Potential Computational Linguistics Resources for Lojban

Ben Goertzel
March 6, 2005

1 Introduction

Lojban (Cowan, 1998; Nicholas and Cowan, 2003) is an intentionally-constructed human language whose semantics is explicitly based on formal logic. It is a forking of the language Loglan created by James Cooke Brown in the 1950's; but at present Lojban is much more actively used than classical Loglan, so I will refer almost only to Lojban from here on.

One of the design goals in the creation of Lojban/Loglan was to create a language with the minimum plausible amount of ambiguity. This minimal amount is not zero, which is why Lojban is not equivalent to formal logic. Communication using formal logic would be unrealistically cumbersome for human beings, and perhaps for AI systems as well. The construction of ambiguous expressions, together with the implicit assumption that the listeners/readers will be able to disambiguate based on shared contextual knowledge, is essential to the nature of language itself. But it's clear that all human languages possess a much larger amount of ambiguity than is necessary to achieve compact communication to context-savvy listeners/readers. Lojban appears to come close to the necessary minimum. For instance, each Lojban word has only one meaning (no lexical-level semantic ambiguity), and Lojban syntax is completely specified by a tractably-sized formal grammar.

One of the motivations underlying the creation of Lojban/Loglan was to make communication between humans and computers easier. The conceptual basis here seems quite clear: computers deal well with formal logic and with unambiguous knowledge representations generally, so a language which is logic-like and relatively unambiguous should be particularly amenable to computational processing. So far, however, this conceptual promise has not been fulfilled: Lojban/Loglan have not been used within AI research at all, so far as I know.

Part of the reason for the lack of use of Lojban/Loglan within AI is simple lack of familiarity: almost no AI researchers have heard of these obscure constructed languages. Part of it is the practical, application-driven nature of much current AI research (in both academia and industry): it’s a lot easier for funding sources to see the value of making AI systems understand languages with more than a few hundred speakers. And part of the reason, I believe, is the lack of any professional-quality computational linguistics resources for Lojban/Loglan.

I’ve recently come to the conclusion that it would be valuable to use Lojban within my own AI project, Novamente (Looks et al, 2004). For my general opinion on the utility of computational linguistics resources for AI, see my essay *Post-Embodied AI* (Goertzel, 2004). In short, I don’t think these resources can be used as a substitute for experientially acquired language facility, but I do think they can be used as a kind of scaffolding, which an AI system can use to support its experiential building-up of linguistic knowledge.

After investigating the current state of Lojban tools and resources, however, I’ve concluded that in order to do this really effectively, it will be necessary to first create some generic Lojban computational-linguistics tools and resources, of a non-Novamente-specific nature. These tools and resources fall into two categories:

- Those aimed at making it easier to parse Lojban text into semantic relationships inside an AI system, or to generate Lojban text from semantic relationships inside an AI system
- Those aimed at automating translation between English and Lojban

The purpose of this document is to describe these tools and resources, which I believe are required to make Lojban really useful from a computational linguistics perspective.

At the end I will give some rough estimates regarding the amount of work that will be required to create these tools and resources. The conclusion is that there are roughly two human-years of work here, assuming the humans involved are highly competent and appropriately knowledgeable. About 10 months is for the Lojban-only work, and the rest for translation-oriented work. While this may seem like a lot of work, it’s remarkably little compared to what would be necessary to do a similar job for any natural human language (or for a constructed language like Esperanto, which lacks Lojban’s logical foundation).

2 Features of Lojban Grammar

The Lojban language is small, but not so small that it can be thoroughly presented in one section of a brief paper. Here I’ll describe only a few key features of Lojban grammar, which are necessary for describing the tools and resources to be presented in later sections.

The best introduction to Lojban is (Nicholas and Cowan, 2003), and the reader is referred there to get a basic grounding in Lojbanic concepts and syntax. However, that book is quite incomplete and (Cowan, 1999) is the reference for a more complete understanding.

Lojban begins with about 1300 root words, all of which have five letters (e.g. “cukta” for book, “vecnu” for sell). It then contains simple rules for combining root words to form compounds, which are called lujvo. For instance, the word for “nurse” is a compound “kurmikce”, formed from “kurji” (take care of) and “mikce” (medic).

There are no parts of speech as such – every word is treated as a predicate, for instance instead of a noun meaning person, there is a word denoting the predicate is_person. In Lojban lingo, a predicate is called a selbri, and its arguments are called sumti. Both selbri and sumti may be either simple or compound.

There is also a mechanism for creating ambiguous combinations of words called tanru – for instance “melbi tavla” (“beautiful talker”), which could mean either someone who’s beautiful and is talking, or someone who talks beautifully. The capability to leave some combinations ambiguous in their semantics is essential to Lojban’s usability – if every relationship had to be specified explicitly as in predicate logic, then creating Lojban sentences would be intolerably laborious.

The bulk of Lojban syntax is pushed into “function words,” which are called cmavo. These deal with preposition and pronoun type functions, but also with tense, grouping and a variety of other phenomena.

A very simple example sentence is

```
mi tavla le vecnu
```

where

```
me = I
tavla = talk
venu = sell
```

The cmavo “le” indicates that a seller is meant rather than an act of selling, so the sentence means “I talk to the seller.” The convention

$$\texttt{argument}_1\ \texttt{predicate}\ \texttt{argument}_2\ \texttt{argument}_3\ \ldots$$

is generally used although other orderings are accepted. When orderings are potentially ambiguous, appropriate cmavo must be inserted to avoid the ambiguity.

On the other hand

```
mi ca ba’o tavla le vecnu be lo since
```

means “I have now talked to the seller of snakes,” where “since” means is_snake, “be” is a cmavo that binds an argument to an instance of a predicate, and “ca ba’o” are tense-marking cmavo.

3 Design for an AI-Friendly Lojban Parser

Unlike any natural human language, Lojban grammar is fully formally specified. Three equivalent formal Lojban grammars are available online: one in yacc format, one in BNF format, and one (by far the most elegant, and the only one that deals adequately with elidable terminators) in PEG (Parsing Expression Grammar) format (Powell, date unspecified). However, none of these grammars is really ideal from an AI perspective. In this section I'll present an alternate, complementary approach to specifying Lojban grammar, which I believe will be more useful in an AI context.

From an AI perspective, what one wants from a grammar is not just the capability to assess which sentences are grammatical and which are not. One wants a grammar to map sentences into parse structures that can then be naturally mapped into semantic structures. Of course, different AI systems create different sorts of internal semantic structures, and the grammar ideally suited to one AI system might not be ideally suited to another. However, there are some general properties that make grammars useful for semantic mapping, so this kind of specialization is not as critical as one might think.

In my own AI projects, we have carried out semantic mapping based on two types of grammars: Webmind used a standard lexicalized feature-structure grammars (XTag; Doran et al, 1994) and Novamente uses a link grammar (Sleator and Temperley, 1993). Both of these approaches are adequate, but we have found link grammars (which are significantly less fashionable within the linguistics community) easier to work with. It is also possible to generate a standard lexicalized feature structure grammar from a link grammar (though the tool that Sleator and Temperley have created for this purpose, and released on the link parser website, is overly simplistic and flawed). The root difference between link grammars and standard lexicalized feature structure grammars is that the former only involves syntactic relationships between individual words, whereas the latter explicitly involves groupings of words into phrases, and syntactic relationships between phrases. The absence of explicit phrase groupings makes link grammar significantly simpler.

The approach we propose here for mapping Lojban parses into semantic structure is loosely inspired by link grammars, in its focus on links between words. However, due to Lojban's particularly simple grammatical structure, the subtleties of the link grammar prove unnecessary in a Lojban context.

Before we get to the elegant part, there is a bit of mess to be dealt with. When one digs into the details of the Lojban grammar rules, some irritating subtleties arise, which we propose to work around via an innovative design strategy in which the grammar is broken into two parts. Specifically, there is a lot of complexity introduced by two factors

- the preponderance of elidable cmavo (function words that may be included for completeness, but may be omitted in cases where this doesn't result in any ambiguity)
- the presence of permutation operators (se, te, ve, xe) which reorder the arguments of a brivla

These factors make the formal grammar a lot more complicated than it would be otherwise, even though they are basically irrelevant from a semantic perspective, in the sense that if one removed permutations and banned elidablity, Lojban would have the same expressive power (although it would become more annoying to speak).

What we suggest is to divide the Lojban parsing process into three stages:

- **Stage 1**: Transformation of a Lojban sentence into a "normalized" Lojban sentence, via insertion of all elided cmavo, and removal of all permutation operators (via enaction of the permutations and insertion of zo'e in any places left blank).
- **Stage 2**: Transformation of the normalized Lojban sentence into predicate-argument form, using Lojban grammar rules
- **Stage 3**: Mapping of a Lojban predicate-argument relationship into a set of syntactic relationships that are well-prepared for semantic mapping

The separation allows the semantic-mapping component to live in a simplified world in which the more annoying aspects of Lojban syntax don't exist. This will allow the link grammar like semantic mapping rules to involve an extremely simple set of linking patterns.

Unfortunately, it seems there is no simple and general way to carry out Stage 1. In general, it seems that to fill in the elided cmavo correctly, it may be necessary to completely parse the sentence using the full Lojban grammar, or something very close to this. In nearly all practical cases, the job of filling in elided cmavo is very simple, but there are also more complex cases. Thus, our suggestion is that both Stage 1 and Stage 2 be carried out using a modification of the PNG grammar for Lojban. It shouldn't be too hard to utilize these grammar rules to create a modified grammar capable of performing Lojban normalization and predicatization.

As an example, the sentence

```
mi djica le nu mi klama le zarci
```

("I want to go to the store") would be normalized to

```
mi cu djica le nu mi cu klama le zarci ku vau kei ku vau
```

The latter has the same semantics, has an easier structure from the perspective of forma parsing, and would be much more annoying to type or speak.

Note that the reversal of the normalization process can be carried out using the same grammatical rules. This is useful for Lojban language production. If one has a process for transforming logic expressions inside one's AI into normalized Lojban sentences, then one can apply a Lojban denormalizer to make the output more humanly palatable.

Given a normalized Lojban sentence, the construction of an equivalent "Lojban predicate-argument relationship" follows easily from the rules of Lojban grammar; for instance, in the above example one finds:

```
djica( mi, le ( nu( klama( mi, le (zarci) )) ) )
```

If one has a tanru joining two Lojban words (the above example sentence contains no tanru), then one may predicatize this by positing a tanru predicate, so that e.g. the tanru "nixli ckule" ("girls school") becomes

```
tanru(nixli, ckule)
```

Finally, let's return to the subject of language generation. We've already mentioned half of the Lojban language generation process: denormalization. The other half is the process of taking logic expressions and generating normalized Lojban expressions from them. This can be carried out using the grammar rules used for parsing.

3.1 Mapping Predicatized Lojban into AI Knowledge Representations

Now, the mapping of this sort of predicate relationship into semantic structures of use for AI reasoning is going to be different for different AI systems.

For instance, in Novamente

```
le $X
```

maps into the Novamente nodes and links denoted

```
ConceptNode: $Y
InheritanceLink $Y #X
```

where #X is a shorthand for the ConceptNode linked by a WordSenseLink from the WordNode for $X.

On the other hand, in Novamente

```
                         nu $X
```

indicates that $X inherits from Abstract rather than Specific (both of these are particular ConceptNodes).

In all, the above sentence maps into Novamente nodes and links as:

```
EvaluationLink
      #djica
      ListLink $M $Z

Inheritance $Z Abstract

$Z =
HypotheticalLink
      EvaluationLink
            #klama
            ListLink $M $ZZ

Inheritance $ZZ #zarci

Inheritance $ZZ Specific

Inheritance $M CurrentSpeaker
```

This sort of mapping can be carried out by a set of simple transformation rules that are called "semantic mapping schema" in Novamente lingo. Examples of such schema are the above-specified rules for "le" and "nu." The number of schema required is not going to be very large – there is one dealing with selbri/sumti relationships, some dealing with special pragmatic terms like "mi" and so forth, and then a number dealing with particular cmavo. The total number of these semantic mapping schema needed for complete coverage of Lojban semantics will likely be in the range of 100-300.

For example, what about tanru? Inside Novamente, for instance, the tanru "nixli ckule" would be represented initially and generically as follows:

```
AssociativeLink $X nixli
InheritanceLink $Z ckule
```

(a very simple and elegant semantic transformation schema). This says that what we have is a kind of school that is somehow associated with girls. The process of disambiguating this sort of relationship may be subtle and is discussed briefly in Appendix 1 of (Goertzel, 2005).

3.2 Comments on Data Interchange Formats

This section gives some suggestions at the software engineering level, regarding the formatting of information generated at various stages of the parsing process. It seems desirable that the different stages of the parsing process mentioned above should be written to input and output data in XML format. (We also advocate the use of SOX, a simplified XML format which is more human-readable and is easily translatable into standard XML.)

The basic unit of processing, we suggest, should be the Context, which is a set of Lojban sentences. The idea is that sentences may refer to other sentences and terms within the Context, but not to sentences or terms outside the Context. I.e., a Context is assumed referentially closed.

At the start of the processing pipeline a Context just contains a list of sentences; then as it passes through the pipeline it acquires more and more information. For instance, a simple context involving only two sentences might initially look like

```
<context>
        <sentence>
                <text> blah blah </text>
        </sentence>
        <sentence>
                <text> blah blah </text>
        </sentence>
</context>
```

After normalization, it would look like

```
<context>
        <sentence>
                <text> blah blah </text>
                <normalizedText> blah blah </normalizedText>
        </sentence>
        <sentence>
                <text> blah blah </text>
                <normalizedText> blah blah </normalizedText>
        </sentence>
</context>
```

After predicatization, it might look like

```
<context>
        <sentence>
                <text> blah blah </text>
                <normalizedText> blah blah </normalizedText>
                <predicatizedText> blah blah </predicatizedText>
```

```
	</sentence>
</context>
```

A special notation must used for referring to words within the link tags, in the predicatized versions. For instance, the notation

```
_blah_m_n
```

might be used to refer to the n'th occurrence of the word "blah" in the m'th sentence in the context.

4 LojLink

In conjunction with Object Sciences Corp., my company Novamente LLC has created a software system called INLINK (Goertzel et al, 2005), which interacts with the user to help the user enter English language sentences into the Novamente AI system, in a way that ensures Novamente has correctly interpreted the sentences. A simplified and modified version of this framework may be useful for Lojban.

Much of what the English INLINK system does is not needed for Lojban. For instance, word sense disambiguation is not an issue for Lojban, and nor is parse selection a major problem (since Lojban parsing is straightforward). However, there are still significant portions of INLINK functionality that will be very useful as aids for communicating with computers using Lojban. Thus it seems to make sense to create a LojLink system as a form of LojLink. This idea is explored in some detail in (Goertzel, 2005).

One use of LojLink is that humans may not always use Lojban correctly. LojLink could show the user the normalized Lojban version of his input, which could then quickly be stupidity-checked, to be sure that no grammar mistakes were made.

Next, realistically the user is not likely to know all Lojban words by heart. For this purpose, the Lojban-WordNet mapping described in Section 6 below should be very useful. The user will be able to browse WordNet to find the concept he wants, and then see if there is any Lojban word corresponding to this concept or any other closely related concepts. Also, he can do a keyword search of the Lojban dictionary. If none of these yields results, then he can optionally create a new word and store it in the LojLink database for other users to see.

There is also the issue of reference resolution. Lojban reference resolution is simpler than in natural languages but it's still far from unambiguous. INLINK allows the user to explicitly specify referents for words, which allows unambiguous understanding of interreferential sentences, and also gives AI systems data from which to learn how to do reference resolution in a fully automated way.

Finally, LojLink should be a valuable resource for correcting errors in the Lojban-English translation process. The translator, if built along the general principles described in Section 7 below, will often output several possible translations for a given Lojban sentence. The user will be able to rank these in the LojLink interface, and this feedback will allow both human and automated improvements to the translator.

5 Lojban FrameNet

Once Lojban has been parsed according to the above-proposed Lojban link parser, what we have is a collection of syntactic relationships between Lojban words. Now how does this help an AI system to actually understand Lojban?

To a large extent, obviously, this is an AI problem rather than a Lojban problem. But it's possible to create linguistic resources that will be very helpful to a broad range of AI systems in correctly interpreting Lojban texts. Primarily, I suggest a resource that I call Lojban FrameNet. This name is inspired by the English linguistic resource FrameNet (Baker et al, 1998), although the details I propose for Lojban FrameNet are not closely modeled on the English FrameNet, which is considerably more complicated.

The basic idea of Lojban FrameNet is to create a database that contains, for each argument-position of each brivla, an indication of the semantic relationship between the brivla and the sumti in that argument-position. These "indications" are drawn from a fixed vocabulary of relationship-types. Of course, some artistry is required in creating the list of relationship-types. The English FrameNet uses a very large set of relationship types, which is useful for some purposes but not others. On the other hand the SUMO knowledge repository (Niles and Pease, 2001) uses a very small set of relationship types (around a dozen), which is insufficient granularity for most purposes.

In our work on the INLINK project, we have created a linguistic resource called LARDict (Logical Argument Relationship Dictionary), which is a list of a couple hundred relationship types, corresponding to relationships denoted by English prepositions or English subject-argument relationships. LARDict has not yet been placed online but our intention is to do so. We suggest to use the LARDict to create the Lojban FrameNet.

Basically, then, the task of creating the Lojban FrameNet is to associate an entry from the LARDict with each argument position of each brivla.

The easiest way to do this may be to create an ontology of brivla, based on their argument-structure semantics. For instance

- brivla with only one argument, whose argument has the LARDict semantic role subjAgent, can be grouped into the semantic category "Intransitive Actions."

- brivla with only one argument, whose argument has the semantic role subjDescription, can be grouped into the semantic category “Descriptions”
- brivla with two arguments, where the first argument has the semantic role subjAgent and the second argument has the role objPatient, can be grouped into the semantic category “Transitive Actions”

Since all arguments are optional in Lojban, potentially some brivla may belong to multiple semantic categories, none of which are subcategories of each other.

Some brivla may not belong to any higher-level semantic category, but the majority of brivla will fall into a small number of large categories.

An example frame entry for a brivla is as follows. The English-Lojban dictionary entry for “kill” is

```
  kill catra: x1 (agent) |-s/slaughters/murders x2 by action/method x3
```

might look like:

```
kill
arg1: Agent
arg2: Patient
arg3: Process OR Action OR Object
```

Or it could be more compactly represented as

```
kill
Inheritance: TransitiveAction
arg3: Process OR Action OR Object
```

in terms of the supercategory

```
TransitiveAction
arg1: Agent
arg2: Patient
```

In XML format, this might look like:

```
<frame>
<word>kill</word>
<inheritance>TransitiveAction</interaction>
<arg pos=3> Process </arg>
<arg pos=3> Action </arg>
<arg pos=3> Object </arg>
</frame>

<frame>
```

```
<category>TransitiveAction</category>
<arg pos=1> Agent </arg>
<arg pos=2> Patient</arg>
</frame>
```

6 Tools for Aiding Lojban/English Translation

From a sufficiently ambitious AI perspective, creating specific tools for Lojban-English intertranslation is unnecessary. Because if Lojban conversation with humans is successfully used to teach an AI to think at a human level, then this AI will be able to learn English (or any other natural human language) in the same manner that humans learn second languages – without any special tools, just via verbal instruction plus practice and experience.

However, this overambitious perspective is too simplistic. There is a lot of English text out there, and an in-development, progressively-learning AI system certainly has a lot to gain from attempting to digest it. It may well make sense to raise a baby AI as a bilingual English/Lojban speaker. In this case, resources for easy translation between English and Lojban will be extremely valuable. Also, automated translation of English texts into Lojban will be useful for providing Lojban-but-not-English-literate AI's with literature to read. And of course Lojban/English translation resources may be valuable outside the AI context.

Fortunately, the construction of such translation tools seems straightforward, though moderately time-consuming. 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 (Fellbaum, 1998). In almost all cases, each Lojban word corresponds naturally to a single WordNet sense; and in the other cases the correct move is to map the word into a brief English phrase.

Once the Lojban-WordNet mapping is complete, the next steps are a Lojban FrameNet – English mapping, and then concrete translation tools.

6.1 Anglicizing the Lojban FrameNet

Given a Lojban-WordNet mapping, the Lojban FrameNet will actually be a valuable resource for English as well as for Lojban. The Lojban frames will be fairly simply mappable into English in nearly all cases. However, these mappings will have to be checked out by hand, one by one.

E.g. to map the Lojban frame for “kill” given above into English, it suffices to attach part-of-speech information, e.g.

```
POS(kill) = TransitiveVerb
arg1 = Subject
arg2 = Object
```

English syntactic rules then generate a variety of equivalent expressions for

```
kill(arg1, arg2)
```

e.g.

```
arg1 kills arg2
arg2 is killed by arg1
```

and their analogues for different tenses (since English, unlike Lojban, lacks a tense-neutral form).

6.2 Pragmatics of Translation Between Lojban and English

Given the above two mappings (Lojban-WordNet, and Lojban FrameNet - English, automatic translation from Lojban to English will be relatively straightforward (because generating passable though not elegant English from predicate logic expressions is a reasonably well solved problem). The Achilles heel of these translations is going to be Lojban tanru. It should be easy to translate Lojban into a kind of almost-grammatical pidgin English that keeps Lojban tanru intact, e.g.

```
le melbi tavla
```

(melbi=beautiful, tavla=talker) would be translated as

```
the beautiful talker
```

and

```
le tavla be do bei melbi
```

(do=you) would be translated as

```
the (talker to you) beautiful
```

A collection of specialized heuristic rules could be created that would succeed in transforming most but not all commonly constructed tanru into fully grammatical

English. Making typical tanru into *elegant* English would be a lot more difficult, but fortunately is not necessary for AI purposes.

Symmetrically, when English sentences are correctly parsed using some framework like INLINK that outputs semantic node and link structures in predicate-logic-compatible form, the translation of these sentences into Lojban will be relatively straightforward as well. Of course, this is less useful than the reverse process. The main use would seem to be internal and cognitive. If an AI system has learned to think in a Lojban-ish way, then for it to understand English intuitively, its best strategy may be to internally translate the English-derived logic expressions it forms from parsing English into Lojban-ish logic expressions.

In translating English into Lojban, special rules would need to be created to map compound English expressions into tanru in natural ways. These should be simpler than the rules in the opposite direction.

It seems that a passable job of translation in both directions can be done without involving any deep AI reasoning, simply by using the above-described linguistic resources and some heuristic rules. This is quite different from the situation with translating between two natural languages. Of course, the heuristic rules and linguistic resources can be loaded into an AI system and then improved upon via experience.

7 Notes on Project Planning

Finally in this section I'll summarize the various subprojects that have been suggested above and give rough estimates of the amount of work that seems to be involved in each of them.

The projects are divided into two phases:

- Phase 1, which involves creating an infrastructure enabling Lojban to be used to communicate with AI systems
- Phase 2, which consists of creating tools for effective Lojban-English translation

Phase	**Task**	**Rough Time Estimate**
1	Lojban normalizer	1-2 months (for someone who knows a little Lojban as well as linguistics and programming)
1	Lojban predicatization	1-2 months (for a good linguist)
1	Lojban link parser	2 months (for a good

		programmer with linguistics knowledge); 1 month if the Novamente parser is customized
1	Lojban de-normalizer	1 month (programmer w/ linguistics knowledge)
1	Lojban FrameNet	2-3 months (by someone with moderate linguistics knowledge)
1	LojLink	2 months (for a member of the INLINK team)
2	Lojban WordNet for cmavo and gismu	1 month (could be done by a bright high school student)
2	Lojban WordNet for lujvo	4 months (ditto)
2	Lojban FrameNet – English mapping	6 months (ditto)
2	Lojbanish – Englishish Logical Expression Mapper	3 months for simple version (for someone expert in Lojban, linguistics & programming)

According to these estimates, for highly competent and appropriately trained individuals, Phase 1 is about 11 months of work altogether, and Phase 2 is about 14 more months.

Furthermore, Phase 1 is parallelizable in that the Lojban FrameNet can be built simultaneously with the other Phase 1 tools, and the Lojban link grammar can be built in parallel with the normalizer/denormalizer. So really Phase 1 can be completed in 4 months by 3 people.

Phase 2 is even more highly parallelizable and could be completed in probably 3-4 months by a team of 4 people.

To minimize management and coordination difficulties, the ideal team for this project is probably one programmer, one linguist, and 2-3 linguist's assistants to help with the WordNet and FrameNet mappings. With this team the project could be wrapped up in probably 9 months or so – an amazingly short amount of time for a comprehensive computational-linguistic treatment of a language, but of course this is due to the fact that Lojban was specifically designed for amenability to mathematical and computational analysis.

Of course, the project of creating computational linguistics tools and resources for Lojban would not end with the completion of these initial projects – the use of these tools and resources will doubtless suggest a host of improvements to the tools and resources, and potentially improvements to the language itself.

References

- Baker, Collin F., Fillmore, Charles J., and Lowe, John B. (1998): The Berkeley FrameNet project. In *Proceedings of the COLING-ACL*, Montreal, Canada.
- Doran Christy, B. Srinivas, Beth Ann Hockey, Martin Zaidel and Dania Egedi. (1994). XTAG System - A Wide Coverage Grammar for English , Coling '94
- Goertzel, Ben (2004). Post-Embodied AI, Frontier Number Four , www.frontiernumber4.com
- Goertzel, Ben (2005). LojLink: A Novel Approach to Knowledge Management
- Goertzel, Ben, Moshe Looks, Michael Ross, Cassio Pennachin and Hugo Pinto (2005). NL Comprehension via Integrative AI and Human-Computer Interaction, submitted for publication
- Looks, Moshe, Ben Goertzel and Cassio Pennachin (2004). Novamente: An Integrative Architecture for Artificial General Intelligence. *Proceedings of AAAI Symposium on Achieving Human-Level Intelligence through Integrated Systems and Research*, Washington DC, August 2004
- Fellbaum, Christine (1998). WordNet: An Electronic Lexical Database, MIT Press.
- Nicholas, Nick and John Cowan (2003). *What Is Lojban?* , The Logical Language Group
- Niles, I., and Pease, A. 2001. Towards a Standard Upper Ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems* (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine, October 17-19, 2001.
- Powell, Robin Lee (date unspecified). Issues with the Lojban Formal Grammar. Online at http://www.digitalkingdom.org/~rlpowell/hobbies/lojban/grammar/
- Sleator, Daniel and Davy Temperley. Parsing English with a Link Grammar. *Third International Workshop on Parsing Technologies*, Tilburg, The Netherlands, 1993