Lexical Inference Mechanisms for Text Understanding and Classification

Elizabeth Figa
School of Library and Information Sciences
University of North Texas
P.O. Box 311068
Denton, Texas 76203
E-mail: efiga@lis.admin.unt.edu
WWW: http://courses.unt.edu/efiga

Paul Tarau
Department of Computer Science
University of North Texas
P.O. Box 311366
Denton, Texas 76203
E-mail: tarau@cs.unt.edu
WWW: http://www.cs.unt.edu/∼tarau

Abstract. This paper describes a framework for building *story traces* (compact global views of a narrative) and *story projections* (selections of key elements of a narrative) and their applications in text understanding and classification. Word and sense properties are extracted using the WordNet lexical database enhanced with Prolog inference rules and a number of lexical transformations. Inference rules are based on navigation in various WordNet relation chains (hypernyms, meronyms, entailment and causality links, etc.) and derived relations expressed as boolean combinations of node and edge properties used to direct the navigation. The resulting *abstract story traces* provide a compact view of the underlying narrative's key content elements and a means for automated indexing and classification of text collections. *Ontology driven projections* act as a kind of "semantic lenses" and provide a means to select a subset of a narrative whose key sense elements are subsumed by a set of concepts, predicates and properties expressing the focus of interest of a user. Finally, we discuss applications of these techniques in text understanding, classification of text collections and answering questions about a text.

1 Introduction

Natural language communication patterns present in a narrative usually bring together emergent properties beyond their underlying morphologic, syntactic and semantic building blocks. Abstraction mechanisms, based on implicit information provided by semantic and syntactic properties of the lexical material of narrative can be used to manage the complexity of the underlying ontology.

This paper describes our experiments in extracting such abstract skeletons from stories (understood here as anything from a news headline to a detailed possibly higly analytical description of a series of events). Our abstract skeletons are helpful in determining what a given story is *about* – as well as the *dynamics* of stories – the chains of temporal transitions stories describe. At an abstract level, stories share experiences in interpreting events and changes in time with focus on explanation and prediction. As such, they are improvised, ad-hoc *theories* trying to convey highly condensed forms of reusable experience. The skeletons we extract summarize this experience in the form of patterns and traces – usable for understanding and classifying them, and answering queries about them. Our skeletons – called abstract story traces – summarize a story as a sequence of abstractions derived from its key concepts, predicates and properties. In addition to the abstract story traces, we also need the ability to automatically focus on the lexical text elements (sentences or paragraphs) which are the most relevant with respect to a fixed ontology – a set of concepts, predicates and properties of interest to a given human or computer agent. We call these "semantic lenses" *ontology driven projections*. They provide a new form of dynamic computer assisted reading of a large corpora of narratives, because they zoom in upon (and "bring close") the objects of interest. This becomes particularly important in text mining, in automated text indexing and text classification.

2 Refactoring WordNet as a Lexical Knowledge Processor

The first step in building this functionality is quick access to the common sense knowledge provided by the WordNet database [1]. This database is also available in Prolog form (see `http://www.cogsci.princeton.edu/~wn`) and is therefore ready to be used as part of a rule-based inference system.

A lexicon consists of a set of word meanings and their semantic relationships. A systematic representation of the English lexicon based in psycholinguistic considerations has been put together in the database WordNet [2].

In a long-term collaborative effort that began in 1985, an interdisciplinary group at Princeton developed WordNet as a "machine readable lexical database organized by meanings". WordNet is not a dictionary – dictionaries help to settle issues of word use or sense priority – but they do not address issues of synchronic organization of lexical knowledge as WordNet does [1]. WordNet divides the lexicon into five categories: *nouns, verbs, adjectives, adverbs* (which are connected by various semantic and lexical realtions), as well as *functions words* (which are not included, as they are assumed part of the syntactic component of the language).

Each of these lexical structures reflects a different way of categorizing experience. WordNet organizes lexical information in terms of word meanings rather than word forms in a way similar to a thesaurus.

WordNet maps word forms and word meanings as a many-to-many relation, which indicates that some forms have several different meanings, and some meanings can be expressed by different forms. Two difficult problems of lex-

icography, polysemy and synonymy, can be viewed as complementary aspects of this mapping. WordNet itself defines *polysemy* as the ambiguity of individual words or phrases that can be used in different contexts to express two or more different meanings and *synonymy* as the semantic relation that holds between two words that can (in a given context) express the same meaning. The common sense opposite of synonymy is antonymy. There are other semantic relationships (and their reverses) covered by WordNet: hypernymy/hyponymy and meronymy/holonomy. A hyponym is a meaning that acquires all the features of its hypernym, which is a more generic concept; for example, oak is a hyponym of tree. Meronymy is the relation of being part of; for example, arm is a meronym of body.

These relations are defined in WordNet between *meanings* instead of being defined beetween *words or word phrases*. Meanings are represented by *integers* called *synsets*) associated to sets of words and word phrases collectively defining a sense element (concept, predicate or property and also usable for indexing.

So, for example, the meaning identifier (synset) `Id=100011413` maps to the following list of words and word phrases: `[[animal], [animate,being], [beast], [brute], [creature], [fauna]]`, which collectively define a common *meaning*.

2.1 WordNet for Efficient Bidirectional Word Phrase to Meaning Mapping

We have refactored the set of predicates provided by WordNet closely following the WordNet relation set (see `http://www.cogsci.princeton.edu/$\sim$wn/doc.shtml`) to support bidirectional constant time access to the set of *meanings* associated to a given word phrase (indexed by a unique head word) and for the set of word phrases and relations associated to a given (unique) meaning.

The refactored WordNet Prolog database contains the following basic binary relations.

- `w/2`: head word to meanings
- `i/2`: meaning to words
- `l/2`: meaning to meaning links (relations from the meaning to another meaning)
- `g/2`: meanings to definitions and examples
- `r/2`: reversed meaning to meaning links
- `t/2`: toplevel meanings with their syntactic roles (nouns or verbs)
- `e/2`: exception variant words to meanings
- `k/2`: known words in definitions and examples with occurrence counts
- `n/2`: new words in definitions and examples with occurrence counts

For a given word we have a list of meanings:

```
?- w(humble,Meanings).
Meanings=[
```

```
  302269648,201415418,301839431,
  201414096,302162242,301551117,109699111
).
```

For a meaning, we have a number of alternative words or word phrases, with attributes, among which the last one provides frequency of occurence in a corpus of texts (useful for disambiguation):

```
?-i(302269648,WordInfo).

WordInfo=[
  f(1,[humble],s,1,3),
  f(2,[low],s,7,0),
  f(3,[lowly],s,1,1),
  f(4,[modes],s,5,0),
  f(5,[small],s,3,155)]).
```

Links (like the sim/1 synonymy link) are collected on a list:

```
?-l(302269648,Links).
  Links=[sim(302269385)]).
```

Definitions and examples originally present in WordNet are preparsed so that they can be processed efficiently, if needed, at runtime.

```
?-g(302269648,DefinitionAndExamples).

DefinitionAndExamples=[
  def([low,or,inferior,in,station,or,quality]),
  ex([a,humble,cottage],
  ex([a,lowly,parish,priest]),
  ex([a,modest,man,of,the,people]),
  ex([small,beginnigs])]).
```

Note that multiple syntactic categories can be present for words like “humble” (v=verb, and a=adverb).

```
i(201415418,[f(1,[humble],v,1,1)])
...
i(301839431,[f(1,[humble],a,2,1)])
...

i(201414096,[
  f(1,[humiliate],v,1,2),
  f(2,[mortify],v,3,0),
  f(3,[chagrin],v,1,0),
  f(4,[humble],v,2,1),
  f(5,[abase],v,1,0)]).
```

Note also the presence of reversed relations like (hyponyms=reverse hypernyms) and reverse meronyms which are precomputed to support high performace graph walk operations using BinProlog [3] and Jinni 2002's [4] built-in database indexing mechanisms, to provide constant time access to edges related to a given node for a given relation.

For instance, her is an example for the reverse hypernymy relation hyp/1):

```
r(201414096,[hyp([201414281,201414446])])
```

We have also precomputed mappings from word variants to related meanings, based on the dictionary entry they belong.

```
e(bagging,[
  []/201170312,[]/201175343,
  []/202141610,[]/201805468,
  []/202139264]).
```

Finally, we have precomputed "toplevel" nouns and verbs, (meaning which do not have further hypernym links).

```
t(100001742,noun).
i(100001742,[
  f(1,[entity],n,1,11),
  f(2,[physical,thing],n,1,0)
]).

t(200001742,verb).
i(200001742,[
    f(1,[breathe],v,1,25),
    f(2,[take,a,breath],v,1,3),
    f(3,[respire],v,30),
    f(4,[suspire],v,2,0)]
).
```

This refactoring provides sets of facts with the following properties:

- given a head word (or a word phrase), we can extract in constant time all related information about their meanings (used in efficient parsing of textual data to lists of synsets)
- given a meaning, we can extract in constant time all relations and related meanings or words provided by the WordNet database

Overall, our refactoring simplifies WordNet while providing an efficient inference engine through Prolog rules that can digest the information contained in these basic relations. WordNet is extracted from a set of lexicographer files which put together a variety of different views on how meanings are organized. The level of precision it provides is statistical in nature. We will reduce some of this ambiguity by lifting to more general concepts following hypernym links (going "up" in the WordNet hierarchy via hypernyms, entailments and/or causality links) – or by selecting subtraces from a given ontology using hyponym and meronym links of subsumed concepts.

2.2 From Words and Phrases to Meanings

WordNet provides a many-to-many relationship from words and word phrases to their independent senses called *synsets*. The first step after tokenizing and splitting a story in sentences is to compute the list of meanings associated to various words and word phrases. Although we extract maximal word phrases from a given story, the attached meaning lists contain multiple synsets – as a result of lexical ambiguity.

```
?- tokens2synsets([the,story,goes,on],Is).
Is=[
 [the]-[],
 [story]-[102932667,105358470,105455815,105590213,105650568,106020196],
 [goes,on]-[200271654,200621853,201209119,201565591,202113765]
]
```

2.3 Using WordNet Definitions and Examples as a Training Set

As part of the refactoring process, we have parsed the WordNet definitions of meanings (synsets) as well as their usage examples. They provide a fairly large corpora of natural language text usable for learning properties of sentences ranging from word frequencies to synset disambiguation hints [5, 6].

2.4 Using depth-K graph walks

In the directed graph G of meaning nodes N_G we can see various WordNet relations as elements of a set of edges E_G. We implement our abstraction operators as depth-K walks in such graphs. Besides the primitive WordNet relations, we use *closures*[1] as edge generators. Such closures are used, for instance, to generate edges leading to synsets which are hyponyms of a given set of synsets – to project a view of a story restricted to a given ontology.

3 Story Abstractions

The view of stories as *programs* usable to *replay* or understand chains of external events suggests a technique similar to Abstract Interpretation [7] which has been used to elegantly infer program properties in the field of Programming Languages. The technique consists of lifting properties from a *concrete domain* – in this case the lexical material of a story, seen as a sequence of word phrases – to an *abstract domain* obtained by following an abstraction operator and then propagating back the results of the (easier) analysis from the (smaller) abstract domain to the more complex concrete domain.

[1] Our closures are predicate name+argument combinations, which receive two graph nodes as extra arguments to make-up a callable predicates.

3.1 Abstraction through the WordNet Hypernym Hierarchy

This is a simple but useful abstraction mechanism – with obvious applications to indexing. It consists of *lifting* words and word phrases extracted from the story transcripts to more general equivalents obtained by following upward links in the WordNet hypernym hierarchy – as in the following example:

```
?- lift_word(spy,1,S).
S=[[[secret,agent],
   [intelligence,officer],
   [intelligence,agent],
   [operative],...]];

no
?- lift_word(spy,2,S).
S=[[[perceiver],[observer],[beholder],[agent],...]]
];

no
?- lift_word(spy,3,S).
S=[[[person],[individual],[someone],[somebody],[mortal],[human],...]];
```

3.2 K-level Noun Abstractions

Hypernym relations are more meaningful for nouns than for other syntactic categories. Noun-related synsets form relatively deep (up to 10-12) hierarchies coming from fairly reliable common sense and natural sciences classifications. By restricting a story trace to nouns, one can get an approximation of what the story is about – at different levels of abstractions.

3.3 Verb Abstractions

Pure verb traces (obtained by selecting only verb sequences) provide an abstract view of a story's dynamics. Like Web links, WordNet graphs exhibit *small-world* structure (strong clustering and small diameter). At a deep enough level (5-10), all stories will map to sequences like `[act] [transfer] [change] [rest]` and similar verbs, organized as collections of story-independent patterns. Such patterns indicate dramatic intensity and can be used to spot out climactic points in a story.

Causality/Entailment Verb Abstractions WordNet provides a `cs(Cause,Effect)` and an `ent(Action,Consequence)` relation applying only to verbs. By using them in the context of a story, on can derive hints about what might happen next or explain why something has happened.

Answering *Why* Questions through Causality Abstractions Causal relations provide possible explanations, and as such, answers to *why* questions about a story or generate explanatory sentences usable in abstracts.

Following the Logic of a Story using Entailment Abstractions Entailment relations provide predictions for future states and can help reducing polysemy by selecting meanings which fit in the entailment neighborhood of previously seen verbs, as the story advances. This is consistent with the fact that, to some extent, plots develop as possibilities are selected from the set of consequences of what happened. This allows following the logic of a story and possibly answering some "what-if" questions involving contrafactual developments and, in particular, questions about possible alternate endings courses of events.

3.4 Customizing Abstraction Operators

Depending on the lexical category of a given meaning, different abstraction operators can be used for fine tuning the results to what a human reader would consider relevant. Our customized abstraction operator follows the following algorithm, detailed per lexical category:

- **nouns**: hypernym links
- **verbs**: hypernym links, and hypernym links following causality and entailment links
- **adjectives**: attribute links to nouns followed by hypernym links and reverse attribute links back to adjectives
- **adjective satellites**: synonymy links to adjectives, followed by adjective abstractions as previously described
- **adverbs**: pertinence links to adjectives from which adverbs have been derived, followed by adjective abstractions and links back to adverbs

Abstraction operators have to deal with a fairly large set of possible meanings associated to each sentence. As we have not yet implemented an ambiguity reducing module using the global context of a narrative, the output contains a superset of what a human reader would pick as the most relevant skeleton describing a story at a given level of abstraction:

```
INPUT:

As Picasso said, computers are useless;
they can only give you answers.


[as]/k(adverb,2,wnet)
[Picasso]/k(noun,1,wnet)
[said]/k(verb,1,exc)
[(,)]/other
[computers]/k(noun,1,new)
[are]/k(verb,1,exc)
[useless]/k(adjective,1,wnet)
[(;)]/other
[they]/k(pronoun,1,fun)
```

```
  [can]/k(verb,1,wnet)
  [only]/k(adverb,4,wnet)
  [give]/k(verb,1,wnet)
  [you]/k(pronoun,1,fun)
  [answers]/k(noun,1,new)
  [.]/other

level = 0
  [[equally],[as],[every,bit]]
  [[Picasso],[Pablo,Picasso]]
  [[submit],[state],[put,forward],[say],[tell],[express]]
  [[computer],[computing,machine],[computing,device],...]
  [[exist],[be]]
  [[useless]]
  [[can],[tin],[tin,can],[put,up],[fire],[give,notice],...]
  [[merely],[simply],[just],[only],[but],...]
  [[give],[yield],[afford]]
  [[answer],[solution],[result],[resolution],...,[response]]

level = 1
  [[painter],[sculptor],[sculpturer],[carver],[statue,maker]]
  [[propose],[suggest],[advise],[denote],[refer],[express],...]
  [[machine],[expert]]
  [[good],[bad],[positive],[negative]]
  [[container],[preserve],[keep],[remove]]
  [[state],[say],[tell],[transfer],...]
  [[reaction],[response],[statement],[reply]]

level = 2
  [[artist],[creative,person]]
  [[mean],[intend],[declare]]
  [[person],[individual],[someone],[somebody],[mortal],[human],[soul],[device]]
  [[instrumentality],[instrumentation],[prepare]]
  [[express],[verbalize],[verbalise],[utter],...]
  [[message],[content],[subject,matter],[substance],[speech,act],
   ...,[bodily,function],[activity]]

level = 3
  [[creator]]
  [[convey],[impart],[state],[say],[tell]]
  [[organism],[being],[causal,agent],[cause],...]
  [[artifact],[artefact],[translate],[transform]]
  [[give],[support],[transfer],[pay],...]
  [[act],[human,action],[human,activity],[communication],
    [organic,process],[biological,process]]
```

One can notice the relatively smooth generalization of meenings. Intuitively, the more general descriptions are templates for a larger set of sentences - suggesting the possibility to use them for recognizing or generating similar sentences, as

the sequence of abstractions preserves the natural syntactic order of the original sentence.

```
level = 4
  [[person],[individual],[someone],[somebody],[mortal],[human],[soul]]
  [[communicate],[intercommunicate],....]
  [[entity],[physical,thing],[living,thing],...]
  [[whole],[whole,thing],[unit],[object],[physical,object],[change],[alter]]
  [[give],[support],[transfer],[pay],...]
  [[social,relation],[natural,process],[natural,action],[action],[activity]]

level = 5
  [[organism],[being],[causal,agent],[cause],[causal,agency]]
  [[interact]]
  [[whole],[whole,thing],[unit],[object],[physical,object]]
  [[entity],[physical,thing],[object],[physical,object]]
  [[arrange],[fix,up],[give],[support],[transfer],...]
  [[relation],[process]]

level = 6
  [[entity],[physical,thing],[living,thing],[animate,thing]]
  [[act],[move]]
  [[entity],[physical,thing],[object],[physical,object]]
  [[entity],[physical,thing]]
  [[agree],[hold],[concur],[concord],...]
  [[abstraction],[phenomenon]]

level = 7
  [[object],[physical,object]]
  [[entity],[physical,thing]]
  [[agree],[hold],[concur],[concord],...]
```

One can see that as abstraction levels increase, story traces become more and more generic – which makes them usable as a semantics driven grammar for recognizing or generating story elements.

4 Ontology Driven Semantic Projections

Semantic relations can be used as selectors to extract relevant parts of a document. We will describe first a number of techniques using basic WordNet relations.

4.1 Concept Specific Abstract Semantic Traces

The previous traces have been driven by syntactic concept classifications. We will explore now the traces generated by restricting the focus of interest to a given concept's semantic neigborhood.

Hyponym Traces Given a concept C we can look at its hyponyms and their contexts occurring in a text. This specializes a text by only retaining concepts subsumed by a given concept.

Meronym Traces Given a concept C we can look at its meronyms and their contexts occurring in a text. This specializes a text by only retaining concepts which are "parts" of the entities refered to by a given concept.

Specifying Ontologies as Formulas about Meanings An ontology driven semantic projection specializes a text with respect to a given ontology. We represent an ontology as *a set of synsets and projection rules* describing how to collect sentences whose content can be derived from the synsets. This generalizes hyponym or meronym traces by allowing the user to fine-tune through a set of rules what kind of links will be followed for each concept or verb.

A query like

```
?-project('bible.txt',[noun:[reptile],verb:[punish]])
```

indicates interest in file **bible.txt** for sentences involving **reptile** (a noun) and **punish** (a verb), and it will pick, in a few seconds from the 4 MBytes ascii version of the Bible sentences like:

```
['","{,3,:,2,},The,woman,said,to,the,serpent,(,),",
 Of,the,fruit,of,the,trees,of,the,garden,we,may,eat,(,),
 {,3,:,3,},but,of,the,fruit,of,the,tree,which,is,in,
 the,middle,of,the,garden,(,),God,has,said,(,),',
 You,shall,not,eat,of,it,(,),neither,shall,you,
 touch,it,(,),lest,you,die,.
]

[ ",{,3,:,14,},Yahweh,God,said,to,the,serpent,(,),",
  Because,you,have,done,this,(,),cursed,are,you,above,
  all,livestock,(,),and,above,every,animal,of,the,field,.
]
```

Ontology description formulas are passed in the recursive graph walk algorithm and applied to meaning sets. The predicate `tcall/2` applies a closure like `noun`, `verb` to check attributes of a synset or to generate links to be followed, belonging to a specific relation like hypernymy, meronymy etc.

```
% applies concept/attribute closure
tcall((R1,R2),X):-!,tcall(R1,X),tcall(R2,X).
tcall((R1;R2),X):-!,tcall(R1,X);tcall(R2,X).
tcall(not(R),X):-!, \+(tcall(R,X)).
tcall(Cond,X):-nonvar(Cond),call(Cond,X).

% applies a relation closure generating edges
```

```
tcall((R1,R2),From,To):-!,tcall(R1,From,To),tcall(R2,From,To).
tcall((R1;R2),From,To):-!,tcall(R1,From,To);tcall(R2,From,To).
tcall(not(R),From,To):-!, \+(tcall(R,From,To)).
tcall(Rel,From,To):-nonvar(Rel),call(Rel,From,To).
```

4.2 Dominant Theme Projection

We obtain the *dominant theme* of a story by collecting the most frequent 1-level noun abstractions occuring in the story. This gives a good approximation of what the story "is about". We can now use the concept subsumption projection to pick the sentences where hyponyms of the dominating concepts occur.

5 Extracting Information from Texts

Texts (narratives, stories) are vehicles for the transmission of cultural knowledge and information. News is often reported as stories. Medical records reflect stories of patient's problems and the history of treatment. Every day conversation (even conversations that are wire-tapped) report snippets of life (or perhaps criminal conspiracies) in a compact structure that associates social reasoning with a form of mapping that the human brain understands as a story pattern. Stories are structured from their inception as a basic skeleton (or storyline) that is further developed by the addition of a narrative world, a plot, characters, action, and resolution toward an ending, etc. Stories adopt rules, and even if broken, rules of tellability define minimums of acceptability for a narrative to retain story properties allowing it to also be sense- making and coherent. While most stories may not have perfect sematic consistency, humans can easily understand them by combining partial meanings.

The content of texts is structured within an architecture as set in its context. Folktales, headlines, news, emails, etc., all have *ontologies* that retain some levels of consistency. These ontologies are set within networks or systems that are complex and do not easily allow for intertexual comparisons. Sowa defines *an ontology*, as a

> "catalog of the types of things that are assumed to exist in a domain of interest `D` from the perspective of a person who uses a language `L` for the purpose of talking about `D`. The types in the ontology represent the predicates, word senses, or concept and relation types of the language `L` when used to discuss topics in the domain `D`" (see `http://users.bestweb.net/~sowa/ontology`).

To achieve "cataloging" (hierarchical organization) of text content, a reformatter type of mechanism is needed. We draw upon the idea of ontologic type suggested by Sowa by using the refactored Wordnet Prolog rule base to develop domain knowledge about storytelling to extract their patterns. The lexicon is the bridge between a language and the knowledge expressed in that language. A lexicon must also support uses of language and as such, it must contain syntactic

information, semantics, and enough information to enable recognition of the context, the topic, and the logical inferences necessary. Sowa [8] developed what he calls "the canonical formation rules", first stated in terms of conceptual graphs and then generalized to other forms of knowledge representation. He shows that the rules of inference of logic define a generalization hierarchy over the terms of any logic-based language. Our goal is to offer realistic but abstract representations of text content via ontology-driven projections and abstract story traces with applications to text mining, summarization and classification.

Story traces can help a developer understand the basic core of a story under development because it gives a compact view of story elements filtered at a given level of abstraction, while retaining the sequence of events. They can also be compared with traces of other stories as a way to focus on similar or different concept development and goals. We are able to trace sequences, patterns, chains of content that indicate causality and entailment.

The use of ontology driven projections with a text serves as an advanced form of information retrieval to find sections of a story with particular themes or ideas. These "semantic lenses" can search by inference rules for concepts or themes that typical word searches or strings of terms might otherwise miss. They also project what is embedded conceptually or thematically in a narrative so that that a developer can confirm what is in the story or reveal what might not be known to be imbedded in a story.

Both story traces and projections provide deeper insight into what a story "is about". As well, they have potential to reveal both the core elements and broad concepts/themes within stories.

6 Applications

We plan to use our abstract story traces to build a library of *text patterns*, by extracting sentence patterns and then organizing them into intersentential abstraction chains – patterns that give a synthetic view of a complete narative story.

Patterns will be also used for answering questions about a story and to automatically generate story indexes and a story database. The technique will also provide story classification combined with clustering of similar traces. By restricting our focus to *linear stories* (sequences of natural language sentences about chronologically ordered events) and using them as a training set to extract causal and logical entailment rules, we plan to obtain *story line predictors* usable, for instance to generate new stories or alternative endings of existing stories.

7 Conclusion

The operations described in the paper use a psycholinguistics inspired lexical database, WordNet, and a number of derived relations and inference mechanisms to extract high-level information about narratives as a compact view of their

key content elements. Story projections provide selections of key story elements which work as a kind of "semantic lenses" focusing on sentences related to a given ontology. These techniques have applications in text understanding, classification of digital text collections, story generation, and story-related question answering.

References

1. George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Five papers on WordNet. CSL Report 43, Cognitive Science Laboratory, Princeton University, July 1990.
2. C. Felbaum. *Wordnet, an Electronic Lexical Database for English.* Cambridge: MIT Press, 1998.
3. Paul Tarau. BinProlog 9.x Professional Edition: Advanced BinProlog Programming and Extensions Guide. Technical report, BinNet Corp., 2002. Available from http://www.binnetcorp.com/BinProlog.
4. Paul Tarau. Inference and Computation Mobility with Jinni. In K.R. Apt, V.W. Marek, and M. Truszczynski, editors, *The Logic Programming Paradigm: a 25 Year Perspective*, pages 33–48. Springer, 1999. ISBN 3-540-65463-1.
5. X. Li, S. Szpakowicz, and S. Matwin. A wordnet-based algorithm for word sense disambiguation. In *Proc. of the 14th IJCAI*, pages 1368–1374, Montreal, Canada, 1995.
6. Rada Mihalcea and Dan I. Moldovan. A highly accurate bootstrapping algorithm for word sense disambiguation. *International Journal on Artificial Intelligence Tools*, 10(1-2):5–21, 2001.
7. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM Symp. Principles of Programming Languages*, pages 238–278, 1977.
8. John F. Sowa. Relating templates to logic and language. In M. T. Pazienza, editor, *Information Extraction: Towards Scalable, Adaptable Systems, LNAI 1714*, page 76. Springer-Verlag, 1999.