation system”, which is obtained almost for free. Here we make use of the fact that Lojban predicates have been defined in many languages. In each of these definitions, the same “slot notation” is used, in which a variable takes the place of the constituent assuming the respective semantic role. The prototype works as follows:

1. Perform a semantic parse of the sentence in the source language using the head word as target.
2. For the resulting predicate, look up the definition in the target language.
3. Substitute the argument constituents from the source sentence.
4. For each single argument token perform a dictionary lookup in the target language, and replace it if it was found.

Figure 5.1 shows an illustration of this system to translate the simple example sentence into Japanese. The resulting translation is of course not correct (for instance with respect to the word order – in Japanese, the main verb is usually at the end of a sentence). This prototype would have to address the same issues discussed above (the handling of complex constituents, and possible disambiguation of atomic expressions). Nevertheless, it would be interesting to see how far such a primitive translation system could be improved, or if this multilingual frame-semantic information in general can be useful in supporting existing MT systems.

Grammar induction of a Lojban grammar

In Sections 2.2.3 and 4.1.1 we have mentioned some of the issues pertaining to parsing Lojban text. Whereas Lojban is fully and unambiguously defined, the provided EBNF grammar is insufficient for some use cases, such as obtaining the predicate-argument representations needed in this work. Furthermore, as the language evolves, formal grammars need continuous curation and adaptation to the language. Despite Lojban's intended goal of being trivially parsable, obtaining a robust parser which yields a high-level representation of the language is an open problem. In this theses, a syntactic parse tree was converted to a semantic representation through implementing each grammar rule of the language one-by-one. However, implementing all these rules by hand is a formidable effort; furthermore, the robustness of the system is still dependent on the syntactic parser, which is based on a complex hand-crafted grammar.

An alternative to this approach is the automatic inference of a regular or context-free grammar, known as *grammar induction* (De la Higuera, 2010). In this field, the objective is to learn a formal grammar, given some information of a language, which can for instance be strings or parse trees. Using a semantic representation of Lojban text as labeled input data, it could be evaluated if a formal grammar that generates this tree structure can be discovered automatically.

Using Lojban for AMR annotations

It was already motivated that Lojban could function as an annotation language for natural language text. However, the task of assigning Lojban predicates to English text has proven to be even more difficult than traditional frame semantic parsing tasks – mainly because the latter are designed specifically for the annotation of English text, and Lojban on the other hand uses a unique set of predicates attempting to be independent of English or any other natural language.

In order to still make use of the advanced language features of Lojban, it may be interesting to replace the inventory of Lojban predicates with an existing semantic inventory. For this, PropBank is a prime candidate, as it also relies on well-defined *indexed argument slots*. Consider for example the sentence "*The boy didn't want to go home*". Instead of translating it to Lojban completely (`lo nanlu cu pu na djica lo nu ri klama fe lo zdani`), it would be plausible to replace the predicates with PropBank rolesets. Here, we simply assume *boy-n.01* and *home-n.01* to exist in NomBank, and determine the correct senses for "*want*" and "*go*" which are "*want-v.01*" and "*go-v.02*". The arguments are then substituted according to the role indexes of the respective rolesets. These are defined as

want-v.01	(sense: desire)
Arg_0	wanter
Arg_1	thing wanted
Arg_2	beneficiary
Arg_3	in-exchange-for
Arg_4	from

go-v.02	(sense: self-directed motion)
Arg_0	goer
Arg_1	journey
Arg_2	start point
Arg_3	end point

The complete statement could then be encoded as

`lo` *boy-n.01* `cu pu na` *want-v.01* `lo nu ri` *go-v.02* `fo lo` *home-n.01*

where the argument indices of the Lojban slots $x_1 \dots x_5$ are simply remapped to $Arg_0 \dots Arg_4$. The semantic content of this statement is very similar to an encoding of the sentence in AMRL, containing well-defined argument assignments, polarity, and tense.

```
(w / want-01
  :ARG0 (b / boy)
  :ARG1 (g / go-01
         :ARG0 b
         :ARG3 (h / home))
  :polarity -
)
```

It would be interesting to see if Lojban could be used in place of such a meaning representation language, of if the two representations are in fact equivalent and can be translated. Analogously of obtaining FrameNet annotations from parallel English-Lojban text (discussed in Section 4.3), it may be feasible to extract complete AMRL annotations in a similar manner.

Appendix

A Documentation of Lojbanlib

The code developed for this thesis has been integrated into a single project *Lojbanlib*[1], which is made publicly available. More comprehensive documentation can be found on the project page.

Architecture

Lojbanlib is written in *Scala*, and uses functional style for nearly all purposes. It is a Maven project organized into the following sub-projects

- `lojbanlib-core` Lojban-related core functionality, including Lojban parsers, Lojban utilities and data
- `lojbanlib-nlp` NLP-related functionality, including wrappers for existing tools
- `lojbanlib-lsp` Main components for Lojban Semantic Parsing, and related functionality
- `lojbanlib-tools` Standalone tools, such as the graphical annotator
- `lojbanlib-brivlasis` The "brivlasis" REST webservice

The library relies on only some configuration files in *YAML*[2] format to be present in the working directory, which specify remaining paths for the data.

- `settings.yaml` the primary settings, required by `lojbanlib-core`
- `annotator.yaml` settings required for running the annotator
- `brivlasis.yaml` settings required for running the "brivlasis" webservice

The general setup for running Lojbanlib is the following

1. A *JDBC*[3] database path has to specified in `settings.yaml` (by default, *sqlite* is used)
2. The "source" data, such as the dictionary files, word lists, parallel corpora, etc. have to be available at the paths specified in `settings.yaml`
3. The executable `lojbanlib.data.creators.CreateAll` has to be run, which processes the source data, builds models and stores this data it into the database (this process takes many hours)
4. Subsequent components (LSP, aligment, etc.) access the preprocessed data from the database (and serialized models) and do not require the "source" data

File formats

The Lojbanlib library generally makes use of two data representations, which can be converted back and forth

1. A *plaintext* format, into which data can be printed and parsed (with the intend to be human-readable)
2. Serialized Java objects (with the intend to improve performance by statically performing preprocessing steps)

For all plaintext files used by Lojbanlib, the encoding specified in `settings.yaml` is assumed (default: "*utf-8*"). The following plaintext formats are supported *PlaintextWKM* format, *PlaintextParallelCorpus* format, *PlaintextAnnotatedCorpus* format and *PlaintextAlignment* format. Other file formats which are supported are those of *hunalign*, *GIZA++*, and XML file formats specified by FrameNet, PropBank, WordNet, SEMAFOR, and the *ARFF* format used by WEKA.

[1] The project page is availabe at `https://github.com/hintz/lojbanlib.git`
[2] YAML: a human-readable data serialization standard: `http://www.yaml.org/`
[3] Java Database Connectivity (JDBC) provides universal access to any database: `http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/`

B Algorithms and procedures

B.1 Corpus preprocessing

Tokenization

Lojban has been designed to be processed easily. The tokenization is near-trivial, and for word-tokenization a simple whitespace tokenizer was sufficient in most cases. However, a set of tokenizers have been defined which were used depending on the corpus and use case.

- *WhitespaceTokenizer*: splitting at the regular expression `/\s+/`
- *LojbanWordTokenizer*: extracts all substrings matching `/[\w'.,]+/`, which is a more strict tokenization only allowing correctly spelled Lojban words.
- *CamxesWordTokenizer*: applies the camxes syntactic parser, and returns the literals of all leaves using *in-order* traversal. This tokenizer therefore only allows syntactically correct expressions to be tokenized. It further breaks up *cmavo clusters*, which are sets of cmavo words which are allowed to be written without any whitespace in between. For example, the string "`piremucu'o`" gets correctly tokenized as [`pi`, `re`, `mu`, `cu'o`] (which has the meaning "x_1 *has a probability of 0.25*", and may be used by a Lojban speaker to say something like "maybe"). It also removes pause tokens, so that "`la .alis.`" gets tokenized to [`la`, `alis`].
- *LojbanSentenceTokenizer*: splits at any token matching the regular expression `/\.i[\w']*/` or line breaks. This identifies the *cmavo* `.i` used to denote the beginning of a new sentence. In contrast to natural languages, this is not an approximation but always yields the correct sentence splitting.

Language identification

Although the Lojban corpora are already of high quality, there are some instances in which non-Lojban text is interlaced. The objective of the following preprocessing step is to filter out this noise. The following procedure was applied:

1. Tokenize the Lojban input, on word level and sentence level.
2. For each sentence s, label each token $t \in s$ as **D** if it is contained in the dictionary.
3. Use a sliding window of odd size n, and iterate over the n-grams in s. Annotate the middle token in the sliding window as **L** if the majority of n tokens in the window are annotated with **D**.
4. Trim all tokens that are not annotated as **L** from the beginning and end of s. If tokens in the middle of the sentence remain, which are not labeled as **L**, discard s, otherwise return the remaining non-empty sequence of Lojban tokens.

This algorithm thus identifies substring of Lojban if the majority of tokens in a sliding window are contained in the dictionary. As an important effect of this, any non-Lojban text at the beginning or end of a sentence gets removed. This is especially important for the `chatlog` corpus, because all lines begin with time-stamp and a user name, which is filtered out by this processing step.

corpus	sentences	alignment quality (base)	alignment quality (realigned)
`wizard_oz`	3213	0.4496	0.6432
`alice`	2464	0.2316	0.4567
`die_verwandlung`	2030	- 0.0062	0.1491
`little_prince`	1634	0.5700	0.8939
`snowwhite`	252	0.5377	0.6336

Table B.1.: Alignment quality of corpora as given by hunalign

B.2 Alignment of Lojban corpora

In order to obtain properly aligned parallel corpora, a number of existing tools was evaluated on the data. The final toolchain, yielding the highest result in alignment quality, is described in the following.

1. The English and Lojban corpora were manually cleaned of any formatting and special characters, so that they only contain equivalent plaintext.
2. For the book corpora, the boundaries between chapters were manually annotated as *anchor points*, a standard procedure to improve sentence alignment.
3. Word and sentence tokenization was performed for both sides.
4. For English, the tokenizer bundled with the Stanford toolchain (Klein and Manning, 2003) is used for word and sentence segmentation. As alignment tokens *untagged lemmas* were selected (using unlemmatized word forms or POS-tags seemed to impact the alignment result negatively).
5. For Lojban tokenization is trivial, as defined above (Section B.1).
6. The word / sentence tokenized input is fed into the sentence alignment tool *hunalign* (Varga et al., 2007). This tool is suited for most languages, and performs reasonably well for Lojban. It is intended for processing small corpora, and employs a multi-pass alignment strategy. First, it is supplied with a bilingual dictionary, translating Lojban to English. The alignment algorithm uses this information to bootstrap a first alignment, and falling back to a Gale-Church strategy (comparing sentence length) otherwise. Then, an improved automatic dictionary is built based on this first alignment, and a second pass is performed. Hunalign also reports a numeric alignment quality, which assumes the 1 for a perfect alignment, and a value < 1 for bad quality. These quality scores for the book corpora are shown in Table B.1. There seems to be no correlation between alignment quality and corpus length. The `die_verwandlung` corpus stands out as being unalignable; the reason for this may be that English was not the source language for translation (the original is German), or that the author translated the text very freely.
7. Given the sentence-aligned output of *hunalign*, the parallel corpora are then fed to the word-alignment tool *GIZA++*[1] (Och and Ney, 2003) to obtain a word-level alignment.
8. In the use case of using the parallel corpus as evaluation data for the LSP system, the aligned sentences were then converted back into their original form (including whitespace, casing, etc.).

[1] GIZA++ is available at `https://code.google.com/p/giza-pp/`, accessed June 2014.

B.3 Greedy limited alignment selection

Algorithm 1 Greedy algorithm for *LimitedAligner*(n, m)

```
function LimitedSelect($M : O_1 \times O_2 \to \mathbb{R}, n : \mathbb{N}, m : \mathbb{N}$)
    result $\leftarrow \{\}$
    count$O_1 \leftarrow \{\}$
    count$O_2 \leftarrow \{\}$
    $R \leftarrow$ sorted list of tuples $O_1 \times O_2$ from $M$
    while $R \neq \emptyset$ do
        $(e_1, e_2) \leftarrow$ HEAD$(R)$
        count$O_1(e_1) \leftarrow$ count$O_1(e_1) + 1$
        count$O_2(e_2) \leftarrow$ count$O_2(e_2) + 1$
        if count$O_1(e_1) \leq n \wedge$ count$O_2(e_2) \leq m$ then
            result $\leftarrow$ result $\cup \{(e_1, e_2)\}$
        end if
        $R \leftarrow R \setminus \{(e_1, e_2)\}$
    end while
    return result
end function
```

B.4 Inferring span ranges based on Language Models

Algorithm 2 Span-inferring algorithm for Lojban predicate definitions

The LM-based inferring of “slash spans”, discussed in Section 3.4.1 works as follows.

1. For a given definition string d, replace all argument slots “x_n” with the word “*this*”
2. Compute all possible groupings of “/” tokens and the resulting span permutation set S_d
3. For each span permutation $s \in S$, compute all expansions of d resulting in a set of expanded strings E_s
4. For each span permutation s, compute its mean perplexity mp, normalized by the length of the sentence

$$mp(s) = \frac{1}{|E_s|} \sum_{e \in E_s} \left(\frac{perplexity(e)}{length(e)} \right)$$

where $perplexity(e)$ is the perplexity score according to a language model.

1. For the definition string d, chose the span permutation $s_{best} := \arg\min_{s \in S_d} mp(s)$

B.5 Alignment-based parsers

Algorithm 3 Trivial FrameNet parser for Lojban prose

Select an alignment of brivla B to FrameNet frames F, and an alignment of brivla slots S to frame elements FE, to define mapping functions

$$map_{\text{FRAME}} : B \to F \cup \{\emptyset\} \qquad \text{and} \qquad map_{\text{FE}} : B \times F \times S \to FE \cup \{\emptyset\}$$

where $\emptyset$ denotes a missing alignment

```
function JboFrameNetParser($s$ : Sentence)
    $parsed \leftarrow$ semantic parse of $s$
    for each annotation with brivla $b$ and sumti arguments $A$ in $parsed$ do
        Generate the mapped frame
        $f \leftarrow map_{\text{FRAME}}(b)$
        if $f \neq \emptyset$ then
            Generate the subset of mappable frame elements
            $fe \leftarrow \left\{ map_{\text{FE}}(b, f, s) \mid s \in A \right\}$
            Annotate($s, f, fe$)
        end if
    end for
end function
```

Algorithm 4 SEMAFOR & Alignment-based semantic parser for Lojban

Select an alignment of lexical units LU to a set of brivla B, and an alignment of frame elements FE to brivla slots S to to define the mapping functions

$$map_{\text{LU}} : LU \to \mathscr{P}(B) \qquad \text{and} \qquad map_{\text{FE}} : LU \times B \times FE \to S \cup \{\emptyset\}$$

```
function SemaforBasedLSP($s$ : Sentence)
    $parsed \leftarrow$ frame semantic parse of $s$ using the SEMAFOR parser
    for each target unit $tu$ with frame $f$, and evoked frame elements $E$ in $parsed$ do
        Covert $tu$ into a lemma with FrameNet POS-tag
        $lu \leftarrow (poslemma(tu, s), f)$

        Obtain the set of possible brivla candidates
        $B_c \leftarrow map_{\text{LU}}(lu)$
        if $B_c = \emptyset$ then
            Fall back to frame-level alignment
            $B_c \leftarrow \bigcup_{l \in \text{Lemmas}} \left( map_{\text{LU}}(l, f) \right)$
        end if

        Select the best-matching brivla $b \in B_c$
        $b \leftarrow \arg\max_{b \in B_c} \left( \frac{2 \cdot |Aligned(b) \cap E|}{|Aligned(b)| + |E|} \right)$

        sumti $\leftarrow \{ map(lu, b, fe) \mid fe \in E \}$
        Annotate($s$, $(b, sumti)$)
    end for
end function
```

C Manual alignment annotations

Format

The manual annotations produced for the alignment task, are given in the following plaintext formats.

1. Frame + argument level

   ```
   [brivla][TAB][FrameName]
   [TAB][arguments]
   ```

   where `[arguments]` is a TAB-separated list of the format

   ```
   [slot name]=[frame element name]
   ```

   For unalignable slots, the right hand side of = is an empty string, whereas unalignable brivla are given in the format

   ```
   brivla [TAB] -
   ```

   Lines beginning with a "#" character are comments.

2. Frame + LU level

   ```
   [brivla][TAB][FrameName]
   [TAB][argument level]
   ```

   where `[argument level]` is a ";" separated list of FrameNet lexical units, for which the alignment holds.

Argument level annotations

```
# clite: x1 is polite/courteous/civil in matter x2 according to
      standard/custom x3.
clite Social_interaction_evaluation
  x1=Evaluee x2=Behavior x3=

# cadzu: x1 walks/strides/paces on surface x2 using limbs x3.
cadzu Self_motion
  x1=Self_mover x2=Area x3=

# basna: x1 emphasizes/accentuates/gives emphasis/stress/accent to x2 by
      (action) x3.
basna Convey_importance
  x1=Speaker x2=Message x3=Means

# ckafi: x1 is made of/contains/is a quantity of coffee from source/bean/grain
      x2.
ckafi Food
  x1=Food x2=

# cfari: x1 [state/event/process] commences/initiates/starts/begins to occur;
      (intransitive verb).
cfari Process_start
  x1=Process_start

# citsi: x1 is a season/is seasonal [cyclical interval], defined by
      interval/property x2, of year(s) x3.
# Note: difficult to assign. Argument level does not work, but Frame level is
      still correct
citsi Calendric_unit
  x1= x2= x3=

# cabna: x1 is current at/in the present of/during/concurrent/simultaneous
      with x2 in time.
# Note: FN has alternatives, hard to model
cabna Temporal_collocation
  x1=Trajector_entity x1=Trajector_event x1=Trajector_period x2=Landmark_entity
      x2=Landmark_event x2=Landmark_period

# blaci: x1 is a quantity of/is made of/contains glass of composition
      including x2.
blaci Substance
  x1=Substance x2=

# carmi: x1 is intense/bright/saturated/brilliant in property (ka) x2 as
      received/measured by observer x3.
# Note: Misaligned?
carmi Location_of_light
  x1=Figure x2=Manner x3=

# barja: x1 is a tavern/bar/pub serving x2 to audience/patrons x3.
barja Buildings
  x1=Building x2= x3=

# cladu: x1 is loud/noisy at observation point x2 by standard x3.
cladu Sound_level
  x1=Entity x2= x3=Degree

# cilta: x1 is a thread/filament/wire [shape/form] of material x2.
cilta Connectors
  x1=Connector x2=

# bitmu: x1 is a wall/fence separating x2 and x3 (unordered) of/in structure
      x4.
bitmu Architectural_part
  x1=Part x2= x3= x4=Whole

# bolci: x1 is a ball/sphere/orb/globe [shape/form] of material x2; x1 is a
      spherical object [made of x2].
# Note: This aligns the material argument for once.
bolci Shapes
  x1=Shape x2=Substance

# citka: x1 eats/ingests/consumes (transitive verb) x2.
citka Ingestion
  x1=Ingestor x2=Ingestibles

# badydi'u: b1 is a defensive building (castle, fortress etc.) for protection
      of b2 from b3 (enemies, danger).
badydi'u Buildings
  b1=Building b2= b3=

# backla: b1=k1 goes beyond destination b2=k2 from origin k3 via route k4
      using means/vehicle k5.
# Note: fix role aliases in lojban sumti extraction
backla Motion
  b1=Theme b2=Goal k3=Source k4=Path k5=Manner
```

```
# cfika: x1 is a work of fiction about plot/theme/subject x2/under convention
       x2 by author x3.
# Notes: all sumti assigned, but not with cores
cfika Text
  x1=Text x2=Topic  x3=Author

# certu: x1 is an expert/pro/has prowess in/is skilled at x2 (event/activity)
       by standard x3.
certu Expertise
  x1=Protagonist x2=Skill x2=Knowledge x3=Judge

# bredi: x1 is ready/prepared for x2 (event).
bredi Activity_ready_state
  x1=Protagonist x2=Activity

# bruna: x1 is brother of/fraternal to x2 by bond/tie/standard/parent(s) x3;
       [not necess. biological].
bruna Kinship
  x1=Ego x2=Alter x3=

# canre: x1 is a quantity of/contains/is made of sand/grit from source x2 of
       composition including x3.
canre Substance
  x1=Substance x2=Type x3=Constituents

# ba'urnoi: n1 is a spoken/uttered message about subject n2 uttered by n3 to
       intended audience n4.
ba'urnoi Text_creation
  n1=Text n2= n3=Author n4=Addressee

# bilma: x1 is ill/sick/diseased with symptoms x2 from disease x3.
bilma Medical_conditions
  x1=Patient x2=Symptom x3=Ailment

# ckeji: x1 feels ashamed/mortified/humiliated under conditions x2 before
       community/audience x3.
ckeji Emotion_directed
  x1=Experiencer x2=Circumstances x3=Expressor

# ckaji: x1 has/is characterized by property/feature/trait/aspect/dimension x2
       (ka); x2 is manifest in x1.
ckaji Distinctiveness
  x1=Entity x2=Feature

# ba'ostu: s1 is a nursery where b1 grows to size/form b2 from b3.
ba'ostu Building_subparts
  b1= b2= b3= s1=Building_part

# cisma: x1 smiles/grins (facial expression).
cisma Facial_expression
  x1=Possessor

# cedra: x1 is an era/epoch/age characterized by x2
       (event/property/interval/idea).
cedra Calendric_unit
  x1= x2=Salient_event

# carna: x1 turns about vector x2 towards direction x3, turning angular
       distance / to face point x4
carna Moving_in_place
  x1=Theme x2=Fixed_location x3=Direction x4=Angle

# ba'urdu'u: d1 whines/bitches about d2 by uttering b2.
ba'urdu'u Complaining
  d1=Complainer d2=Complaint d2=Topic b2=Complaint

# carvi: x1 rains/showers/[precipitates] to x2 from x3; x1 is precipitation
       [not limited to 'rain'].
carvi Precipitation
  x1=Precipitation x2=Place x3=Cause

# bagyce'a: c1 is a bow that shoots arrow c2 from string c3, and is made of
       material b3.
bagyce'a Weapon
  c1=Weapon c2= c3=Part b3=Material

# clira: x1 (event) is early by standard x2.
clira Relative_time
  x1=Focal_occasion x1=Focal_participant x2=Landmark_occasion

# bende: x1 is a crew/team/gang/squad/band of persons x2 directed/led by x3
       organized for purpose x4.
bende Aggregate
  x1=Aggregate x2=Individuals x3=Container_possessor x4=

# barda: x1 is big/large in property/dimension(s) x2 (ka) as compared with
       standard/norm x3.
barda Size
  x1=Entity x2= x3=Standard

# blabi: x1 is white/very-light colored [color adjective].
blabi Color
  x1=Color

# bajra: x1 runs on surface x2 using limbs x3 with gait x4.
bajra Self_motion
  x1=Self_mover x2=Area x3= x4=

# ba'armo'a: x1 is a pattern of marks x2 arranged according to structure x3 on
       x4 of material x5.
ba'armo'a Pattern
  x1=Pattern x2= x3= x4= x5=

# cfila: x1 (property - ka) is a flaw/fault/defect in x2 causing x3.
# Note: not the best frame
cfila Judgment
  x1= x2=Evaluee x3=Result

# bandu: x1 (event) defends/protects x2 (object/state) from
       threat/peril/potential x3 (event).
bandu Defend
  x1=Defender x2=Victim x3=Assailant

# canja: x1 exchanges/trades/barters commodity x2 for x3 with x4; x1, x4 is a
       trader/merchant/businessman.
# Note: nice example
canja Exchange
  x1=Exchanger_1 x2=Theme_1 x3=Theme_2 x4=Exchanger_2

# cinba: x1 (agent) kisses/busses x2 at locus x3.
# Note: Bodypart_of_agent is not correct for x3, sadly
cinba Manipulation
  x1=Agent x2=Entity x3=Locus

# ckire: x1 is grateful/thankful to/appreciative of x2 for x3 (event/property).
ckire Judgment_direct_address
  x1=Communicator x2=Addressee x2=Topic x3=Reason

# cizra: x1 is strange/weird/deviant/bizarre/odd to x2 in property x3 (ka).
cizra Idiosyncrasy
  x1=Entity x2= x3=Idiosyncrasy

# cikna: (adjective:) x1 is awake/alert/conscious.
cikna Being_awake
  x1=Protagonist

# clani: x1 is long in dimension/direction x2 (default longest dimension) by
       measurement standard x3.
clani Dimension
  x1=Object x2=Dimension x3=Measurement

# ba'urtadji: t1 is b1's pronunciation of utterance b2 under conditions t3.
ba'urtadji Spelling_and_pronouncing
  b1=Speaker b2=Sign t1=Part_of_form t3=Context

# bacru: x1 utters verbally/says/phonates/speaks [vocally makes sound] x2.
# Note: x2 is a sound, not a topic or a message!
bacru Statement
  x1=Speaker x2=

# bratu: x1 is hail/sleet/freezing rain/solid precipitation of
       material/composition including x2.
bratu Precipitation
  x1=Precipitation x2=

# cerni: x1 is a morning [dawn until after typical start-of-work for locale]
       of day x2 at location x3.
cerni Calendric_unit
  x1=Relative_time x2=Whole x3=

# bargu: x1 arches/curves over/around x2 and is made of x3; x1 is an arch
       over/around x2 of material x3.
bargu Shapes
  x1=Shape x2=Substance x3=

# cakla: x1 is made of/contains/is a quantity of chocolate/cocoa.
cakla Food
  x1=Food

# bunda: x1 is x2 (def. 1) local weight unit(s) [non-metric], standard x3,
       subunits [e.g. ounces] x4.
bunda Measure_mass
  x1=Stuff x2=Count x3=Unit x4=

# cabra: x1 is apparatus/mechanism/device/equipment for function x2
       controlled/[triggered] by x3 (agent).
cabra Gizmo
  x1=Gizmo x2=Use x3=User

# bradi: x1 is an enemy/opponent/adversary/foe of x2 in struggle x3.
bradi Taking_sides
  x1= x2=Cognizer x2=Side x3=Issue

# claxu: x1 is without/lacking/free of/lacks x2; x1 is x2-less.
claxu Possession
  x1=Owner x2=Possession

# betri: x1 is a tragedy/disaster/tragic for x2.
betri Catastrophe
  x1=Undesirable_Event x2=Undergoer
```

```
# bancu: x1 exceeds/is beyond limit/boundary x2 from x3 in property/amount x4
       (ka/ni).
bancu Surpassing
 x1=Profiled_item x2=Attribute x3=Standard_item x4=Extent

# citri: x1 is a history of x2 according to x3 (person)/from point-of-view x3.
citri History
 x1=Topic x2=Domain x3=

# bacycripu: c1 is a bridge to the beyond across c2 between c3 and c4.
bacycripu Roadways
 c1=Roadway c2=Path c3=Source c4=Goal

# bevri: x1 carries/hauls/bears/transports cargo x2 to x3 from x4 over path
       x5; x1 is a carrier/[porter].
bevri Carry_goods
 x1=Distributor x2=Goods x3=Place x4= x5=

# ciksi: x1 (person) explains x2 (event/state/property) to x3 with explanation
       x4 (du'u).
ciksi Statement
 x1=Speaker x2=Topic x3=Addressee x4=

# bunre: x1 is brown/tan [color adjective].
bunre Color
 x1=Color

# cipra: x1 (process/event) is a test for/proof of property/state x2 in
       subject x3 (individ./set/mass).
cipra Operational_testing
 x1= x2=Tested_property x3=Product

# benji: x1 transfers/sends/transmits x2 to receiver x3 from
       transmitter/origin x4 via means/medium x5.
benji Transfer
 x1=Donor x2=Theme x3=Recipient x4= x5=Means

# ba'argau: x1 (agent) marks x3 with mark(s) x2 of material x4.
ba'argau Sign
 x1=Indicator x2= x3=Indicated x4=

# birje: x1 is made of/contains/is a amount of beer/ale/brew brewed from x2.
birje Food
 x1=Food x2=Constituent_parts

# cigla: x1 is a/the gland [body-part] secreting x2, of body x3; x2 is a
       secretion of x1.
cigla Observable_body_parts
 x1=Body_part x2= x3=Possessor

# berti: x1 is to the north/northern side [right-hand-rule pole] of x2
       according to frame of reference x3.
berti Part_orientational
 x1=Part x2=Whole x3=

# bartu: x1 is on the outside of x2; x1 is exterior to x2.
bartu Part_inner_outer
 x1=Part x2=Whole

# cilmo: x1 is moist/wet/damp with liquid x2.
# Note: easy
cilmo Being_wet
 x1=Item x2=Liquid

# betfu: x1 is a/the abdomen/belly/lower trunk [body-part] of x2; [metaphor:
       midsection].
betfu Observable_body_parts
 x1=Body_part x2=Possessor

# birti: x1 is certain/sure/positive/convinced that x2 is true.
birti Certainty
 x1=Cognizer x2=Content x2=Topic

# boxfo: x1 is a sheet/foil/blanket [2-dimensional shape/form flexible in 3
       dimensions] of material x2.
boxfo Shapes
 x1=Shape x2=Substance

# bukpu: x1 is an amount of cloth/fabric of type/material x2.
bukpu Clothing
 x1=Garment x2=Material

# boxna: x1 is a wave [periodic pattern] in medium x2, wave-form x3,
       wave-length x4, frequency x5.
boxna Moving_in_place
 x1=Theme x2= x3=Path_shape x4= x5=Periodicity

# catlu: x1 looks at/examines/views/inspects/regards/watches/gazes at x2.
catlu Perception_active
 x1=Perceiver_agentive x2=Phenomenon

# ba'urxausku: b1 (agent) eloquently speaks/verbally expresses x1
       (sedu'u/text/lu'e concept) for audience c3, good/beneficial/nice for x2
       by standard x3.
ba'urxausku Expressing_publicly
 b1=Communicator c3=Addressee x1=Content x2=Manner x3=

# citno: x1 is young/youthful [relatively short in elapsed duration] by
       standard x2.
citno Age
 x1=Entity x2=Circumstances

# cinla: x1 is thin in direction/dimension x2 by standard x3; [relatively
       short in smallest dimension].
cinla Body_description_holistic
 x1=Individual x2= x3=Degree

# ba'orzu'e: z1 grows b1 for purpose/goal z3 to size/form b2 from b3.
# Note: Nice example
ba'orzu'e Cause_expansion
 z1=Agent b1=Item z3=Purpose b2=Result_size b3=Initial_size

# cinri: x1 (abstraction) interests/is interesting to x2; x2 is interested in
       x1.
cinri Mental_stimulus_stimulus_focus
 x1=Stimulus x2=Experiencer

# bersa: x1 is a son of mother/father/parents x2 [not necessarily biological].
bersa Kinship
 x1=Alter x2=Ego

# carce: x1 is a cart/carriage/wagon [wheeled vehicle] for carrying x2,
       propelled by x3.
carce Vehicle
 x1=Vehicle x2= x3=Means_of_propulsion

# briju: x1 is an office/bureau/work-place of worker x2 at location x3.
briju Building_subparts
 x1=Building_part x2= x3=Place_holder

# canko: x1 is a window/portal/opening [portal] in wall/building/structure x2.
canko Connecting_architecture
 x1=Part x2=Whole

# cidni: x1 is a/the knee/elbow/knuckle [hinged joint, body-part] of limb x2
       of body x3.
# Notes: here, Subregion can be aligned, for other examples, it can't.
cidni Observable_body_parts
 x1=Body_part x2=Subregion x3=Possessor

# cimni: x1 is infinite/unending/eternal in property/dimension x2, to degree
       x3 (quantity)/of type x3.
cimni Duration_attribute
 x1=Eventuality x2= x3=Degree

# ckule: x1 is school/institute/academy at x2 teaching subject(s) x3 to
       audien./commun. x4 operated by x5.
ckule Locale_by_use
 x1=Locale x2=Relative_location x3= x4= x5=Container_possessor

# ciska: x1 inscribes/writes x2 on display/storage medium x3 with writing
       implement x4; x1 is a scribe.
ciska Text_creation
 x1=Author x2=Text x3=Manner x4=Instrument

# bacyde'i: d1 is a tusk of d2 protruding beyond the mouth b3 by amount b4
bacyde'i Observable_body_parts
 d1=Body_part d2=Possessor b3=Attachment b4=

# cirla: x1 is a quantity of/contains cheese/curd from source x2.
cirla Food
 x1=Food x2=Constituent_parts

# bilni: x1 is military/regimented/is strongly organized/prepared by system x2
       for purpose x3.
bilni Military
 x1=Force x2=Possessor x3=Goal

# banli: x1 is great/grand in property x2 (ka) by standard x3.
banli Desirability
 x1=Evaluee x2=Parameter x3=Comparison_set

# ckasu: x1 ridicules/mocks/scoffs at x2 about x3 (property/event) by doing
       activity x4 (event).
ckasu Judgment_communication
 x1=Communicator x2=Evaluee x3=Topic x4=Expressor

# cfine: x1 is a wedge [shape/form/tool] of material x2.
cfine Shapes
 x1=Shape x2=Substance

# badri: x1 is sad/depressed/dejected/[unhappy/feels sorrow/grief] about x2
       (abstraction).
badri Emotion_directed
 x1=Experiencer x2=Topic x2=Event

# cecmu: x1 is a community/colony of organisms x2.
cecmu Aggregate
 x1=Aggregate x2=Individuals
```

```
# cinmo: x1 feels emotion x2 (ka) about x3.
cinmo Feeling
  x1=Experiencer x2=Emotion x3=Cause

# bajli'a: b1 runs away from c2 via route c3 on surface b2 using limbs b3 with
       gait b4.
bajli'a Fleeing
  b1=Self_mover c2=Source c3=Path b2= b3= b4=

# ba'usku: s1 (agent) says s2 (sedu'u/text/lu'e concept) for audience s3
       through expressive medium s4.
ba'usku Statement
  s1=Speaker s2=Message s3=Addressee s4=Means

# calku: x1 is a shell/husk [hard, protective covering] around x2 composed of
       x3.
calku Part_inner_outer
  x1=Part x2=Whole x3=

# casnu: x1(s) (mass normally, but 1 individual/jo'u possible)
       discuss(es)/talk(s) about topic/subject x2.
casnu Discussion
  x1=Interlocutor_1 x1=Interlocutor_1 x2=Topic

# catni: x1 has authority/is an official in/on/over matter/sphere/persons x2
       derived on basis x3.
catni Leadership
  x1=Leader x2=Governed x3=

# catra: x1 (agent) kills/slaughters/murders x2 by action/method x3.
catra Killing
  x1=Killer x2=Victim x3=Means x3=Cause x3=Instrument

# burna: x1 is embarrassed/disconcerted/flustered/ill-at-ease about/under
       conditions x2 (abstraction).
burna Emotion_directed
  x1=Experiencer x2=Circumstances

# botpi: x1 is a bottle/jar/urn/flask/closable container for x2, made of
       material x3 with lid x4.
botpi Containers
  x1=Container x2=Contents x3=Material x4=Part

# basti: x1 replaces/substitutes for/instead of x2 in circumstance x3; x1 is a
       replacement/substitute.
basti Replacing
  x1=Agent x2=Old x3=New

# banro: x1 grows/expands [an increasing development] to size/into form x2
       from x3.
banro Change_position_on_a_scale
  x1=Item x2=Final_value x3=Initial_value

# budjo: x1 pertains to the Buddhist culture/religion/ethos in aspect x2.
budjo People_by_religion
  x1=Person x2=Persistent_characteristic

# cange: x1 is a farm/ranch at x2, farmed by x3, raising/producing x4;
       (adjective:) x1 is agrarian.
cange Locale_by_use
  x1=Locale x2=Relative_location x3= x4=Use

# cacra: x1 is x2 hours in duration (default is 1 hour) by standard x3.
cacra Measure_duration
  x1=Process x2=Count x3=Unit

# baktu: x1 is a bucket/pail/can/deep, solid, wide-topped container of
       contents x2, made of material x3.
baktu Containers
  x1=Container x2=Contents x3=Material

# batci: x1 bites/pinches x2 on/at specific locus x3 with x4.
# Note: Only a very distant frame, real frame for 'biting' does not exist
batci Measure_by_action
  x1= x2=Entity x3= x4=

# cecla: x1 launches/fires/shoots projectile/missile x2, propelled by x3
       [propellant/propulsion].
cecla Shoot_projectiles
  x1=Agent x2=Projectile x3=Firearm

# cafne: x1 (event) often/frequently/commonly/customarily occurs/recurs by
       standard x2.
cafne Frequency
  x1=Event x2=

# balvi: x1 is in the future of/later than/after x2 in time sequence; x1 is
       latter; x2 is former.
balvi Temporal_collocation
  x1=Trajector_event x1=Trajector_entity x1=Trajector_period x2=Landmark_event
        x2=Landmark_entity x2=Landmark_period

# badgau: g1 causes event b1 which defends/protects b2 (object/state) from
       threat/peril/potential b3 (event).
badgau Defend
  g1=Defender b1= b2=Victim b3=Assailant

# cenba: x1 varies/changes in property/quantity x2 (ka/ni) in amount/degree x3
       under conditions x4.
cenba Similarity
  x1=Entity_1 x2=Differentiating_fact x3=Degree x4=Circumstances

# blanu: x1 is blue [color adjective].
blanu Color
  x1=Entity

# cicna: x1 is cyan/turquoise/greenish-blue [color adjective].
cicna Color
  x1=Entity

# catke: x1 [agent] shoves/pushes x2 at locus x3.
catke Manipulation
  x1=Agent x2=Entity x3=Locus

# cando: x1 is idle/at rest/inactive.
cando Being_active
  x1=Agent

# cfipu: x1 (event/state) confuses/baffles x2 [observer] due to [confusing]
       property x3 (ka).
cfipu Experiencer_obj
  x1=Stimulus x2=Experiencer x3=Depictive

# canci: x1 vanishes/disappears from location x2; x1 ceases to be observed at
       x2 using senses/sensor x3.
canci Departing
  x1=Theme x2=Source x3=

# binxo: x1 becomes/changes/converts/transforms into x2 under conditions x3.
binxo Becoming
  x1=Entity x2=Final_state x3=Circumstances

# ckape: x1 is perilous/dangerous/potentially harmful to x2 under conditions
       x3.
ckape Risky_situation
  x1=Dangerous_entity x2=Asset x3=Situation

# cilre: x1 learns x2 (du'u) about subject x3 from source x4 (obj./event) by
       method x5 (event/process).
cilre Education_teaching
  x1=Student x2=Teacher x3=Manner x4=Material x5=Means

# cirko: x1 loses person/thing x2 at/near x3; x1 loses property/feature x2 in
       conditions/situation x3.
cirko Losing
  x1=Owner x2=Possession x3=Place

# birka: x1 is a/the arm [body-part] of x2; [metaphor: branch with strength].
birka Observable_body_parts
  x1=Body_part x2=Possessor

# banzu: x1 (object) suffices/is enough/sufficient for purpose x2 under
       conditions x3.
banzu Sufficiency
  x1=Item x2=Enabled_situation x3=Circumstances

# bilga: x1 is bound/obliged to/has the duty to do/be x2 in/by
       standard/agreement x3; x1 must do x2.
bilga Being_obligated
  x1=Responsible_party x2=Duty x3=Condition

# bebna: x1 is foolish/silly in event/action/property [folly] (ka) x2; x1 is a
       boob.
bebna Mental_property
  x1=Protagonist x2=Practice

# ckini: x1 is related to/associated with/akin to x2 by relationship x3.
ckini Cognitive_connection
  x1=Concept_1 x2=Concept_2 x3=Specification

# bapli: x1 [force] (ka) forces/compels event x2 to occur; x1 determines
       property x2 to manifest.
bapli Causation
  x1=Cause x2=Effect

# bemro: x1 reflects North American culture/nationality/geography in aspect x2.
bemro Origin
  x1=Entity x2=

# bloti: x1 is a boat/ship/vessel [vehicle] for carrying x2, propelled by x3.
bloti Vehicle
  x1=Vehicle x2= x3=Means_of_propulsion

# badna: x1 is a banana/plantain [fruit/plant] of species/breed x2.
badna Food
  x1=Food x2=Type

# cifnu: x1 is an infant/baby [helpless through youth/incomplete development]
       of species x2.
```

```
cifnu People_by_age
  x1=Person x2=Ethnicity

# cidja: x1 is food/feed/nutriment for x2; x1 is edible/gives nutrition to x2.
cidja Food
  x1=Food x2=

# ciste: x1 (mass) is a system interrelated by structure x2 among components
       x3 (set) displaying x4 (ka).
ciste Gizmo
  x1=Gizmo x2= x3= x4=

# clite: x1 is polite/courteous/civil in matter x2 according to
       standard/custom x3.
clite Custom
  x1=Protagonist x2=Behavior x3=Society

# basna: x1 emphasizes/accentuates/gives emphasis/stress/accent to x2 by
       (action) x3.
basna Place_weight_on
  x1=Agent x2=Consideration x3=Action

# clupa: x1 is a loop/circuit of x2 [material].
clupa Shapes
  x1=Shape x2=Substance

# cabna: x1 is current at/in the present of/during/concurrent/simultaneous
       with x2 in time.
cabna Simultaneity
  x1=Profiled_event x2=Landmark_event

# cabna: x1 is current at/in the present of/during/concurrent/simultaneous
       with x2 in time.
cabna Relative_time
  x1=Focal_occasion x2=Landmark_occasion

# barja: x1 is a tavern/bar/pub serving x2 to audience/patrons x3.
barja Locale_by_use
  x1=Locale x2= x3=

# backla: b1=k1 goes beyond destination b2=k2 from origin k3 via route k4
       using means/vehicle k5.
backla Path_shape
  b1= b2=Goal k1= k2=Goal k3=Source k4=Road k5=Means

# backla: b1=k1 goes beyond destination b2=k2 from origin k3 via route k4
       using means/vehicle k5.
backla Traversing
  b1=Theme b2= Goal k1=Theme k2=Goal k3=Source k4=Path k5=Vehicle k5=Means

# bredi: x1 is ready/prepared for x2 (event).
bredi Activity_prepare
  x1=Agent x2=Activity

# ba'urnoi: n1 is a spoken/uttered message about subject n2 uttered by n3 to
       intended audience n4.
ba'urnoi Statement
  n1=Message n2=Topic n3=Speaker n4=Addressee

# ba'urnoi: n1 is a spoken/uttered message about subject n2 uttered by n3 to
       intended audience n4.
ba'urnoi Chatting
  n1= n2=Topic n3=Interlocutor_1 n4=Interlocutors

# cisma: x1 smiles/grins (facial expression).
cisma Making_faces
  x1=Agent

# carna: x1 turns about vector x2 towards direction x3, turning angular
       distance / to face point x4
carna Change_direction
  x1=Theme x2= x3=Direction x4=Angle x4=Goal

# ba'urdu'u: d1 whines/bitches about d2 by uttering b2.
ba'urdu'u Communication_noise
  d1=Speaker b2=Topic d2=Message

# clira: x1 (event) is early by standard x2.
clira Temporal_subregion
  x1=Sub_part x2=

# bende: x1 is a crew/team/gang/squad/band of persons x2 directed/led by x3
       organized for purpose x4.
bende Organization
  x1=Organization x2=Members x3=Container_possessor x4=Purpose

# skicu: x1 tells about/describes x2 (object/event/state) to audience x3 with
       description x4 (property).
skicu Statement
  x1=Speaker x2=Topic x3=Addressee x4=Depictive

# gunta: x1 (person/mass) attacks/invades/commits aggression upon victim x2
       with goal/objective x3.
gunta Invading
  x1=Invader x2=Land x3=Purpose

# gunta: x1 (person/mass) attacks/invades/commits aggression upon victim x2
       with goal/objective x3.
gunta Committing_crime
  x1=Perpetrator x2= x3=Reason

# cizra: x1 is strange/weird/deviant/bizarre/odd to x2 in property x3 (ka).
cizra Typicality
  x1=Entity x2= x3=Feature

# darlu: x1 argues for stand x2 against stand x3; [an opponent is not
       necessary].
darlu Evidence
  x1=Proposition x2=Support x3=

# bradi: x1 is an enemy/opponent/adversary/foe of x2 in struggle x3.
bradi Hostile_encounter
  x1=Side_1 x2=Side_2 x3=Issue
# klama: x1 comes/goes to destination x2 from origin x3 via route x4 using
       means/vehicle x5.
klama Motion
  x1=Theme x2=Goal x3=Source x4=Path x5=Carrier


# klama: x1 comes/goes to destination x2 from origin x3 via route x4 using
       means/vehicle x5.
klama Arriving
  x1=Theme x2=Goal x3=Source x4=Path x5=Mode_of_transportation

# klama: x1 comes/goes to destination x2 from origin x3 via route x4 using
       means/vehicle x5.
klama Self_motion
  x1=Self_mover x2=Goal x3=Source x4=Path x5=

# citri: x1 is a history of x2 according to x3 (person)/from point-of-view x3.
citri Individual_history
  x1=Events x2=Domain x3=Depictive

# bevri: x1 carries/hauls/bears/transports cargo x2 to x3 from x4 over path
       x5; x1 is a carrier/[porter].
bevri Bringing
  x1=Agent x2=Theme x3=Goal x4=Source x5=Path

# ciksi: x1 (person) explains x2 (event/state/property) to x3 with explanation
       x4 (du'u).
ciksi Explaining_the_facts
  x1= x2=State_of_affairs x3= x4=Fact

# glare: x1 is hot/[warm] by standard x2.
glare Ambient_temperature
  x1=Place x2=Degree

# glare: x1 is hot/[warm] by standard x2.
glare Chemical-sense_description
  x1=Perceptual_source x2=Degree

# cliva: x1 leaves/goes away/departs/parts/separates from x2 via route x3.
cliva Path_shape
  x1= x2=Source x3=Path_shape

# cipra: x1 (process/event) is a test for/proof of property/state x2 in
       subject x3 (individ./set/mass).
cipra Examination
  x1=Examination x2=Examinee x3=Qualification

# benji: x1 transfers/sends/transmits x2 to receiver x3 from
       transmitter/origin x4 via means/medium x5.
benji Sending
  x1= x2=Theme x3=Recipient x3=Goal x4=Sender x5=Transport_means

# berti: x1 is to the north/northern side [right-hand-rule pole] of x2
       according to frame of reference x3.
berti Direction
  x1= x2=Base_position x3=Domain

# berti: x1 is to the north/northern side [right-hand-rule pole] of x2
       according to frame of reference x3.
berti Locative_relation
  x1=Figure x2=Ground x3=

# cmoni: x1 utters moan/groan/howl/scream [non-linguistic utterance] x2
       expressing x3 (property).
cmoni Make_noise
  x1=Sound_source x2=Sound x2=Noisy_event x3=Depictive

# cmoni: x1 utters moan/groan/howl/scream [non-linguistic utterance] x2
       expressing x3 (property).
cmoni Complaining
  x1=Complainer x2=Complaint x3=Depictive

# bartu: x1 is on the outside of x2; x1 is exterior to x2.
bartu Locative_relation
  x1=Figure x2=Ground

# birti: x1 is certain/sure/positive/convinced that x2 is true.
```

```
birti Likelihood
  x1= x2=Hypothetical_event

# catlu: x1 looks at/examines/views/inspects/regards/watches/gazes at x2.
catlu Scrutiny
  x1=Cognizer x2=Ground

# cmima: x1 is a member/element of set x2; x1 belongs to group x2; x1 is
       amid/among/amongst group x2.
cmima Aggregate
  x1=Individuals x2=Aggregate

# ba'orzu'e: z1 grows b1 for purpose/goal z3 to size/form b2 from b3.
ba'orzu'e Growing_food
  z1=Grower b1=Food z3=Purpose b2=  b3=

# cinri: x1 (abstraction) interests/is interesting to x2; x2 is interested in
       x1.
cinri Emotion_directed
  x1=Event x2=Experiencer

# cinri: x1 (abstraction) interests/is interesting to x2; x2 is interested in
       x1.
cinri Experiencer_focus
  x1=Topic x1=Content x2=Experiencer

# ckule: x1 is school/institute/academy at x2 teaching subject(s) x3 to
       audien./commun. x4 operated by x5.
ckule Education_teaching
  x1=Institution x2=Place x3=Subject x4=Student x5=

# ckasu: x1 ridicules/mocks/scoffs at x2 about x3 (property/event) by doing
       activity x4 (event).
ckasu Judgment
  x1=Expressor x2=Evaluee x3=Topic x4=Manner

# badri: x1 is sad/depressed/dejected/[unhappy/feels sorrow/grief] about x2
       (abstraction).
badri Stimulus_focus
  x1=Experiencer x2=Property x2=Circumstances

# ba'usku: s1 (agent) says s2 (sedu'u/text/lu'e concept) for audience s3
       through expressive medium s4.
ba'usku Text_creation
  s1=Author s2=Text s3=Addressee s4=Instrument

# ba'usku: s1 (agent) says s2 (sedu'u/text/lu'e concept) for audience s3
       through expressive medium s4.
ba'usku Communication
  s1=Communicator s2=Message s3=Addressee s4=Medium

# casnu: x1(s) (mass normally, but 1 individual/jo'u possible)
       discuss(es)/talk(s) about topic/subject x2.
casnu Speak_on_topic
  x1=Speaker x2=Audience

# bongu: x1 is a/the bone/ivory [body-part], performing function x2 in body of
       x3; [metaphor: calcium].
bongu Observable_body_parts
  x1=Body_part x2= x3=Possessor

# basti: x1 replaces/substitutes for/instead of x2 in circumstance x3; x1 is a
       replacement/substitute.
basti Take_place_of
  x1=New x2=Old x3=Context

# banro: x1 grows/expands [an increasing development] to size/into form x2
       from x3.
banro Becoming
  x1=Entity x2=Final_state x3=Initial_state

# banro: x1 grows/expands [an increasing development] to size/into form x2
       from x3.
banro Expansion
  x1=Item x2=Result_size x3=Initial_size

# cmene: x1 (quoted word(s)) is a/the name/title/tag of x2 to/used-by
       namer/name-user x3 (person).
cmene Referring_by_name
  x1=Name x2=Entity x3=Speaker

# cmene: x1 (quoted word(s)) is a/the name/title/tag of x2 to/used-by
       namer/name-user x3 (person).
cmene Name_conferral
  x1=Name x2=Entity x3=Speaker

# catke: x1 [agent] shoves/pushes x2 at locus x3.
catke Cause_motion
  x1=Agent x2=Theme x3=Subregion

# cfipu: x1 (event/state) confuses/baffles x2 [observer] due to [confusing]
       property x3 (ka).
cfipu Stimulus_focus
  x1=Stimulus x2=Experiencer x3=Parameter
```

```
# binxo: x1 becomes/changes/converts/transforms into x2 under conditions x3.
binxo Cause_change
  x1=Entity x2=Final_category x2=Final_value x3=Circumstances

# ckape: x1 is perilous/dangerous/potentially harmful to x2 under conditions
       x3.
ckape Being_at_risk
  x1=Harmful_event x1=Dangerous_entity x2=Asset x3=Situation

# ckape: x1 is perilous/dangerous/potentially harmful to x2 under conditions
       x3.
ckape Run_risk
  x1=Protagonist x2=Asset x3=Circumstances

# cilre: x1 learns x2 (du'u) about subject x3 from source x4 (obj./event) by
       method x5 (event/process).
# Note: aligns nicely!
cilre Coming_to_believe
  x1=Cognizer x2=Evidence x3=Topic x4=Evidence x5=Means

# cilre: x1 learns x2 (du'u) about subject x3 from source x4 (obj./event) by
       method x5 (event/process).
cilre Memorization
  x1=Cognizer x2=Pattern x3= x4= x5=Means

# tavla: x1 talks/speaks to x2 about subject x3 in language x4.
tavla Statement
  x1=Speaker x2=Addressee x3=Topic x4=

# bilga: x1 is bound/obliged to/has the duty to do/be x2 in/by
       standard/agreement x3; x1 must do x2.
bilga Be_in_agreement_on_action
  x1=Party_1 x2=Obligation x3=Circumstances

# damba: x1 fights/combats/struggles with x2 over issue x3 (abstract); x1 is a
       fighter/combatant.
damba Quarreling
  x1=Arguer1 x2=Arguer2 x3=Issue

# damba: x1 fights/combats/struggles with x2 over issue x3 (abstract); x1 is a
       fighter/combatant.
damba Point_of_dispute
  x1= x2= x3=Question

# darxi: x1 hits/strikes/[beats] x2 with instrument [or body-part] x3 at locus
       x4.
darxi Impact
  x1=Impactor x2=Impactee x3= x4=Subregion

# darxi: x1 hits/strikes/[beats] x2 with instrument [or body-part] x3 at locus
       x4.
darxi Hit_or_miss
  x1=Agent x2=Target x3=Instrument x4=Target_location

# bloti: x1 is a boat/ship/vessel [vehicle] for carrying x2, propelled by x3.
# Note: Probably misaligned
bloti Cause_motion
  x1=Theme x2= x3=Agent

# lenku: x1 is cold/cool by standard x2.
lenku Ambient_temperature
  x1=Place x2=Degree

# lenku: x1 is cold/cool by standard x2.
lenku Subjective_temperature
  x1=Experiencer x2=Degree

# clupa: x1 is a loop/circuit of x2 [material].
clupa Path_traveled
  x1=Theme x2=

# cmaci: x1 is a mathematics of type/describing x2.
cmaci Fields
  x1=Activity x2=Type

# skicu: x1 tells about/describes x2 (object/event/state) to audience x3 with
       description x4 (property).
skicu Communicate_categorization
  x1=Speaker x2=Item x3= x4=


# gunta: x1 (person/mass) attacks/invades/commits aggression upon victim x2
       with goal/objective x3.
gunta Attack
  x1=Assailant x2=Victim x3=Purpose

# darlu: x1 argues for stand x2 against stand x3; [an opponent is not
       necessary].
darlu Reasoning
  x1=Arguer x2=Support x3=Content

# cusku: x1 (agent) expresses/says x2 (sedu'u/text/lu'e concept) for audience
       x3 via expressive medium x4.
cusku Statement
  x1=Speaker x2=Message x3=Addressee x4=
# glare: x1 is hot/[warm] by standard x2.
glare Temperature
  x1=Entity x2=Degree

# cliva: x1 leaves/goes away/departs/parts/separates from x2 via route x3.
cliva Departing
  x1=Theme x2=Source x3=Path

# cmoni: x1 utters moan/groan/howl/scream [non-linguistic utterance] x2
       expressing x3 (property).
cmoni Communication_noise
  x1=Speaker x2=Message x3=

# cmima: x1 is a member/element of set x2; x1 belongs to group x2; x1 is
       amid/among/amongst group x2.
cmima Membership
  x1=Member x2=Group

# cmila: x1 laughs (emotional expression).
cmila Make_noise
  x1=Sound_source

# jamna: x1 (person/mass) wars against x2 over territory/matter x3; x1 is at
       war with x2.
jamna Hostile_encounter
  x1=Side_1 x2=Side_2 x3=Issue

# cmene: x1 (quoted word(s)) is a/the name/title/tag of x2 to/used-by
       namer/name-user x3 (person).
cmene Being_named
  x1=Name x2=Entity x3=Speaker

# cnebo: x1 is a/the neck [body-part] of x2; [metaphor: a relatively narrow
       point].
cnebo Observable_body_parts
  x1=Body_part x2=Possessor

# fapro: x1 opposes/balances/contends against opponent(s) x2 (person/force
       ind./mass) about x3 (abstract).
fapro Taking_sides
  x1=Cognizer x2=Side x3=Issue

# cmana: x1 is a mountain/hill/mound/[rise]/[peak]/[summit]/[highlands]
       projecting from land mass x2.
cmana Natural_features
  x1=Locale x2=Constituent_parts

# cmalu: x1 is small in property/dimension(s) x2 (ka) as compared with
       standard/norm x3.
cmalu Size
  x1=Entity x2= x3=Standard

# tavla: x1 talks/speaks to x2 about subject x3 in language x4.
tavla Discussion
  x1=Interlocutor_1 x2=Interlocutor_2 x3=Topic x4=Language

# damba: x1 fights/combats/struggles with x2 over issue x3 (abstract); x1 is a
       fighter/combatant.
damba Hostile_encounter
  x1=Side_1 x2=Side_2 x3=Issue

# darxi: x1 hits/strikes/[beats] x2 with instrument [or body-part] x3 at locus
       x4.
darxi Cause_harm
  x1=Agent x2=Victim x3=Instrument x4=Body_part

# lenku: x1 is cold/cool by standard x2.
lenku Temperature
  x1=Entity x2=

# xanka: x1 is nervous/anxious about x2 (abstraction) under conditions x3.
xanka Emotion_directed
  x1=Experiencer x2=Topic x3=Circumstances

# xanka: x1 is nervous/anxious about x2 (abstraction) under conditions x3.
xanka Fear
  x1=Experiencer x2=Topic x3=Circumstances

# tcati: x1 is made of/contains/is a quantity of tea brewed from leaves x2.
tcati Food
  x1=Food x2=Constituent_parts

# cnisa: x1 is a quantity of/contains/is made of lead (Pb); [metaphor: heavy,
       malleable, soft metal].
cnisa Substance
  x1=Substance

# cnano: x1 [value] is a norm/average in property/amount x2 (ka/ni) among
       x3(s) (set) by standard x4.
cnano Typicality
  x1=Entity x2=Feature x3=Comparison_set x4=

# mulno: x1 (event) is complete/done/finished; x1 (object) has become whole in
       property x2 by standard x3.
mulno Activity_finish
  x1=Activity x2= x3=

# mulno: x1 (event) is complete/done/finished; x1 (object) has become whole in
       property x2 by standard x3.
mulno Activity_done_state
  x1=Activity x2= x3=

# tcidu: x1 [agent] reads x2 [text] from surface/document/reading material x3;
       x1 is a reader.
tcidu Reading
  x1=Reader x2=Text x3=

# tcidu: x1 [agent] reads x2 [text] from surface/document/reading material x3;
       x1 is a reader.
# Note: Not exactly correct, but no better brivla currently defined
tcidu Reading_aloud
  x1=Speaker x2=Text x3=

# cnemu: x1 (agent) rewards x2 [recipient] for atypical x3 (event/property)
       with reward/desserts x4.
cnemu Rewards_and_punishments
  x1=Agent x2=Evaluee x3=Reason x4=Means x4=Instrument

# cnino: x1 is new/unfamiliar/novel to observer x2 in feature x3 (ka) by
       standard x4; x1 is a novelty.
cnino Familiarity
  x1=Entity x2=Cognizer x3=Context x4=Degree

# liste: x1 (physical object) is a list/catalog/register of sequence/set x2 in
       order x3 in medium x4.
liste Text
  x1=Text x2=Components x3= x4=Medium

# simlu: x1 seems/appears to have property(ies) x2 to observer x3 under
       conditions x4.
simlu Appearance
  x1=Phenomenon x2=Characterization x3=Perceiver_passive x4=Circumstances

# traji: x1 is superlative in property x2 (ka), the x3 extreme (ka; default ka
       zmadu) among set/range x4.
traji Desirability
  x1=Evaluee x2=Parameter x3=Degree x4=Comparison_set

# ralte: x1 retains/keeps/holds x2 in its possession.
ralte Storing
  x1=Agent x2=Theme

# djuno: x1 knows fact(s) x2 (du'u) about subject x3 by epistemology x4.
djuno Certainty
  x1=Cognizer x2=Content x3=Topic x4=

# djuno: x1 knows fact(s) x2 (du'u) about subject x3 by epistemology x4.
djuno Awareness
  x1=Cognizer x2=Content x3=Topic x4=

# djuno: x1 knows fact(s) x2 (du'u) about subject x3 by epistemology x4.
djuno Familiarity
  x1=Cognizer x2= x3=Entity x4=

# preti: x1 (quoted text) is a question/query about subject x2 by questioner
       x3 to audience x4.
preti Questioning
  x1=Message x2=Topic x3=Speaker x4=Addressee

# tikpa: x1 kicks [hits with x1's foot/feet x4] x2 in/at locus x3, using x1's
       foot/feet x4.
tikpa Cause_harm
  x1=Agent x2=Victim x3=Body_part x4=Instrument

# bajykla: k1 runs to destination k2 from origin k3 via route k4 using limbs
       b3 with gait b4.
bajykla Self_motion
  k1=Self_mover k2=Goal k3=Source k4= Path b3= b4=

# kakne: x1 is able to do/be/capable of doing/being x2 (event/state) under
       conditions x3 (event/state).
kakne Capability
  x1=Entity x2=Event x3=Circumstances

# jalge: x1 (action/event/state) is a result/outcome/conclusion of antecedent
       x2 (event/state/process).
jalge Causation
  x1=Effect x2=Cause
```

Lexical unit level annotations

```
ba'argau Sign
  mark.v
ba'armo'a Pattern
  pattern.n
ba'orzu'e Cause_expansion
  grow.v
ba'orzu'e Growing_food
  grow.v
ba'ostu Building_subparts
  nursery.n
ba'urdu'u Communication_noise
  whine.v
ba'urdu'u Complaining
  whine.v;bitch.v
ba'urnoi Chatting
  speak.v
ba'urnoi Statement
  speak.v;message.n
ba'urnoi Text_creation
  utter.v;speak.v
ba'urtadji Spelling_and_pronouncing
  pronounce.v
ba'urxausku Expressing_publicly
  expression.n;express.v
ba'usku Communication
  say.v
ba'usku Statement
  say.v
ba'usku Text_creation
  say.v
backla Motion
  go.v
backla Path_shape
  pass.v
backla Traversing
  pass.v
bacru Statement
  speak.v;say.v
bacycripu Roadways
  bridge.n
bacyde'i Observable_body_parts
  mouth.n
badgau Defend
  defend.v;defender.n
badna Food
  banana.n
badri Emotion_directed
  sad.a
badri Stimulus_focus
  sad.a
badydi'u Buildings
  fortification.n;fortress.n;castle.n;building.n
bagyce'a Weapon
  bow.n
bajli'a Fleeing
  run away.v
bajra Self_motion
  run.v
bajykla Self_motion
  barge.v;hasten.v;hurry.v;jog.v;leap.v;run.v;rush.v;sprint.n;sprint.v
baktu Containers
  bucket.n
balvi Temporal_collocation
  future.a
bancu Surpassing
  exceed.v
bandu Defend
  defend.v
banli Desirability
  great.a
banro Becoming
  grow.v
banro Change_position_on_a_scale
  grow.v
banro Expansion
  grow.v
banzu Sufficiency
  suffice.v;enough.adv
bapli Causation
  force.v
barda Size
  big.a;large.a
bargu Shapes
  curve.n
barja Buildings
  bar.n;tavern.n;pub.n
barja Locale_by_use
  pub.n
bartu Locative_relation
  out.prep
bartu Part_inner_outer
  outside.a
basna Convey_importance
  emphasize.v
basna Place_weight_on
  emphasize.v
basti Replacing
  replace.v;substitute.v
basti Take_place_of
  replace.v;substitute.v
batci Measure_by_action
  bite.n;pinch.n
bebna Mental_property
  foolish.a;foolishness.n
bemro Origin
  american.a
bende Aggregate
  team.n;crew.n
bende Organization
  club.n
benji Sending
  send.v
benji Transfer
  transfer.n;transfer.v
bersa Kinship
  son.n
berti Direction
  north.adv
berti Locative_relation
  north.prep
berti Part_orientational
  north.a;northern.a
betfu Observable_body_parts
  belly.n;trunk.n
betri Catastrophe
  tragedy.n;disaster.n
bevri Bringing
  carry.v;haul.v;bear.v;transport.v
bevri Carry_goods
  carry.v
bilga Be_in_agreement_on_action
  agreement.n
bilga Being_obligated
  duty.n;should.v;must.v
bilma Medical_conditions
  ill.a;sick.a;disease.n
bilni Military
  military.a;military.n
binxo Becoming
  become.v
binxo Cause_change
  convert.v;transform.v
birje Food
  beer.n
birka Observable_body_parts
  arm.n
birti Certainty
  certain.a;sure.a;positive.a
birti Likelihood
  certain.a;sure.a
bitmu Architectural_part
  wall.n
blabi Color
  white.a
blaci Substance
  glass.n
blanu Color
  blue.a
bloti Cause_motion
  propel.v
bloti Vehicle
  boat.n;vessel.n;ship.n
bolci Shapes
  ball.n
bongu Observable_body_parts
  -
botpi Containers
  bottle.n;jar.n;flask.n;urn.n;container.n
boxfo Shapes
  sheet.n
boxna Moving_in_place
  wave.v
bradi Hostile_encounter
  struggle.n
bradi Taking_sides
  oppose.v;against.prep;side.n;side.v;opposition_((act)).n;opposition_((entity))
         .n
bratu Precipitation
  hail.n;sleet.n;hail.v;precipitation.n;rain.n
bredi Activity_prepare
  ready.v;prepare.v
bredi Activity_ready_state
  ready.a
briju Building_subparts
  office.n
bruna Kinship
  brother.n
budjo People_by_religion
  buddhist.n
bukpu Clothing
```

```
  clothes.n
bunda Measure_mass
  pound.n
bunre Color
  brown.a
burna Emotion_directed
  embarrassed.a;flustered.a;disconcerted.a
cabna Relative_time
  simultaneous.a
cabna Simultaneity
  concurrent.a;simultaneous.a
cabna Temporal_collocation
  current.a;now.adv
cabra Gizmo
  apparatus.n;equipment.n;device.n
cacra Measure_duration
  hour.n
cadzu Self_motion
  walk.v;walk.n
cafne Frequency
  often.adv;frequently.adv;frequent.a
cakla Food
  chocolate.n
calku Part_inner_outer
  shell.n
canci Departing
  vanish.v;disappear.v
cando Being_active
  inactive.a
cange Locale_by_use
  farm.n;ranch.n
canja Exchange
  exchange.n;trade.n;exchange.v;trade.v
canko Connecting_architecture
  window.n
canre Substance
  sand.n
carce Vehicle
  cart.n;carriage.n
carmi Location_of_light
  brilliant.a;bright.a
carna Change_direction
  turn.v;turn.n
carna Moving_in_place
  turn.v
carvi Precipitation
  rain.n;precipitation.n;shower.n;rain.v
casnu Discussion
  discuss.v;talk.n
casnu Speak_on_topic
  discuss.v
catke Cause_motion
  shove.v;push.v
catke Manipulation
  push.v
catlu Perception_active
  look.v;look.n;gaze.n
catlu Scrutiny
  look.v
catni Leadership
  authority.n;officer.n;official.n
catra Killing
  kill.v;murder.n;slaughter.n;murder.v
cecla Shoot_projectiles
  shoot.v;launch.n
cecmu Aggregate
  community.n;colony.n
cedra Calendric_unit
  era.n;age.n
cenba Similarity
  vary.v
cerni Calendric_unit
  morning.n;day.n
certu Expertise
  expert.n;expert.a;prowess.n;skilled.a
cfari Process_start
  commence.v;begin.v;start.v
cfika Text
  fiction.n
cfila Judgment
  fault.n
cfine Shapes
  wedge.n
cfipu Experiencer_obj
  confuse.v;baffle.v
cfipu Stimulus_focus
  confusing.a
cicna Color
  cyan.a
cidja Food
  food.n
cidni Observable_body_parts
  knee.n;elbow.n;limb.n;body.n
cifnu People_by_age
  baby.n;infant.n
```

```
cigla Observable_body_parts
  gland.n
cikna Being_awake
  awake.a;conscious.a
ciksi Explaining_the_facts
  explain.v;explanation.n
ciksi Statement
  explain.v;explanation.n
cilmo Being_wet
  moist.a;wet.a
cilre Coming_to_believe
  learn.v
cilre Education_teaching
  learn.v
cilre Memorization
  learn.v
cilta Connectors
  thread.n;wire.n
cimni Duration_attribute
  eternal.a;infinite.a
cinba Manipulation
  kiss.v
cinla Body_description_holistic
  thin.a
cinmo Feeling
  emotion.n;feel.v
cinri Emotion_directed
  interest.n
cinri Experiencer_focus
  interested.a
cinri Mental_stimulus_stimulus_focus
  interesting.a
cipra Examination
  test.n
cipra Operational_testing
  test.n;test.v
cirko Losing
  lose.v
cirla Food
  cheese.n
ciska Text_creation
  write.v
cisma Facial_expression
  smile.n
cisma Making_faces
  smile.v;grin.v
ciste Gizmo
  system.n
citka Ingestion
  eat.v;ingest.v;consume.v
citno Age
  young.a
citri History
  history.n
citri Individual_history
  history.n
citsi Calendric_unit
  year.n;season.n
cizra Idiosyncrasy
  weird.a
cizra Typicality
  odd.a
ckafi Food
  coffee.n
ckaji Distinctiveness
  characteristic.a;characterize.v;aspect.n
ckape Being_at_risk
  danger.n
ckape Risky_situation
  dangerous.a;danger.n
ckape Run_risk
  peril.n
ckasu Judgment
  mock.v
ckasu Judgment_communication
  ridicule.v;ridicule.n;mock.v;scoff.v
ckeji Emotion_directed
  ashamed.a
ckini Cognitive_connection
  relationship.n;related.a
ckire Judgment_direct_address
  grateful.a
ckule Education_teaching
  school.v
ckule Locale_by_use
  school.n;institute.n
cladu Sound_level
  loud.a
clani Dimension
  long.a
claxu Possession
  lack.v
clira Relative_time
  early.a
clira Temporal_subregion
```

```
  early.a
clite Custom
  custom.n
clite Social_interaction_evaluation
  polite.a;civil.a;courteous.a
cliva Departing
  leave.v;depart.v
cliva Path_shape
  leave.v
clupa Path_traveled
  circuit.n
clupa Shapes
  circuit.n
cmaci Fields
  mathematics.n
cmalu Size
  small.a;tiny.a;little.a
cmana Natural_features
  mountain.n;hill.n;land.n
cmene Being_named
  name.n
cmene Name_conferral
  name.v
cmene Referring_by_name
  name.n
cmila Make_noise
  laugh.v
cmima Aggregate
  group.n
cmima Membership
  member.n;belong.v
cmoni Communication_noise
  moan.v;groan.v;scream.v
cmoni Complaining
  moan.v
cmoni Make_noise
  moan.v;scream.v
cnano Typicality
  average.a;ordinary.a
cnebo Observable_body_parts
  neck.n
cnemu Rewards_and_punishments
  reward.n;reward.v
cnino Familiarity
  new.a;unfamiliar.a
cnisa Substance
  -
cusku Statement
  comment.v;declare.v;describe.v;explain.v;mention.v;note.v;remark.v;report.v;
        say.v;speak.v;state.v;suggest.v;talk.v;tell.v
damba Hostile_encounter
  struggle.v;fight.n;struggle.n;fight.v;altercation.n;battle.n;battle.v;bout.n;
        brawl.n;brawl.v;clash.n;clash.v;combat.n;conflict.n;confront.v;
        confrontation.n;fighting.n;wrangling.n
damba Point_of_dispute
  issue.n
damba Quarreling
  fight.n;fight.v
darlu Evidence
  argue.v;argument.n;mean.v;substantiate.v;support.v
darlu Reasoning
  argue.v;reason.v
darxi Cause_harm
  hit.v;strike.v
darxi Hit_or_miss
  hit.v;hit.n
darxi Impact
  hit.v;strike.v;hit.n
djuno Awareness
  understand.v;understanding.n;know.v;knowledge.n;knowledgeable.a;believe.v;
        think.v;imagine.v;knowledge.n;knowledgeable.a
djuno Certainty
  know.v;believe.v;doubt.v;doubtful.a;doubt.n
djuno Familiarity
  know.v;acquainted.a;familiar.a
fapro Taking_sides
  oppose.v;opponent.n;against.prep;opposition_((act)).n;opposition_((entity)).n;
        side.n;side.v
glare Ambient_temperature
  hot.a;warm.a
glare Chemical-sense_description
  hot.a
glare Temperature
  hot.a;warm.a
gunta Attack
  attack.n;attack.v
gunta Committing_crime
  commit.v
gunta Invading
  invade.v
jalge Causation
  because of.prep;because.c;causative.a;cause.n;cause.v;lead_(to).v;reason.n;
        result.n;result_(in).v;since.c
jamna Hostile_encounter
  war.n;war.v
kakne Capability
  able.a;capable.a;ability.n
klama Arriving
  come.v;get.v;return.v;visit.v
klama Motion
  go.v;fly.v;glide.v;move.v;travel.v
klama Self_motion
  head.v;proceed.v;run.v;rush.v;walk.v
lenku Ambient_temperature
  cold.a;cold.n
lenku Subjective_temperature
  cold.a
lenku Temperature
  cold.a
liste Text
  list.n
mulno Activity_done_state
  done.a;finished.a;through.a
mulno Activity_finish
  complete.v;completion.n;finish.v
preti Questioning
  question.n;question.v;questioning.n;interrogate.v;query.n;query.v;quiz.v
ralte Storing
  keep.v
simlu Appearance
  appear.v;seem.v;look.v
skicu Communicate_categorization
  describe.v;description.n;depict.v;depiction.n;portray.v
skicu Statement
  describe.v;tell.v;address.v;mention.v;note.v;observe.v,remark.v;say.v;speak.v;
        talk.v
tavla Discussion
  talk.n;discuss.v
tavla Statement
  talk.v;speak.v;address.v;say.v;tell.v
tcati Food
  tea.n
tcidu Reading
  devour.v;peruse.v;pore.v;read.v;reader.n;scan.v;skim.v
tcidu Reading_aloud
  read.v;read out.v
tikpa Cause_harm
  kick.v
traji Desirability
  amazing.a;astonishing.a;astounding.a;excellent.a;execrable.a;extraordinary.a;
        fabulous.a;good.a;great.a;incredible.a;magnificent.a;marvellous.a;
        outstanding.a;sensational.a;splendid.a;super.a;superb.a;superlative.a;
        wonderful.a
xanka Emotion_directed
  nervous.a;agonized.a;agony.n;anxious.a;desolate.a;despair.n;frightened.a
xanka Fear
  nervous.a;scared.a
```

Bibliography

Baker, C., Ellsworth, M., and Erk, K. (2007). Semeval'07 task 19: frame semantic structure extraction. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pp. 99–104, Prague, Czech Republic. ACL '07.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, pp. 86–90, Montreal, Canada. ACL '98.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013). Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 178–186, Sofia, Bulgaria. ACL '13.

Bär, D., Biemann, C., Gurevych, I., and Zesch, T. (2012). Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pp. 435–440, Montreal, Canada. NAACL '12.

Bateman, J. A. (1997). Enabling technology for multilingual natural language generation: the kpml development environment. *Natural Language Engineering*, 3(01):15–55.

Biemann, C. and Riedl, M. (2013). Text: Now in 2d! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

Bishop, C. M. (2006). *Pattern recognition and machine learning*, volume 1. Springer.

Brown, J. C. (1989). *LOGLAN 1: a logical language*. THE LOGLAN INSTITUTE, INC. Available at `http://www.loglan.org/Loglan1/`, accessed June 2014.

Carreras, X. and Màrquez, L. (2005). Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pp. 152–164, Sydney, Australia. ACL '05.

Chen, D., Schneider, N., Das, D., and Smith, N. A. (2010). Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pp. 264–267, Uppsala, Sweden. ACL '10.

Cowan, J. W. (1997a). *The Complete Lojban Language*. Logical Language Group Inc.

Cowan, J. W. (1997b). *The Lojban Reference Grammar*. Logical Language Group Inc. Pre-final version available at `http://www.lojban.org/tiki/The+Lojban+Reference+Grammar`, accessed June 2014.

Crocker, M. W. (1996). *Computational Psycholinguistics: An Inter-Disciplinary Approach to the Study of Language*, volume 20. Springer.

Das, D., Chen, D., Martins, A. F., Schneider, N., and Smith, N. A. (2014). Frame-semantic parsing. *Computational Linguistics*, 40:1(1):9–56.

Das, D., Martins, A. F. T., and Smith, N. A. (2012). An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, SemEval '12, pp. 209–217, Jeju Island, Korea. ACL '12.

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010a). Probabilistic frame-semantic parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 948–956, Los Angeles, United States. HLT '10.

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010b). SEMAFOR 1.0: A probabilistic frame-semantic parser. Technical Report CMU-LTI-10-001, Carnegie Mellon University.

Das, D. and Smith, N. A. (2011). Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume*

1, pp. 1435–1444, Portland, United States. HLT ’11.

De la Higuera, C. (2010). *Grammatical inference*, volume 96. Cambridge University Press Cambridge.

De Marneffe, M.-C. and Manning, C. D. (2008). The stanford typed dependencies representation. In *Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pp. 1–8, Manchester, United Kingdom. COLING ’08.

De Saussure, F. (1916). Nature of the linguistic sign. *Course in general linguistics*.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. (2004). Ontology matching: A machine learning approach. In *Handbook on ontologies*, pp. 385–403. Springer.

Dorogovtsev, S. N. and Mendes, J. F. F. (2001). Language as an evolving word web. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1485):2603–2606.

Erk, K. and Padó, S. (2006). SHALMANESER - A Toolchain For Shallow Semantic Parsing. In *Proceedings of LREC ’06*, Genoa, Italy. LREC ’06.

Euzenat, J. and Shvaiko, P. (2007). *Ontology matching*, volume 18. Springer.

Fillmore, C. J. (1976). Frame semantics and the nature of language. *Annals of the New York Academy of Sciences*, 280(1):20–32.

Fleischman, M., Kwon, N., and Hovy, E. (2003). Maximum entropy models for framenet classification. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pp. 49–56, Sapporo, Japan. EMNLP ’03.

Ford, B. (2004). Parsing expression grammars: a recognition-based syntactic foundation. In *ACM SIGPLAN Notices*, volume 39, pp. 111–122. ACM SIGPLAN ’04.

Fürstenau, H. and Lapata, M. (2009). Graph alignment for semi-supervised semantic role labeling. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 1, pp. 11–20, Singapore, Republic of Singapore. EMNLP ’09.

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Goddard, C. (2008). Natural semantic metalanguage: The state of the art. *Cross-linguistic semantics*, 102:1–34.

Goertzel, B. (2005a). LojLink: A Novel Approach to Knowledge Management. Unpublished. Available online at `http://www.goertzel.org/new_research/LojLink.pdf`, accessed June 2014.

Goertzel, B. (2005b). Potential Computational Linguistics Resources for Lojban. Unpublished. Available online at `http://www.goertzel.org/new_research/lojban_AI.pdf`, accessed June 2014.

Goertzel, B. (2006). Lojban++: An Efficient, Minimally Ambiguous User-Friendly Natural-Like Language for Human-Computer, Computer-Computer and Human-Human Communication . Unpublished. Available online at `http://www.goertzel.org/papers/lojbanplusplus.pdf`, accessed June 2014.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics*, volume 2, pp. 539–545, Nantes, France. COLING ’92.

Johansson, R. and Nugues, P. (2007). Lth: semantic structure extraction using nonprojective dependency trees. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pp. 227–230, Prague, Czech Republic. SemEval ’07.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2nd edition edition.

Kay, P. and Kempton, W. (1984). What is the sapir-whorf hypothesis? *American Anthropologist*, 86:65–79.

Kilgarriff, A., Rychly, P., Smrz, P., and Tugwell, D. (2004). Itri-04-08 the sketch engine. *Information Technology*, 105:116.

Kingsbury, P. and Palmer, M. (2003). Propbank: the next level of treebank. In *Proceedings of Treebanks and Lexical Theories*, volume 3, Váxjó, Sweden.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pp. 423–430, Sapporo, Japan. ACL '03.

Koomen, P., Punyakanok, V., Roth, D., and Yih, W.-t. (2005). Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pp. 181–184, Ann Arbor, United States. ACL '05.

Lin, D. (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, volume 2, pp. 768–774, Montreal, Canada. ACL '98.

Litkowski, K. (2004). Senseval-3 task: Automatic labeling of semantic roles. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pp. 9–12, Barcelona, Spain. ACL '03.

McCord, M. C., Murdock, J. W., and Boguraev, B. K. (2012). Deep parsing in watson. *IBM Journal of Research and Development*, 56(3.4):3–1.

Meyer, M., Andrews, C., and Itz, M. (2005). A natural language interface using first-order logic. Master's thesis, Worcester Polytechnic Institute, United States. Bachelor's thesis. Available at `https://www.wpi.edu/Pubs/E-project/Available/E-project-121705-011216/unrestricted/NaturalLanguageInterfaceUsingFOL.pdf`, accessed June 2014.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). The nombank project: An interim report. In *HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, pp. 24–31, New York City, United States. HLT '04.

Mihalcea, R., Corley, C., and Strapparava, C. (2006). Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence*, volume 6, pp. 775–780. AAAI '06.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Moschitti, A., Pighin, D., and Basili, R. (2008). Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Nezhadi, A. H., Shadgar, B., and Osareh, A. (2011). Ontology alignment using machine learning techniques. *International Journal of Computer Science & Information Technology*, 3(2):139.

Nicholas, N. (1993). A lojban-to-prolog semantic analyser. Unpublished. Available at `http://www.lojban.org/files/papers/lojban_parser_paper`, acccessed June 2014.

Nicholas, N. (1996). Lojban as a Machine Translation Interlanguage in the Pacific. In *Fourth Pacific Rim International Conference on Artificial Intelligence: Workshop on Future Issues for Multilingual Text Processing*, pp. 31–39, Cairns, Australia. PRICAI '96.

Nicholas, N. (2003). *What Is Lojban*. Logical Language Group Inc.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Palmer, M. (2009). Semlink: Linking propbank, verbnet and framenet. In *Proceedings of the Generative Lexicon Conference*, pp. 9–15, Pisa, Ital. GL '09.

Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Pease, A., Niles, I., and Li, J. (2002). The suggested upper merged ontology: A large ontology for the semantic web and its applications. In *Working notes of the AAAI-2002 workshop on ontologies and the semantic web*, volume 28, Edmonton, Canada. AAAI '02.

Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, pp. 38–41, New York City, United States. HLT '04.

Potts, T. C. (2007). *Structures and Categories for the Representation of Meaning*. Cambridge University Press.

Pradhan, S., Hacioglu, K., Ward, W., Martin, J. H., and Jurafsky, D. (2005). Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pp. 217–220, Sydney, Australia. ACL '05.

Pradhan, S. S., Ward, W., Hacioglu, K., Martin, J. H., and Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. In *HLT-NAACL*, pp. 233–240, New York City, United States. HLT '04.

Punyakanok, V., Roth, D., Yih, W.-t., and Zimak, D. (2004). Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, p. 1346, Geneva, Switzerland. COLING '04.

Robin Turner, N. N. (2003). Lojban For Beginners. Available at `http://www.tlg.uci.edu/~opoudjis/lojbanbrochure/lessons.pdf`, accessed June 2014.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.

Ruppenhofer, J., Ellsworth, M., Petruck, M. R., Johnson, C. R., and Scheffczyk, J. (2006). *FrameNet II: Extended Theory and Practice*. International Computer Science Institute, Berkeley, California.

Ruppenhofer, J., Sporleder, C., Morante, R., Baker, C., and Palmer, M. (2010). Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 45–50, Uppsala, Sweden. ACL '10.

Schuler, K. K. (2005). *Verbnet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, Faculties of Computer and Information Science of the University of Pennsylvania. PhD thesis.

Shi, L. and Mihalcea, R. (2004). An algorithm for open text semantic parsing. In *Proceedings of the 3rd Workshop on RObust Methods in Analysis of Natural Language Data*, pp. 59–67, Barcelona, Spain. ROMAND '04.

Shi, L. and Mihalcea, R. (2005). Putting pieces together: Combining framenet, verbnet and wordnet for robust semantic parsing. In *Computational Linguistics and Intelligent Text Processing*, pp. 100–111. Springer.

Speer, R. and Havasi, C. (2004). Meeting the Computer Halfway: Language Processing in the Artificial Language Lojban. In *Proceedings of MIT Student Oxygen Conference, MIT*, Ashland, United States.

Stallard, D. (2000). Talk'n'travel: A conversational system for air travel planning. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pp. 68–75, Seattle, United States. ANLP '00.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pp. 159–177, Manchester, United States. CoNLL '08.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., and Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958.

Thompson, C. A., Levy, R., and Manning, C. D. (2003). A generative model for semantic role labeling. In *Machine Learning: ECML 2003*, pp. 397–408. Springer.

Titov, I. and Klementiev, A. (2012). A bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 12–22, Avignon, France. EACL '12.

Toutanova, K., Haghighi, A., and Manning, C. D. (2005). Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 589–596, Sydney, Australia. ACL '05.

Varga, D., Halácsy, P., Kornai, A., Nagy, V., Németh, L., and Trón, V. (2007). Parallel corpora for medium density languages. *Amsterdam Studies in the Theory and History of Linguistic Science, Series 4*, 292:247.

Wirick, B. (2005). Lojban as a tool for encoding prose on the semantic web. Master's thesis, Faculty of California Polytechnic State University, San Luis Obispo. Available at `http://users.csc.calpoly.edu/~gfisher/classes/590/reference/theses/wirick.pdf`, accessed June 2014.

Yi, S.-T., Loper, E., and Palmer, M. (2007). Can semantic roles generalize across genres? In *HLT-NAACL*, pp. 548–555, Rochester, United States. HLT '07.