A Model-Based Multilingual Natural Language Parser — Implementing Chomsky's X-bar Theory in ModelCC

Luis Quesada, Fernando Berzal, and Juan-Carlos Cubero

Department of Computer Science and Artificial Intelligence, CITIC,
University of Granada, Granada 18071, Spain
{lquesada,fberzal,jc.cubero}@decsai.ugr.es

Abstract. Natural language support is a powerful feature that enhances user interaction with query systems. NLP requires dealing with ambiguities. Traditional probabilistic parsers provide a convenient means for disambiguation. However, they incorrigibly return wrong sequences of tokens, they impose hard constraints on the way lexical and syntactic ambiguities can be resolved, and they are limited in the mechanisms they allow for taking context into account. In comparison, model-based parser generators allow for flexible constraint specification and reference resolution, which facilitates the context consideration. In this paper, we explain how the ModelCC model-based parser generator supports statistical language models and arbitrary probability estimators. Then, we present the ModelCC implementation of a natural language parser based on the syntax of most Romance and Germanic languages. This natural language parser can be instantiated for a specific language by connecting it with a thesaurus (for lexical analysis), a linguistic corpus (for syntax-driven disambiguation), and an ontology or semantic database (for semantics-driven disambiguation).

Keywords: Natural languages, disambiguation, query parsing.

1 Introduction

Lexical ambiguities occur when an input string simultaneously corresponds to several token sequences [10], which may also overlap. Syntactic ambiguities occur when a token sequence can be parsed into several parse trees [7]. A common approach to disambiguation consists of performing probabilistic scanning (i.e. probabilistic lexical analysis) and probabilistic parsing (i.e. probabilistic syntactic analysis), which assign a probability to each possible parse tree. However, existing techniques for probabilistic scanning and parsing present several drawbacks: probabilistic scanners may produce incorrect sequences of tokens due to wrong guesses or to occurrences of words that are not in the lexicon, and probabilistic parsers cannot consider relevant context information such as resolved references between language elements, and both probabilistic scanners and parsers

impose hard constraints on the way lexical and syntactic ambiguities can be resolved.

Model-based language specification techniques [8] decouple language design from language processing. ModelCC [13,12] is a model-based parser generator that includes support for dealing with references between language elements and, thus, instead of returning mere abstract syntax trees, ModelCC is able to obtain abstract syntax graphs and consider lexical and syntactic ambiguities.

In this paper, we explain how ModelCC supports probabilistic language models and we present the implementation of a natural language parser. Section 2 provides an introduction to probabilistic parsing techniques and to the model-based language specification techniques employed by the ModelCC parser generator. Section 3 explains the probabilistic model support in ModelCC. Section 4 describes the implementation of a natural language parser. Finally, Section 5 presents our conclusions and pointers for future work.

2 Background

In this section, we provide an analysis of the state of the art on probabilistic parsing and on model-based language specification.

2.1 Probabilistic Parsing

There are many approaches to part-of-speech tagging using probabilistic scanners and for language disambiguation using probabilistic parsers.

Probabilistic scanners based on Markov-like models [9] consider the existence of implicit relationships between words, symbols or characters found close in sequences, and irrevocably guess the type of a lexeme based on the preceding ones. When using such techniques, a single wrong guess renders the whole parsing procedure irremediably erroneous, as no correct parse tree that uses a wrong token can be found.

Probabilistic scanners based on lexicons [7] assign probabilities to a lexeme belonging to different word classes from the statistical analysis of lexicons. Scanning a lexeme that belongs to a particular word class but never belonged to that class in the training lexicon provides wrong scanning results, which, in turn, render the whole parsing procedure useless.

Probabilistic parsers [11] compute the probability of different parse trees by considering token probabilities and grammar production probabilities, which are empirically obtained from the analysis of linguistic corpora. The probability of a symbol is defined as the product of the probability of the grammar rule that produced the symbol and the probabilities of all the symbols involved in the application of that rule. The probability of a parse tree is that of its root symbol. These techniques do not take context into account.

Probabilistic lexicalized parsers [4,2] associate lexical heads and head tags to the grammar symbols. Grammar rules are then decomposed and rewritten to include the different combinations of symbols, lexical heads, and head tags.

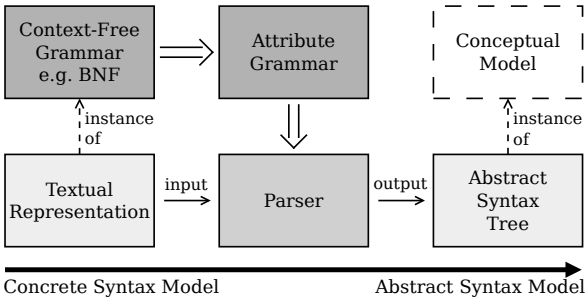

Fig. 1: Traditional language processing.

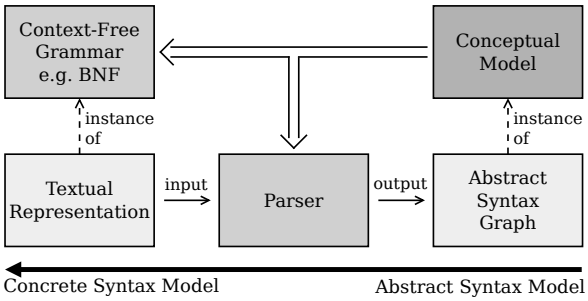

Fig. 2: Model-based language processing.

Different probabilities can be associated to each of the new rules. When using this technique, the grammar significantly expands and a more extensive analysis of linguistic corpora is needed to produce accurate results. It should be noted that this technique is not able to consider relevant context information such as resolved references between language elements.

Conventional probabilistic scanners and parsers do not allow the use of arbitrary probability estimators or statistical models that take advantage of more context information.

2.2 Model-Based Language Specification

In its most general sense, a model is anything used in any way to represent something else. In such sense, a grammar is a model of the language it defines. The idea behind model-based language specification is that, starting from a single abstract syntax model (ASM) that represents the core concepts in a language, language designers can develop one or several concrete syntax models (CSMs). These CSMs can suit the specific needs of the desired textual or graphical representation for language sentences. The ASM-CSM mapping can be performed, for instance, by annotating the abstract syntax model with the constraints needed to transform the elements in the abstract syntax into their concrete representation.

A diagram summarizing the traditional language design process is shown in Figure 1, whereas the corresponding diagram for the model-based approach is

Constraints on...	Annotation	Function
Patterns	@Pattern	Pattern definition for basic language elements.
	@Value	Field where the input element will be stored.
Delimiters	@Prefix	Element prefix(es).
	@Suffix	Element suffix(es).
	@Separator	Element separator(s).
Cardinality	@Optional	Optional elements.
	@Minimum	Minimum element multiplicity.
	@Maximum	Maximum element multiplicity.
Evaluation order	@Associativity	Element associativity (e.g. left-to-right).
	@Composition	Eager or lazy composition for nested composites.
	@Priority	Element precedence level/relationships.
Composition order	@Float	Element member position may vary.
	@FreeOrder	All the element members positions may vary.
References	@ID	Identifier of a language element.
	@Reference	Reference to a language element.

Table 1: Summary of the metadata annotations supported by ModelCC.

shown in Figure 2. It should be noted that ASMs represent non-tree structures whenever language elements can refer to other language elements, hence the use of the 'abstract syntax graph' term.

ModelCC [13,12] is a parser generator that supports a model-based approach to the design of language processing systems. Its starting ASM is created by defining classes that represent language elements and establishing relationships among those elements. Once the ASM is created, constraints can be imposed over language elements and their relationships as the metadata annotations [6] shown in Table 1 in order to produce the desired ASM-CSM mappings.

Although probabilistic language processing techniques and model-based language specification have been extensively studied, to the best of our knowledge, there are no techniques that allow model-driven probabilistic parsing. In the next section, we explain ModelCC's support for probabilistic language models.

3 Probabilistic Parsing in ModelCC

ModelCC combines model-based language specification with probabilistic parsing by allowing the specification of arbitrary probabilistic language models.

Subsection 3.1 introduces ModelCC support for probabilistic language models and presents ModelCC's *@Probability* annotation. Subsection 3.2 discusses the use of contextual information in ModelCC probabilistic parsers. Subsection 3.3 explains how symbol probabilities are computed.

3.1 Probabilistic Language Models

ModelCC's *@Probability* annotation allows the specification of probability values for language elements and language element members. Probability values can be

specified for syntactic elements of the languages and for lexical components, in which case it should be noted that the lexical analyzer behaves as a part-of-speech tagger in natural language processing.

Such probability values can be specified using three alternatives: a probability value as a real number between 0 and 1, a frequency as an integer number, or a custom probability evaluator that computes the probability value from the analysis of the language element and its context.

Since ModelCC supports lexical and syntactic ambiguities and the combination of language models, one of the main novelties of ModelCC with respect to existing techniques is that it allows the modular specification of probabilistic languages, that is, it is able to produce parsers from composite language specifications even when some of the language elements overlap or conflict.

ModelCC also supports alternative models for the representation of uncertainty (e.g. possibilistic models, models based on Dempster-Shafer theory, or any other soft computing models), provided that an evaluation operator for language element instances is provided and an evaluation operator for the application of grammar rules is provided. Optionally, a casting operator that translates the estimated value in one model into a value valid for a different kind of model allows the specification of modular languages even when different mechanisms for representing natural language ambiguities are employed for different parts of the language model.

3.2 Context Information

ModelCC provides context information that custom probability evaluators and constraints can take into account when processing a language element.

The context information includes the current syntax graph and the parse graph symbol corresponding to the language element being evaluated. If the language element instance is a reference, the context information also includes the referenced language element instance, its corresponding parse graph symbol, and the context graph, which is the smallest graph that contains both the reference and the referenced object.

It should be noted that, from this information, it is possible to compute traditional metrics such as the distance between the reference and the referenced object in the input or in the syntax graph and whether the reference is anaphoric, cataphoric, or recursive.

However, in contrast to existing probabilistic parsing techniques, ModelCC also allows the specification of complex syntactic constraints, semantic constraints, and probability evaluators that use extensive context information such as resolved references between language elements.

3.3 Probability Evaluation

The probability of a particular parse graph G for a sentence $w_{1:m}$ of length m is defined as the product of the probabilities associated to the n instances of language elements E_i in the parse graph G:

$$P(G|w_{1:m}) = \prod_{i=1}^{n} P(E_i|w_{s_i:e_i}) \tag{1}$$

Given a language element E that represents a part-of-speech tag and a word w, the lexical analyzer acts as a POS tagger and provides $P(E|w)$.

Given a language element E with $M_1..M_n$ members in its definition, some of which are optional, the probability $P(E|M_{1:n})$ is computed as follows. Let $OPT(E)$ be the set of optional elements for E. Assuming that their appearance is statistically independent, we can estimate the probability of E given its observed elements O:

$$P(E|O_{1:k}) = P(E) \prod_{\substack{M_i \in OPT(E),\\ M_i \in O_{1:k}}} P(M_i|E) \prod_{\substack{M_j \in OPT(E),\\ M_j \notin O_{1:k}}} (1 - P(M_j|E)) \tag{2}$$

Given an ambiguous sentence $w_{1:n}$, its disambiguation is done by picking the parse graph $\hat{G}$ with the highest probability for that sentence:

$$\hat{G}(w_{1:m}) = \arg\max_{G} \{P(G|w_{1:m})\} \tag{3}$$

We now present the ModelCC implementation of a natural language parser.

4 A Natural Language Parser

In this section, we present a model-based specification for a probabilistic natural language parser. Subsection 4.1 outlines the natural language features. Subsection 4.2 provides the ModelCC ASM specification of the natural language. Subsection 4.3 explains how the natural language can be instantiated. Subsection 4.4 presents a sample English language parser.

4.1 Natural Languages in Terms of Chomsky's X-bar Theory

Chomsky's X-bar theory [3] claims that certain human languages share structural similarities. Our language supports this theory by comprehending these common structures, which we proceed to explain.

In our model, a sentence consists of a clause (i.e. a complete proposition), a clause can be either a simple clause or the coordinate clause composite that creates a compound sentence, a simple clause consists of an optional nominal phrase and a verbal phrase, and a coordinate clause composite consists of a set of clauses and an optional floating coordinating conjunction.

In our natural language model, a complement is a phrase used to complete a predicate construction, and a head is a complement that plays the same grammatical role as the whole predicate construction.

Our natural language model supports nominal, verbal, adverbial, adjectival, and prepositional complements.

Nominal complements comprise nominal phrases, nominal composites, and nominal clauses. A nominal phrase consists of an optional determiner, a noun, and an optional set of complements. A nominal composite consists of an optional determiner, a set of nominal complements and an optional floating conjunction. A nominal clause consists of an optional determiner, an optional subordinating conjunction and a subordinate clause. Nouns comprise common nouns, proper nouns, and pronouns. Pronouns, in turn, reference nouns and proper nouns.

Verbal complements comprise verbal phrases and verbal composites. A verbal phrase consists of a set of floating verbs and an optional floating preposition. A verbal composite consists of a set of verbal complements and an optional floating conjunction.

Adverbial complements comprise adverbial phrases, adverbial composites, and adverbial clauses. An adverbial phrase consists of an adverb. An adverbial composite consists of a set of adverbial complements and an optional floating conjunction. An adverbial clause consists of an optional subordinating conjunction and a subordinate clause.

Adjectival complements comprise adjectival composites and adjectival clauses. An adjectival composite consists of a set of adjectival complements and an optional floating conjunction. An adjectival clause consists of an optional subordinating conjunction and a subordinate clause.

Prepositional complements comprise prepositional phrases and prepositional composites. A prepositional phrase consists of a floating preposition and a head. A prepositional composite consists of a set of prepositional complements and an optional floating conjunction.

It should be noted that this natural language embraces Romance languages such as Spanish, Portuguese, French, and Italian, as well as Germanic languages such as English and German.

4.2 Specification of the Natural Language ASM

In order to implement our natural language parser using ModelCC, we first provide a specification of the language ASM as a set of UML class diagrams shown in Figures 3, 4, 5, 6, and 7. Adjectival complements and adverbial complements can be specified in a manner similar to nominal complements. As it can be observed from the figures, the model-based specification of the natural language matches the language description.

As the specified model is an abstract syntax model, it does not correspond to any particular language. The ASM is more like the Mentalese language postulated by the Language Of Thought Hypothesis [5]. In the next subsection, we explain how different fully-functional natural language parsers can be instantiated from this model by defining additional language-specific constraints.

4.3 Specification of the Natural Language CSM

In order to implement a parser for a particular natural language, the ASM-CSM mapping has to be specified. A pattern matcher is assigned to each lexical

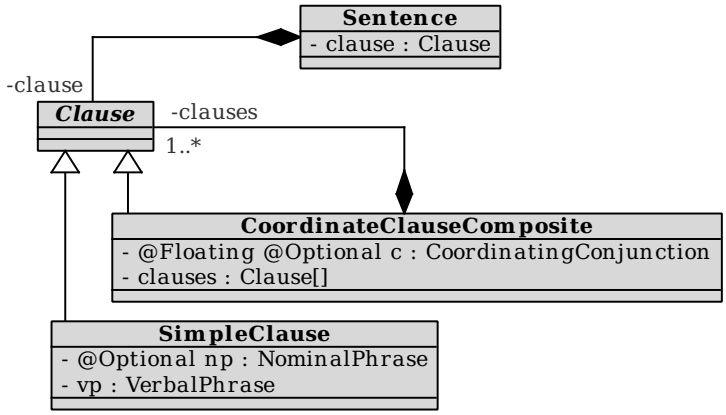

Fig. 3: ModelCC specification of the sentence and clause elements of our natural language.

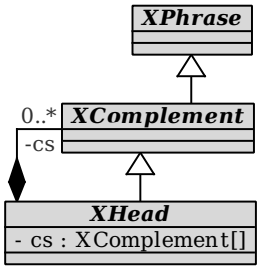

Fig. 4: ModelCC specification of the phrase, head, and complement elements of our natural language.

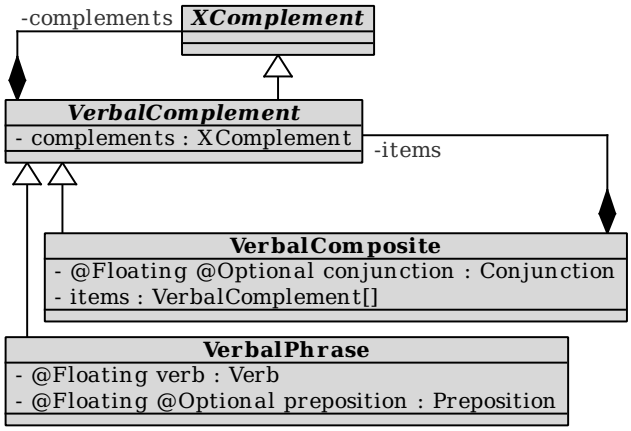

Fig. 5: ModelCC specification of the verbal complement language elements in our natural language.

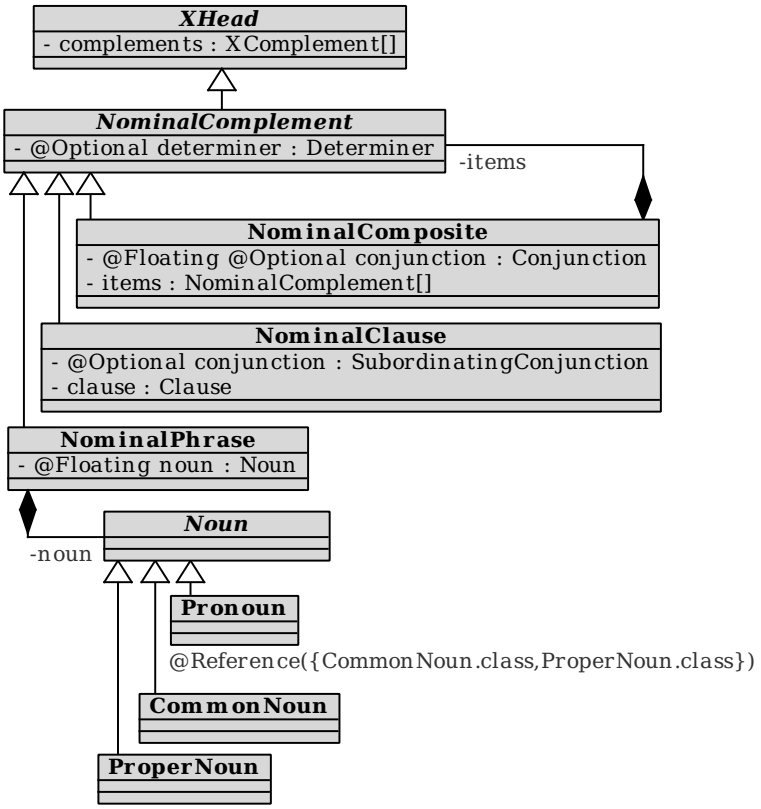

Fig. 6: ModelCC specification of the nominal complement language elements in our natural language.

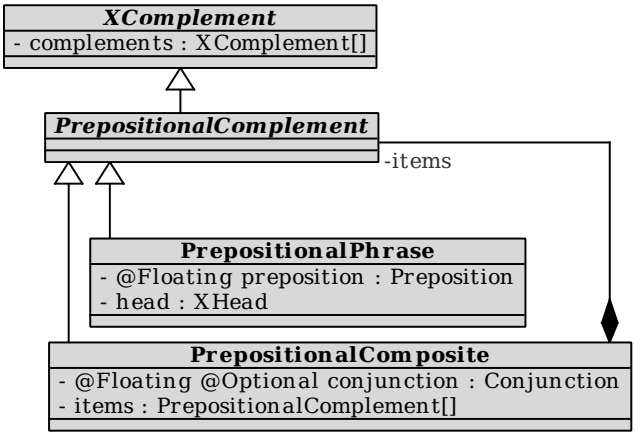

Fig. 7: ModelCC specification of prepositional complement language elements in our natural language.

component of the language model. For this purpose, ModelCC's *@Pattern* annotation allows the specification of custom pattern matchers that can consist of regular expressions, dictionary lookups, or any suitable heuristics. Such pattern matchers can easily be induced from the analysis of lexicons.

ModelCC supports lexical ambiguities apart from syntactic ambiguities, so the specified pattern matchers can produce different, and even overlapping, sets of tokens from the analysis of the input string.

After specifying the pattern matchers for the lexical components of the language, language-specific constraints are imposed on syntactic components of the language model. For this purpose, ModelCC's *@Constraint* annotation allows the specification of methods that evaluate whether a language element instance is valid or not. These constraints can be automatically induced from the analysis of linguistic corpora and, as explained in Subsection 3.2, these constraints can take into account extensive context information, which can even include resolved references between language elements.

Finally, in order to produce a probabilistic parser, probability evaluators are assigned to the different language constructions. For this purpose, ModelCC's *@Probability* annotation allows the specification of the probability evaluation for language elements. These probabilities can also be estimated from the analysis of linguistic corpora, although heuristics could also be used.

4.4 Example: Parsing an English Sentence

We have implemented an English parser by specifying an ASM-CSM mapping from the language ASM.

We have defined pattern matchers that query *wiktionary.org* to perform the lexical analysis. We have approximated probability values derived from the analysis of the Google n-gram datasets to different lexemes and constructions.

As an example, we have parsed the sentence "I saw a picture of New York". The lexical analysis graph for this sentence represents 192 valid token sequences and is shown in Figure 8.

It should be noted that some of the tokens produced by the *wiktionary.org*-based pattern matcher may not be intuitive, but they are indeed valid. For example, the "I" proper noun refers to the ego, and the "New York" adjective refers to a style, particularly of food, originating in New York.

A set of valid parse graphs is then obtained from this lexical analysis graph. The graph in Figure 9 represents the intended meaning, which implies that the speaker did see a photography of New York city at some point in the past. Apart from this meaning, others such as the speaker using a saw to cut a photography of New York in half or the speaker having seen a photography of a recently founded York city can also be found in the valid parse graph set.

The application of probability evaluators deducted from the analysis of linguistic corpora allows the parser pinpointing the intended meaning.

Furthermore, there exist multiple language thesauri annotated with specific semantic knowledge such as the 20 question game [1] database, which have information on whether a word refers to an animal, vegetable, mineral or other

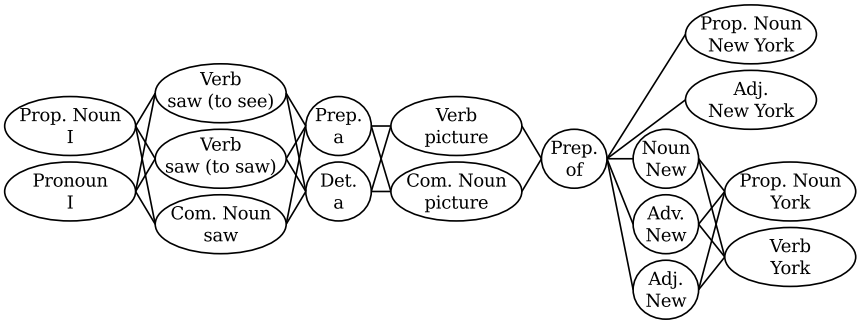

Fig. 8: Lexical analysis graph for the sentence "I saw a picture of New York".

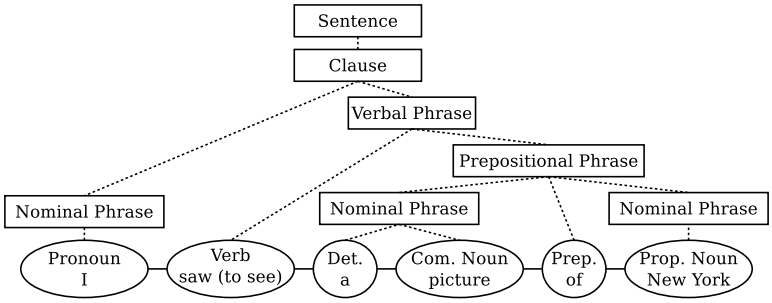

Fig. 9: Correct parse graph for the sentence "I saw a picture of New York".

(such as specific actions, processes, or abstract concepts), on whether if a word refers to real or imaginary concepts, and on the look and feel or effect of most concepts. Such a database can be combined with our language parser by means of probability evaluators to figure out semantic constraints that greatly narrow the valid parse graph set by discarding absurd cases.

A specific ontology designed for a query answering system can also be used to fine tune the natural language parser for a particular application. For example, let us consider the transitive verb "to rub" and the direct object "table". Since the verb implies a physical effect on the direct object and the direct object has several meanings, such as a piece of furniture or as a data matrix, the latter interpretation is discarded as it cannot undertake a physical effect.

5 Conclusions and Future Work

Lexical and syntactic ambiguities are always present in natural languages. NLP-based query answering systems require dealing with ambiguities. A common approach to disambiguation consists of performing probabilistic scanning and probabilistic parsing. Such techniques present several drawbacks: they may produce wrong sequences of tokens, they impose hard constraints on the way ambiguities can be resolved, and they only take advantage of small amounts of context information.

ModelCC is a model-based parser generator that supports lexical ambiguities, syntactic ambiguities, the specification of constraints, and reference resolution. ModelCC solves the aforementioned drawbacks of existing techniques.

In this paper, we have described ModelCC support for probabilistic language models. We have also described the ModelCC implementation of a natural language parser, and we have provided an English-language instantiation of it.

In the future, we plan to research on the automatic induction of probabilistic language models, syntactic constraints, and semantic constraints.

Acknowledgments

Work partially supported by research project TIN2012-36951, "NOESIS: Network-Oriented Exploration, Simulation, and Induction System".

References

1. 20q. http://www.20q.net.
2. E. Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proc. AAAI'97*, pages 598–603, 1997.
3. N. Chomsky. *Remarks on nominalization. R. Jacobs and P. Rosenbaum (eds.), Readings in English Transformational Grammar*, pages 184–221. 1970.
4. M. Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637, 2003.
5. J. A. Fodor. *The Language of Thought*. Crowell Press, 1975.
6. M. Fowler. Using metadata. *IEEE Software*, 19(6):13–17, November 2002.
7. D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2nd edition, 2009.
8. A. Kleppe. Towards the generation of a text-based IDE from a language metamodel. volume 4530 of *Lecture Notes in Computer Science*, pages 114–129, 2007.
9. A. A. Markov. *Dynamic Probabilistic Systems (Volume I: Markov Models). Howard, R. (ed.), Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain*, pages 552–577. John Wiley & Sons, 1971.
10. J. R. Nawrocki. Conflict detection and resolution in a lexical analyzer generator. *Information Processing Letters*, 38(6):323–328, 1991.
11. H. Ney. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2):336–340, 1991.
12. L. Quesada. A model-driven parser generator with reference resolution support. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, pages 394–397, 2012.
13. L. Quesada, F. Berzal, and J.-C. Cubero. A language specification tool for model-based parsing. In *Proceedings of the 12th International Conference on Intelligent Data Engineering and Automated Learning. Lecture Notes in Computer Science*, volume 6936, pages 50–57, 2011.