# APPROVAL SHEET

**Title of Thesis:** Learning By Reading: Automatic Knowledge Extraction through Semantic Analysis

**Name of Candidate:** Jesse Cole English
Doctor of Philosophy, 2010

**Thesis and Abstract Approved:** _____
Sergei Nirenburg
Professor
Department of Computer Science and
Electrical Engineering

**Date Approved:** _____

# Curriculum Vitae

**Name:** Jesse Cole English

**Degree and date to be conferred:** Doctor of Philosophy, August 2010

**Secondary Education:** James M. Bennett High School, Salisbury, Maryland

**Collegiate institutions attended:**

University of Maryland Baltimore County, Doctor of Philosophy in Computer Science, 2010.
University of Maryland Baltimore County, Master of Science in Computer Science, 2006.
Salisbury University, Bachelor of Science in Computer Science and Mathematics, 2004.

**Major:** Computer Science

**Professional publications:**

Akshay Java, et al. *Using a Natural Language Understanding System to Generate Semantic Web Content*. International Journal on Semantic Web and Information Systems (IJSWIS) 3(4) Article 03.

Sergei Nirenburg, Tim Oates, Jesse English. *Learning by Reading by Learning to Read*. Proceedings of ICSC-07, August 2007.

Jesse English, Sergei Nirenburg. *Ontology Learning from Text Using Automatic Ontolgical-Semantic Text Annotation and the Web as the Corpus*. Proceedings of the AAAI 2007 Spring Symposium Series on Machine Reading, March 2007.

Sandor Dornbush, et al. *XPod: A Human Activity Aware Learning Mobile Music Player*. Proceedings of the Workshop on Ambient Intelligence, 20th International Joint Conference on Artificial Intelligence (IJCAI), January 2007.

Jesse English, Michael Wiacek, Mohamed Younis. *CORE: Coordinated Relocation of Sink Nodes in Wireless Sensor Networks*. Proceedings of the 23rd Biennial Symposium on Communication, May 2006.

**Professional positions held:**

Research Assistant, Institute for Language and Information Technologies (2004 – 2010).

On-site Technical Support, McEnroe Voice and Data (2001 – 2004).

# ABSTRACT

**Title of Thesis:** Learning By Reading: Automatic Knowledge Extraction through Semantic Analysis

Jesse Cole English, Doctor of Philosophy in Computer Science, 2010

**Thesis directed by:** Sergei Nirenburg, Professor
Department of Computer Science and
Electrical Engineering

To support rich semantic analysis of text, traditional natural language processing tools require access to a cache of static knowledge with both broad coverage and deep meaning. Acquiring this knowledge by hand is so expensive and error-prone, it has been dubbed the ”knowledge acquisition bottleneck”. In this work, we present a method for reducing the impact of this bottleneck by automating the knowledge acquisition task using the novel approach of bootstrapping a machine learner with a fully-realized semantic analysis engine, creating a life-long learner.

We present an overview of our learner: a system that automatically produces lexical and ontological knowledge resources by building a corpus of raw texts from the web, semantically analyzing them to the best ability of the existing engine, and extracting the word meanings. We expand on this overview by presenting a series of experiments in chronological order, each evolving on the previous one as we explore the possibilities presented by our methodology.

Finally, we explore a series of improvements to our system: we discuss a variety of changes to individual components, as well as complete methodological shifts. These discussions will set the stage for the next round of interesting experiments in pursuit of a fully automatic language learner.

# Learning By Reading: Automatic Knowledge Extraction through Semantic Analysis

by

Jesse Cole English

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science
2010

*For Sarah*

## ACKNOWLEDGMENTS

I would like to thank my graduate advisor Dr. Sergei Nirenburg for his passion and guidance through the course of this work: it has been both an inspiring and humbling experience that has shaped who I am, as much as it has shaped what I will do. I would also like to thank the members of my committee, Dr. Marge McShane, Dr. Tim Oates, Dr. Yelena Yesha and Dr. Nick Cassimatis for the valuable feedback. Further, I'd like to thank Dr. Stephen Beale and Matt Rodatus for their assistance.

I take this opportunity to also thank my friends who have journeyed this road with me: Dr. Craig Gerardi, Allen Stone, Dr. Akshay Java, Benjamin Johnson and Don Dimitroff. Enduring this trial together has been much more bearable than standing alone.

Finally, I would like to thank all of my family and friends who have been unbelievably supportive. To my mother Helene, my father Chris, my brother Grayson, and my grandparents who all have given me a lifetime of encouragement; to my friends, especially Steven Blankenship and Youssef Mahmoud who have both helped me keep my eye on the goal, in their own ways; and to my wife, Sarah, to whom this work is dedicated. Thank you.

# TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

**Chapter 1**

# INTRODUCTION

Lexical and ontological resources often provide the foundation of the static knowledge used by semantic analysis engines in natural language processing (NLP). Lexical resources act as a mediator between words, and the complex semantic meanings described in an ontology. Using these two resources in conjunction, along with a set of processors, algorithms, heuristics, and microtheories, a semantic analysis engine can convert raw text into a text meaning representation (TMR), a machine-tractable description of the input sentence.

Acquisition of these static knowledge resources has traditionally been a manual operation, requiring specially trained ontologists as well as input from domain experts. Manual acquisition, however, is both costly and error-prone, and can be far too slow to maintain broad coverage in a rapidly changing environment (such as the medical or technological domains). As wide coverage is often required to enable quality NLP, the problem of sufficient acquisition has been termed ”the knowledge acquisition bottleneck”: further progress in NLP using traditional methods and static knowledge is often halted by a lack of coverage in the knowledge itself.

In contrast to manual acquisition, and partially as a result of the knowledge acquisition bottleneck, many statistics-based NLP techniques have been developed. Using string-based heuristics, such as bag-of-words, these techniques often focus on knowledge-lean

approaches, attempting to assimilate meaning from text without any existing prior knowledge. An alternative to this approach, one that maintains the knowledge-rich traditional NLP techniques, involves automating knowledge acquisition to a point that the bottleneck's influence is negligible. Often, this is presented in the context of a particular domain, or targets only limited knowledge resources and may include manual intervention or verification of the results.

In this work we propose and present a technique for fully automating knowledge acquisition for an existing semantic analysis engine. The system presented uses an existing NLP engine as input to produce analyses of texts. The results of these analyses are collated, analyzed and filtered, and are presented as candidate knowledge to be added back into the existing system, producing a lifelong learner. We present several hypotheses related to our proposed system, notably that an existing NLP engine can be used to bootstrap a lifelong learner, and that quality inputs will have a greater positive impact on automatically learned knowledge than quantity of inputs. We will also present our implementation of the proposed system, and our findings following a series of experiments.

The remainder of this work is organized as follows: in the next chapter we discuss the motivation behind the research. In Chapter 3, we present OntoSem, an existing semantic analysis engine with a collection of manually acquired static knowledge resources. Chapter 4 reports on prior work in the field of automatic knowledge acquisition for a variety of tasks and systems. In Chapter 5, we will discuss our proposed experimental pipeline at a coarse level. Chapter 6 presents a detailed look at the preprocessing phases required by our experiments prior to learning. In Chapter 7 we will explore the full experimental process undertaken in this research, spotlighting the evolution of the system over time. In Chapter 8, we will present the results of our experiments and discuss the findings they suggest. Finally, Chapter 9 will propose future research directions for each of the learner's components, as well as for the entire process itself.

**Chapter 2**

# MOTIVATION

Traditional natural language processing can require access to knowledge in order to support a variety of tasks, including reasoning, question / answering and machine translation. The knowledge available must be broad in coverage, as well as sufficiently detailed to be of benefit to the task at hand. Often, this knowledge takes the form of ontological and lexical resources, with varying degrees of complexity. These resources assist the NLP system by providing context and imposing structure on corpora and databases, resulting in a system capable of understanding (given the level of depth required for the task).

Regardless of the challenge, it is clear that providing this knowledge to the NLP system is a costly, slow and error-prone process that has been labeled the ”knowledge acquisition bottleneck”. In (Brewster *et al.* 2005), the need to automate the task of knowledge acquisition was addressed: ”Excavating knowledge by hand from the substrate of human artifacts (whether a collection of minds or texts) is extremely costly. It is costly in time, it is costly in financial terms and it is very error prone. It has consequently been a foundational tenet of ontology building that we must automate the task ideally in its entirety.” Although the authors implicate the process of ontology construction specifically, the message holds true for all knowledge acquisition: manually acquiring lexical, ontological or other knowledge resources takes immense amounts of time, requiring skilled computational linguists

(and often domain experts as well), and is subject to a variety of errors (including misunderstandings, typos, or divergent paths of knowledge development).

The cost in financial terms can be staggering: it has been estimated that it would take a dedicated team of twenty individuals four years to produce 100,000 ontological concepts for OntoSem (see Chapter 3: OntoSem), and another thirty individuals that same time to produce 250,000 lexicalized word senses using that ontology (these estimates do not include the necessary management and support infrastructures for such a venture). Although estimates on the number of English language words vary, it is not uncommon to find numbers reaching as high as 750,000 distinct word-senses (e.g., (www.askoxford.com )), excluding slang, common acronyms, and technical and regional vocabularies (e.g., the medical field has an immense vocabulary peculiar to its task). To further complicate the task, language is an ever-evolving entity: new words are added (formally and informally) to a language each year as domains grow, and popular culture shifts. The task is not a simple open and shut case, but requires life-long attention, further raising the impracticality of a completely manual solution to knowledge acquisition.

At present, manual knowledge acquisition for a system like OntoSem involves a series of procedures requiring an individual who has been specifically trained to work within the system and the interfaces available. A target word must first be selected by the acquirer (likely one chosen to support the current task). Once the word-sense is known, the acquirer must check to see if any synonyms for the word already exist, in which case the word can simply be added as such. Assuming the word is new to the system, the acquirer must then hunt through the hierarchical ontology for the correct matching concept for the word: if one exists, then the lexical entry for the word can be created (involving mapping the word to the concept, and detailing the syntactic structure imposed by the word-sense); if a suitable concept does not exist, then one must be acquired before continuing. Acquiring an ontological concept involves understanding how the concept interacts with the rest of the

world-view currently described in the ontology. A suitable location for the new concept must be found (this involves identifying where subsumption links should be added), and then the acquirer must decide which property / filler pairs from the parent(s) should be inherited by the new concept, and which should be ignored. Following this, new property / filler pairs can be added manually to fully describe the new concept: this process can be arduous and involves heavy thinking and understanding of the interactions described within the ontology. Once the new concept has been finished, the acquirer may return to the word-sense and acquire the lexical entry as described above.

To assist in this process, a full suite of user interfaces designed for this task has been created in the DEKADE (Development, Evaluation, Knowledge Acquisition and Demonstration Environment) system (see Appendix B: Tools and Interfaces). This custom software, created for OntoSem facilitates knowledge acquisition at a convenience level (e.g., custom editors for acquisition with built-in validators to warn the user when semantically invalid knowledge has been entered), but does not assist in making knowledge-based decisions, leaving that for the acquirer. Other systems have also been developed specifically to aid in knowledge acquisition, such as the Protege system (Gennari *et al.* 2003), whose sole purpose is to facilitate manual acquisition of knowledge in a system-independent way. Protege, which has undergone a series of complete revamps stands as an example of how complicated the task of assisting manual acquisition can be, suggesting the true task is immeasurably more complex.

Understandably, the desire to automate the task of knowledge acquisition is often held strongly by proponents of traditional natural language processing. Although complete and flawless automation of acquisition is not currently realistic, using automated techniques to assist acquirers in either broadening coverage, increasing quality of existing coverage, or in simply reducing the cost of manual acquisition is desirable. An automated learner that can suggest candidate concepts to be post-process edited by hand would assist in both increas-

ing coverage and reducing the cost of acquisition; alternatively a learner that scrutinizes existing (even intentionally under-developed) concepts and corrects or adds to the property / filler pairs would improve quality without the need to dedicate additional man-hours for the same task.

In (Brewster *et al.* 2005), the authors argue that ontology building is akin to civil engineering: the right groundwork must be laid prior to detailed construction. This claim fits well with one of the major hypotheses presented and tested in this work: we propose that a life-long learner can be constructed from an existing NLP toolset, bootstrapped by its own (manually acquired) knowledge-base. This system would be able to re-bootstrap itself after each learning cycle, with the improved knowledge base allowing for finer grained learning in the next round. In order to test this, the right groundwork must first be laid; for our purposes, we will use the existing OntoSem system as our groundwork and mutual-bootstrapper. Using the manually acquired knowledge already present in OntoSem enables us to automate language learning from a point further ahead than the starting line.

Although fully-automatic, high-quality ontological and lexical resource learning may still be out of reach, the need to alleviate the arduous task of manual knowledge-acquisition is strong. Continuing to produce all knowledge for complex NLP systems entirely by hand is far too costly a proposition, and must be aided in some way by autonomous language-learners.

## Chapter 3

# ONTOSEM

### 3.1 Introduction: What is OntoSem?

OntoSem is a natural language processing suite primarily based on the theory of ontological semantics. Using a machine-tractable view of world-knowledge (a semantic ontology) and a database of lexical mappings, OntoSem can perform a deep semantic analysis of input text, extracting language-independent meaning from the input as well as accurately model intelligent agents in a virtual environment.

The primary goal of OntoSem is to accurately produce thorough semantic analyses of arbitrary input text (including factual articles, fiction and prose, and dialog). This task is performed by a set of sub-processors which link to both the ontology and lexicon, as well as lesser static knowledge resources. Input text is part-of-speech tagged, marked with syntactic structures, and then semantically analyzed to produce a text meaning representation (TMR).

In this chapter, we'll discuss each component of the OntoSem system that relates to the work presented. In the next section we'll describe each of the static knowledge resources available to OntoSem; in section 3.3 we'll discuss TMRs (including what a TMR is, how it relates to the static knowledge, and how it can be used in this work); in section 3.4 we'll present OntoSem's text analysis pipeline, touching on each component that is used to fully

process text into semantic meaning; finally, section 3.5 will detail OntoSearch, a tool used by OntoSem (and this work) for judging distances across the semantic ontology.

## 3.2 Static Knowledge Resources

In order to support language analysis and intelligent agent modeling, OntoSem requires access to an array of static knowledge resources. These resources are largely manually acquired and are interdependent. Primarily, there are two main resource types used by OntoSem, however there are several supporting resources as well (which do not factor into the work presented in future chapters). The primary resource, a language-independent semantic ontology, encapsulates a world-view model using a hierarchical network of semantically-rich concepts. The lexicon, a resource of word-senses, both links to elements of the ontology and describes the syntactic nature of words. Additionally, an onomasticon (a lexicon of proper nouns), a fact repository (an extension of the ontology for particular instances of knowledge) and scripts (a series of events from the ontology in chronological or logical order) all support language processing and agent modeling in OntoSem. Currently, each of these resources is manually acquired using custom interfaces designed for the task (see Appendix B: Tools and Interfaces).

### 3.2.1 Ontology

The language-independent semantic ontology used by OntoSem consists of a collection of concepts organized hierarchically and interconnected by a set of property / filler pairs to create a dense network of world knowledge. Each concept is defined by a set of property / filler pairs, which fall into one of two categories: relations link the concept to another concept in the ontology (examples include AGENT, and THEME), and attributes describe the concept internally (examples include HEIGHT and COLOR). Every concept in

the ontology is linked to at least one other through a subsumption link (multiple inheritance is allowed), with all concepts having either OBJECT or EVENT (the two most general concepts) as an ancestor.

Each concept makes use of of the subsumption links to inherit property / filler pairs from parent concepts: at the child concept level, a decision to include a particular property / filler pair from the parent or to override or remove it entirely can be made. This allows for descriptions such as "a penguin is a bird, but it does not fly" to bear deep meaning. The concept PENGUIN can be made a child of the concept BIRD, and inherit all property / filler pairs. Then the single property / filler pair AGENT-OF / FLY, can be removed, resulting in a semantic description of a flightless bird. Table 3.1 shows a snippet of semantic description of the concept HUMAN, a well-developed root of a large sub-tree of ontological concepts describing various people:

| **Property** | **Filler** | **Notes** |
|---|---|---|
| IS-A | PRIMATE | (this is a relation) |
| AGE | 0 - 130 | (this is an attribute on an absolute scale) |
| CHARISMA | 0 - 1 | (this is an attribute on a relative scale) |
| LOCATION | PLACE | (this is an inherited relation) |
| HAS-OBJECT-AS-PART | LEG | (this is an inherited relation) |

Table 3.1. A partial description of the ontological concept HUMAN.

The ontology is a language-independent resource, meaning it has no association with any particular language, although the names of the concepts and properties are often English words for the sake of clarity. The ontology is designed without language in mind in order to facilitate multi-lingual lexicons and machine translation tasks. Essentially, the ontology is built in a self-defined universal language: each property is defined and used in the ontology, creating a meta-description out of the ontology's backbone. At the time of this writing, there are over 8,300 concepts in the ontology.

The ontology defines the underlying knowledge structure required for the remainder of the static knowledge resources. Language dependency is gained by associating a lexicon with the ontology, factual instance information is gained by layering a fact repository on top of the ontology and complex event sequences are gained by creating scripts from the ontology. Finally, TMRs, the language-independent meanings of input text are generated from instances of ontological concepts: the entire process of language analysis and agent modeling requires a broad coverage, well-defined, language-indendent semantic ontology.

### 3.2.2 Lexicon

The semantic lexicon used by OntoSem defines a list of lexical units, each with a complete syntactic, semantic, and morphological description. A single lexical entry details how that word can be used, structurally, in a sentence, as well as how it ties into the ontology. Semantically, it specifies what concept is defined in the ontology that encapsulates the entry's meaning, and optionally provides modifications to property / filler pairs for further specification. Example 3.1 shows a lexical entry:

```
woman-n1
     syn-struc
          root      $var0
          cat       n
     sem-struc
          HUMAN
               AGE         > 15
               GENDER      female
```

**Example 3.1:** A simple lexical entry; HUMAN is a concept in the ontology; this lexical entry is overriding the AGE and GENDER properties, supplying more specified fillers of its own.

Lexical entries also contain lists of synonyms, hyponyms and abbreviations, as well as any specific directives for OntoSem to follow when using the entry in an analysis (see

3.4.4 Micro-theories and Meaning Procedures). Although the lexicon is currently only developed in English, separate lexicons could be developed for other languages and could be tied directly into the same ontology. The lexicon consists of stemmed (or root) words: alternate forms of words are converted to their basic forms in the preprocessing phase of analysis (see 3.4.1 Preprocessing). At the time of writing, there are over 17,000 manually acquired lexical entries available to OntoSem.

### 3.2.3 Onomasticon and Fact Repository

In addition to the primary knowledge resources (the ontology and lexicon), a handful of other knowledge resources also support analysis and agent modeling in OntoSem. These resources do not play a significant role in the research presented here, and so will be briefly discussed.

The onomasticon is a lexicon of proper nouns: most of the fields used in full lexical entries are not present, leaving only a sem-struc (which is often a simple mapping) and a list of synonyms. A sample onomasticon entry is shown in Example 3.2:

```
united_states-1
     sem-struc
          NATION
     synonyms
          america, united_states_of_america, u.s., u.s.a.,
          us, usa
```

**Example 3.2:** A sample onomasticon entry with a simple ontological mapping.

Onomasticon entries are treated as simple noun entries in the lexicon, but are moved to a special resource due to their quantity. At the time of this writing, there are over 275,000 onomasticon entries (mostly automatically populated) available to OntoSem.

The fact repository (FR) is a collection of ontology concept instances, modified (often

by a TMR) and tied to a specific frame. FR entries encapsulate a specific mention of a ontological concept: e.g., HUMAN is the ontological concept describing a person, whereas HUMAN-FR15 is a particular instance of HUMAN, and will have a set of property / filler pairs that identify uniqueness. The FR is used to capture long-term memory of modeled agents, and to retain factual information for reference resolution purposes (see 3.4.5 Reference Resolution).

To continue the example: HUMAN-FR15 may have been derived from a conversation with an agent, wherein the agent was told: "Carl will be dining with us tonight." The TMR produced by analysis will include a HUMAN frame, as shown in Example 3.3:

```
HUMAN
     AGENT-OF      EAT-MEAL
     HAS-NAME      Carl
```

**Example 3.3:** A frame generated in the TMR, prior to inclusion in the FR.

If the agent wished to retain this knowledge for future use (long-term memory), the HUMAN frame could be added to the FR, by appending the FR postfix: e.g., HUMAN-FR15. During future discussion, the agent may be informed: "Carl will be bringing his sister to dinner." The reference resolution engine would now be able to identify the instance of "Carl" in the second sentence with the existing FR instance (HUMAN-FR15). Further processing would result in the following modifications to the FR shown in Example 3.4:

## 3.3 TMRs

Text meaning representations (TMRs) are machine-tractable formulations of the meaning of texts. A TMR is created by OntoSem as a result of semantic analysis: raw input text is mapped to lexical senses through syntactic structures, which are then tied in to

```
HUMAN-FR15
     AGENT-OF          EAT-MEAL-3
     HAS-NAME          Carl
     HAS-SIBLING       HUMAN-16

HUMAN-16
     GENDER            female
     HAS-SIBLING       HUMAN-15
     AGENT-OF          EAT-MEAL-3
```

**Example 3.4:** The FR has been significantly modified after the second sentence was analyzed by the agent. A relationship between "Carl" and his "sister" now exist, and both are shown to be attending the same dinner.

the ontology via the sem-struc field. A TMR consists of a collection of frames generated as instances of ontological concepts, with restrictions placed on the fillers of properties to match the semantic content of the input sentence.

Each OBJECT and EVENT found in a sentence is given an instance in the TMR, and the relations between them are formulated as property / filler pairs in the respective instances. For example, the text "Joe saw the ball." would generate two OBJECTs, a HUMAN (for "Joe") and a BALL (for "ball"). An EVENT would be generated for "saw", VISUAL-EVENT, and its two primary properties would be filled: its AGENT would be HUMAN and its THEME would be BALL. Specifically, its AGENT would be HUMAN-15 (the instance of "Joe" in the TMR) and its THEME would be BALL-22 (the instance of "ball" in the TMR). Similarly, the text "Joe saw Sally." would result in two HUMAN instances (each with a different postfix instance number).

Each TMR frame that was directly generated from a lexical token contains a pointer to that raw text in the form of the TEXTPOINTER attribute. Additional syntactic phenomena are also recorded in the TMR frames in the form of property / filler pairs (although they are not formal properties defined in the ontology). These attributes do not carry any semantic

information, but are useful for debugging purposes and were involved in one of the phases of the learner described later.

Unknown words in a TMR generate a flagged frame, either *UNKNOWN-OBJECT* or *UNKNOWN-EVENT*, depending on both syntax and semantics (see 3.4.3: Semantic Linking and Analysis). These frames will often have as many property / fillers defined as traditional (known) frames, however will not have as much specific semantic information to fall back on from the ontology; in other words, knowing a frame is derived from MAMMAL is more informative than knowing it is derived from OBJECT, even if the property / fillers from the TMR (via semantic context) are the same. As an example, the following two input texts would both generate the same TMR frame for ”ape” and ”aaa”, only the former would be derived from the concept MAMMAL and the latter would be labeled an *UNKNOWN-OBJECT*:

1. The ape ate the banana.

2. The aaa ate the banana.

Example 3.5 shows a sample TMR (abbreviated for space), for the text ”John purchased his new car, and lost its keys.”

The expressive power of the TMR is only limited by the coverage and depth of the ontology: more concepts (with deeper descriptions) linked to lexical entries will result in more accurate TMRs from text. The process of generating a TMR is covered in the next section.

## 3.4 Processing Text to TMRs

For OntoSem to convert text into a TMR, it must first pass the text through a variety of processors prior to the final semantic analysis. Once the text has been prepared, semantic

```
HUMAN-19
     HAS-NAME                 John
     TEXTPOINTER              "John"

BUY-23
     AGENT                    HUMAN-19
     THEME                    AUTOMOBILE-30
     TEXTPOINTER              "purchased"
     FROM-SENSE               purchase-v1

AUTOMOBILE-30
     AGE      0.2
     HAS-OBJECT-AS-PART       KEY-45
     TEXTPOINTER              "car"
     FROM-SENSE               car-n1

LOSE-37
     AGENT                    HUMAN-19
     THEME                    KEY-45
     TEXTPOINTER              "lost"
     FROM-SENSE               lose-v4

KEY-45
     TEXTPOINTER              "keys"
     FROM-SENSE               key-n1
```

**Example 3.5:** A sample TMR showing multiple interconnected frame instances.

linking and analysis can occur, resulting in a TMR. Following this, further processing can be done in order to resolve inter-sentence reference. This section describes each of the phases of analysis that a text passes through in order to produce a TMR.

### 3.4.1 Preprocessing

The first phase of processing done by OntoSem on a raw text input is termed ”preprocessing”. This phase focuses on preparing the text for more in-depth analysis by part-of-speech (POS) tagging each token, performing named entity recognition and dividing the

text into sentences (for individual analysis). Part-of-speech tagging is done following a stemmer that turns each lexicon token into its root word. A large database of tokens and POS markers is used to associate each possibility for the word. Named entity recognition is also performed: proper nouns (both found directly in a database, as well as unknown tokens presumed to be proper nouns) are identified and labeled. Sentence breakers are then located using a variety of heuristics, including searching for common punctuation markings. Additionally, a variety of other special-case tags are added to lexical tokens, including time-markers (e.g., ”11:52 am”), and currency markers (e.g., ”$4.15”).

The results from the preprocessing phase of a text include a set of sentences, containing POS-marked lexical roots, as well as additional special-case markings. These results are now ready to be syntactically analyzed by the next phase.

### 3.4.2 Syntactic Analysis

Following preprocessing, OntoSem uses the POS markings on each lexical token to calculate possible syntactic structures (e.g., verb and noun phrases). The result is a (small) set of sentence-wide phrase structures, containing increasingly smaller sub-phrases spanning all of the tokens in the sentence. The possible phrase structures are then passed on to the linking and semantic analysis stage.

### 3.4.3 Semantic Linking and Analysis

The final stage of TMR production requires an input text to be tokenized, stemmed, POS marked and spanned with syntactic arcs. At this point, the semantic lexicon is introduced to the analysis: using both the syntactic structures and the semantic structures of a given lexical entry (the syn-struc and sem-struc fields), each token is associated with an existing lexical entry. In other words, using the root word from the lexical entry to match a small set of all entries to a given token in the text, the system then attempts to match a

particular entry from that set based on the known syntactic restrictions (and, if necessary, semantic restrictions).

During this phase, if multiple sentence-wide phrases were suggested, all but one will be dropped. This is due to the application of syntactic selectional restrictions: individual lexical entries use the syn-struc field to restrict how they can be, syntactically, used in a sentence. In order to force one in to place, it may impose restrictions on other lexical tokens, forcing a choice amongst all of the possible phrase choices presented.

If ambiguity still exists, the sem-struc field can also be used: the system can make a repeated attempt to apply semantic restrictions in an effort to logically remove choices. In other words, the analyzer performs a ”if sense-1 is chosen for token-1, then token-2 cannot be sense-2 due to semantic restrictions imposed by sense-1”. The combination of syntactic and semantic restrictions is intended to remove enough ambiguity to result in a clear choice.

Eventually, each token will have a single lexical entry applied to it, and only one sentence-wide phrase will be retained. Using the sem-struc fields, the analyzer can now begin to construct the TMR by instantiating a frame for each lexical token. Recall that the sem-struc field points to an existing ontological concept (and applies restrictions of particular properties of that concept). A frame for each token is created as an instance of the concept referenced in the sem-struc field, and is then populated with property / fillers generated as relations between the other TMR frames. These relations are determined by the restrictions imposed by both the syn- and sem-strucs of the lexical entries selected: using syntactic structures such as subject / object pairs, property / filler pairs can be create (often AGENT and THEME).

By means of example, consider the text ”The man ate the salad.” In this case, the verb, ”ate” has a subject (”the man”) and and object (”the salad”), both of which would be proposed during syntactic analysis. The lexical entry for ”ate”, eat-v1, shown below, uses two internal variables ($VAR2 and $VAR3) to associate the subject of the action with

the filler of the AGENT property, and the object of the action with the filler of the THEME property:

```
eat-v1
     syn-struc ((SUBJECT ((ROOT $VAR2) (CAT NP)))
          (ROOT $VAR0) (CAT V)
          (DIRECTOBJECT ((ROOT $VAR3) (OPT +) (CAT NP))))
     sem-struc (INGEST
          (AGENT (VALUE ^$VAR2))
          (THEME (VALUE ^$VAR3)))
```

**Example 3.6:** The lexical structure for eat-v1 associates the subject and object of the verb with the AGENT and THEME of its concept (INGEST).

As a result, the TMR produced would contain a frame generated from ”ate”, using eat-v1 as the lexical entry:

```
INGEST-49
     AGENT            HUMAN-12
     THEME            SALAD-88
     TEXTPOINTER      "ate"
     FROM-SENSE       eat-v1
```

**Example 3.7:** The TMR frame created from the word ”ate” in ”The man ate the salad.”

Continuing in this manner, the entire TMR is created by associating lexical entries with POS marked tokens, applying syntactic and semantic restrictions, and generating frames from semantically constrained ontological concepts. The result is a machine-tractable span of the meaning of the text, written in the language of the ontology.

### 3.4.4 Micro-theories and Meaning Procedures

Unfortunately, full semantic analysis is not always as simple as token → POS → syntax → lexical entry → TMR frame. Often, complex semantic structures are included: simile, metaphor, prose, ambiguity, ellipsis, non-literal language, etc. all create a far more complex environment than can be handled easily as described in the previous section. In order to manage these phenomena, a complex collection of micro-theories have been developed for OntoSem, each designed to handle very specific and complicated input.

Triggering these meaning procedures can be the job of either the analyzer itself (if it identifies an area of text that suggests a certain micro-theory) or the job of lexical entries, which have been designed to detect these phenomena and include directives to the analyzer to fire a meaning procedure. As OntoSem is a research platform, many meaning procedures are currently un- or under-developed; lexical entries may trigger their usage, but the results are often barely modified. These pose a particular problem in the research presented here, as we needed to essentially ignore the results of these unfinished processes.

### 3.4.5 Reference Resolution

Reference resolution takes two forms in OntoSem: intra-sentence reference resolution is part of the standard semantic analysis needed to produce a TMR. Inter-sentence reference resolution is an optional module that spans multiple input texts. Intra-sentence reference resolution aims to resolve ambiguities in reference such as pronouns. In the sentence ”Bob went to the store, and then he got on the bus.”, it is necessary to properly link ”Bob” and ”he” as the same agent. The resultant TMR, without reference resolution, would contain two HUMAN frames, one for the token ”Bob” (being the AGENT of ”going to the store”) and one for the token ”he” (being the AGENT of ”getting on the bus”). Ideally, there should be only one HUMAN frame encapsulating both of these: intra-sentence reference resolution

aims to provide this level of analysis.

Inter-sentence reference resolution, on the other hand, aims to span corefering frames across multiple sentences in a text. Image the example ”Bob went to the store. Afterwards, he got on the bus.” Here, a model that analyzes the two existing TMRs would be needed to properly link the frame created from ”Bob” and the frame created from ”he”. Although intra-sentence reference resolution is used (by default) in the experiments presented in this work, inter-sentence reference resolution is not; Chapter 9: Future Research Directions discusses one way that inter-sentence reference resolution could be used to increase the quality of output presented by the learner in this work.

## 3.5 OntoSearch

OntoSearch (Onyshkevych 1997), a component function of OntoSem, bears special attention here (as it was specifically used in the first Experiment reported in Chapter 7: Experimental Setup and Evaluation Processes). OntoSearch is a trained network crawling algorithm designed to report the distance between two concepts in the ontology by finding the shortest weighted path crossing property / filler pairs. OntoSearch does not exclusively use subsumption links (although it can): rather, it takes all relations into account as possible paths to join two concepts. Each link that is crossed applies a weight to the traversal score, resulting in a normalized ”proximity” value.

The weights applied by each path are automatically determined during a training phase that analyzes the current state of the ontology as a directed cyclic graph. The result of this training phase is a set of property / weight pairs which are used to penalize the search algorithm as it traverses each link.

## 3.6 Conclusion

In this chapter we have introduced OntoSem, a semantic analysis and agent modeling system capable of producing text meaning representations in the language of the ontology from raw text using a series of preprocessors, a semantic lexicon and an ontology of world knowledge. We have touched on each of the static knowledge resources, including those not relevant to the work presented here.

We have introduced the TMR, a collection of ontological concept instances describing the semantic state of a text and provided examples of how one is created, as well as covered, in depth, the text analysis process. Finally, we touched on OntoSearch, an internal component of OntoSem that was externalized for the purposes of our experimentation. In Chapter 5: Experiment Overview, we will discuss how OntoSem and its resources will be used in our language learning experiments.

Chapter 4

# PRIOR WORK

## 4.1 Introduction

Language learning from text, the focus of this research, relates to a wide array of other research areas, including information retrieval, machine learning, language processing and question/answering. This chapter presents a literature survey that touches on each of these fields, with a focus on work that heavily relates to the learning-by-reading paradigm, and is therefore comparable in scope, or technique to the research we present in the coming chapters.

Rather than a straightforward chronological directory of previous publications, we've identified four major components to related research that influenced both our work and the field as a whole. We'll present and motivate each of these four components, and then discuss the highlights of the prior work for each. We will continue this metaphor as we discuss the research presented in this document, in Chapter 7: Experimental Setup and Evaluation Processes, and then in Chapter 8: Results and Discussion.

## 4.2 Relevant Research Components

In a more confined area of research, a survey of previous publications may suffice by enumerating the research's hypothesis, followed by a summarized explanation of the

process, and finally the results of the experimentation. As language learning from text has application in many related fields, we cannot use this simplistic method of presenting previously described work. Rather, we will need to modify these three basic components, and present a necessary fourth discussion point as well.

For our purposes, as language learners, information retrievers, or machine learners, often ”hypothesis” is not the most appropriate description of our research’s starting point. Publications in these fields often focus more on a ”goal”, although the distinction in terms may be largely semantic. It is important for us to present what, in exact terms, a piece of prior work was hoping to achieve. The researchers, in most cases, have more of a target (e.g. to take unknown words in a corpus, and use them to automatically populate an existing ontology) and less of a supposition of how things may work (e.g. will semantic analysis assist in language learning?). This is not to say that a conclusion cannot be drawn from the research, nor that a hypothesis was not the original motivation for the research; rather that a more appropriate starting point in a conversation on the history of language learning in related fields is more directed when oriented from the perspective of a goal, rather than a hypothesis.

Our second component does not break from convention nearly as much as our use of ”goal” rather than ”hypothesis”. We will need to discuss various techniques used in language learning. These vary from manual annotation, to automatic semantic analysis, and many points in between. The more information retrieval-heavy research leans largely on statistical filtering methods, and may depend exclusively (or largely) on simple syntactic parse trees. Much research across the spectrum uses widely available resources such as WordNet. It is important to look, historically, at the directions that were taken, even when the goals were not entirely similar. Each publication discussed relates, in some way, to language learning, and uses some novel approach or combination of prior experimental processes to produce results.

Third, we need to address a component of our research that is largely unique to our field: the corpus. The input text that can be used in this field of research is largely arbitrary and flexible. Although some publicly available corpora do exist, there is no widely accepted standard. This problem stems from the myriad goals that researchers in this field apply themselves to, and has a direct impact on what corpus is appropriate to use: the goal could be part of an on-going attempt to improve particular results on a pre-determined corpus; likewise the goal could be to directly improve a domain-specific static knowledge resource, and so might use textbooks or publications from within that domain; further, the goal could be to provide question/answering support on any subject, and thus will need to use an open corpus, such as the web. The corpus that is used has a large impact on how an experiment can be run, and as importantly, how it can be evaluated.

The final component that we will discuss is the evaluation process presented in each publication. This is significantly different from the more common discussion of evaluation results. We are not so interested in how well the experiment performed, but are more interested in how evaluation was carried out. Largely, this is impacted by the corpus chosen: a manually annotated corpus that belongs to a challenge set allows results from an individual experiment to be relatively easily calculated. However, a more open corpus coupled with a goal of extending an existing static knowledge resource provides a much greater challenge to the evaluation process. It is important to note when manual evaluation is required, when automatic evaluation is preferred, and how the two can be combined to display a more accurate value of the described experimental process.

In the following sections, we will discuss each of these components in turn, describing an evolution in the methodology used in each by referencing publications covering the last 25 years of research in this and its related fields. In section 4.3 we will discuss the various goals that researchers have pursued in these fields; in section 4.4 we will present the myriad techniques used to achieve language learning goals set forth in the prior section; in section

4.5 we will discuss the various corpora used in these experiments, and in the final section, 4.6, we will investigate how evaluation was performed under a variety of experimental circumstances.

## 4.3 Prior Research: Goals

This section will describe a variety of goals commonly set forth in the literature that relate to our language learning research. Many of these goals have direct parallels with the research presented in the coming chapters, while others are tangent, but not unrelated. The intention is to present a variety of focuses that have been pursued, and to show their evolution over time. We will only discuss the aims of each experiment in this section; in the following sections we will discuss in greater detail the techniques, data, and evaluation methods employed by the researchers to forward their individual research.

### 4.3.1 Goals: Syntax or Semantics

Automatic knowledge extraction from text, at a very coarse grain, can refer to (commonly) one of a pair of very high level goals: syntactic learning, and semantic learning. As discussed in Chapter 3: OntoSem, the differences between syntactic knowledge (and its representation) and semantic knowledge (and its representation) are vast. Although commonalities can occur, depending upon implementation, by and large syntactic knowledge deals with the mechanical structures of a sentence: part-of-speech tagging, phrase identification, and synonymy to name a few. Semantic knowledge, on the other hand, describes the actual meaning of a word, phrase, or concept in either a human- or machine-readable form. Learning automatically from text can refer to either learning syntax, or semantics.

Although our research focuses primarily on semantic learning, it is important to inspect where prior research in syntactic learning crosses our path. In the case of syntax,

rules for automatically part-of-speech tagging words can be learned from a bootstrapped system as demonstrated in (Brill 1995). (Krymolowski & Roth 1998) showed that semantic knowledge could be used to enhance a preposition phrase attachment engine, an interesting take on building syntactic resources from semantic ones (rather than the more common reversed direction). Verb subcategorization lexical entries have been learned using an existing parser in (Basili, Pazienza, & Vindigni 1999). And syntactic patterns were also shown to be part of a trinary system of equilibrium (along with a growing set of documents and an ontology) in (Iria *et al.* 2006).

Learning semantics is often considered more challenging, but equally more interesting in terms of usefulness. Chapter 2: Motivation, described many possible uses for semantic knowledge. Often, learning in this field further focuses on learning lexical semantics, or ontological semantics: the research may be interested simply in the meaning of the word, or they may be interested in where a word belongs in a hierarchy of other words (whose contents may be either light or heavy in semantic description). Lexicons have been shown to be automatically populated with semantic knowledge in (Zernik & Dyer 1985), (Mooney 1987), (Cardie 1996), (Thompson & Mooney 1998), (Hahn & Marko 2002), (Gupta 2005), and (Kohomban & Lee 2005).

The goal presented in (Zernik & Dyer 1985) was to automatically augment a phrasal lexicon from figurative phrases. An example provided was ”David took on Goliath.”; the system’s intent was to extract the hidden meaning of the figurative phrase ”took on”, in this case implying ”challenged” or ”faced”. (Mooney 1987) aimed to learn the meaning of new words or phrases presented in narrative, by understanding (through language parsing) the actions taken by the characters. (Cardie 1996), (Thompson & Mooney 1998), and (Hahn & Marko 2002) directed their focus on learning the meaning of unknown words through language parsing of text, using context to assimilate the semantics of new knowledge; although their goal was primarily similar, the methods employed varied (and are discussed in

the next section). (Gupta 2005) also aimed to learn unknown words through text analysis, however the resources used included OntoSem, which the research we present intimately involves. (Kohomban & Lee 2005) took a different approach to lexical learning, by leveraging existing semantic resources (WordNet) and attempting to map lexical terms to the pre-defined semantic classes.

Ontology learning, a subset of semantic learning, gained particular attention with the buzz surrounding the semantic web (Palmer 2001). However, even prior to the popularity presented by the prospects of a knowledge-rich internet source, ontology learning was already in full swing: semantic classes (ontology frames, without necessarily subsumption links) were targets or knowledge learners including: (Mooney 1987), who associated learned words with new semantic classes from narrative; (Gomez, Hull, & Segami 1994), in which semantic knowledge was constructed from encyclopedic text; (Clark & Porter 1997) which constructed views of existing semantic information automatically to facilitate question/answering; linking previously unknown words into existing semantic classes as demonstrated in (Hahn & Schnattinger 1997); learning new semantic classes from language processing ((Esposito *et al.* 2000), (Reinberger 2004)); (Iria *et al.* 2006) showed how to simultaneously learn semantic classes while maintaining a complete set of syntactic patterns and expanded corpus documents; finally (Clark & Harrison 2009) presented automatic semantic class population coupled with assessing its use in various NLP tasks.

As important as filling the contents of semantic classes, automatically populating an ontology's structure (subsumption links) from text, has been an equally popular task. In 1993, (Futrelle & Gauch 1993) displayed how unknown words can be constructed into a binary tree, using similarity clusters. (Hahn & Schnattinger 1997) populated semantic classes and linked them in to an existing ontology, and extended the research in (Hahn & Marko 2002); (Gupta 2005), (Iria *et al.* 2006), and (Betteridge *et al.* 2009) also displayed ontology construction using semantically rich classes. However, building ontological structures

without heavy semantic information in each class has also been a common goal: (Faure & Nedellec 1999) showed how this could be done using syntax parsing and user intervention; (Iwańska, Mata, & Kruger 2000) also produced taxonomic knowledge, fully automatically, and without any traditional NL tools; (Maedche, Maedche, & Volz 2001), (Ogata & Collier 2004), and (Seo, Ankolekar, & Sycara 2004) all learned ontologies using light parsing and data mining; and (Cimiano, Staab, & Tane 2003) used formal concept analysis to construct a taxonomy of domain specific words.

### 4.3.2 Enhanced Parsing and Reasoning

In the previous section, we described research whose primary motivation was the augmentation of static knowledge. The use of the knowledge after its extraction was a secondary goal, and was not primarily emphasized. This section will enumerate research in language learning whose primary goal is to enhance existing parsing or reasoning engines. The researchers have focused on an existing task, and have attempted to improve their current results by automatically improving the the static knowledge at their disposal.

(Riloff 1996) aimed to automatically produce domain-specific dictionaries for directed information extraction tasks: the goal was to automatically assemble the requested information by generating patterns and templates. In (Craven *et al.* 2000), information retrieval was used to populate probabilistic statements of knowledge using web data, effectively providing a weighted answering system. Question/answering systems (or similar knowledge bases) have also been supported by automatic language learning in (Frank, Farquhar, & Fikes 1999), (Aussenac-Gilles, Bibow, & Szulman 2000), (Clark *et al.* 2007) and (Bobrow *et al.* 2009). An intriguing line of research that attempted to learn language by associating it with non-linguistic action was presented in (Fleischman & Roy 2005). Finally, the task of text classification was approached using a form of language learning in both (Lavelli, Magnini, & Sebastiani 2002) (who put a spin on the task, focusing on individual words,

rather than entire text) and (Ankolekar 2003) where ontologies and the semantic web were applied.

## 4.4 Prior Research: Techniques

In the prior section, we touched on a variety of goals that drive research in the broad application areas of language learning. Some research focuses purely on the learning of language (syntactic or semantic), while other research uses language learning to further a separate, yet related, task. In this section, we will investigate, in depth, the techniques used in the language learning tasks presented in the previous section. Language learning from text takes many forms: the learning can be knowledge-lean, with a focus on statistical classifiers, or it can be knowledge-heavy, relying on already robust knowledge bases and NLP toolsets; it can be directed in scope, relying on the nuances of a particular domain of text, or it can be broad in coverage, attempting to learn knowledge from general texts.

### 4.4.1 NLP-light and Semantic-free Language Learning

Historically, language learning (often to support information retrieval) has been well supported by research that uses little to no traditional NLP tools, and may operate without any deep semantic analysis. Research in this vein tends to rely heavily on statistical filtering and clustering techniques, heuristics applied to light syntactic parsing (possibly tagging only), and sometimes little to no bootstrapped static knowledge beyond what is needed to facilitate the syntactic processing. Although our research shies away from this direction, it is important to note the history of research in this area.

Often, research in language learning that is NLP-light is also domain specific: not having access to a general purpose language parser can necessitate the need to focus on a particular domain of texts for constructing light parsers and learning heuristics. In the

case of (Riloff 1996), the target text was domain specific, however the heuristics produced were domain independent. Requested domain specific information was extracted by finding untagged target words, performing a syntactic analysis on that word, and then using the domain independent heuristics to locate trigger words and assume semantic knowledge from them. (Thompson & Mooney 1998) removed the syntactic parsing entirely from the equation, and constructed lexicons from hand annotated corpuses using greedy heuristics.

Further work in the field of domain specific, NLP-light language learning focused on constructing semantic dependencies, often in the form of ontological hierarchies: (Faure & Nedellec 1999) used syntax parsing only to construct an ontology by clustering relations from a parser, then presenting a user with a validation request as each level of the ontology was assembled. In (Cimiano, Staab, & Tane 2003), selectional restrictions were imposed by verbs in order to create a list of relationships between words using only a statistical parser and probability calculations (for noise reduction); these restrictions were converted into a taxonomy of domain-specific terms. (Reinberger 2004) aimed to extract not only an ontological hierarchy, but to also populate its frames with relations, using shallow parsing, and unsupervised learning on domain-specific texts. Selectional restrictions were applied to find subject-verb and object-verb relations, which converted into semantic constraints. (Ogata & Collier 2004) focused on constructing an ontological hierarchy only, using typing information (e.g. ”a human is a type of mammal”), and then having the results validated by domain experts. The same year, (Seo, Ankolekar, & Sycara 2004) also focused on constructing an ontological hierarchy, but rather than using typing information, four statistical feature selectors were employed (mutual information, chi-squared, Markov blanket, and information gain (Duda, Hart, & Stork 2001)). The input corpus was scanned for common ”feature” words, which were eventually combined into an ontology.

Rarely, language learning in a domain uses untraditional techniques, as in the research presented in (Fleischman & Roy 2005). Two players entered a virtual world, where one

player commanded a second player through a series of navigational challenges. The instructions and actions taken were recorded, and the actions were mapped to a custom ”ontology” of terms. Statistical correlations were then applied between the instruction text, and the resulting actions, effectively mapping language to an ontology. Producing a knowledge base from domain independent (or general purpose) text, without the aid of deep semantic processing can be particularly tricky: the introduction of noise from polysemous words due to the open nature of the text can often confuse statistic methods that rely heavily on string matching techniques. Regardless, the lure of a general purpose language learner without the overhead of comprehensive NLP tools is strong, and much research has been done in this area, particularly in the vein of improving the parsing mechanisms through language learning.

In (Futrelle & Gauch 1993), a binary tree of words was automatically constructed by combining the context of the target word in a text (defined as a window of two words to either side of the target) with the the mutual information of the context words in order to generate a similarity vector. These vectors were used to construct a binary tree, which was shown to emulate simsets and semantically similar clusters. (Brill 1995) attempted to enhance the parser itself by automatically learning tagging rules from a manually annotated corpus by tagging the corpus and then calculating the error, which provided a delta of changes that could be applied to the tagging rules themselves. (Krymolowski & Roth 1998) enhanced a propositional phrase attachment engine using the semantic knowledge from WordNet (Miller 1995) to automatically grow the list of features applicable to the engine.

Although these three examples focus on immediately improving parsing capabilities, much effort has been put into actual knowledge extraction for static resources, which often doubles as an improvement to parsing as well. (Frank, Farquhar, & Fikes 1999) used an existing manually-annotated knowledge-rich web site, and converted it into a knowledge

base and a formal ontology by using a custom parser, and a series of rules that push the knowledge from the parser into the hand-made ontology. In (Basili, Pazienza, & Vindigni 1999), verb subcategorization lexical entries were learned using inductive learning (Duda, Hart, & Stork 2001) on the parses that were produced by a lexicon-free parser. The parses were clustered, which allowed for the construction of lightweight syn- and sem-struc fields for the verbs. Semantic knowledge is also extracted, using a lightweight NLP system in (Esposito *et al.* 2000), which makes use of context free grammars and an information extraction learner. The knowledge learned is also re-checked against future information to affirm its correctness. (Clark & Harrison 2009) automatically populated a database of tuples of the form (subject ”helicopter” ”land”) using a broad coverage phrase parser; the frequency of each tuple was used to calculate its mutual information, which was used to assess the usefulness of the learned knowledge in other NLP tasks.

(Iwańska, Mata, & Kruger 2000) showed that taxonomic structures could be constructed fully automatically from open text using no real parsing or pre-existing knowledge bases by building a dozen patterns patterns such as ”X such that Y”. (Maedche, Maedche, & Volz 2001) also constructed taxonomic structures, but used a light NLP system, supplemented with regular expressions and statistical clustering over web data to produce a standardized ontology that could then be manually modified for correctness. Eschewing a fully define ontology, (Lavelli, Magnini, & Sebastiani 2002) aimed to group terms into lexical clusters, similar to text categorization, using machine learning and information retrieval techniques (such as boosting (Duda, Hart, & Stork 2001) with manual intervention) on a subset of nouns from WordNet.

Using an existing ontology as a form of decision tree (Duda, Hart, & Stork 2001), (Gupta 2005) performed latent semantic analysis (LSA) (Landauer & Dumais 1997) and single value decomposition (SVD) (Will 1999) to identify similar words and produce a similarity score, directly from raw text. Using a semantic lexicon (a mapping of lexical terms

to semantic frames in an ontology), the lexicon was expanded by comparing target words to existing words at each branch of the ontology, descending until reaching a leaf. (Kohomban & Lee 2005) also aimed to map lexical terms to semantic classes, using WordNet, rather than the semantic ontology used by OntoSem in the previous example. Three classifiers were trained on an annotated data set, and used a context window (three words to either side of the target) to suggest a top level concept for the target word. Finally, (Betteridge *et al.* 2009) demonstrated a never-ending language learner which self enhances its ontology by identifying noun phrases from web text that are members of a semantic class. Using the bootstrapped knowledge found in the ontology, the system locates new knowledge by querying the web, and the results are fed back into the system, restarting the cycle.

### 4.4.2 NLP-rich and Deep Semantic Language Learning

In contrast to research that has developed to meet the needs of language learners without the overhead of an extensive NLP toolset, learners have also evolved using existing deep semantic resources and comprehensive (sometimes general-purpose) language parsers. Research of this nature often relies on a heavily bootstrapped system, complete with static knowledge (lexical and ontological, extensive pre- and post-processors and heuristics.

Although infrequent, domain-specific language learning using deep semantic understand has been investigated. As techniques in language learning using classic NLP often require both a general-purpose parser, and knowledge, it is often less necessary to target a particular domain of knowledge as it can be when using a more statistically driven method (as described in the previous section). However, (Gomez, Hull, & Segami 1994) targeted a domain on animal eating habits, and ran a learner over encyclopedic texts on the topic. A series of inference rules were created to directly assist the domain; the example given referred to the phrase ”digs out”, which has an implied meaning of ”to eat” when the agent and theme are both animals. It seems that manual intervention was allowed between each

step in the processing to filter out erroneous parses. In (Craven *et al.* 2000), the construction of a web site (particularly, a university web site containing faculty pages) was used in conjunction with NLP and custom heuristics to construct a domain-specific ontology describing the information contained within the site. They assumed a given web page to be a class instance in the ontology, and a hyperlink to be a relation between two instances. (Hahn & Marko 2002) used text parsing to identify unknown words, and applied selectional restrictions using known words to construct both a lexical and ontological view of the new word, which was added to the existing static knowledge. In order to provide logic for a knowledge base, (Clark *et al.* 2007) manually simplified input text to a controlled language, which was then parsed into a logical form which integrated with an existing knowledge base on the target domain; the results provided effectively a question/answering system in logical form.

More traditionally, the flexibility granted by having robust NLP tools and/or a deep semantic analysis encourages a more general-purpose language learner. (Zernik & Dyer 1985) worked on automatically augmenting a phrasal lexicon by parsing input text, and flagging any phrases that were not properly understood, and disambiguating using increasing specificity of context. Lexical entries and ontological concepts were learned from narratives in (Mooney 1987), where a parser identified goals set forth by the characters, and then attempted to explain their actions, leading to broadened static knowledge resources.

(Cardie 1996) aimed to learn both syntactic and semantic structures by identifying unknown or ambiguous words, and creating a traditional machine learning classification task to handle them; the entire task was performed with only a few dozen ontological frames and a lexicon of just over one hundred function words. Similarly, (Hahn & Schnattinger 1997), and (Bobrow *et al.* 2009) also sought to learn general purpose lexical and semantic knowledge from raw text. (Hahn & Schnattinger 1997) linked unknown words to an existing ontology by parsing text, and applying selectional restrictions which proposed a set of

learning hypotheses for the target word. Increasing evidence improved a credibility score to the hypothesis, eventually suggesting a proposed location of the target word in the ontology. It was noted that the research did not account for polysemous words. (Bobrow *et al.* 2009) aimed to facility question/answering by parsing text into an abstract representation that could be mapped to an existing knowledge base: WordNet.

In (Clark & Porter 1997), it was shown that an existing knowledge base can be manipulated to provide reasoning to a question/answering system by automatically constructing new information by building a different view over existing data. High level concepts were combined and then recursively refined until a script containing a path from the question to the answer was constructed; the process was highly reflective of constraint satisfaction problems. Existing tools were leveraged in (Aussenac-Gilles, Bibow, & Szulman 2000) to allow a knowledge engineer to nearly manually elicit desired knowledge from a corpus by effectively stringing them together. Finally, (Iria *et al.* 2006) introduced a new paradigm in text learning by giving equal emphasis to the ontology, the corpus, and the learning patterns (analysis techniques), using each as resources in a lifetime learner. By maintaining an equilibrium between the three, which is handled by event triggering when one is changed, the system was demonstrated to continue to extract new knowledge from text.

## 4.5 Prior Research: Corpus

Regardless of goal or technique used in language learning research, one component of the experimental process is required: an input corpus. In order to extract knowledge from text, one must have a set of documents to be scanned, parsed, analyzed and filtered. In this section we'll discuss the various corpora used in the research presented in the prior sections. In some cases the input is raw, and general purpose, other times it is hand selected, manually annotated, and domain specific. The choice of corpus is often related to the goals

and techniques of the researchers, but sometimes may be dictated by external parties (as in the case of open competitive ”tasks”).

Rarely, no formal corpus is provided. This may be due to a lack of a formal experiment itself, as in the cases of (Aussenac-Gilles, Bibow, & Szulman 2000), (Betteridge *et al.* 2009), and (Bobrow *et al.* 2009). However, the absence of a formal corpus may be due to the unusual nature of the experiment: (Zernik & Dyer 1985), which focused on learning figurative phrases seemed to derive input from a dialog between the system and the a native speaker. (Mooney 1987) provided no corpus, but did provide a few examples of the narrative text that was targeted for language learning, including a narration of a kidnapping event. In (Clark & Porter 1997), again no formal corpus is given, as the goal is to rearrange existing knowledge bases, rather than to learn directly from text. A sample question that was provided as input to the system was: ”What is the cost of the equipment required for microbe-soil-treatment?”; rather than scanning an input corpus of text for the answer, an existing database of information was crawled to produce a response. (Fleischman & Roy 2005) constructed the training corpus and the experimental input directly from the interaction between the two players of the game; although they provided no formal corpus, it is clear that constructing one was a large part of the unusual experiment.

When a formal corpus is referenced or included, it often takes one of three forms: rarely, a hand selected corpus is chosen, often when dealing within a particular domain. Frequently, an annotated corpus is used: these corpora tend to be from various tasks (either current or past), and have been manually annotated to provide a solid method for evaluation. Using these corpora allows researchers to more easily compare their findings to those of other researchers in the field. Finally, there is a growing trend in recent years to use raw, un-annotated text, often from the web or a similarly open and accessible resource.

Examples of hand selected corpora include: (Futrelle & Gauch 1993), which aimed to create a binary tree of terms from the biology domain. The corpus was manually built

by a specialist using titles and abstracts from publications in the field. It was noted that the corpus, being 200,000 words in size, was small compared to those used by contemporaries. (Thompson & Mooney 1998), also working in a limited domain (geography), created 250 questions manually and paired them with hand-annotated logical representations. Similarly, (Seo, Ankolekar, & Sycara 2004) selected a subset of texts from the commonly used Reuters-21578 corpus that were relevant to the domain of interest, and used the newly formed corpus to construct an ontology using statistical feature selections. In (Clark *et al.* 2007) logic was learned for a knowledge base by feeding the system six hand selected pages from a chemistry textbook (again, domain specific) which had been manually modified into 280 simplified language sentences.

More commonly, large scale, domain unspecific corpora that have been manually annotated (often by external groups) are used in order to easily provide comparable results. An early example, (Brill 1995), uses the very frequently used Penn Treebank tagged Wall Street Journal corpus, which consists of over one million words; the researchers used a subset of the corpus to train a part-of-speech tagger using the annotated features, and were able to test the learnt rules on the remainder of the corpus. (Krymolowski & Roth 1998) also used the Penn Treebank corpus to enhance a preposition phrase attachment engine, another use of that corpus to learn methods for language parsing, rather than directly supplementing static knowledge resources. A third use of the Penn Treebank corpus for the purposes of augmenting an existing parser is shown in (Basili, Pazienza, & Vindigni 1999) where verb subcategorization lexical entries are automatically populated.

Other editions of the Reuters corpus (including Reuters-21598) were used in (Lavelli, Magnini, & Sebastiani 2002), (Ankolekar 2003) and (Clark & Harrison 2009). (Lavelli, Magnini, & Sebastiani 2002) aimed to categorize lexical terms using the first volume of the Reuters corpus, which was specifically designated for such a task. (Ankolekar 2003) used the annotations provided to focus on four domains (choosing the words ”cocoa”, ”copper”,

”cotton” and ”nat-gas”), and empirically studied the effect the semantic web has on text learning. (Clark & Harrison 2009) automatically populated an ontology using both the Reuters corpus as well as the British National Corpus (BNC), and evaluated their system with a third, the annotated Brown corpus. The Brown corpus (-1 and -2) were also used by (Kohomban & Lee 2005) who mapped learned words to coarse grained semantic classes in WordNet.

Another commonly used pair of annotated corpora, MUC (Message Understanding Conferences) and Tipster were employed in (Cardie 1996) in order to learn syntactic and semantic structures. Various MUC corpora were also used in (Riloff 1996) in a domain specific learning task, while (Frank, Farquhar, & Fikes 1999) parsed the highly marked up CIA's World Fact Book (in XML format) into a query-able knowledge structure.

(Craven *et al.* 2000) constructed a unique corpus to suit the unusual nature of their research, attempting to construct a knowledge base using both the content and the structure of a web site as input. They gathered over 8000 web pages from multiple faculty computer science departments, then hand-annotated the text using their custom ontology. The pages were gathered using existing index pages, and were limited to a small (2000 word) vocabulary.

Recently, the use of large-scale un-annotated texts as input to language learners has increased, largely due to the accessibility of web data, along with a growing interest in annotating that data to supplement the notion of the semantic web. However, prior to easily accessible large-scale web data, (Gomez, Hull, & Segami 1994) aimed to extract semantic data from open (albeit domain specific) text using the World Book Encyclopedia as a corpus. (Hahn & Schnattinger 1997) also sought to learn semantic information from text, aiming to map new words into an existing ontology, using open text from information technology magazines (a second domain specific case). A third example of earlier open, but still domain specific texts is presented in (Faure & Nedellec 1999), where recipes were

used to study basic language learning, and technical patents were used to study highly domain specific language learning. Taxonomic knowledge was learned by (Iwańska, Mata, & Kruger 2000) from raw, un-annotated articles from Time Magazine and the Wall Street Journal. (Hahn & Marko 2002) reported automatic lexical and ontological augmentation from a small set (48) of domain specific texts, whose origin was not noted. Ontology learning including relation extraction was demonstrated in (Reinberger 2004) using the medical domain texts available from MedLine. Two corpora were constructed by querying MedLine for the search strings ”hepatitis A” & ”hepatitis B” (for the first corpus) and ”blood” for the second corpus.

Using the web, rather than an independently distributed corpus, even for domain specific learning has become increasingly popular recently. (Esposito *et al.* 2000) acquired domain specific texts from the web, focusing on learning semantic definitions from commerce texts that were publicly available. Similarly, (Maedche, Maedche, & Volz 2001) learned ontologies from un-annotated domain specific web texts, which were noted to be populated with both markup and web junk. In the same vein, (Cimiano, Staab, & Tane 2003) also learned ontologies from un-annotated domain specific web texts, coming from a publicly available corpus on travel, totaling roughly one million words. Using a more restricted web search, (Ogata & Collier 2004) constructed three data sets on molecular biology by combining abstracts from publications in the field using targeted key words, and then constructed an ontological hierarchy automatically using statistical clustering. (Gupta 2005) used a combination of offline texts (the Wall Street Journal) and an online resource (Google’s ”define” functionality), along with the lexical resources available in the OntoSem system to construct a corpus of texts for lexical learning. Finally, (Iria *et al.* 2006) demonstrated a never-ending language learner that automatically augmented its internal corpus by querying the web for open, raw texts as needed.

## 4.6 Prior Research: Evaluation

The fourth component of prior research that we will discuss is the evaluation metric used to assess the quality of the experimental results. The focus of this section will be on how evaluation was performed, and not on individual results. Evaluating the results of a language learning experiment can often be a tricky task, however it is necessary to properly report the quality of results in a way that both emphasizes their qualities and spotlights their failings. Evaluation often depends on the goal of the research: an experiment focused on augmenting an existing parser can evaluate its findings by comparing parses before and after augmentation (assuming such a comparison metric has already established). An experiment that focuses on learning syntactic or semantic structures without a directed use for them has a tougher time providing a solid evaluation metric. Evaluation can compare to prior findings in the field, to prior findings amongst the research group, or even to a manually crafted standard.

In some cases, no formal evaluation is provided at all. This may indicate that the work was not easily evaluated, or was not ready for evaluation, as in (Aussenac-Gilles, Bibow, & Szulman 2000), (Esposito *et al.* 2000) and (Betteridge *et al.* 2009). Sometimes, informal evaluation is given in a very subjective tone, as in (Zernik & Dyer 1985) where the only evaluation provided was a claim that "...RINA can learn a variety of phrasal verbs and idioms." (Faure & Nedellec 1999) also gives no evaluation, but suggests that the ontologies learned by their system could be plugged in to an existing pipeline, which could show efficiency improvements. The difficulty of providing an independent and automatic evaluation of learned ontologies was emphasized. Similarly, (Frank, Farquhar, & Fikes 1999) suggested that the real use of their automatically parsed knowledge bases was in inference tasks (such as question/answering), however no inference engine was implemented or used to demonstrate evaluation of the main line of research. Although (Iria *et al.* 2006)

does not present a formal evaluation of their unique equilibrium system, they do discuss a likelihood metric which suggests whether an object in their system (ontology, document, or phrase parser) belongs to a given domain. Finally, (Bobrow *et al.* 2009) provides an evaluation metric, with no associated formal results: questions are presented to the question/answering system whose knowledge base was automatically populated, facilitating an evaluation of the knowledge by comparing the responses of the system to known answers.

Rarely, a formal evaluation will be eschewed in exchange for discussion of preliminary results, sometimes accompanied by a few examples of output. Often this indicates that the system is not yet ready for a formal evaluation, but the authors would prefer not to ignore this necessary point entirely. (Mooney 1987) follows the narrative example provided throughout the report into preliminary results, but makes no claims to the quality of the output. In (Clark & Porter 1997), the question/answering system described is demonstrated with one reasonable example, but no extensive evaluation.

The most common form of evaluation in the field of language learning stems directly from the corpus used during the experimentation. A corpus that has been previously annotated as part of past research, a competition or ”task”, or simply as a means to provide fair comparisons between research teams, makes for an easy source of evaluation metrics. With the input to the system also serving as an oracle, researchers are free to conveniently, and automatically, calculate precision, recall or a host of customized numeric evaluation attributes. Further, these results can be compared to previous work by other researchers using the same corpus and targeting the same goals. Examples of this method of evaluation include (Brill 1995), (Krymolowski & Roth 1998) and (Basili, Pazienza, & Vindigni 1999) who all used the Penn Treebank Wall Street Journal annotated corpus. Of these, (Basili, Pazienza, & Vindigni 1999) investigates further evaluation by demonstrating the improvement of the parser after the learner's task, also by comparing the results to the oracle provided by the corpus. (Cardie 1996) used the annotated MUC and Tipster corpora

for their lexical and semantic learning experiments, and were able to provide simple evaluation using the accessible correct data available. Also, (Kohomban & Lee 2005) used the annotated SemCor corpus (Brown-1 and -2) to evaluate their experiment in knowledge acquisition; the popular corpus allowed the results to be compared to other research. Additionally, a baseline best guess, which assumed a perfect classifier but an imperfect subsumption crawler was calculated and used as a comparison metric.

In (Lavelli, Magnini, & Sebastiani 2002), the annotated Reuters corpus was used for text categorization, and was divided into a pair typical training and test sets. Precision and recall were calculated and averaged. The authors noted that a baseline for their research did not exist to compare to. (Ankolekar 2003), also using the annotated Reuters corpus, focused heavily on evaluation, their goal being to identify how ontologies affect text classification. Using the oracle provided by the corpus, they were able to calculate results for both a feature selection, and classification experiment. Evaluating the feature selection experiment involved calculating how many documents were required input to achieve an expected coverage of the target ontology; they found that beyond a certain threshold, expanding coverage took an exponential increase in input size. The classification task was evaluated by calculating precision and recall using varying feature coverage in an attempt to identify an area of maximum gain.

Occasionally, researchers would prefer to follow conventional evaluation paradigms, but do not have a pre-tagged corpus available. In these situations, it is often the solution to manually annotate the corpus used as part of the task itself. (Riloff 1996) created a hand annotated answer sheet to compare the results of their information extraction task, which was fueled by the language learning experiment presented. In this was, they could judge the improvement of quality in the IE system, following the automatic population of knowledge. (Thompson & Mooney 1998) also used a hand-annotated results sheet, and added a further level of comparison by judging their findings against other work in the field.

In (Craven *et al.* 2000), the unique corpus constructed for the research (a set of interlinking web pages from CS departments) was hand annotated, and the results were evaluated using leave-one-out four-fold cross validation.

(Hahn & Marko 2002), focusing on automatic augmentation of both a lexicon and ontology calculated precision and recall on the collection of learned words by manually creating gold standards based on the corpus to compare to. (Ogata & Collier 2004) pre-labeled the input data manually to act as an oracle as well, assisting in evaluating their automatically produced ontological hierarchies. Very similarly, (Seo, Ankolekar, & Sycara 2004) also pre-labeled the input to map to a custom ontology for evaluation. And in (Clark & Harrison 2009), a hand-annotated corpus was coupled with human evaluation on a subset of the tuples generated to provide multiple vectors of evaluation.

Beyond a lack of evaluation, minimal evaluation, pre-existing oracles, or hand-annotated corpora, some evaluation methods are often too unique to properly classify. Often, the more unusual the research direction, the more likely the evaluation method must be customized in order to properly demonstrate the quality of the results presented. An early example can be found in (Futrelle & Gauch 1993), where the binary tree of words constructed was compared to entries in Roget's Thesaurus, in order to judge the quality of the simsets presented by the learner. (Gomez, Hull, & Segami 1994) evaluated a system of knowledge extraction on encyclopedic texts by applying a question/answering system, and judging the expected results from posed questions, such as: "Which birds eat nectar?". In (Reinberger 2004), the proposed knowledge used the UMLS (Unified Medical Language System) as an oracle, but conceded that not all of the words learned in the research could be evaluated, as some were not present in the UMLS. The research presented in (Fleischman & Roy 2005), which tied language to non-linguistic responses that were mapped to an ontology was evaluated by entering text into the system, and determining if the correct ontological responses were returned: essentially, they were testing to see if the game could

play itself with only natural language instructions provided. And in (Clark *et al.* 2007), the knowledge base crafted was tested via a question/answering system that was fed AP chemistry questions that were known to be answerable using the existing controlled logic. The results provided were further compared to the results of the same experiment prior to the inclusion of the automatically populated knowledge base, in order to demonstrate a baseline of improvement.

When working in ontology construction, (Hahn & Schnattinger 1997) showed that evaluation could be performed by judging the gap between where a word was placed in the structure, compared to where it should have been placed, traversing subsumption links to calculate a distance. (Iwańska, Mata, & Kruger 2000) took a different approach, having human judges deem if the learned taxonomic knowledge was accurate on an A, B, or C scale. (Maedche, Maedche, & Volz 2001), on the other hand, compared the ontology produced with a gold-standard ontology, and judged precision and recall based on inclusion in the ontology, and not on accuracy of placement. In (Cimiano, Staab, & Tane 2003), the automatically constructed ontology was compared to five manually created ontologies using the same term list, and found vast differences in evaluation results; they concluded by demonstrating that the manually created ontologies did not really agree with one another, and thus may be a poor mechanic for evaluating an automatically created one. Finally, (Gupta 2005) used leave-one-out cross validation on the corpus of words, calculating precision as the number of words that were correctly classified into the ontology. The lexical word had to map to the correct ontological frame in order to be counted as a positive example.

In this chapter, we've explored a broad set of published research goals with ties in automatic language learning, from knowledge base construction for question/answering systems to pure ontology construction. We've discussed the techniques used by researchers, ranging from shallow parsing and statistical clustering to semantic analysis and ontological ordering. We've examined the various corpora used in these experiments, including pub-

lished, hand-annotated texts, open web documents, and custom collections of data. Finally, we've investigated the evaluation techniques used to quantitatively and qualitatively judge the results of the presented work, including the use of existing oracles, hand-crafted answer sheets, and human evaluators. In the next chapter we'll discuss the techniques that will be presented in this research.

Chapter 5

# EXPERIMENT OVERVIEW

## 5.1 Introduction

Automated language learning can take many forms and have many different goals, as evidenced in the previous chapter. To construct a unique language learning experiment, we aimed to build our foundation on the unique tool available to us: OntoSem. Using OntoSem as a base, we began to formulate hypotheses and eventually constructed a skeletal experimental pipeline, which would be fully realized over twelve individual experiments described in Chapter 7: Experimental Setup and Evaluation Processes.

OntoSem exposes a special set of processes that allowed us to investigate language learning in a more intuitive sense: rather than needing to rely on traditional, statistical approaches which are heavily based on string matching, we could leverage the deep semantic meaning OntoSem would provide, allowing us to direct our machine-learning algorithms towards knowledge, rather than towards characters. Essentially, we could approach language learning in much the same way humans would: when encountering an unknown word, contextual clues can be leveraged to construct a meaning. This foundation allowed us to propose several advanced hypotheses in the field as well as design a robust test suite for experimentation.

However, OntoSem was not the only prerequisite for our experiments: we would also

need access to an open, domain-independent and queryable corpus, as well as a toolset for streamlining the experimentation and visualizing our results. The web provided a solution to the first requirement, however the second would require a series of engineering tasks that were necessary to further the research.

In this chapter we'll discuss an overview of our goals and experiments: in the following section we'll present our hypotheses for language learning; in section 5.3, we'll describe a coarse-grained view of the experimental pipeline; finally, we will also discuss all of the prerequisites for the proposed experiments, and expose the solutions we found or constructed for each.

## 5.2 Hypotheses

Parallel to our language learning research, our work on the OntoSem system aims to produce (among other things) deep semantic representations (TMRs) from raw text using a manually populated semantic lexicon and ontology. Naturally, the knowledge acquisition bottleneck presents an expectedly difficult problem. Automating knowledge acquisition, a commonly shared goal, takes on particular difficulties when attempting to apply common research efforts (often aimed at mapping strings into WordNet, or clustering similar texts together) to a system that, by definition, defies conventional means of text analysis.

As a result of this observation, a custom solution to language learning must be approached, and with it, a series of experimental goals. We put forth three hypotheses for language learning when using a knowledge-rich semantic analysis engine, which, while similar to some goals covered in the previous section, describe anticipated (and desired) behavior in such a learning environment with unique foundations.

Initially, we put forth that a semantic analysis engine, in any state of development, could use itself to bootstrap a learning algorithm that would further expand on the engine's

own static knowledge resources. More specifically, we hypothesized that OntoSem could provide its own semantic analysis (involving its lexical and ontological resources) to learn the meanings of unknown words, and formulate them into new lexical and ontological resources, creating a self-supporting language learner.

Our second hypothesis, more a corollary to the first, states that broader coverage of static knowledge in the semantic analysis engine will result in broader coverage in learned knowledge and / or increased accuracy in learned knowledge. For our purposes, this implies that the more lexical and ontological resources OntoSem has available when bootstrapping the learner, the better the learned knowledge (candidate concepts) will be.

Finally, we propose that although a quantity of inputs can assist to statistically drown noise from a corpus, when dealing with a semantic analysis engine, higher quality inputs will result in significantly increased accuracy of outputs from the learner. Specifically, we suggest that if OntoSem is provided with a handful of contentful, accurate, and clean texts, the TMRs produced will be far more valuable to a language learner than those produced by OntoSem when given a large collection of malformed, inaccurate or noisy texts. In this section, we'll discuss each of these hypotheses in detail.

### 5.2.1 A Self-bootstrapping Lifelong Learner

Our primary hypothesis proposes that a semantic analysis engine (in our case OntoSem), at any level of development beyond basic functionality, can be used to bootstrap a learner whose output can be fed back in to the engine, thus improving its own performance and dodging the knowledge acquisition bottleneck. OntoSem, at any given point, has a standing ontology, lexicon, and collection of processes, algorithms and microtheories that combine to facilitate the production of a TMR from raw text. Although the input provided to OntoSem may not be fully covered by the static knowledge, a TMR will be generated regardless, implying a key caveat to this hypothesis: the semantic analysis engine must not

fail on any input. Although the methods used by individual implementations may vary, an approximation of OntoSem's capability to relax constraints on unexpected inputs, effectively generalizing the unknown word, is required.

Presuming a TMR, or equivalent machine-tractable representation of a text can be produced, and unknown words in that text can be generalized using selectional (semantic) restrictions, syntactic restrictions, or similar techniques, a learner can be introduced that will collate the data presented into a semantic representation of the unknown word. This candidate knowledge can then be turned around, and placed back in to the semantic analysis engine, broadening its coverage, and allowing for more accurate learning to continue indefinitely. OntoSem produces TMRs from raw text, which are comprised of a set of frames, each an instance derived from an existing ontological concept. These frames are populated with property / filler pairs, exactly as concepts in the ontology, thus making the job of the learner one of organizing this information, and eliminating the need to engineer mapping functions between data representations.

By converting a collection of the contents of the TMR frames generated from the unknown target word into a set of candidate concepts (using clustering techniques, heuristics and decision trees), the proposed learner can inject its results directly back into OntoSem's ontology, giving the semantic analyzer broader coverage of knowledge. Doing so implies a greater capacity to understand the context of other unknown words, fueling a lifelong learner. Intuitively, when presented with a sentence full of unknown words, a human reader would be hard-pressed to understand the meaning; when presented with a sentence with only one unknown word, the contextual clues assist the reader in understanding. Once some definition of the unknown word has been formalized, the next encounter (the original sentence of only unknown words) will be somewhat easier to follow. By means of example, assume the following three sentences (Example 5.1):

The first sentence, when taken alone, is completely nonsensical. Three unknown

1. The BBB AAA the CCC.
2. The baker baked the CCC.
3. The BBB was her youngest brother.

**Example 5.1:** Three unknown words in three sentences.

words, AAA, BBB, and CCC have no known context and can, at best, be syntactically guessed. The inclusion of the second sentence give substantial context to the unknown word CCC: selectional restrictions on ”baked” as well as ”baker” imply that CCC is likely a cake or other baked good. A learner could reasonably decipher this, given the rest of the text, and could now plug this meaning into the first sentence:

1. The BBB AAA the BAKED-GOOD.
2. The baker baked the BAKED-GOOD.
3. The BBB was her youngest brother.

**Example 5.2:** One of the unknown words has been resolved.

The first sentence is still too vague, although common syntactic structures might allow the strong guess that BBB is a noun, and AAA is a verb. The third sentence also includes the unknown word BBB, as well as context that suggests that BBB may mean ”boy” (other possibilities include any SOCIAL-ROLE, such as ”president”, although these are less likely). Although the choice of ”boy” is more of a guess than ”cake” was for the previous sentence, it is the job of the learner to factor multiple inputs into its final candidate frame, keeping polysemy of words in mind. Substituting again results in the following sentences:

At this point, the semantic analysis engine has enough contextual clues to guess at the meaning of AAA, likely ”ate”. This system of repeated bootstrapping and learning de-

1. The BOY AAA the BAKED-GOOD.
2. The baker baked the BAKED-GOOD.
3. The BOY was her youngest brother.

**Example 5.3:** The second unknown words has been resolved.

picts our hypothesis well: OntoSem, in its current form, should be able to produce enough knowledge through contextual clues to allow a learner to define an unknown word in the form of an ontological concept (and associated lexical entry), which can be plugged back in to the system in order to assist the next round of learning.

### 5.2.2 Broader Coverage Improves Results

Our second hypothesis, derived from the first, states that broader coverage in the bootstrapped system will produce improved (either broadened or increased in accuracy) learned results. Intuitively, this implies ”the more you know, the more you can learn”; in other words, if the context surrounding an unknown word is known, the unknown word can be more easily deciphered using selectional restrictions (and other techniques).

The examples presented in the previous section, Examples 5.1, 5.2 and 5.3, already demonstrate this hypothesis well: had BAKED-GOOD and BOY already been known, the learner would have been able to jump immediately to the best guess for the unknown word AAA (”ate”). In order to get there, however, the other unknown words had to first be learned, and were done so in an environment where they were the sole ambiguity. In other words, the example shows that the capacity of the learner to put forth candidate concepts for new words is expanded as the learner’s bootstrapped static knowledge expands.

This presents a ”chicken and egg” problem: the learner requires some threshold of knowledge in order to learn knowledge. However, the situation is not as bad as it may seem.

Again, Example 5.3 shows that the threshold of knowledge is variable, depending on the target word. Extremely domain specific, monosemous words (such as technical or medical terms) may require a high level of very bootstrapped knowledge in the same domain, as the texts are likely to contain other specific words. General purpose, monosemous words may require the least amount of bootstrapped knowledge, as they will be found in common text, and the input texts could be cherry-picked based on existing coverage. General purpose, polysemous words may require more bootstrapped knowledge than monosemous ones, but only to insure that the coverage is sufficient to help disambiguate word-senses.

From this, we can gather that the key to successful learning lies in selecting the correct target words (and input corpus) given the current state of the system. Choosing words whose context is likely to include just as many unknown words as known words will result in poor results, whereas choosing words whose context is likely to be easily disambiguated should result in high quality output.

### 5.2.3 Quality Inputs Rather than Quantity of Inputs

Our final hypothesis claims that, given a language learning system based on a semantic analysis engine, quality of inputs (in the form of both raw text, and later TMRs) is more important than quantity of inputs. Often in this field, highly statistical methods are employed for language learning, where a true semantic analysis engine is not available. As a result, increasing the quantity of the input corpus has been shown to help smooth out noise and erroneous data introduced by open texts. Although we do propose means to assist in noise reduction using traditional machine learning techniques, we hold that starting with fewer high-quality inputs can produce better results than more low-quality inputs.

Primarily, this stems from the method OntoSem uses to disambiguate unknown words in an input text. In an isolated sentence, OntoSem's only capacity to guess at the meaning of unknown input involves analysis of the surrounding context. If the context is clear,

specific, and meaningful (a high quality input text), OntoSem will have less trouble disambiguating the meaning of the unknown word using selectional restrictions. Alternatively, if the context is vague, messy, or erroneous (factually), OntoSem may select entirely incorrect relations for the resultant TMR. Although this noise can be reduced if it is an outlier, it will start to have a negative impact if no discernible patterns emerge in the rest of the corpus. Adding more low-quality texts will only increase the distance between the candidate concept and the gold-standard ”correct” knowledge.

The nature of our proposed experiments uniquely positions this research to test this hypothesis in a way that would be challenging to do without the use of a tool similar to OntoSem. Although we are convinced that traditional machine learning and statistical approaches have their uses (even throughout our experimentation), we also believe that focusing on a cleaner corpus, at the cost of size, will produce stronger candidate concepts for inclusion in the existing static knowledge resources.

## 5.3 Experimental Pipeline

In order to address our hypotheses using OntoSem as our semantic analysis engine, we needed to create an experimental pipeline that learned target words using the TMRs produced by OntoSem and returned them as new concepts and lexical entries to be added to the existing static knowledge. The pipeline would need to include a variety of preprocessing phases, the TMR creation, and the actual learning; it would need to be tolerant of poor input, but ideally very capable when presented with a high quality corpus. At a glance, the pipeline would need to be supplied with, or construct, a corpus of raw texts, process them with OntoSem, extract the knowledge from the TMRs and formulate candidate concepts for inclusion in the static knowledge.

Rather than using a hand-crafted or externally provided corpus, for the first phase

of the pipeline we opted to construct a corpus automatically based on the needs of the learner: given a target word, the system would query common search engines for web pages containing that word, and would extract their contents into raw text usable by OntoSem. Throughout our experimentation, we would fine tune this first step, touching on the target words chosen, the queries issued to the search engines, the size of the corpus, and how to include smart filters in order to trim the fat.

Following the construction of a raw text corpus, the second phase of the experimental pipeline involves turning those raw texts into TMRs. This phase would be handled entirely by OntoSem, resulting in a collection of TMRs stored in a database whose format is designed to support querying of TMR frames. Effectively, this phase is a black-box to the learning process: the details of the methods that a semantic analyzer uses to produce TMRs are not relevant to the learner itself, as the experimental pipeline only assumes a set of inputs and outputs at this time.

The third phase of the pipeline would involve extracting the property / filler pairs from the TMR frames generated from the target word. Each frame may contain semantically useful information derived by OntoSem from context, and will be clues as to the meaning(s) of the target word. This knowledge would need to be extracted for further processing by the learner.

With the knowledge extracted from the TMR, the learner would then need to intelligently cluster the data presented into multiple word-senses. The querying will not distinguish between word-senses, so any polysemous words would have multiple meanings mixed together in the collection of TMRs. Using the context of the texts (in the form of property / filler pairs in the TMRs), a set of clever heuristics and algorithms would be needed to assign the property / filler pairs for the target word into groups, approximating word-sense disambiguation.

Finally, each cluster of property / filler pairs (now a candidate concept) would need to

be placed into the existing ontology by locating a suitable existing parent concept to attach to. Following this, the candidate could then inherit all values from the parent (a standard operation in our existing manual acquisition of knowledge) and a lexical stub could be created and attached to the new concept. Learning for the target word would be considered complete at this phase, and the learner could start over with a new target word. This phase, as well as the previous phases, are described in more detail in Chapter 6: Experiment Preparations: Corpus Building and Preprocessing, as well as Chapter 7: Experimental Setup and Evaluation Processes. Chapter 9: Future Research Directions explores improvements to each of these phases, as well as proposes a pipeline completely different from the one discussed here and used in the experiments presented later.

## 5.4 Prerequisites

In order to construct the experimental pipeline described in the previous section, we would need several prerequisites: a semantic analysis engine, an open and queryable corpus, and a toolset for streamlining the experimentation development, analysis and evaluation.

The first prerequisite would be treated as a black-box: OntoSem. We would use our existing, custom-built, in-house implementation of a semantic analysis engine to both produce TMRs from text, and operate as a baseline and bootstrapper using its own static knowledge resources (a hand-crafted semantic ontology and lexicon). Additionally, OntoSem would also serve a variety of other functions, including an initial evaluation metric and a means of filtering raw texts prior to analysis.

With the first prerequisite covered, we would also need a second black-box system which would query an open corpus. Choosing the web as the corpus provided both a near limitless amount of data to pull from, as well as a plethora of existing systems to satisfy our

prerequisite. Using common search engines (such as Google and Yahoo!) allowed us to craft intelligent queries over the breadth of the web without any implementation, providing instant fuel for our text analysis needs.

Our final prerequisite could not be black-boxed: we needed to develop a custom set of tools (scripts, programming objects, and user interfaces) to assist in crafting, executing and evaluating a range of experiments. Such a toolset did not exist in the form that was necessary to carry out the complex experiments planned, so a system would need to be developed from scratch in order to fully support the research. Appendix B: Tools and Interfaces discusses DEKADE, the THP Interface and the THP Processor in more details. The first of these systems would be created as a general purpose suite of support tools for our research using OntoSem and its static knowledge resources; the latter two would be created on top of DEKADE to directly support the language learning research presented here.

## 5.5 Conclusion

In this chapter we presented the hypotheses that provide the foundation of our research, an experimental pipeline constructed to test these hypotheses, and enumerated the technical requirements for building such experiments. We proposed that a semantic analysis engine could be used to bootstrap a language learner, resulting in the engine being enhanced with new static knowledge. We also put forth that more initial coverage of knowledge would allow for broader coverage of learned knowledge and that higher quality input texts are more important (in terms of impact on quality of results) than higher quantity when working with a deep semantic engine (instead of a heavily statistics-based learning system).

We discussed our proposed experimental pipeline at a coarse level of detail: a target

word would trigger a query to the web, resulting in a set of texts to be semantically analyzed; relevant knowledge would then be extracted from the resultant TMRs, clustered into word-senses and placed into the ontology. In order to perform the proposed experiment, we would need an existing semantic analysis engine (OntoSem), access to an open and queryable corpus (Google and Yahoo!) and a toolset for streamlining the experimental process (DEKADE and the THP Processor). In the next chapter, we'll discuss, in detail, the preprocessing phase of the learner (leading up through the creation of a corpus of TMRs). Following that, in Chapter 7: Experimental Setup and Evaluation Processes, we'll detail each experiment rigorously, spotlighting the evolution in the experimental pipeline as each experiment completed, as well as discussing the means of evaluation used for each set of experiments.

Chapter 6

# EXPERIMENT PREPARATIONS: CORPUS BUILDING AND PREPROCESSING

## 6.1 Introduction

Before language learning can begin, we must provide an input corpus to the learner in the form of a collection of TMRs. In order to produce this corpus, a series of preprocessing phases, each integral to the learner's success (as evidenced in 8.7) must be executed. We will need to begin with a target word, and from it produce a set of TMRs for the learner to digest. The descriptions in this chapter highlight each stage of preprocessing at a general level, and all details pertain to the last set of fully-automatic experiments run (8.7); for a complete chronological description of how each phase of the preprocessing step evolved over time, see Chapter 7: Experimental Setup and Evaluation Processes.

In this chapter we'll discuss the three main preprocessing stages leading to a complete TMR corpus: in the next section, we'll discuss selecting a target word (or set of target words) for the learner to focus on; in section 6.3, we'll present the steps required to construct a raw text corpus from the set of target words chosen; finally, section 6.4 will present converting the raw text corpus into a corpus of TMRs to be processed by the learner.

## 6.2 Target Word Selection

Selecting a target word (or collection of words) is the first step the learner must complete prior to any further processing. The target word provides the seed for all corpus construction and a goal for the learner to meet. Selection of a target word can be either manual or automatic, neither or which directly impacts the methods of all further stages of the learner (although the actual word chosen can greatly impact the quality of the results).

Manually selected words can be chosen arbitrarily, or with a direct purpose: in our experimentation, we often selected words based on their polysemy (or lack thereof). In some cases, we selected words who had at least one domain-interdependent meaning with the other target words, while in other cases the words selected were largely random. In principle, the method of manually selecting target words can be any: if components of the learner (or the learner's capabilities itself) are being tested, then target words should be chosen to spotlight those experiments. In a practical environment, target words may be chosen simply because they are needed for the task at hand: the learner, in principle, is meant to enhance or replace manual acquisition, so having an acquirer choose a target word for the learner based on the need to add that word to the static knowledge is a perfectly viable means of selection.

Target words can also be selected automatically. As the semantic analyzer encounters a word that is either completely unknown, or is used in a sense that appears to be unknown (there are no sufficient semantic constraints that fill the lexical form of the word used), the learner can be triggered to investigate and populate the static knowledge with new data about the target word. Although we performed no rigorous experiments using this technique (as we needed to spotlight the learner's capabilities in each experiment), this means of selecting target words was demonstrated as a proof-of-concept, and is discussed (with results) in Appendix A: Automated Target Word Selection Test.

## 6.3 Raw Text Corpus

Regardless of the method chosen to elect the the target words, following their selection the preprocessing of the raw text corpus begins. In order to produce a collection of raw texts including a target word, the web is queried, and the results are locally archived, cleaned of markup and web junk, broken into sentences and filtered based on relevance, structure and complexity.

### 6.3.1 Queries

Although the source of the data can, in principle, vary, our experiments used the web as a source of raw input accessed through common search engines (Google and Yahoo!) via accessible APIs. The querying module would issue a query to the search engine a number of times until a threshold of input web pages was found. The query often took the form of simply the target word as it was presented: this meant that although both the popular search engines provided stemming (in order to reduce the word down to its root), further stages in this process would ignore these suggestions (this issue is discussed further in Chapter 9: Future Research Directions).

In the case of the final experiments, we introduced the concept of dependent and independent target words. Independent target words were queried as before, with a simply one word search issued to the search engine. Dependent target words used a logical search to return web pages that both included the target word, as well as one of the independent target words. Both search engines support complex logic as input, so mandating the appearance of one of a set of words along with the target word was handled entirely by the search engine.

The results of the queries returned were in the form of a list of URLs to pages that matched. Each of these pages was then locally archived for faster future access and manip-

ulation by the learner. Given a high speed connection, and a reasonable desired corpus size (the final experiments used a corpus of over 15,000 sentences), the querying and archiving process took a trivial amount of time when compared with the entire learning pipeline, making it one of the most flexible and manipulable phases of the process.

### 6.3.2 HTML Stripping

The cached web pages returned by the query phase of the preprocessor need to be cleaned of extraneous HTML junk, mostly including markup but also navigation and design elements that exist on the page but are not part of the true page contents. In order to simply remove the HTML, we used an open source HTML stripper (HTM 2006) which carves out all HTML tags leaving only raw text behind. Using only this method introduces problems: often web pages contain menus, navigation bars, and hidden search terms that appear to not be part of the page's content, but when all markup is removed are combined with valid information into a jumbled mess.

To address this, we used a series of heuristics developed in (Java *et al.* 2007) prior to HTML stripping designed to detect the presence of common web page meta-data not intended to be part of the actual content. This text is removed completely, following which the HTML stripper is employed, resulting in a set of texts containing contentful web data including the target word. Both techniques employed were off-the-shelf processes which were not customized to our specifications. In many cases, the quality of the output was high, but in some cases web junk still existed. A further set of filtering (see 7.3.4) would help to clean the output.

### 6.3.3 Sentence Breaking

The pages of raw text output by the HTML stripper are then intentionally chunked into sentences prior to semantic analysis. This is done for two practical reasons: first, the

analyzer does not yet have a fully functioning reference resolution engine, making multi-sentence analyses costly and error-prone; further, the filtering that follows this stage is designed to work with single sentences, rather than full texts, and would need to be re-implemented otherwise.

The actual sentence breaking is performed by an external component of the OntoSem system that uses a lexical corpus and a series of heuristics to identify breaks in sentences in a large text, such as common punctuation. The result is a large collection of single sentence texts that can now be filtered in a variety of ways prior to semantic analysis and the production of the TMR corpus.

### 6.3.4 Filtering

The collection of single sentence raw texts must now pass a series of filters before being considered part of the final raw text corpus. Each passing text must contain the target word, must have a high probability of being an English sentence, and must be of medium syntactic complexity.

Any text that does not contain the target word will be removed: at current, this is done via a simple string match, and Chapter 9: Future Research Directions discusses several means of improvement for this filter. This is done to ensure that the final corpus contains only texts referring to some sense of the target word, and does not contain large amounts of unrelated (or related only through reference resolution) texts that would simply add to the complexity and runtime of the learner without producing any further semantic knowledge for the target word.

The second filter employed uses a hidden Markov model (Duda, Hart, & Stork 2001), trained on sections of *Moby Dick* to predict the likelihood that a string of characters is an English sentence. This filter was designed to cover any gaps left behind following the HTML stripper: if the heuristics failed to identify a menu or navigation bar, this filter would

likely identify the ”sentence” as being nothing more than a collection of nouns and would filter it out. Section 7.3.4 further discusses this filter and provides a few examples of texts that would be retained and a few that would be filtered.

The final filter used involves calculating the complexity of the syntactic parse produced by OntoSem for a given text. This complexity value is used by the filter to retain only texts of medium complexity: extremely simplistic sentences tend to have very little, if any, semantically useful knowledge (even though they may contain the target word, it may not be used in any meaningful way); further extremely complex sentences tend to be too challenging for the semantic analyzer to properly handle, resulting in erroneous TMRs which would, in turn, impact the learner's output. This filter can be tuned to select a certain window of texts based on complexity, resulting in semantically useful and contextually clear input texts to be semantically analyzed. Following this filter, the construction of the raw text corpus is considered complete: a collection of single sentences containing the target word, with a high likelihood of being grammatically correct English sentences with a medium level of complexity will be passed to OntoSem for semantic analysis.

## 6.4 TMR Corpus Construction

The TMR corpus creation is handled entirely by OntoSem (see Chapter 3: OntoSem) as a black-box. Each sentence from the raw text corpus is fed, one at a time, to the semantic analyzer which produces a TMR containing (amongst others) a frame generated from the target word as an instance of either OBJECT or EVENT, depending on the selectional restrictions presented by the context surrounding the target word. Each TMR is then indexed and archived into a database for access by the learner. Once all TMRs for the experiment have been generated and stored, the preprocessing phase is complete and the learner can begin to extract the knowledge contained in the TMRs.

## 6.5 Conclusion

The preprocessing phase of the learner does not directly collate any knowledge into the candidate concept; rather, all of the groundwork is laid for the learner to efficiently extract semantic knowledge for the construction of a set of candidate concepts for the target word. Starting from a set of target words, and resulting in a collection of TMRs produced from sentences that are contentful and topic-appropriate, the preprocessing phase exists to ready a corpus of TMRs using as many tricks as are available to eliminate erroneous, malformed, and junky input from raw web data. These techniques and filters were layered in as a need to improve the quality of results arose: in the following chapter, the chronological evolution of the preprocessing phase (interwoven with the learner itself) is detailed, and the reasons for the inclusion of each of its parts are discussed.

Chapter 7

# EXPERIMENTAL SETUP AND EVALUATION PROCESSES

## 7.1 Introduction

The experiments and evaluation metrics used in this research underwent a heavy evolution from the preliminary investigations in this technique through a custom-tailored experiment with supporting evaluation metrics. This chapter presents the history of the evolution that occurred in the experimental and evaluation processes.

Section 7.2 describes, in brief, the baseline experimental setup (see Chapter 6: Experiment Preparations: Corpus Building and Preprocessing for a full description), and discusses the change in philosophy over time. Section 7.3 discusses the experimental process chronologically, explaining what changed at each step, and discussing the implications of those changes. Section 7.4 presents the evaluation process, and shows how it changed over time with the experimentation. Specific experimental results are found in Chapter 8: Results and Discussion.

## 7.2 Experimental Setup Over Time

Initially, the goal of the preliminary experiments was to show proof of concept by leveraging the resources uniquely available (OntoSem), and the defining an appropriate approach. Our approach was to construct a learner that could take unknown raw text in a domain, coupled with a collection of target words to learn, and produce semantically rich definitions of the target words in the form of ontological concepts that could supplement the existing language-independent ontology used by OntoSem.

This preliminary experiments corpus was selected for its unique collection of words that were known to not already exist in the static knowledge of the semantic analyzer. The first corpus we chose consisted of excerpts from *The Lord of the Rings*, by J.R.R. Tolkien. The target words selected were nouns created by Tolkien, and so were not previously addressed by OntoSem: hobbit, orc, ent, etc. The results of this experiment were not fantastic, however, they were encouraging, and provided a useful baseline for improvement in subsequent experiments.

Our next experiment was geared towards learning non-domain specific words from a raw text corpus. This fairly bold goal was strengthened by the backing of OntoSem, which has been constructed specifically for extracting and representing meaning of general text. The first experiment (Experiment 1) was aimed at learning a small collection of unrelated words as single senses. The words were primarily simple nouns, and an equal emphasis was given to learning ontological relations (case-roles, such as AGENT-OF) as was to learning attribute fillers (descriptors, such as HEIGHT). The system took what came to it as absolute, and made no efforts to filter its output based on any threshold (e.g. minimum occurrence count, confidence metrics, etc.) before presenting its results.

The early experiments provided evaluation based on hitting a target in the existing semantic ontology. We compared the learned knowledge to the ”best” match found in the

static knowledge; the premise being that closely located nodes in an ontology are heavily similar due to inheritance of property / filler pairs from common ancestors. The metric we used for this evaluation changed even at this early stage, moving from a previously existing algorithm to a more appropriate custom algorithm.

The results from the first round of experiments were promising, but it was clear that the process, and the goal itself, would need to be refined in order to provide usable results. The amount of input to the system was substantially increased, and a statistical clusterer was included to help minimize the noise that would be introduced. A major philosophical change that occurred at the midpoint of the experimental process was to begin handling, and expecting, multiple-sense target words. The system was prepared to divide the learned knowledge into multiple outputs in a number of ways. Further, many improvements were made to the individual components of the process, which we discuss in detail in the next section.

The final experiment's focus was shifted from producing improved results, to providing a solid bed of results to evaluate. The goal was to use a cascading series of evaluation metrics to identify where along the pipeline precision and recall loss were occurring, in order to provide a solid direction for future research to pursue. The evaluation process itself was redesigned to focus less on seeing how close the learner hit the target in the ontology, to how close the learned knowledge lined up to a human-created gold standard.

Following the preliminary experiment, all of the subsequent experiments followed the same general setup pattern. Although some changes were made as the experiments continued (discussed in the next section), the system mostly followed a standard bootstrapping methodology:

1. A corpus of raw texts was constructed from the web using the target words as input to the search engine.

2. The raw texts were cleaned of excess web artifacts, broken into sentences, and filtered, resulting in a corpus of single sentences containing only the target word.

3. These sentences were parsed and semantically analyzed by OntoSem, leaving a corpus of automatically created TMRs for the learner to use as input.

This process is described in much more detail in Chapter 6: Experiment Preparations: Corpus Building and Preprocessing.

## 7.3 Experiment Overviews

In this section we discuss, in depth, the setup and process involved in each experiment in a chronological order, starting with the first experiment following the preliminary research. The experiments are divided into four groups based on the level of change in setup that occurred between individual experiments.

### 7.3.1 Experiment 1

The first experiment following the preliminary research, (see (English & Nirenburg 2007b)), was designed to capture the overall goals of the project: it was to learn general purpose words, using an open corpus of raw text as input. We tasked the learner with learning four target words: *pundit*, *CEO*, *hobbit*, and *song*. The learner, at that point in time, was not accounting for multiple word senses, and so was looking to construct only one candidate frame per target word. As input, the learner was fed 2802 raw text sentences from the web containing the target words. The preprocessing step used is described in detail in Chapter 6: Experiment Preparations: Corpus Building and Preprocessing.

Leveraging OntoSem's ability to gracefully handle unknown inputs through a process of analyzing and assigning selectional restrictions (see Chapter 3: OntoSem), the final input to the learner consisted of a corpus of TMRs, each containing one instance of the

unknown target word mapped to a best guess in the ontology by OntoSem. Each frame in the TMR corresponding to the target word contained a list of properties describing the frame's interaction with other elements of the TMR.

The list of properties actually present in any one TMR is likely to be small (because only a few will be referred to in a single sentence). The learner then had to collate the knowledge extracted from processing individual sentences. Given a list of TMRs, the learner searched through each one, finding all properties associated with the instances of the candidate frame, and collating them into a single frame for the corresponding candidate ontological frame. When collating the fillers of each property of the candidate frame, the system filtered out weaker constraints if stronger constraints had been attested. For example, if among the fillers of the AGENT-OF property of the candidate frame the system finds READ, ACTIVE-CONGNITIVE-EVENT and MENTAL-EVENT, it will retain only READ because it is a descendent of the other two frames in the ontology. The weaker constraints can, in principle be retained in the OntoSem ontology because the latter uses multi-level constraints to support robust disambiguation processes. Technically, the constraint READ may appear in the SEM facet of the property AGENT-OF in the candidate frame while MENTAL-EVENT may appear as the filler of the RELAXABLE-TO facet of the same property.

After a candidate frame has been constructed from the collated results of the associated TMR frames, the final step was to find a suitable location for the frame in the existing ontology. Doing so will result in a closure for the new frame, as it was able to inherit a substantial amount of existing properties from its new parent in the ontology. In Experiment 1 we used the OntoSearch algorithm (Onyshkevych 1997) to provide a comparison metric between the candidate frame and each existing frame in the ontology. OntoSearch works by finding the shortest weighted path between two frames along any connections (not just subsumption). It is important to note that OntoSearch will use any available connection to find a path between two frames, and this is not an appropriate measure of similarity for

these experiments; this is discussed more in the next section.

The final results of the learner for the first experiment consisted of four candidate ontological frames, one for each of the input target words. The frames were populated with properties extracted directly from texts that included the target words, and were heavily supplemented with properties inherited from the candidate's presumed parent in the ontological hierarchy.

The first experiment was flawed in two ways, both of which were addressed as the experimentation continued. Importantly, the first experiment did not account for multiple word senses, which caused the results to contain fairly confusing collections of properties. Additionally, the comparison metric used to find a suitable place for the candidate in the hierarchy of the ontology was not appropriate for the task, and needed to be replaced (see section 7.3.2).

### 7.3.2 Experiments 2 & 3

Our second and third experiments used identical setups, but with different inputs, and were a natural evolution from the first experiment. The second experiment maintained a control environment by using the same four target words as the first experiment: pundit, CEO, hobbit, and song. The third experiment broadened the second by maintaining the same experimental setup, but substituting the original four target words for twelve words, many of which were chosen intentionally as single sense words: *brontosaurus*, *cherimoya*, *deport*, *depose*, *diplodocus*, *obey*, *pledge*, *spartan*, *stegosaurus*, *syrup*, *triceratops*, and *wigger*. The second experiment used the same 2802 input sentences as the first experiment, while the third experiment used a new corpus of 4703 input sentences.

Beyond minor improvements to logic, we didn't modify the main learning algorithm in the second and third experiments. Both experiments still depended on OntoSem as a black-box to produce a corpus of TMRs including on-the-fly lexicon entries for unknown words.

The properties that were generated were collated and used as the input to the comparison metric in order to find the candidate frame a suitable home in the existing ontological hierarchy.

The major change in both of these experiments was in the comparison metric. Experiment 1 had used OntoSearch, a pre-existing ontology-based distance calculation that leverages the interdependence of the properties and frames in OntoSem's ontology to find the shortest path between two frames (see Chapter 3: OntoSem for a complete description of OntoSearch). Interestingly, OntoSearch doesn't use only subsumption paths to navigate the tree, but will also apply a weighted traversal penalty to cross any property that connects two frames, thus making the traversable network very dense. As an example, if a frame contained the property / filler pair AGENT / HUMAN, OntoSearch would consider moving along the AGENT path to the node HUMAN, with an appropriate penalty applied. In this way OntoSearch does a very good job at finding the shortest distance between two existing frames, but this is not an cognitively plausible portrayal of similarity between a candidate and an existing ontological concept.

Primarily, this metric is a poor calculation for similarity due to its fundamental design. OntoSearch assumes (and was designed for) a well constructed, fully realized ontology. The assignment of weights for path traversal is measured assuming that both frames are well formed and exist in the ontology: which implies that the ontology is well formed around those frames; in other words, OntoSearch assumes that any two given frames contain certain necessary relations, including subsumption links, which the candidate frame will not yet have. When one of the frames involved is not yet a part of the ontology, OntoSearch's base assumption no longer holds. We cannot use a system that expects the target to already be well formed in the ontology in order to then find a place to set it down; as this is effectively a contradiction, it thus needed to be addressed.

In order to replace OntoSearch with a more appropriate measuring tool in our pro-

cess, we needed to develop a custom comparison metric that would maintain OntoSearch's familiarity with the ontology but would remove its dependency drawbacks. Experiment 2 introduced the first replacement comparison metric to the process, reported in (English & Nirenburg 2007a), the new concept comparison heuristic allowed the system to effectively look at two frames in isolation, and produce a numeric equivalency score that reflected the semantic hierarchy intrinsic to the properties the frames were constructed of.

**Custom frame comparison metric** The comparison metric used in Experiments 2 and 3 took into account the number of property names that each frame had in common as well a measure of the similarity of filler sets for the shared properties. We then proposed to take a simple average of these two values to produce a similarity value. Formally: Let C1 denote an existing frame and C2, the candidate frame; let Pt be the total number of properties defined in either concept (union of properties) and Ps, the total number of properties shared by the two concepts (intersection of properties); and let Vi denote the vector of computed value pairs for all values in C1 and C2 with property i; let Viv stand for the combined results of value set comparisons for property i; and let Vgt stand for the total number of Viv values greater than 0.0. We can then compute a value of the intersection of the sets of properties defined for each of the compared concepts as P = Ps / Pt and the quality of the intersection of value sets for the defined properties as:

$$V = \frac{sum(i = 0, Ps; Viv)}{Vgt} \tag{7.1}$$

The simple averaging of the two values yields:

$$Similarity = \frac{P + V}{2} \tag{7.2}$$

The above calculation yields an average result of similarity given the known similari-

ties of any two properties defined in each frame (defined above as Vi). Calculating each Vi, however, requires weighing each defined filler in the candidate frame's property i with each defined filler in the other frame's property i. To do this, we must be aware of what types of filler can be present in a property, and must also have an understanding of the structure of the ontology.

In order to calculate Vi, we needed to allow relaxation in compared values. It is very subjective to say that a giraffe's height has a relative value of 0.8, or a person only speaks to other people. Although these may be recorded as facts in the ontology, OntoSem is aware of the necessary flexibility of these relations, and so a comparison metric must as well. For numeric attributes, there must be an allowed give before two values are considered significantly far apart to no longer be equal. For semantic fillers (frames defined in the ontology that are used as fillers), it is not good enough to see that exactly the frame names match: subsumption must also be incorporated into the metric, with a weighted penalty applied to each IS-A link that is traversed.

Fillers can be of six distinct types, depending on the property that is defined. Numeric attribute properties only allow fillers of the types literal number, relative number, and numeric range. Most case roles, and other semantically deep relations only allow fillers of the types frame and set (of frames). Additionally, a small set of properties, known as literal attributes, also allow for fillers of the type literal text. When comparing any two fillers in property i, Table 7.1 was used to calculate the similarity score. All similarity scores in property i were averaged, resulting in a Vi, used above.

We can summarize Table 7.1, broadly, by examining the comparison types listed. In the case of literal text, only a case-insensitive match results in a positive response. In the case of all numbers, they must be within a range of each other as defined by a tolerance value (defaulted to 10%) in order to have a positive response. Matching for numeric ranges and names of ontological concepts are both complex enough to warrant separate discussion

| Value 1 Type | Value 2 Type | Comparison Metric |
|---|---|---|
| Lit. Text | Lit. Text | Case-insensitive match = 1.0, else 0.0 |
| Lit. Text | Lit. Num | 0.0 |
| Lit. Num | Lit. Num | Num 1 within (Num 2 * tolerance) of Num 2 = 1.0, else 0.0 |
| Lit. Num | Rel. Num | 1.0 |
| Lit. Num | Num Range | Num 1 <> (Range 2 * tolerance) = 1.0, else 0.0 |
| Lit. Num | Frame | 0.0 |
| Rel. Num | Rel. Num | Num 1 within (Num 2 * tolerance) of Num 2 = 1.0, else 0.0 |
| Rel. Num | Num Range | 1.0 |
| Rel. Num | Frame | 0.0 |
| Num Range | Num Range | See 7.3.2, & Figure 7.1 |
| Num Range | Frame | 0.0 |
| Set | Frame | 0.0 |
| Frame | Frame | See 7.3.2, & Algorithm 1 |

Table 7.1. Comparisons of filler types.

below.

**Custom comparison metric special case: numeric ranges** Although most of the comparison metrics between two fillers could be easily summed up as shown in Table 7.1, comparing two numeric ranges needed some extra handling. In principle, any two numeric ranges can be in three possible cases together:

Case 1: The ranges do not intersect.

Case 2: One range encapsulates the other range.

Case 3: The ranges (partially) intersect.

For our purposes, the fourth possible case where the ranges are identical, is handled by case 2. In order to calculate similarity, each of these cases must be processed separately. After first identifying which case two numeric ranges belong to, the following algorithm for matching was applied:

Case 1:
- Let d = the shortest distance between the two ranges (the distance between the upper bound of the lower range and the lower bound of the upper range)
- Let t = the sum of both ranges (the total distance covered by both ranges combined)
- if (d / t) >= tolerance, then 1.0, else 0.0

Case 2:
- 1.0

Case 3:
- Let d = the maximum distance overlapped by the two ranges (the total distance shared by both ranges)
- Let t = the total union distance of the two ranges (the total distance covered by both ranges combined)
- if (d / t) >= tolerance, then 1.0, else 0.0

Again, in this case, the default tolerance has been set to 10%. To make these cases clearer, Figure 7.1 shows a graphical breakdown of the range similarities.

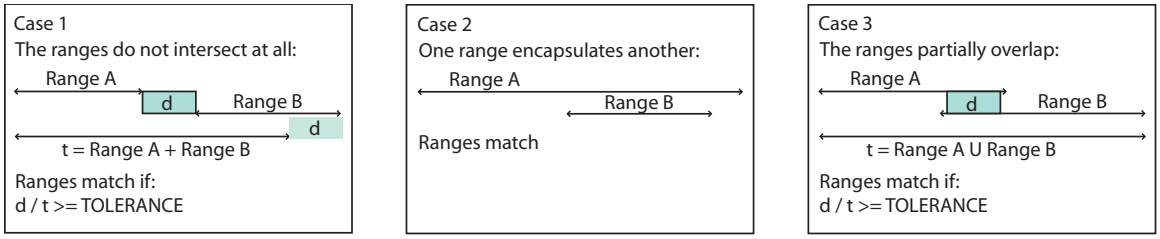


FIG. 7.1. Three cases for numeric range comparison.

**Custom comparison metric special case: concept names** The second case where special attention was warranted was when comparing two fillers containing concept names. In this case, we are looking to see how closely related the two concepts are by

inspecting the ontology's inheritance structure. If the two frames representing concepts are the same, then a maximum value (1.0) can be returned, and the metric can continue on to the next property / filler pair. However, if the frames differ, they could be very closely related in an ancestor/descendant relationship. For our purposes, it is important to know that MEDICAL-WORK-ROLE, and PHYSICIAN are in a parent/child relationship, and so should have a very high similarity score (much higher than MEDICAL-WORK-ROLE and MOTION-EVENT, for instance).

In order to process a similarity score between two frames based solely on subsumption, we look at the ancestry tree of each frame in turn. Algorithm A ascends recursively up the inheritance tree, checking to see if the current parent is contained in the ancestry tree of the other frame. When a match is found, the number of IS-A paths that were traversed is returned, and then averaged with the results of the same process applied to the second frame. A penalty is then applied for each subsumption link that was crossed, with the final output scoring a perfect match as 1.0, and distant matches as a 0.0.

The weighted penalty is designed to prevent the metric from reporting any similarity once the distance between two frames is great enough that their should be no further implied similarity. We selected the weight 0.2, particularly, based on the design of OntoSem's ontology. Beyond five subsumption links, enough generality has been introduced that nearly all similarity between two frames has been intentionally removed, aside from the most basic semantic descriptions (in other words we've exited anything that could be considered a "local sub-tree"). The entire process is described in Algorithm 1 below:

### 7.3.3 Experiments 4 - 8

Following the improvement to the comparison metric introduced in Experiments 2 and 3, it became clear that a major component of the process needed to be reconsidered. Until the fourth experiment, the learner made a naive assumption that all words had only a single

```
compare_frames (frame f1, frame f2)
     distance1 = find_closest_ancestor_distance(f1, f2)
     distance2 = find_closest_ancestor_distance(f2, f1)

     avg = (distance1 / distance2) / 2
     penalty = avg * 0.2

     result = 1.0 - penalty
     if (result < 0.0)
          return 0.0
     if (result > 1.0)
          return 1.0
     return result

find_closest_ancestor_distance (frame f1, frame f2)
     smallest = max_integer_value
     for all parent in f1.parents
          height = find_height_in_ancestors(parent, f2)
          if (height == -1)
               height = find_closest_ancestor_distance(parent, f2)
          if (height != -1)
               smallest = min(smallest, height)
     return smallest

find_height_in_ancestors (frame f1, frame f2)
     for all parent in f2.parents
          if (parent == f1)
               return 1
          else if (parent.parents == NIL)
               return -1
          else
               height = find_height_in_ancestors(f1, parent)
               if (height == -1)
                    return -1
               else
                    return height + 1
     return -1
```

**Algorithm 1:** Calculating weighted similarity of frames in a hierarchy.

meaning, or sense. This led to a number of problems, most notably a skew in the results as multiple senses of one word were condensed into a single meaning. The resulting frame would be a hybrid of the two senses, generally with a bias towards the more commonly used meaning. We needed to address this issue, and so the learner needed to account for multiple word senses.

In addition to addressing the concerns presented by words with multiple meanings, the learner was also improved in nearly every other area: corpus building, preprocessing, semantic analysis, candidate frame construction and placement were all improved over the course of these five experiments.

Experiment 4 ramped up the size of the corpus, to include 15416 sentences, and introduced a new set of target words: *buoy*, *coronary*, *datum*, *dub*, *license*, *nail*, *retake*, *vent* and *version*. With just a few exceptions, these words were chosen specifically to reflect multi-sense words. The fourth experiment introduced a method of statistical clustering of the proposed property / fillers as a method of suggesting the number (and contents) of multiple candidate frames from a single target word. Additionally, Experiment 4 also introduced a new method of locating a home in the ontology for a candidate frame, using a tree-crawling methodology. Each of these improvements are discussed further below.

The fifth and sixth experiments introduced a new set of sentences as the corpus (15188 texts) was built from a new collection of target words: *dragster*, *truck*, *freeway*, *skid*, *curb*, *chicken*, *jackknife*, *bottleneck*, *rubberneck*, and *hitchhike*. These words were selected again as words that primarily have multiple meanings, but further, were domain specific. It is important to note that the corpus the original texts were selected from is not domain specific, but is still an open corpus. The target words were selected in order to study the relationship between learned words that have some interdependence on previously learned words. As such, the second half of the corpus was selected to contain at least one of the target words and a second target word that had already been selected from the first half. As an example,

a text in the corpus may contain only the target word truck, but no text in the corpus contains only the target word hitchhike (instead, it contains hitchhike and at least one of the first five target words as well).

The fifth and sixth experiments, in addition to moving to the final corpus for the remainder of the experiments, also began to use a newer, and substantially improved version of OntoSem for the semantic analysis (TMR corpus building) phase. Parallel research, at the time, focused on improving the semantic analyzer within OntoSem as well as expanding coverage of the lexical and ontological resources, resulting in a system that was more capable of handling the input texts used for remainder of the experiments.

Experiments 7 and 8 continued to use the same corpus and target words as the previous two experiments, but added improvements to the late preprocessing phase, focusing mostly on removing excess TMR junk (debugging properties) from the initial corpus. Further, the eighth experiment also improved the tree-crawling location algorithm introduced in the fourth experiment by focusing the crawler's attention on the semantically relevant parts of the ontological tree, and away from the engineering and debugging areas that are required of the system, but do not represent any formal knowledge.

We will discuss each of the improvements listed above in the following sections, including an analysis of their impact at a fundamental level. Individual experimental results are discussed in Chapter 8: Results and Discussion.

**EM clustering and multiple word sense handling** One of the fundamental drawbacks in the learner's process in the first three experiments was the absence of handling for multiple word senses. In order to address this issue, we implemented a statistical clusterer into the learner for the purposes of dividing the extracted properties into multiple candidate frames. The intuition was that the different properties would be present in different senses of the word, due to the inherent descriptive nature of the properties themselves.

Prior to using the statistical clusterer, a few, more naive clustering algorithms are run over the input TMR frames. Any frames of the same type (generated from the same ontological frame) are temporarily clustered together. Intuitively, we are assuming that if OntoSem thought the target word in sentence X was an instance of HUMAN, and that the same target word in sentence Y was also an instance of HUMAN, but that target word in sentence Z was an instance of MENTAL-EVENT, then the target words in sentence X and Y are probably the same word sense (certainly different from the one present in sentence Z) and should be clustered together.

After this basic clustering technique, any clusters that fail to meet a threshold of expressiveness will be passed into the statistical clusterer to be further processed. For our purposes, a threshold of expressiveness refers simply to the absolute depth of the frame in the ontological tree. In many cases, when OntoSem does not know a word, it will fall back to the most basic description it can give (OBJECT or EVENT). If OntoSem can't do any better than this, we need to further distinguish between the newly formed cluster of OBJECTs. However, if OntoSem was able to make a more precise guess, we can rest on its decision. We chose an absolute depth of 4 as our threshold; any cluster formed of frames whose depth in the ontology was greater than 4 passed muster, all other clusters were pushed into the statistical clusterer for further differentiation.

The statistical clusterer, an implementation of the EM algorithm (Dempster, Laird, & Rubin 1977) as provided by the WEKA toolkit **??**, expected a collection of TMR frames as its input, and returned a collection of collections of TMR frames, essentially clustering the input frames into separate groups. A TMR frame consists of a list of property / filler pairs, which are used to describe the frame via semantic relations to the other frames in the text. It stands to reason that two texts that use a word in a similar way will therefore have TMR frames for that word that share common property / filler pairs. This is the intuition behind using the statistical clusterer to produce a set of candidate frames spanning multiple senses

of a target word.

The EM algorithm, when used for clustering, can be viewed as assigning each point in a distribution to a group consisting of other points, much like drawing a few ovals around a scattering of rocks in the sand. Eventually, all of the rocks belong to at least one oval, and ideally there are less ovals than rocks. Effectively, EM is taking a set of multi-dimensional points, and dividing them into groups based on relative distance.

This method can be applied to property / filler pairs in a set of TMR frames by making each unique property / filler pair an attribute (or vector in multi-dimensional space). The number of dimensions will be equal to the number of distinct property / filler pairs in the collection of TMR frames, and any frame that has that property / filler pair will have the value of 1 for the corresponding vector; similarly an absence of the property / filler pair will result in a 0 value for the corresponding vector. This allows us to collapse the collection of TMR frames and their descriptions down to a 2-dimensional matrix: one axis represents each TMR frame, the other represents each attribute (unique property / filler pair), and each cell represents that frame's corresponding vector value (in other words: does the frame have the property / filler?). A partial example of what this would look like is shown in Table 7.2:

| | **FRAME A** | **FRAME B** | **FRAME C** |
|---|---|---|---|
| AGENT / HUMAN | 1 | 0 | 1 |
| AGENT / FORCE | 0 | 1 | 0 |
| THEME / MENTAL-EVENT | 1 | 0 | 0 |
| BENEFICIARY / HUMAN | 0 | 0 | 1 |

Table 7.2. Sample multi-TMR frame input to EM clusterer.

The example in Table 7.2 illustrates two possible clusters that EM might form, from the three input TMR frames. The only shared property / filler pair, AGENT / HUMAN, belongs to both frames A and C, and since no other property / filler pairs are shared, the clusterer would decide to couple frames A and C, resulting in two clusters: the first con-

taining all three properties defined in both A and C, and the second containing the property defined in B. The learner would take this knowledge and presume that there are two word senses of the target word, and would proceed from there. In Chapter 8: Results and Discussion, a comparison of our use of EM as a clusterer to suggest word senses is compared to a more traditional ”bag of words” technique commonly used in the field.

**Tree-crawling location metric** Until the fourth experiment, after the learner had finished compiling the candidate frame, the next step it performed was an exhaustive search of the ontology, comparing each frame to the candidate (using the custom similarity metric described in 7.3.2), and producing the existing frame that ranked the highest as a suggestion for the parent of the candidate. The intuition here is that high similarity indicates a high probability that two frames are in a parent-child relationship, thus if a candidate highly matches existing frame X, the candidate may be situated as a child of X, inheriting its properties rather than defining them independently.

The exhaustive approach used was tolerable when the similarity metric employed was OntoSearch; the runtime of OntoSearch is small, as it is a trained algorithm intended to work specifically over OntoSem’s ontology space. Running an exhaustive search over the thousands of frames took a comparatively small time (on the order of minutes). However, after the introduction of the custom comparison metric discussed above, the runtime of an exhaustive search became unmanageable (on the order of days), due to the complexity of each individual search.

In order to speed up the search, we decided that an exhaustive search needed to be replaced with a more intelligent, tree-crawling algorithm. The premise behind the new search procedure was to travel the ontology’s hierarchy, starting from the root, and comparing the candidate to each node. The subtree that had the highest similarity would win out, and the process would recurse until either no children were left, or none of the children compared

more favorably than the current frame. In other words, we hoped that if a candidate frame belonged deep in the OBJECT tree, that it would compare at least somewhat more favorably to OBJECT than EVENT (the two children of ALL, the root), and thus would skip the entire EVENT tree in its search, effectively halving* (not exactly, OntoSem's ontology is not a binary tree) the search time. If this assumption held, then large subtrees of OBJECT could also be removed, forcing the comparison metric to only run a comparative handful of times, and still result in the same conclusion that the exhaustive method provided. Algorithm 2 shows a breakdown of the algorithm used to crawl the ontological tree:

```
find_location_in_tree (frame candidate)
     return crawl_tree(candidate, ALL, 0)

crawl_tree (frame candidate, frame location, double best)
     best_location = location

     for all child in location.children
          result = compare_frames(candidate, child)
          if result > best
               best = result
               best_location = child

     if best_location != location
          crawl_tree(candidate, best_location, best)

     return best_location
```

**Algorithm 2:** A tree-crawling algorithm for finding a suitable location for a candidate concept in the ontology.

At this stage, the tree-crawling algorithm had a few drawbacks: notably, it did not well account for tied scores amongst children, and would simply select the first available highest score. Further, if none of the children of the current frame beat the current frame's score, even if some tied, the algorithm would halt on the current frame, rather than investi-

gating grandchildren. Overall, however, the implementation of this search metric resulted in a necessary improvement in processing time, at a small cost to precision that will be addressed in section 7.3.4.

In the eighth experiment, the tree-crawling algorithm was slightly improved with the introduction of a blacklist of subtrees in the ontology that the algorithm was not allowed to recurse down. This was done to insure that the algorithm did not travel down paths that were either under development by in-house acquirers, or were flagged as debugging or engineering-specific ”knowledge” (data stored in the same format for a software-related reason, but not actually formal knowledge).

**Improved junk filtering in the TMR frame extraction process** The final phase of preprocessing that presents a collection of TMR frames to the learner to be analyzed involves extracting all of the valid TMR frames from the database that match the target word. Prior to Experiment 8, this had been done by simply taking all matching TMR frames (by the ”textpointer” attribute), leaving them unmodified and handing them off to the learner with no further processing. The eighth experiment introduced a few simple filtering techniques to hand off a better collection of TMR frames to the learner.

The first such filtering to be implemented was a simple stop-list of properties that should not be included in a TMR frame intended for the learner. As the learner uses every property at its disposal to find a match in the ontology, any erroneous properties will throw this search off. All properties that are included for debugging or engineering purposes (as in, all properties that exists in the TMR frame but are intended for human consumption and not as strict knowledge) needed to be removed.

In addition to removing junk populated into the TMR frame by OntoSem itself, we also implemented a small improvement to the clustering process in an attempt to mitigate the amount of junk that presented itself due to our use of an open corpus. The web, as

discussed in Chapter 6: Experiment Preparations: Corpus Building and Preprocessing, presents plenty of challenges when used as a corpus, one of which includes the myriad array of erroneous junk. Thanks to the use of the statistical clusterer, however, much of this junk ends up isolated in its own cluster(s). Beginning with the eighth experiment, we implemented a simple threshold to the amount of well formed properties that must exist in a cluster before it can be considered a valid candidate. From observation, it appeared that requiring at least two knowledge-rich properties weeded out the vast majority of junk without sacrificing much useful information. As can be seen in Chapter 8: Results and Discussion, when a very small corpus is used as the input to the learner, this can actually result in loss of knowledge.

### 7.3.4 Experiments 9 - 11

The major improvements to the learner's process introduced in Experiments 4 through 8 focused on supporting multi-sense words and reducing overall processing time while using the custom comparison metric. The next three experiments looked back to the earliest stages of preprocessing in an attempt to provide even OntoSem with a better starting corpus to produce TMRs from, by removing large amounts of web junk from the raw text corpus. Two improvements were also made to the tree-crawling algorithm, providing it with a degree of flexibility which allowed the algorithm to more easily avoid being caught in a local maxima, a common phenomenon encountered in hill climbing problems.

The ninth experiment used the same starting raw text corpus as the previous four experiments had used: a collection of 15188 texts built from ten target words: *dragster*, *truck*, *freeway*, *skid*, *curb*, *chicken*, *jackknife*, *bottleneck*, *rubberneck*, and *hitchhike*. Experiment 9 included two major improvements to the tree-crawling algorithm: first, the algorithm would now start at multiple sub-tree roots in the ontology, rather than the overall parent ALL, and would select the best match from amongst all of the proposed candidates. Sec-

ond, the algorithm would now consider any tie values proposed by multiple children (also a child and parent), and recurse down all possible sub-trees to find a more optimal location for the candidate. Both of these improvements were designed to assist the tree-crawling algorithm in avoiding ruts in the ontology that may either be under- or over-developed.

Experiment 10 maintained the experimental setup, corpus, and target words of Experiment 9, but introduced a major enhancement to the preprocessing step that greatly reduced the final raw corpus size. Experiment 10 put forth only 4768 of the original 15188 raw texts to be semantically analyzed after running the original corpus through a pair of filters designed to detect both the probability that the text is an English sentence, and that its complexity (as measured by OntoSem's estimation of how complex a text is to syntactically analyze) is within a "normal" range (not too simple, nor too complex). These filters were introduced in order to further thin the raw text arriving from the web into a closer approximation of a manually created, domain specific corpus.

Experiment 11 furthered the experiment before it, maintaining the now 4768 raw texts as input, as well as the same ten target words, but introduced a further improvement to the tree-crawling algorithm, as well as a host of enhancements and software fixes to all of the various systems used by the learner (its own logic suite, OntoSem's semantic analyzer, etc.). Although the improvements made in the ninth experiment helped dramatically reduce the tree-crawling algorithm's likelihood of being nested into a local maxima, the problem persisted due to the intentionally designed nature of the semantic ontology. To address this, a flexibility was given to the tree-crawler, allowing it to attempt to find its way past a possible solution, in order to verify that the subtree it was currently in did not vastly improve the results of the comparison metric further down in the inheritance.

**Tree-crawling algorithm improvement: pre-defined starting points** The original tree-crawling algorithm's design had a major flaw: it did not account for the problem of

local maxima. Essentially, the algorithm would move further down a sub-tree along any path that produced a higher comparison calculation than the previous path. However, due to the intentionally designed nature of OntoSem's ontology, certain frames are intentionally over- or under-specified, to either act as an important root of a sub-tree, or as more of a placeholder or organizational frame. In both cases, the algorithm could halt, when a more suitable location might be just beyond the next frame: in the case of over-specification, if none of the children were better suited than the root of the sub-tree, the algorithm would halt; in the case of under-specification, if the algorithm tried to move beyond a frame to its child, but found that child to have very few comparable properties, the algorithm would halt at the parent frame.

For our purposes, this problem meant that the algorithm would frequently get stuck very high up in the ontological hierarchy, never making it to the more interesting and descriptive frames below. In order to sidestep this problem, in Experiment 9, we introduced a change to the algorithm that allowed it to start at multiple locations throughout the tree, and compare the best results from each sub-tree to find the overall match.

The tree-crawling algorithm was originally designed to allow the learner to find a suitable location in the ontology for the candidate without having to perform a brute-force comparison over every existing frame. In keeping with this, we did not want the multiple starting points to be too numerous, so it was important to select only a few that well captured the major sub-trees in the ontology. To do this, we selected a dynamic list of frames who each had in excess of 25 locally defined (not inherited) property / fillers. Intuitively, the more local properties a frame has, the more intentional, manual effort has been put into its definition, meaning it is likely the head of a sub-tree that will then inherit most of the properties that were manually defined. Selecting 25 as a cutoff point meant that only 38 of the nearly 9000 frames would be used as starting points, less than 0.5%.

The frames selected in this way represented a spectrum of mid-level frames, effec-

tively weeding out the excessively generic and the overly specific. Some examples of frames selected by this process include: corporation, animal, creative-work, and ingestible. As a result, the tree-crawling algorithm was able to avoid some of the highest level local maxima, however still fell prey to these problem frames, and would need a pair of further improvements in order to avoid them completely. Algorithm 3 shows a breakdown of the improved tree-crawling algorithm:

```
find_location_in_tree (frame candidate)
     best = 0
     best_location = NIL

     starting_points = database_select("frames with > 25
                       local property / fillers")
     for starting_point in starting_points
          result = crawl_tree(candidate, starting-point, 0)
          if (result.best > best)
               best = result.best
               best_location = result.best_location

     return best_location

crawl_tree (frame candidate, frame location, double best)
     best_location = location

     for all child in location.children
          result = compare_frames(candidate, child)
          if result > best
               best = result
               best_location = child

     if best_location != location
          crawl_tree(candidate, best_location, best)

     return best_location, best
```

**Algorithm 3:** A tree-crawling algorithm for finding a suitable location for a candidate concept in the ontology, starting from multiple points.

**Tree crawling algorithm improvement: multiple-branching** To further address the local maxima problem the tree-crawling algorithm suffered from, Experiment 9 also introduced a second alteration to the algorithm which intended to dodge the problem of tied values in the original algorithm. Prior to the ninth experiment, the tree-crawling algorithm moved down the hierarchy one step at a time, looking at each child of the current frame, and if one was deemed better than the rest (and also better than the current frame), the algorithm would recurse, otherwise it would halt on the current frame. This introduces a major issue in precision: recall Algorithm 2 from section 7.3.3, importantly this snippet from the **crawl_tree** function shown in Algorithm 4:

```
...
for all child in location.children
     result = compare_frames(candidate, child)
     if result > best
          best = result
          best_location = child
...
```

**Algorithm 4:** A snippet of the original tree-crawling algorithm.

As can be seen, the code iterates through the frame's children, only assigning the child to the "best" value if its result beats the previous one. This does not, however, address a tie-value situation, where more than one (possibly all) of the children of a frame have an equal comparison value to the candidate, which is also higher than the current frame's comparison value.

Our solution, in the case of tie values, was to set them aside, and investigate all of their combined children as if the algorithm had recursed normally. If an eventual better value could be obtained in that way, then the algorithm could continue as normal. If no better value is eventually found, then an arbitrary member of the tie value set is selected. Intuitively, we are allowing the tie value children frames a pass, in order to verify that one

of their eventual descendants is a better match for the candidate without given immediate preference to the first frame listed as a child. That preference is held only until all other frames tied in value with that child fail to present a descendant with a better comparison value to the candidate. Algorithm 5 shows the enhanced tree-crawling metric, including the changes introduced in the previous section:

**English sentence probability filter** Until the tenth experiment, the preprocessing step, in particular the raw text corpus builder remained largely unchanged. Primarily, this process had been responsible for gathering raw text from the web that matched the target word, extracting only the relevant sentences, cleaning any excess web artifacts (such as HTML and other markup), and forwarding that collection to OntoSem to be semantically analyzed (for a detailed description of this process, see Chapter 6: Experiment Preparations: Corpus Building and Preprocessing).

One of the major benefits to using OntoSem as a semantic analyzer also presented itself as a drawback throughout the experimental process: OntoSem attempts to handle unexpected input gracefully, and will do the best it can with what it is given. This aspect was not only beneficial, but necessary to the experimental pipeline. However, it often caused an originally unforeseen problem: if truly poor input was fed to OntoSem, it would do the best it could, but would never fail to produce some sort of TMR. Using the web as a corpus meant that any amount of unexpected input could be fed to OntoSem under the guise of a being a well formed English sentence; this is simply not the case.

Often, after removing markup and running the sentence splitter on an input text, the remaining text flagged as a sentence that included the target word may actually have been effectively an artifact of the web-space that was not originally accounted for. A primary example includes web page menus. After the page passed through the HTML stripper, and was run through the sentence splitter, frequently one of the proposed ”sentences” simply

```
find_location_in_tree (frame candidate)
     best = 0
     best_location = NIL

     starting_points = database_select("frames with > 25
                       local property / fillers")
     crawl_tree(candidate, starting_points, 0)

     return best_location

crawl_tree (frame candidate, frame[] locations, double parent_best)
     best_locations[] = NIL
     best = 0

     for all location in locations
          result = compare_frames(candidate, location)
          if result > best
               best = result
               best_locations.append(location)

     if parent_best >= best
          return NIL

     if best_locations.itemcount = 1
          return best_locations[1]

     best_descendant = crawl_tree(candidate,
                                  best_locations.children, best)
     if best_descendant != NIL
          return best_descendant

     return best_locations[1]
```

**Algorithm 5:** The enhanced tree-crawling algorithm.

contained every word found in, sometimes extensively long, web page menus. If the menu happened to contain the target word, this ”sentence” would be passed on to OntoSem, and the resulting TMR would contain large amounts of bad data that would have an eventual negative impact on the learner.

In order to reduce the amount of poor input sent to OntoSem, a filter was introduced following the sentence splitter that would judge the likelihood of a text being an English sentence. The filter employed was a custom implementation of a hidden Markov model which was trained on a few chapters of the novel *Moby Dick*. Each sentence passed through the filter was judged by how well the model generates n-grams for it. Any sentence that generated a result of 0.5 or greater was retained. A few examples of texts and their scores are given below:

- 0.91: ”And, more trucks often mean less visibility for other drivers who impatiently race around a truck without always seeing what is clearly in front of them. ”
- 0.51: ”For the 1962 map of these freeways, Click here.”
- 0.27: ”Trucks for Sale — Trailers for Sale — New & Used — TruckSalesRUs.com-”
- 0.00: ”–ATS.”

The resulting collection of raw text was substantially reduced in size, with over 6000 texts being filtered out of the corpus as a result of passing through the hidden Markov model. This modification addressed the problem of having very poorly formed input from the web, however another form of web junk also needed to be filtered out before proceeding with the learner.

**Parse complexity filter** Junk from the web presented itself not only in the form of poor input characters, and poorly formed English sentences, but also semantically poor

texts as well. Texts that are very under- or over-specified can be just as detrimental to the learner if presented as a TMR. The tenth experiment, in addition to filtering texts based on their likelihood of being an English sentence, also introduced a second filter to the raw text corpus building step that removed sentences that were too specific, or not specific enough.

A custom implementation was added to the OntoSem suite of functions that would take a text as input and produce an estimation of the complexity required to produce a syntactic linking structure (one of the intermittent phases of the semantic analysis process). The higher the complexity, the substantially more in depth (and usually lengthy) a text would be. Similarly, the lower the complexity, the more simplistic (and short) the text would be. The intuition here was to find a middle ground: a text with a linking complexity that was neither too high nor too low would be a text with some amount of complex information in it (thus making it semantically valuable) but not so much so that relevant information was likely to be lost in an extremely tangled TMR. We were attempting to weed out texts that had nothing useful, while simultaneously weeding out texts that would be too challenging to cleanly extract what useful knowledge was there.

Two example texts are shown below, one with an extremely low syntactic complexity, and one with an extremely high syntactic complexity. These are shown to demonstrate their unsuitability towards the task. The first sample is syntactically simple, but has no useful semantic meaning at all. The second sample is very syntactically complex, and although semantically rich, contains far too many entangling meanings to extract anything relevant to the target word (in this case, ”bottleneck”).

- 14: ”I love that truck!”
- 99995938560: ”Teach your employees how to do everything you do so things don’t bottleneck around your skills.”

The filter selected all texts whose linking complexity landed between 1000 and

100000000. This resulted in a further 5000 texts filtered from the original corpus of 15188, leaving 4768 raw texts to be processed by OntoSem and then by the learner.

**Tree-crawling algorithm improvement: comparison value flexibility** Although the improvements introduced in Experiment 9 greatly enhanced the accuracy of the tree-crawling algorithm, it was still prone to settling into local maxima and halting. To address this problem, in Experiment 11, we implemented a subtle change to the algorithm to allow a degree of flexibility when finding the best match at any stage of the algorithm's recursion. Recall that the algorithm begins its search with a set of heavily defined frames, and for each one recurses down the ontological hierarchy until no children exist that exceed the parent frame's similarity score to the candidate frame. The algorithm handled tie values at this point, but could still halt if the only children of a frame had an equivalent (or lesser) score than the parent frame.

Ideally, the algorithm wouldn't fully halt at this point, but rather would choose to "peak" around the corner; in other words, we'd like the algorithm to continue on, so long as the loss in similarity was very small. To accomplish this, we implemented a flexibility measure into the tree-crawling algorithm's logic. Rather than halting when no children exceeded the parent frame's similarity score, the algorithm would continue if any of the children were within 1%, which was selected in order to account for, on average, two negatively impacting changes to property / filler pairs. In other words, on average, two property / filler pairs could be added, removed, or changed and the similarity measure would remain within 1%. This would allow the tree-crawling algorithm to continue to recurse down the hierarchy, in the hopes that greater similarities would be found deeper in the tree.

The key was to select a very small flexibility value, but not one so small that it would never trigger. Too large of a flexibility value would result in the algorithm recursing when it

should be clear that the candidate frame did not belong in that portion of the hierarchy, and too small a flexibility would result in effectively the same behavior the algorithm displayed prior to the Experiment 11, a propensity towards local maxima. Observation suggested that most often the algorithm halted when no children exceeded the parent frame's similarity score, but they were not worse either. The flexibility value of 1% would insure that tie values across hierarchy levels would be addressed, along with genuinely small disparities in scores. Algorithm 6 illustrates the inclusion of the flexibility value into the tree-crawling algorithm:

Although all three of the improvements to the tree-crawling algorithm substantially increased the number of frames that were compared to the candidate frame, overall only a small fraction were still considered, and each improvement assisted the algorithm in honing down the correct location of the candidate in the ontology while still avoiding a brute force search.

### 7.3.5 Experiment 12

The twelfth experiment deviated from the previous eleven in both scope and direction. While all prior experiments had focused on generating results in the form of higher quality candidate frames and on pinpointing a proposed location in the ontology for those frames, experiment twelve had a different focus. As can be seen in Chapter 8: Results and Discussion, although encouraging, the results from the previous experiments could be substantially improved. The problem, however, lay in blame placement. The learning pipeline is a very large system, with many layers that we had varying levels of control over. Each sub-system's results would have an impact on all subsequent processing, so it became clear that an experiment which would test each of these impacts was necessary.

Experiment 12 used the same experimental setup as the previous experiment, and included the same 4768 raw texts, and the same ten target words: *dragster*, *truck*, *freeway*,

```
find_location_in_tree (frame candidate)
     best = 0
     best_location = NIL

     starting_points = database_select("frames with > 25
                       local property / fillers")
     crawl_tree(candidate, starting_points, 0)

     return best_location

crawl_tree (frame candidate, frame[] locations,
            double parent_best)
     best_locations[] = NIL
     best = 0

     for all location in locations
          result = compare_frames(candidate, location)
          if result > best
               best = result
               best_locations.append(location)
          else if result > (best - (best * 0.1))
               best = result
               best_locations.append(location)

     if parent_best >= best
          return NIL

     if best_locations.itemcount = 1
          return best_locations[1]

     best_descendant = crawl_tree(candidate,
                                  best_locations.children, best)
     if best_descendant != NIL
          return best_descendant

     return best_locations[1]
```

**Algorithm 6:** The tree-crawling algorithm including flexible decision-making.

*skid*, *curb*, *chicken*, *jackknife*, *bottleneck*, *rubberneck*, and *hitchhike*. However, the twelfth experiment was designed to fork at multiple points along the learning pipeline; we would inject manual correction and support at three major junctions, and then test the final results against the fully automatic process. We decided to focus on each of the primary sections of the learner: first, the raw text corpus would be manually cleaned; second the TMRs produced would be corrected to the best standard without any modification to the static knowledge; finally the TMRs would be perfected by manually acquiring any missing or unknown word senses (with the exception of the target word).

This process presented us with a four-way set of values to compare against a manually created gold-standard word sense for each target word. The learner would have the fully automatic results, with no manual intervention; second, a nearly fully automatic process, where the raw text corpus had been manually pruned to include only texts that were clean and relevant; third, a mostly manual process that included the pruned raw text corpus and a manually corrected set of TMRs using the best available knowledge at the time; finally, a set of TMRs whose raw texts were chosen manually, and which were corrected to a near-golden standard, including manual acquisition of any knowledge that was required to complete each TMR.

These four experimental variants would each provide a metric of comparison would allow us to attempt to place blame in up to three categories: 1) the raw text corpus building, 2) the semantic analysis, and 3) the existing static knowledge resources. Although we expected each of these three factors to be responsible for some of the negative results, we wanted to see where the blame most heavily lied, in order to draw attention to future research goals. In principle, very poor input texts (or lack of significant quality input texts) should result in a lack of understanding by the learner. Additionally, semantic analysis is an extremely complex subject that is far from perfected, and will suffer even further if either the input text is bad (by any standards that the analyzer considers), or if the static

knowledge at the disposal of the analyzer is not sufficient to handle the input. By injecting manual correction at each of these levels, we effectively eliminate a factor as the cause of poor results - we can guarantee that the input text is clean, and semantically rich; we can guarantee that the semantic analysis is as correct as possible given the existing static knowledge; finally we can guarantee that the analysis is golden by manually adjusting the static knowledge to fit our needs. At each stage, we can compare the results to the fully automatic results and look for improvements.

**Baseline** In order to provide a basis of comparison for the rest of the experiment, we needed to produce a baseline level of results fully automatically. Using the same setup as the eleventh experiment, we produced a candidate ontology frame for each target word sense without any manual intervention. The same 4768 raw texts were processed by OntoSem, and the matching TMR frames were extracted and processed through the learner, producing the results. We were then able to judge the results of the remaining phases of Experiment 12 in relation to these results by comparing the frames produced by the learner to a set of gold-standard oracle frames manually created for the purposes of evaluation.

**AS-IS** The first modification to the baseline in the twelfth experiment involved manually pruning the raw text input to OntoSem. The automatically constructed raw text corpus consisted of 4768 sentences which came from the web, was cleaned, and passed through multiple filters in an attempt to have a set of easily parsed, moderately complex texts on the topic of the target words. However, many of the texts in the corpus were still poor inputs: either the text itself was not well structured, or it contained web artifacts, or the content was either wrong or misleading. Although the size of the corpus is intended to outweigh the poor inputs, erroneous data going in to OntoSem will have a trickle-down effect on the outputs of each subsequent stage.

To remove these problems from the pipeline, the stage following the baseline had the raw text input corpus manually pruned from the original 4768 texts down to only 92 hand selected texts. The texts selected were unmodified from the original input corpus (thus they were found on the web during the initial search), and were selected as examples of clean, well formed, and contentful texts on the topics of the target words. Samples of texts that were both included (1: the target word is rubberneck) and rejected (2: the target word is curb) are shown below:

1. ”Always slow down and rubberneck when you see an accident or even a person changing a tire.”

2. ”These statistics prompted the community to mobilize and develop Project CURB.”

The second sample shows clearly a poor choice for an input sentence: although the text has no web artifacts, spelling errors, or poor grammar, the noise introduced by the proper noun ”Project CURB” will produce some degree of rippled effects throughout the remainder of the learning process. By removing this text from the corpus, its negative impact is also removed. Intuitively, by hand selecting a raw text corpus, we have emulated a stronger web-crawler and text filterer, thus removing the impact of having an imperfect preprocessing phase from the final results. The AS-IS phase continued through to the final results without any further manual interruption.

**Edited** The second modification to the baseline included the manual work done in the AS-IS phase. In other words, the Edited corpus of TMRs was produced from only the hand-selected raw texts chosen during the first modification to the twelfth experiment, but also included its own manual corrections.

The corpus of TMRs produced from the 92 raw texts by OntoSem was imported into DEKADE (see Appendix B: Tools and Interfaces) to be manually corrected to a ”near

bronze" standard. The TMRs were inspected and modified by hand to produce the most accurate TMR of the original input text given the existing static knowledge resources. It is important to note that the goal of this phase was to reduce the noise generated by OntoSem itself, given that a more correct TMR could be produced with what OntoSem had at its disposal. Although many of the input sentences included missing lexical senses that were not the target words, these senses needed to remain ambiguous in order to emulate the best possible results OntoSem could have produced.

By including the manual corpus selection from the previous phase, the Edited phase resulted in a collection of TMRs that were generated from only semantically useful, moderately complex texts and were hand corrected to the highest standard possible without any modification to the existing static knowledge resources. The Edited phase continued to process automatically after the TMRs were corrected.

**Lexicalized** The final modification to the baseline included all of the prior manual work: the manually pruned raw text corpus, and the TMRs that were corrected to the highest standard available given the knowledge resources. To remove the final source of poor input to the learner, the TMRs needed to be further improved by manually acquiring lexical and ontological resources in order to fill every gap in the TMR corpus with the exception of the target words. This task would produce a "best-of-all-worlds" corpus of input TMRs for the learner: the source material was on point, and the TMRs were completely corrected, with all word senses accurately assigned, leaving only the target word to be learned.

The input texts averaged just over one missing word sense (not including the target word), meaning that most TMRs automatically produced before this phase of manual correction would have a second layer of ambiguity to further confuse the analyzer as it worked around the target word. Removing this layer of ambiguity reduced the error generated on most target words.

The final phase gave us our fourth vector of comparison: we were able to judge the results of a fully automatic process against the results of the same process where each phase had progressively more human interaction; the texts were manually pruned, the TMRs were corrected to the best available standard, and finally the static knowledge was updated (and the TMRs were finalized). These four comparisons were able to shed a good amount of light into where the performance of the learner was degrading. For a complete discussion of these results, see Chapter 8: Results and Discussion.

## 7.4 Evaluation Overviews

In this section we discuss the evaluation processes that were used throughout each experiment's life cycle. Much like the experimental setup itself, we adapted the way evaluation of the results should be performed over time. Initially, the experiments were simply aimed at obtaining results for proof of concept. As a result, evaluation took a back seat to the development of the process. As the experiments matured, evaluation was also addressed, but still with the mentality that the results were simply an indication of the quality of the learner, and not so much as an indication of the quality of the process. The final experiment focused completely on evaluation, and was intended to show how the process itself should be evaluated. Results of the individual experiments will not be discussed in this section, but can be found in Chapter 8: Results and Discussion.

### 7.4.1 Experiment 1

The first experiment was designed to generate further interest in the research by producing tangible results. Primarily, the output was inspected by hand to assess the results produced by the learner in comparison to what a human researcher suspected should be included. This was a non-scientific approach meant purely to identify the plausibility of

the experiments themselves. In order to provide a scientifically based approximation of the quality of the results, we then proceeded to compare the candidate frames with the ”next best available” frames using the OntoSearch (Onyshkevych 1997) algorithm.

Intuitively, we selected the frame in the ontology that was the best representation of the frame that was being learned. This means that a gold standard was not created for the purposes of evaluation, but rather the next best was chosen. As an example, one of the target words was pundit. Rather than acquiring an exact meaning of pundit into the ontology for comparison, we manually located a word with a very similar (but not exact) meaning that already existed: intellectual. We then produced a comparison value by processing the pundit and intellectual frames through OntoSearch. In order to provide a basis of comparison to that value, we then compared pundit with every other frame in the ontology using OntoSearch. This resulted in a list of the highest scoring frames when compared to the target, which allowed us to see how well the desired match (intellectual) stacked up, when compared to the exhaustive list of all comparisons.

Since OntoSearch returns a percentage result, we were able to produce a difference value (how far off from the best match was the desired match), a rank (how many frames matched better than the target), and a percentile (an alternate view of the rank). However, OntoSearch is an inherently flawed comparison metric for our purposes, and this was addressed in subsequent experiments both for the purposes of evaluation (as discussed here) and for the purposes of learning (as discussed in the previous sections).

OntoSearch provided for us a distance value between the target frame and the desired match. By crossing connections in the ontology (property / fillers that associate two frames together) and applying a penalty for each connection, OntoSearch finds the shortest distance between two frames. We had originally intended to argue that the shorter the distance, the higher the similarity, hence our original use of the OntoSearch metric. However, as the experiment progressed it became clear that this is not a good indication of similarity,

and is therefore a bad evaluation metric for the learner. OntoSearch will cross any connection in the ontology's graph that it finds, and therefore any reference the target frame has to the desired match will result in a high score. Conversely, if the target frame has little to no connections to the majority of the ontology (due to the learner's inherent nature), it may not be able to find a clear path to the desired match at all, and could result in a very low score.

OntoSearch assumes that the ontology is well formed, and that both frames input to it are found within the ontology. This assumption is in contradiction with our use of the metric, as the target frame has just been introduced by the learner and is not yet connected to the ontology at all. Even if the method which OntoSearch employes was appropriate for determining similarity, it would still not be proper to use for our purposes as the target frame has no hierarchical connection to the ontology, and is thus missing IS-A links, as well as a tremendous amount of inherited knowledge. OntoSearch cannot be expected to perform adequately under these conditions, so it became clear that although our overall method of evaluation (select a desired match, produce a similarity score and comparing that score to other scores produced by the remainder of the ontology) was sound, the metric we were using to produce similarity scores needed to be replaced.

### 7.4.2 Experiments 2 & 3

The second and third experiments replaced the inappropriate similarity metric OntoSearch with a custom metric described in detail in the previous section. This metric was used during the experimental procedure to locate a suitable place for the candidate frames, but also doubled as a means of evaluation, similarly to how OntoSearch was used in Experiment 1.

The evaluation process used was similar: a desired matching frame that already existed was selected in lieu of producing a gold-standard by hand. The candidate frame was

compared to the desired match using the new similarity metric, producing an ”accuracy” score. In other words, the metric was calculating the similarity of the candidate frame with its intended match - a higher number indicates a higher degree of similarity, and thus better accuracy. In order have a basis of comparison, during both of these experiments the candidate frame was also compared, exhaustively, to the remainder of the frames in the ontology.

Each of these experiments had effectively three vectors to compare for evaluation. First, the frame proposed by the learner (being the existing frame that had the highest comparison score with the candidate). That score could then be compared to the second value: the score of the desired match (the semi-golden-standard existing frame that was hand selected as the target for the candidate frame). Finally, both of these scores could be ranked according to all of the other scores for the remainder of the frames in the ontology, providing an indication as to how far off the proposed frame was from the desired frame in terms of both similarity score difference and in number of intervening frames.

The distinction in the these last two comparisons is important: it is necessary to note that although the proposed frame and the desired frame may both have very close similarity scores (within a few percentage points of each other) it is very meaningful to know if many (or few) other frames scored better than the desired frame but worse than the proposed frame. A high number of intervening frames indicates that the candidate frame was either missing some crucial knowledge or, more likely, had some erroneous knowledge that was very important to a branch of the ontology unrelated to the desired frame. Armed with this data, we were able to address some of the issues with the learner in future experiments.

### 7.4.3 Experiments 4 - 8

The next round of experiments introduced the tree-crawling algorithm into the learner, which had a profound effect on the way evaluation was approached. The tree-crawler was

primarily designed to eliminate the costly operation of locating the best fit frame for the candidate by making a decision at each branch in the ontology, rather than performing an exhaustive search. The tree-crawling algorithm, as well as its output, however, was also usable for evaluation purposes in a broad sense. Intuitively, one can judge the quality of the candidate against the desired frame by seeing at what level the tree-crawling algorithm diverged from the correct path.

Once again, in lieu of a hand-created gold-standard to use for evaluation, we opted to select the closest matched frames for each candidate to compare to. Using the custom similarity metric described earlier, we could produce an ontology-aware similarity score between the candidate and our target. Rather than simply produce this value, using the tree-crawling algorithm, we could also observe where along the path from the ontology's root to the desired frame the algorithm diverged. Intuitively, the shallower this location, the worse the performance. If a frame had troubles making even high-level comparisons (such as selecting OBJECT, rather than EVENT), then certainly the target and desired frames were not similar. This gives us more scrutinizing power over the values that are reported by the custom similarity metric: although the scores may subjectively appear high, it may be that the frame shares many very common property / fillers that would result in a high score with most frames in the ontology. A subjectively lower score may actually disguise positive performance, as it may reflect a hit on the important property / fillers, while missing some of the more broadly available ones.

The previous evaluation techniques (in Experiments 2 and 3) relied on demonstrating an absolute position in relation to the other frames in the ontology. We tested the desired frame against the candidate, and then ranked that similarity value amongst all the other frames: doing so provided us a similar level of investigative strength into the results. With the improvements to the similarity metric used in the fourth experiment on, the exhaustive comparison was no long practical, so replacing it with an investigation into the performance

of the tree-crawling algorithm was a somewhat suitable replacement. We were able to see the similarity score of the desired frame, as well as the similarity score of the frame selected by the learner. Additionally, we could manually investigate the path the tree-crawling algorithm took in order to identify areas needing improvement. Although we do not report any firm results on this evaluation technique, it was used often to ”spot-check” the learner's progress.

One drawback in our evaluation methodology during these experiments was a heavy focus on recall, and less so on precision. We were primarily interested in seeing if the learner could pick up on the target words (recall) and were dismissive of the vast amount of false positives that were produced (precision). Some target words may have had a few senses, and although the learner did find them (as evidenced by high similarity scores and close tree-crawling paths), it also found, in some cases, dozens of other word senses that had no desired match. Although we did not address this problem immediately, this was a primary motivating factor in the setup decisions of the twelfth experiment.

### 7.4.4 Experiments 9 - 11

Starting with the ninth experiment, we made an adjustment to one of the parameters used in the comparison metric. The comparison metric was designed to look at every single property / filler pair in both frames being compared, and produce a similarity score based on possible matches. Naturally, this score needed to include property / filler pairs that were both locally defined and inherited from ancestors in the ontological tree. This was done to insure that two extremely similar frames did not suffer due to having very few locally defined property / filler pairs: very low hanging frames (often leaves in the hierarchy) derive nearly all of their property / filler pairs from inheritance, and have very few (sometimes only a handful) that are locally defined. Two such frames that are siblings would score either extremely high (desired) with the inclusion of inherited property / fillers

in the metric, or extremely low (not desired) without the inclusion of such data.

Although this decision is appropriate for the design as a metric, it does pose a previously unforeseen problem in using the custom comparison metric as a source of evaluation: the learner uses the tree-crawling algorithm in conjunction with the comparison metric to locate a suitable place for the candidate frame in the ontology. In so doing, it enhances the frame with inherited values as it is pushed further down the hierarchy. The result is a semantically well-formed frame with a proposed location in the ontology, suitable for immediate inclusion in the static knowledge. This is problematic to the evaluation process because now the candidate frame has potentially hundreds of property / filler pairs that are identical to its proposed parents, but were not originally found in the TMRs produced by the semantic analysis. These property / filler pairs have been retrofitted on to the candidate during the final stage of the learning process, and heavily slant the evaluation results as shown in Chapter 8: Results and Discussion. It was imperative that we remove the influence of these inherited properties, but needed to retain the influence of any inherited properties the desired frame may have.

In order to achieve a more accurate evaluation using the comparison metric, we needed to compare the candidate frame to the desired frame (or any other frame) using only the candidate frame's local fillers, but without modifying the existing frame at all. This would allow us to judge how much of the full definition of an existing frame (both the local and inherited properties) matched the learned definition of the candidate frame (just the local fillers that were extracted from the TMRs and passed through the learners, and not including the inherited fillers that were added as a result of navigating the candidate frame through the hierarchy using the tree-crawling algorithm).

The results of this change saw a sharp decline in the previously over-inflated similarity scores, resulting in a much more accurate similarity for use in evaluation. As with the previous round of experiments, we continued to produce results that showed the similarity

of the candidate frame to the desired matching frame (which was pre-existing, and not hand-created for the purposes of evaluation), as well the similarity to the highest match as reported by the tree-crawling algorithm (rather than an exhaustive comparison of the entire ontology). The paths that the tree-crawling algorithm chose for any given candidate were also spot checked as another form of unreported manual evaluation, which provided insight into areas needing improvement, and led to changes in methodology in the final experiment.

### 7.4.5 Experiment 12

Unlike the previous eleven experiments, Experiment 12 was designed to focus on evaluation, not results. Rather than manipulating the learner, or the experimental pipeline as we had done previously, instead we maintained the same system, but set up an experiment with the intention of identifying problem areas through a series of cascading evaluations based on the compared relative quality of results. The experiment pipeline consists of several fully automatic, but discrete, stages. Any of these stages could produce less than desirable results, which then would have a trickle-down effect on the remaining parts of the learner. In order to identify which stages were most contributing to the perceived negative results, we decided to focus on evaluating the results that were produced by each stage, and we did so by running the automatic learning process side-by-side with a manually intervened corpus.

To evaluate each process, we still needed to be able to evaluate the individual results that were output by the learner. To do this, we used the same process as in the previous experiments, however, instead of selecting a ”desired frame” (a best fit for the candidate given the current state of the ontology), we manually produced gold-standard frames for each word sense being learned. To evaluate the output, we were able to produce a similarity result using the custom comparison metric described earlier by comparing the candi-

date frame and the gold-standard frame. Additionally, we produced the similarity results from comparing the candidate frame to the ”best-match frame” that was suggested by the tree-crawling algorithm. Additionally, we produced several more comparison values as an enhancement to the comparison metric, in order to have a more accurate precision, recall, and f-measure.

In order to use these similarity values to evaluate various stages of the learner, we needed to provide a set of values to compare to. To do this, at each stage of the learner, we created a branch in the process: down one path the learner continued on in a fully automatic way, and down the other, the results from the stage were manually modified to the best possible quality before the learner was allowed to continue. This branch was injected at three points during the process: 1) after selecting the raw text corpus to process, 2) after producing the automatic TMRs, and 3) after correcting the automatic TMRs. Intuitively, we were attempting to capture the existing drawbacks in the learning process: 1) the web as a corpus is open and subject to noise, and erroneous or misleading data; 2) OntoSem is a vastly complex system that is far from perfect in its production of TMRs; 3) the semantic ontology and lexicon OntoSem uses is a manually created knowledge repository that is (by definition of this experiment) lacking in coverage. We hoped to show that: 1) by hand-selecting the raw texts, and thus eliminating noise and focusing the value of the knowledge present in the texts, we could improve the overall results; 2) by manually correcting the TMRs that were produced by OntoSem to the best standard available, given the knowledge, we could assist the learner in selecting better property / fillers; and 3) by manually acquiring any missing knowledge that would benefit the TMRs, we could pass the learner the best possible input.

The goal of the twelfth experiment was evaluation, and by producing evaluated results along multiple vectors, we were able to layer in another form of evaluation: one that looked at each stage of the learning process as it relates to the whole. There is an important

distinction between this evaluation, and directed evaluation of the stage itself. Rather than evaluating OntoSem based on its ability to produce quality TMRs, or the corpus builder based on its ability to extract meaningful texts, we were evaluating these processes based on their ability to interact with the learner as a whole. Ample evaluation has already been performed on OntoSem (Nirenburg, Beale, & McShane 2004), and much has been written about using the web as a corpus (Kilgarriff & Grefenstette 2003). For our purposes, it is more important to see how using the web as a corpus impacts OntoSem's ability to produce TMRs. The learner is a pipeline, and each process along the path is intrinsically tied to the others. This evaluation was designed to see what impact the quality of each process had on the other processes, not just to evaluate the quality of a process in isolation.

**Precision and Recall** Building on previous methods of evaluation, which were recall-centric, we felt it was necessary to also account for precision as we tried to nail down the problematic processes within the learner. To this end, we enhanced and modified the existing custom comparison metric to now report an array of scores, rather than the single one that had been previously reported.

Until now, the metric had been reporting only one score, which was an average of two internally calculated scores, both of which were targeting recall, rather than precision. For the twelfth experiment, we reported both of those values, along with the originally reported value, plus a newly calculated precision score, and the f-measure of the precision and averaged recall.

Recall had previously been calculated by averaging two specific recall based scores that were internally computed. The first such score focused on the recall of properties (not fillers) found to be matching between the candidate frame and the existing frame. In other words, we were looking to see if the candidate had any filler for a property which the existing frame had a filler for. This test is somewhat designed to anticipate future research:

knowing that a candidate should have fillers for a property is a positive result, regardless of which fillers are currently known (or are erroneously reported). In order to calculate this, we calculate the intersection of shared properties between the candidate and existing frames, and divide by the union of the properties defined in both frames; Equation 7.3 defines *RECALL_PROP*:

$$RECALL\_PROP = \frac{\cap(frame1.properties, frame2.properties)}{\cup(frame1.properties, frame2.properties)} \tag{7.3}$$

The second recall score used to find a more reportable recall was designed to capture recall of the fillers within the properties. To do this, an average of all internally computed property scores was taken for each property in the intersection of properties. In other words, using only the properties shared by the two frames, each property's score was internally calculated (as discussed previously in this chapter), these scores were then averaged, resulting in a recall-based score using matching fillers, rather than just matching properties. Intuitively, we are capturing the similarity of the fillers within each property found in the intersection of properties; although the internally computed value is somewhat precision-based, the only properties that are used are the ones that are shared. Equation 7.4 defines *RECALL_FILL*:

$$RECALL\_FILL = avg(\cap(frame1.properties, frame2.properties).score) \tag{7.4}$$

*RECALL_PROP* and *RECALL_FILL* were used to calculate a more representative recall score which accounted for the two important vectors being captured: recall of properties found, and accuracy of fillers within found properties. Thus, *RECALL* is defined in Equation 7.5:

$$RECALL = avg(RECALL\_PROP, RECALL\_FILL) \tag{7.5}$$

To enhance our evaluative methods, we added a new precision score calculation for the similarity metric to report. To calculate precision, we averaged the internally computed property score for each property defined by the candidate frame. In this case, we are only interested in how correct each property proposed by the learner was, and so we are not interested in intersections with the existing frame's properties. Rather, we want to score each property that is defined in the candidate; if no matches are found for a particular property in the existing frame, the internal score will be a zero, and will reduce the overall computed average precision. A property with a perfect one-to-one match of fillers will have the maximum value of one, and will not penalize the precision score. Equation 7.6 defines *PRECISION*:

$$PRECISION = avg(candidate - frame.properties.score) \tag{7.6}$$

Finally, to combine our *PRECISION* and *RECALL* scores, we calculated a standard f-measure score, using the common algorithm defined in Equation 7.7:

$$F\_MEASURE = \frac{2 * PRECISION * RECALL}{PRECISION + RECALL} \tag{7.7}$$

Using these five scores, with a focus on the f-measure, we were able to more closely examine the results that were produced from each stage of Experiment 12's process. Having a firmer grasp on the evaluation of the candidate frames allowed us to more accurately assess and evaluate the effects of each manually injected change that went in to the learner during the experiment. The results of this evaluation, as well as the evaluations performed for all other experiments are presented in Chapter 8: Results and Discussion.

Chapter 8

# RESULTS AND DISCUSSION

## 8.1 Introduction

In the previous chapter, we introduced and described in detail the setup for each of the twelve distinct experiments, and followed with a discussion on the evolution of our evaluation methodology. In this chapter, we'll present the individual results for each experiment, and motivate a discussion for each, examining the conditions that led to each recordable result. Previously, for the purpose of discussion, we had grouped the experiments by the factors that divided their setup and execution. In this chapter, we'll rearrange the groupings, joining experiments that share a similar corpus and set of target words, in order to more easily compare, side by side, their individual results.

In the next section, we'll reintroduce Experiments 1 and 2, briefly recap their setup and evaluation metrics, and discuss their results, both independently, and jointly. In section 8.3, we'll cover the third experiment, which was highly similar to Experiment 2, but used a different corpus and set of target words. Experiment 4 will be discussed in section 8.4, and will include a comparison between a custom implementation of a word sense clusterer to a commonly used technique known as "bag of words". Section 8.5 will detail Experiments 5, 6, 7, and 8, each of which was an evolution on the previous one. We'll show their results independently and jointly with the preceding experiment, as well as a final set of

all results for a side by side comparison. In section 8.6, we'll cover Experiments 9, 10, and 11, in much the same way as the previous four experiments were reported. Section 8.7 will discuss Experiment 12, whose focus was on evaluation of individual components, each of which will be discussed and shown, with the final results of all branches of the experiment being compared to each other. Finally, we'll conclude the chapter with a recap of the lessons learned from each set of experiments, leading to the discussion on future work, in the following chapter.

Tables 8.1 and 8.2 provides an overview of the changes in the corpus and processes for each of the first eleven experiments (Experiment 12 is intentionally omitted as it does not fit the same paradigm). This is intended as a quick reminder of the previous chapter, as well as a glimpse into the sections ahead. The table details the parameters for each experiment, including which processes were used. Some fields include an "*" by their contents: this implies that the individual process referred to in that experiment changed from the one immediately preceding it. For example, Experiment 6 lists its similarity metric as "Custom *". This means that the custom metric used in Experiment 5 was modified for the sixth experiment (in contrast to Experiment 5, whose similarity metric was unchanged from the fourth experiment). Each of these changes will be detailed in the coming sections.

Throughout this chapter, trends will be used in various charts and diagrams: the trendlines produced are a standard linear function, and the target words are ordered, ascending, by the number of texts in the corpus in which they were contained. In other words, the trendlines show the average change in value as the number of input texts increased. This is done to highlight one of our principle hypotheses, that the quality of coverage of a target word increases as the size of the input corpus increases.

| | # Sentences | #Target Words | #Senses | EM? | Tree-crawler? |
|---|---|---|---|---|---|
| Experiment 1 | 2802 | 4 | 4 | no | no |
| Experiment 2 | 2802 | 4 | 4 | no | no |
| Experiment 3 | 4703 | 12 | 12 | no | no |
| Experiment 4 | 15416 | 9 | 18 | yes | yes |
| Experiment 5 | 15188 | 10 | 17 | yes | yes |
| Experiment 6 | 15188 | 10 | 17 | yes | yes |
| Experiment 7 | 15188 | 10 | 17 | yes | yes |
| Experiment 8 | 15188 | 10 | 17 | yes | yes* |
| Experiment 9 | 15188 | 10 | 17 | yes | yes* |
| Experiment 10 | 4768 | 10 | 17 | yes | yes |
| Experiment 11 | 4768 | 10 | 17 | yes | yes* |

Table 8.1. Overview of experimental parameters (1).

| | Target Word Type | Similarity Metric | Sentence Filters? |
|---|---|---|---|
| Experiment 1 | monosemous | OntoSearch | no |
| Experiment 2 | monosemous | Custom | no |
| Experiment 3 | monosemous | Custom | no |
| Experiment 4 | polysemous | Custom | no |
| Experiment 5 | polysemous | Custom | no |
| Experiment 6 | polysemous | Custom* | no |
| Experiment 7 | polysemous | Custom | yes |
| Experiment 8 | polysemous | Custom | yes* |
| Experiment 9 | polysemous | Custom* | yes |
| Experiment 10 | polysemous | Custom | yes* |
| Experiment 11 | polysemous | Custom* | yes |

Table 8.2. Overview of experimental parameters (2).

## 8.2 Experiments 1 & 2

The first two experiments, as described fully in Chapter 7: Experimental Setup and Evaluation Processes, varied only in the metric used to carry out the evaluation, using the same input corpus and target words. Both experiments focused on a fairly arbitrary selection of four target words, using a corpus that was automatically gathered from the web and semantically analyzed by OntoSem. Neither experiment expected polysemous words, resulting in all learned property / filler pairs being mapped to one uniform candidate concept for each target word.

Experiment 1 (English & Nirenburg 2007b) introduced the four target words: *pundit*, *CEO*, *hobbit*, and *song*. Although the words were not selected on their absence of polysemy, most of the words have only one common meaning. Interestingly, hobbit was the only word with two commonly used meanings: the fictional humanoid species, and the title of a book. Proper nouns were not considered during this (nor any subsequent) experiments, becoming, essentially, another word sense.

The corpus was automatically constructed, and consisted of a total of 2802 sentences containing at least one instance of one of the target words. Each word was passed through the learner, resulting in a candidate concept whose place in the ontology was suggested by calculating its similarity value against each existing concept using OntoSearch, and selecting the highest match as a proposed parent. Evaluation was performed by hand selecting the preferred parent, and determining its similarity score with the candidate. Several values were then computed, including a difference score (simply, the difference in similarity between the proposed parent, and the preferred parent), a rank (how many concepts scored higher than the preferred parent) and a percentile (the rank score, normalized). Table 8.3 shows each word, the number of sentences in the corpus matching that word, the score for the proposed parent(s), the score for the preferred parent, and each of the evaluation scores:

| | song | pundit | CEO | hobbit |
|---|---|---|---|---|
| **# Sentences** | 339 | 453 | 552 | 1458 |
| **Proposed Parent Score** | 0.8 | 0.8 | 0.9 | 0.9 |
| **Preferred Parent Score** | 0.8 | 0.679 | 0.638 | 0.806 |
| **Similarity Difference** | 0 | 0.121 | 0.262 | 0.094 |
| **Rank (out of 6000)** | 12 | 210 | >500 | 18 |
| **Percentile** | 0.2 | 3.5 | >8.3 | 0.3 |

Table 8.3. Experiment 1 results overview.

| **Word** | **Proposed Parents (concepts)** | **Preferred Parent (concept)** |
|---|---|---|
| song | WORD, RECORD-TEXT, OBJECT, 8 others | SONG |
| pundit | TELEVISION, CITIZEN, HUMAN, 12 others | INTELLECTUAL |
| CEO | EVENT | PRESIDENT-CORPORATION |
| hobbit | PUBLISH | HUMAN |

Table 8.4. Experiment 1 proposed and preferred parent(s).

At a glance, it can be seen that the score for the proposed parent increased along with the number of sentences. It should be noted however, that this is neither unexpected, nor a very meaningful value. The score matching the proposed parent is simply the highest match to any concept in the ontology, not necessarily a desired one (Table 8.4 shows the target words, along with the proposed parent concept(s) and preferred parent concept). Recall that OntoSearch, the similarity metric used in this experiment, does not penalize for extra (incorrect) knowledge, but simply finds the shortest distance between two concepts. As the number of texts increases, the volume of relations in the candidate will increase, thus providing OntoSearch with more possible paths to traverse, inevitably leading to higher scores.

It should be noted that the quantity of texts did not seem to have an appreciable effect on the similarity score with the preferred parent (the existing concept hand-selected as the parent of the candidate). The highest matches to the preferred parents were the two target words with the least, and most texts, suggesting that another factor was at play, possibly the

| Word | # Sentences | Proposed Parent Score | Preferred Parent Score |
|---|---|---|---|
| song | 339 | 0.446 | 0.36 |
| pundit | 453 | 0.458 | 0.45 |
| CEO | 552 | 0.448 | 0.417 |
| hobbit | 1458 | 0.493 | 0.493 |

Table 8.5. Experiment 2 results overview.

quality of the input texts for pundit and CEO. Notably, CEO presents an array of problems that were not addressed by this experiment, as a result of being an acronym. Although the most common usage, ”chief executive officer” was the intended meaning, a seemingly limitless supply of alternate meanings of CEO become available from a simple internet search. This almost certainly introduced a large amount of noise, resulting in the poorest showing of results.

The results reported in Table 8.3 are deceptively high, and appear scattered: OntoSearch, as discussed in Chapter 7: Experimental Setup and Evaluation Processes, is not an appropriate similarity metric, and further, is heavily weighted towards higher numbers. It is extremely unusual to see OntoSearch results under 0.7 (this value is a result of internal tuning in OntoSem), and so the results presented need to be taken with a grain of salt. To address the problems presented by using OntoSearch, Experiment 2 introduced the custom comparison metric, specifically designed to produce an accurate similarity measure between a candidate concept and an existing concept. Using the same input texts, and target words, Experiment 2 addressed the evaluation and results presented by the first experiment, by improving on the similarity metric used. Table 8.5 shows the results of the second experiment:

Assuming CEO as an outlier, due to the noise generated in the input corpus, a pattern begins to emerge once an appropriate similarity metric is applied to the results: as the number of input sentences increases, the score to the proposed parent increases (implying

broader coverage), the score to the preferred parent increases (implying increased accuracy) and the difference between the the two decreases (implying reduced error rates). In other words, this experiment showed that increasing the amount of quality input texts produced candidate concepts that contained more meaningful and correct semantic information, and reduced the amount (or impact) of corpus noise. Figure 8.1 shows this pattern of increased quality of results with increased corpus size:

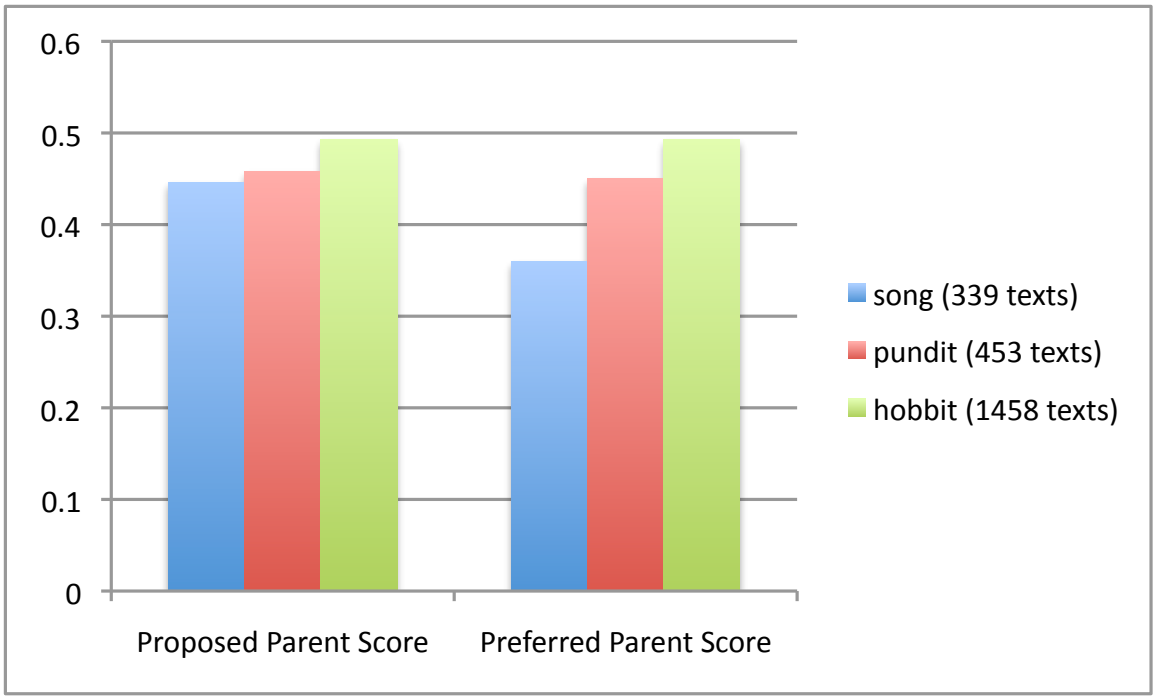


FIG. 8.1. Increased corpus size produces increased quality of results..

The results produced from Experiment 2 were much more accurately evaluated as a result of the inclusion of the custom comparison metric. As mentioned earlier, OntoSearch does not penalize a concept for containing erroneous data; this, however, is captured in the custom comparison metric by producing an accurate similarity of all property / filler pairs in both concepts. Having a property / filler pair in the candidate concept that does not have a match in the preferred parent will result in a penalty to the similarity score,

| Word | # Sen | EXP1 Prop | EXP1 Pref | EXP2 Prop | EXP2 Pref |
|---|---|---|---|---|---|
| song | 339 | 0.8 00 | 0.8 00 | 0.446 | 0.360 |
| pundit | 453 | 0.8 00 | 0.679 | 0.458 | 0.450 |
| CEO | 552 | 0.9 00 | 0.638 | 0.446 | 0.360 |
| hobbit | 1458 | 0.9 00 | 0.806 | 0.493 | 0.493 |

# Sen = The number of input sentences for the target word.
EXP1 Prop = Proposed parent score for Experiment 1.
EXP1 Pref = Preferred parent score for Experiment 1.
EXP2 Prop = Proposed parent score for Experiment 2.
EXP2 Pref = Preferred parent score for Experiment 2.

Table 8.6. Comparison of evaluation results for Experiments 1 & 2.

producing a much more accurate evaluation metric. Table 8.6 shows both sets of scores (from Experiment 1 and Experiment 2), side by side. It can be seen that the inclusion of a penalty both more accurately reflects the actual similarity (reduced overall scores), and clarifies the expected pattern of increased quality of results given a large input corpus.

The results from the first two experiments demonstrated a key hypothesis in our research, and showed that the inclusion of a custom comparison metric was a necessary step towards properly calculating concept similarity (allowing the learner to more accurately suggest proposed parents in the ontology, and allowing more precise and useful evaluation outputs). In the next section, we'll discuss Experiment 3, whose setup and evaluation processes were identical to Experiment 2, but used a different set of target words and a completely new input corpus.

## 8.3 Experiment 3

C.3. Experiment 3

The third experiment (Nirenburg *et al.* 2007) used an identical setup and evaluation technique as Experiment 2, but with a significantly expanded corpus and set of target words selected intentionally due to their monosemy. The new corpus consisted of 4703

| Word | Preferred Parent (concept) |
|---|---|
| depose | DEPOSE |
| wigger | SOCIAL-ROLE |
| obey | OBEY |
| triceratops | DINOSAUR |
| cherimoya | FRUIT-FOODSTUFF |
| brontosaurus | DINOSAUR |
| syrup | PLANT-DERIVED-FOODSTUFF |
| stegosaurus | DINOSAUR |
| spartan | MILITARY-ROLE |
| diplodocus | DINOSAUR |
| deport | BANISH |
| pledge | PROMISE |

Table 8.7. Experiment 3 target words and preferred parents.

input sentences, covering twelve new target words: *brontosaurus*, *cherimoya*, *deport*, *depose*, *diplodocus*, *obey*, *pledge*, *spartan*, *stegosaurus*, *syrup*, *triceratops*, and *wigger*. Table 8.7 shows the target words, along with their preferred parents (existing concepts in the ontology):

Similarly to the first two experiments, in some cases a target word may already have a perfect meaning in the ontology (or even the lexicon). In order to simulate learning, these frames and lexical senses were temporarily removed for the duration of the experiment, and were returned to the static knowledge later to serve as a sort of gold-standard for evaluation. Words that did not have a perfect meaning, as before, were mapped to the next best choice (e.g. ”brontosaurus” to DINOSAUR).

Unfortunately, the results from the experiment follow a less obvious trend than those in Experiment 2, which may be a result of the more unusual (and semantically complex) target words, or possibly a result of a poorly assembled corpus. Table 8.8 shows the results in a raw form:

As the number of sentences increases, we detect a dropping trend, that appears to

| Word | # Sen | Prop Score | Pref Score | Difference |
|---|---|---|---|---|
| depose | 54 | 0.600 | 0.479 | 0.121 |
| wigger | 57 | 0.489 | 0.484 | 0.005 |
| obey | 60 | 0.460 | 0.384 | 0.076 |
| triceratops | 84 | 0.488 | 0.482 | 0.006 |
| cherimoya | 148 | 0.453 | 0.335 | 0.118 |
| brontosaurus | 302 | 0.492 | 0.373 | 0.119 |
| syrup | 322 | 0.465 | 0.423 | 0.042 |
| stegosaurus | 415 | 0.538 | 0.499 | 0.039 |
| spartan | 426 | 0.492 | 0.481 | 0.011 |
| diplodocus | 469 | 0.550 | 0.500 | 0.050 |
| deport | 1043 | 0.485 | 0.409 | 0.076 |
| pledge | 1323 | 0.436 | 0.335 | 0.101 |

# Sen = The number of input sentences for the target word.
Prop Score = Score for the proposed parent.
Pref Score = Score for the preferred parent.
Difference = Similarity difference between scores (proposed and preferred).

Table 8.8. Experiment 3 results overview.

bottom out in the middle and rises back up. This can be better seen in Figure 8.2, which graphs the proposed parent score and the preferred parent score over a change in input size. The trend represents an inverted bell curve, and could indicate the presence of noise in the corpus: the relatively low sentence count target words may have very little noise, the middle sentence count words having the highest ”noise to value” ratio, followed by the high sentence count words that statistically outweigh the noise with semantically truthful knowledge.

The results from the third experiment were very inconclusive, evading most any discernible pattern, leading only to the conclusion that a component of the process caused enough error to throw off some of the results. Although individually, the results are not bad, averaging 43.2% similarity to the preferred concept (Experiment 2 averaged 43.0%, unsurprising as the process did not change at all, indicating consistency), a clearer indication of the effect various inputs have on the final results was desired, and would be directly

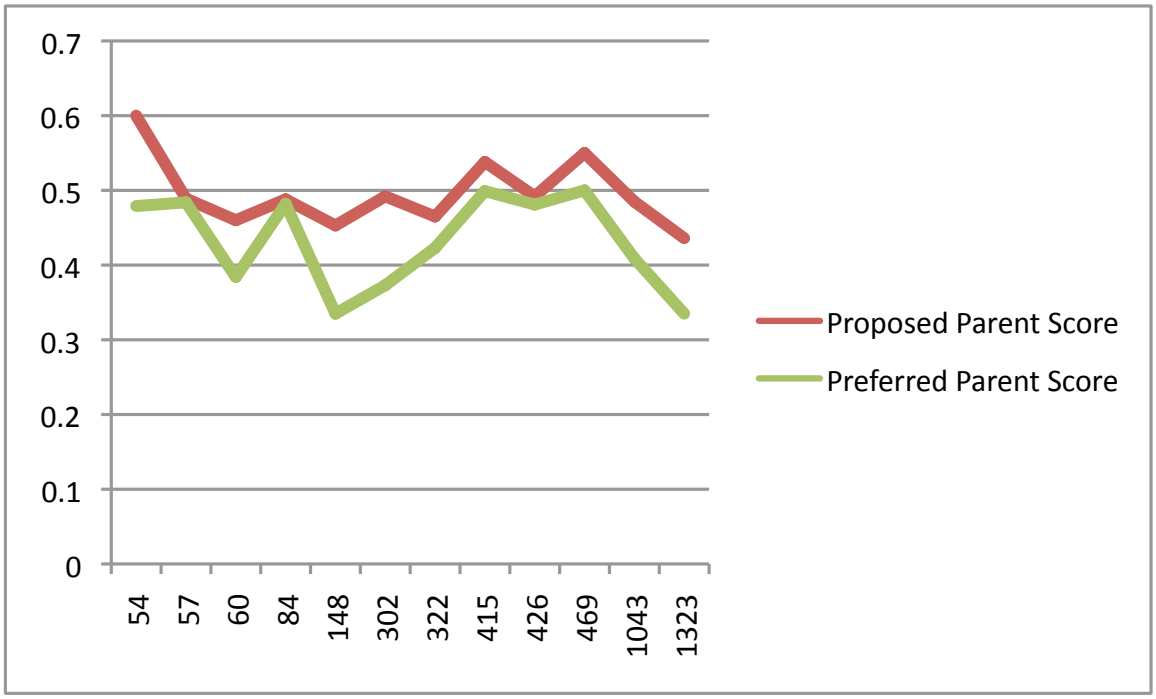


FIG. 8.2. Experiment 3: Proposed Parent And Preferred Parent Scores.

addressed in the twelfth experiment.

## 8.4 Experiment 4

Experiment 4 introduced two large improvements to the learner, both an entirely new addition to the algorithm and a replacement for an existing phase. In order to address polysemous words, which had previously been ignored, we introduced a statistical clusterer using the EM algorithm. Clustering on relations found in candidate frames, we intended to show that commonality between defined property / filler pairs would suggest senses for the target word. Additionally, we also replaced the brute force proposed parent locater (the system responsible for finding the highest similarity concept in the ontology, as compared to the candidate, and suggesting it as the proposed parent) with a tree-crawling algorithm designed to both make more intelligent decisions and substantially reduce processing time

| Word | # Sen | # Senses | Sense Definitions |
|---|---|---|---|
| version | 328 | 2 | • an account of something<br>• a variant of software |
| datum | 338 | 1 | • any level surface, or line used in measuring elevations |
| coronary | 394 | 2 | • pertaining to arteries<br>• a heart attack |
| vent | 419 | 2 | • an opening in a wall for releasing air<br>• to speak out all related thoughts overwhelmingly |
| buoy | 961 | 1 | • a distinctive shaped and marked float |
| nail | 1034 | 3 | • a metal rod used to secure objects together<br>• to attach one object to another using a nail<br>• to hit someone or something |
| license | 1164 | 3 | • formal permission from a government to do something<br>• a certificate, tag, plate, etc. as proof of permission<br>• legal right to use a patent owned by another |
| retake | 1387 | 2 | • to capture a location again<br>• filming or photographing again |
| dub | 1935 | 2 | • to invest with a name, character, dignity or title<br>• to furnish a film or tape with a new soundtrack |

Table 8.9. Polysemous words in Experiment 4.

(see Chapter 7: Experimental Setup and Evaluation Processes). Finally, we introduced a new corpus, containing 15,416 total sentences, with a new set of 9 (some polysemous) words: *buoy*, *coronary*, *datum*, *dub*, *license*, *nail*, *retake*, *vent*, and *version*. Table 8.9 shows each of the target words, the number of sentences in the corpus containing the word, the number of senses anticipated for the word, and a brief definition of each anticipated sense:

In order to objectively judge the quality of the EM-based sense clusterer, we implemented the commonly used bag-of-words (Lewis 1998) algorithm, often used for sense clustering. Bag-of-words, on its own, cannot suggest any more than a number of clusters, and to which cluster a word instance belongs; the EM-based sense clusterer we developed is capable of those features as well, but rather than using simple word counting, more detailed semantic information is used (in the form of property / filler pairs, see 7.3.3). Figure 8.3 reveals that both implementations produce relatively similar results when compared to

the desired sense count (in Table 8.9 above). The custom EM-based clusterer produces, on average, 2.4 more senses than desired, while the standard bag-of-words produces 2.7 more senses:

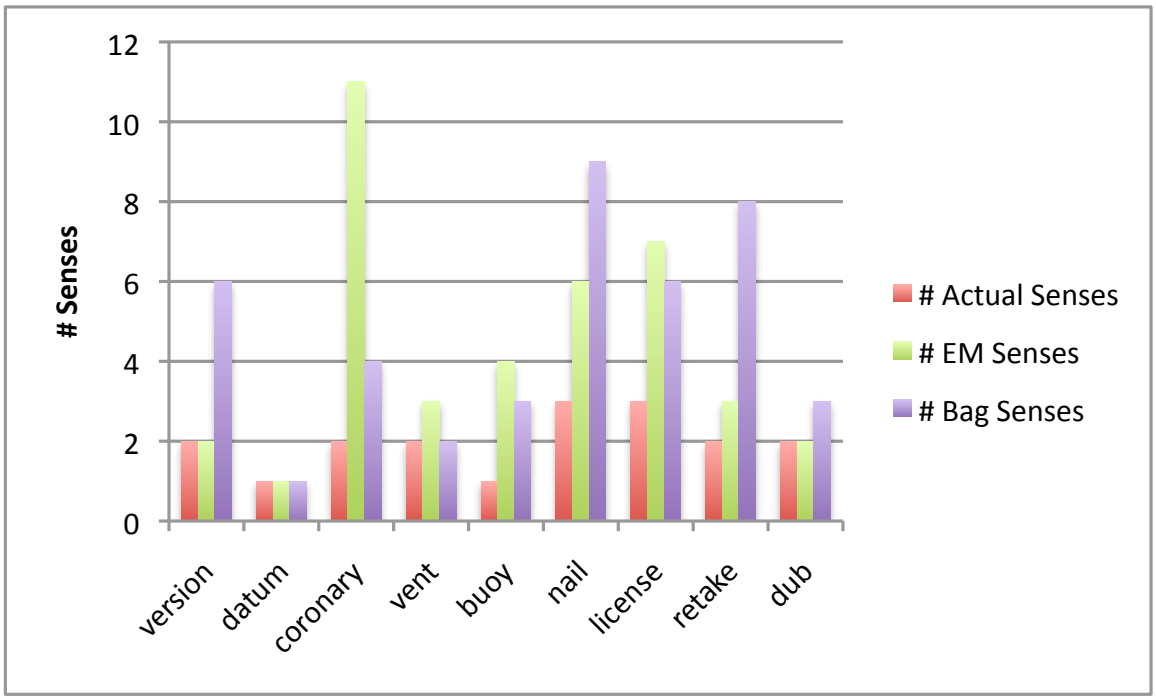


FIG. 8.3. Comparison of sense clusterers to actual (desired) senses.

Evaluation of the EM-based clusterer in light of the entire learner was also performed: taking each cluster that was produced, we manually inspected the property / filler pairs and proposed parent, mapping the candidate to one of the desired senses. Table 8.10 shows these results, along with a basic precision and recall calculation. Precision is calculated as the ”the number of senses suggested that were correct, over the number of senses that were suggested”. Recall is calculated as ”the number of senses suggested that were correct, over the number of desired senses”. Algorithm 1 formalizes these definitions:

Unsurprisingly, the recall score is substantially stronger than the precision score: given enough input, eventually the learner will encounter each of the desired senses, and as the

```
Let Gwi = # of golden senses (desired senses)
          for word i
Let Pwi = # of proposed senses for word i
Let Mwi = # of proposed senses for word i
          that match one of the senses in Gwi

PRECISIONi = Mwi / Pwi
RECALLi = Mwi / Gwi
```

**Algorithm 1:** Definition of precision and recall for EM-based clusterer.

| **Word** | **# Sen** | **Gwi** | **Pwi** | **Mwi** | **Precision** | **Recall** |
|---|---|---|---|---|---|---|
| version | 328 | 2 | 2 | 2 | 1.0 0 | 1.00 |
| datum | 338 | 1 | 1 | 1 | 1.0 0 | 1.00 |
| coronary | 394 | 2 | 11 | 2 | 0.18 | 1.00 |
| vent | 419 | 2 | 3 | 0 | 0.00 | 0.00 |
| buoy | 961 | 1 | 4 | 0 | 0.00 | 0.00 |
| nail | 1034 | 3 | 6 | 3 | 0.50 | 1.00 |
| license | 1164 | 3 | 7 | 1 | 0.14 | 0.33 |
| retake | 1387 | 2 | 3 | 0 | 0.0 0 | 0.00 |
| dub | 1935 | 2 | 2 | 2 | 1.00 | 1.00 |
| **AVG** | 730.3 | 2 | 4.3 | 1.2 | 0.42 | 0.59 |

# Sen = The number of input sentences for the target word.
Gwi = # of golden senses for the word.
Pwi = # of proposed senses (by EM).
Mwi = # of matching senses (Gwi and Pwi).

Table 8.10. EM-based sense clusterer results.

EM-based clusterer uses the semantic analysis as input, these senses will show up as clusters. Although the scores reported are encouraging, more importantly, they are highly indicative of problems at various stages of the learner: poor input or poor semantic analysis (as a result of lack of coverage, logic, or input handling) results in 59% coverage, with a 58% rate of false positives that the final stages of the learner cannot hope to counter. These problems were addressed in the subsequent experiments, as an improved version of OntoSem was deployed for the fifth experiment.

## 8.5 Experiments 5 - 8

The final set of target words was selected during the fifth experiment, and would be used for the remainder of the experiments. Ten target words were selected, and a corpus was constructed of at least five hundred sentences per target word, with the corpus size totaling 15,188 texts. The words selected were chosen primarily due to their polysemous nature, but also for their domain interdependence: *dragster*, *truck*, *freeway*, *skid*, *curb*, *chicken*, *jackknife*, *bottleneck*, *rubberneck*, and *hitchhike* each relate to the automotive domain in at least one word sense, but were selected from a domain-independent corpus. Further, the corpus built from the second half of the words was chosen with a required dependence on the first five words. In other words, each word in the second half (e.g., *chicken*) of the corpus must be paired with a word from the first half (e.g., *dragster*) in a text, in order for the text to be included. An example of the query issued to the search engine, Example 8.1, shows how texts were selected for the later (dependent) target words:

```
chicken AND (dragster OR truck OR freeway OR skid OR curb)
```

**Example 8.1:** A sample query for dependent word corpus construction for Experiment 5.

As a result of the restrictions placed on the search engine, and the filtering processes used during preprocessing, not all of the words resulted in a minimum of five hundred sentences contributed to the corpus. Table 8.11 shows the list of words, the number of sentences containing that word that were found in the final corpus, the number of target senses to be learned, the definitions of each target sense, and the preferred (existing) parent in the ontology to map the target sense to. As can be seen, most of the lower sense-count words resulted in a smaller contribution to the final corpus:

One change that was introduced in the fifth experiment that would later be reversed, was an auto-inheritance method that was inserted into the tree-crawling algorithm: as the candidate frame was pushed down the tree by the algorithm (in search of an existing concept to be rooted under), the candidate automatically inherited all values from the root of the sub-tree that the algorithm was currently visiting, in much the same way that manual acquisition would occur. This introduced a problem in that results were over-inflated due to the overwhelming volume of inherited property / filler pairs that were a guaranteed match. Essentially, this served only to skew the numbers, as the property / filler pairs that were locally defined in the candidate and in an existing frame still weighed in, and had the final say in the evaluation. The results reported in Table 8.12 show each target sense, along with the similarity score for the preferred (existing) parent.

Interestingly, with only one exception, the independent target words (the words whose contribution to the corpus was not dependent on another word) scored higher (often substantially) to their preferred parent than the words whose contribution to the corpus was dependent on another word being in the text (see Figure 8.4). This is likely due to the increased size of the corpus for the independent words, leading to broader coverage of the property / filler pairs. It should be noted however, that the similarity metric does penalize erroneous data, suggesting that the increased corpus for each of the independent words was adding more correct data than not.

| Word | # Sen | Sense Definitions | Preferred Parent |
|---|---|---|---|
| rubberneck | 242 | • to look about or stare with great curiousity | •VOLUNTARY-VISUAL-EVENT |
| bottleneck | 332 | • a narrow entrance or passage-way | •SPACE |
| jackknife | 338 | • a large pocketknife | •KNIFE |
| | | • to have the cab and trailer of a truck form a V during an accident | •ACCIDENT |
| hitchhike | 398 | • to travel by soliciting rides from passing vehicles | •TRAVEL-EVENT |
| dragster | 914 | • an automobile designed and built for drag racing | •AUTOMOBILE |
| | | • a person who races such a vehicle | •SPORTS-ROLE |
| chicken | 1118 | • a domestic fowl | •CHICKEN |
| | | • a contest in which two cars approach each other at high speeds | •SPORTS-COMPETITION |
| curb | 1674 | • a rim of concrete along a street | •ROAD-SYSTEM-ARTIFACT |
| | | • to restrain or check | •RESTRAIN |
| skid | 1678 | • to slip or slide sideways | •SLIDE |
| | | • a low mobile platform for moving goods | •WHEELED-VEHICLE |
| freeway | 1997 | • an express highway with no intersections | •HIGHWAY |
| truck | 6445 | • any of a variety of vehicle for carrying goods | •AUTOMOBILE |
| | | • also called a hand truck, a barrow-like device to move packages | •WHEELBARROW |
| | | • to transport by truck | •TRANSFER-OBJECT |

Table 8.11. Target words and corpus size introduced in Experiment 5.

| Word | # Sen | Preferred Parent | Preferred Similarity |
|---|---|---|---|
| rubberneck | 242 | VOLUNTARY-VISUAL-EVENT | 86.47 |
| bottleneck | 332 | SPACE | 85.84 |
| jackknife | 338 | KNIFE | 90.17 |
| jackknife | 338 | ACCIDENT | 97.50 |
| hitchhike | 398 | TRAVEL-EVENT | 80.28 |
| dragster | 914 | AUTOMOBILE | 94.28 |
| dragster | 914 | SPORTS-ROLE | 99.20 |
| chicken | 1118 | CHICKEN | 85.97 |
| chicken | 1118 | SPORTS-COMPETITION | 70.53 |
| curb | 1674 | ROAD-SYSTEM-ARTIFACT | 97.98 |
| curb | 1674 | RESTRAIN | 97.93 |
| skid | 1678 | SLIDE | 97.87 |
| skid | 1678 | WHEELED-VEHICLE | 94.93 |
| freeway | 1997 | HIGHWAY | 99.39 |
| truck | 6445 | AUTOMOBILE | 99.29 |
| truck | 6445 | WHEELBARROW | 98.25 |
| truck | 6445 | TRANSFER-OBJECT | 99.22 |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Preferred Similarity = Candidate similarity to the preferred parent.

Table 8.12. Skewed similarity results for Experiment 5.

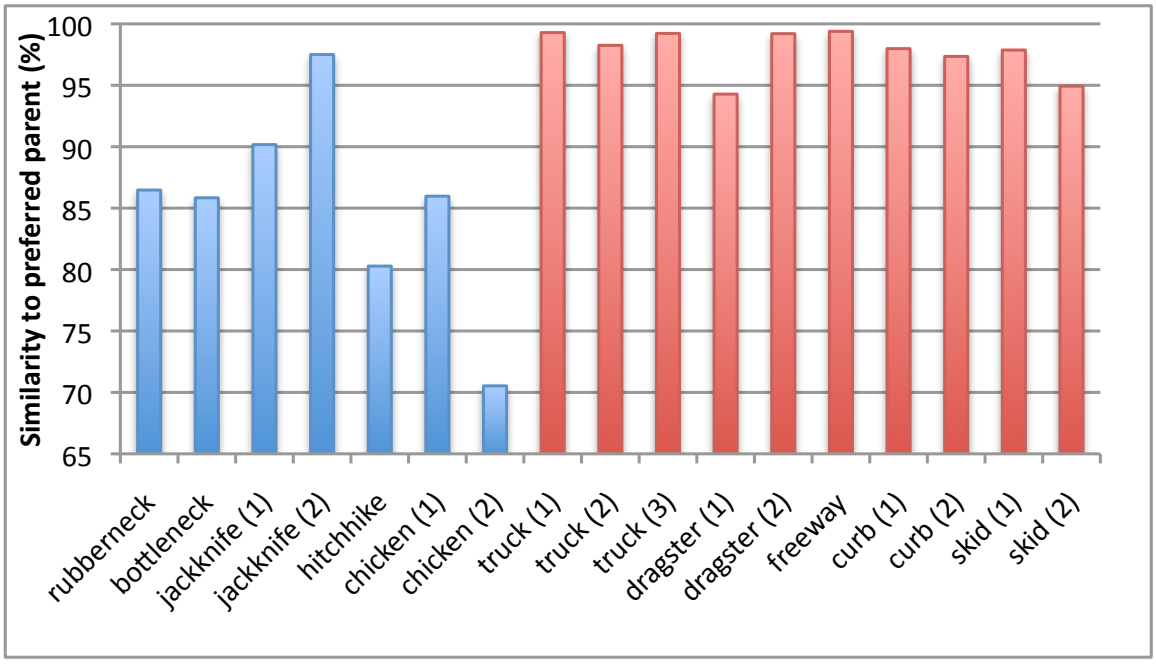


FIG. 8.4. Dependent vs. independent target word senses and their preferred parent scores.

Although the scores reported were relatively encouraging, forgiving the skew introduced by the guaranteed matches of the inherited values, in most cases the preferred parent was not the proposed parent. In other words, although the candidate scored a high similarity match with the preferred parent, another existing frame located by the tree-crawling algorithm was selected as the proposed parent due to a higher similarity score. Table 8.13 shows each target sense, their preferred parent, and the actual parent proposed by the learner:

The highlighted fields in Table 8.13 give a subjective indication as to which target senses had proposed parents that were conceptually similar (or in the case of ”jackknife”, the same) as the preferred parent. In many of these cases, the proposed parent and preferred parent are in close ontological proximity, such as the senses of ”truck” (CARRY is an immediate child of TRANSFER-OBJECT, as is SPORT-UTILITY-VEHICLE to AUTOMOBILE). It is therefore worth noting that, in addition to scoring well against the preferred parent, the proposed parent was often only a few property / filler pairs different, which indicates

| Word | Preferred Parent | Proposed Parent |
|---|---|---|
| rubberneck | VOLUNTARY-VISUAL-EVENT | MENTAL-EVENT |
| bottleneck | SPACE | CAPITAL-CITY |
| jackknife | **KNIFE** | **KNIFE** |
| jackknife | **ACCIDENT** | **ACCIDENT** and JUMP |
| hitchhike | TRAVEL-EVENT | ERR |
| dragster | AUTOMOBILE | HIGHWAY |
| dragster | **SPORTS-ROLE** | **BUS-DRIVER** |
| chicken | CHICKEN | HUMAN |
| chicken | SPORTS-COMPETITION | ACCUSE |
| curb | ROAD-SYSTEM-ARTIFACT | PAINT-BRUSH |
| curb | RESTRAIN | VISUAL-EVENT |
| skid | SLIDE | FIND |
| skid | WHEELED-VEHICLE | BASE-OF-OBJECT |
| freeway | **HIGHWAY** | **ALLEY** |
| truck | **AUTOMOBILE** | **SPORT-UTILITY-VEHICLE** |
| truck | WHEELBARROW | CAMPER |
| truck | **TRANSFER-OBJECT** | **CARRY** |

Table 8.13. Proposed parents for Experiment 5.

generally positive results.

In the sixth experiment, we used the same corpus and target words as in Experiment 5, but moved to a new version of OntoSem, which included better semantic analysis and expanded static knowledge resources. We reprocessed the corpus, creating a new set of TMRs for the learner to explore. Unfortunately, the new set of TMRs introduced more unexpected input to the learner (in the form of debugging information that was misinterpreted as knowledge), which would be addressed in the following experiment. Additionally, we implemented a penalty to the similarity metric that reduced the impact of comparing inherited property / filler pairs, to try to soften the skewed results presented in the fifth experiment. This had the result of reducing the similarity numbers reported substantially, while still being representative of all of the knowledge extracted by the learner. Table 8.14 shows each word sense, its preferred parent, and the similarity score to the candidate:

| Word | # Sen | Preferred Parent | Preferred Similarity |
|---|---|---|---|
| rubberneck | 242 | VOLUNTARY-VISUAL-EVENT | 85.54 |
| bottleneck | 332 | SPACE | 66.81 |
| jackknife | 338 | KNIFE | 88.35 |
| jackknife | 338 | ACCIDENT | 49.41 |
| hitchhike | 398 | TRAVEL-EVENT | 85.57 |
| dragster | 914 | AUTOMOBILE | 41.90 |
| dragster | 914 | SPORTS-ROLE | 57.11 |
| chicken | 1118 | CHICKEN | 56.79 |
| chicken | 1118 | SPORTS-COMPETITION | 57.84 |
| curb | 1674 | ROAD-SYSTEM-ARTIFACT | 46.20 |
| curb | 1674 | RESTRAIN | 84.49 |
| skid | 1678 | SLIDE | 86.83 |
| skid | 1678 | WHEELED-VEHICLE | 43.82 |
| freeway | 1997 | HIGHWAY | 42.50 |
| truck | 6445 | AUTOMOBILE | 41.89 |
| truck | 6445 | WHEELBARROW | 42.18 |
| truck | 6445 | TRANSFER-OBJECT | 85.98 |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Preferred Similarity = Candidate similarity to the preferred parent.

Table 8.14. Experiment 6 similarity scores.

As can be seen, the scores, in absolute terms, have all been reduced, however the values now represented are more indicative of the similarity of the learned knowledge. Although inherited values are still brought to bear, they are not overwhelming the scores. This, however, introduced a new problem: the added junk data from the new TMRs, combined with the reduced impact of the inherited property / filler pairs caused the tree-crawling algorithm to take some unusual paths, often getting stuck in undesired subtrees. Table 8.15 compares the proposed parents for each target word sense from Experiment 5 (which did comparatively well in locating a suitable parent for the candidate) with those for Experiment 6:

The highlighted fields in Table 8.15 suggest proposed parents which were subjectively worse (farther from the preferred parent) than those found in Experiment 5. In some cases, the proposed parent generalized, backing out of an incorrect subtree (as in the case of ROAD-SYSTEM-ARTIFACT, becoming OBJECT rather than PAINT-BRUSH). Unfortunately, most cases resulted in the candidate being placed with a completely incorrect proposed parent (as in the case of AUTOMOBILE changing from SPORT-UTILITY-VEHICLE to MENTAL-EVENT).

In the seventh experiment, we again maintained the same input corpus and target words, but added a series of filters to the TMR extraction phase of the learner, to remove the unwanted metadata from the TMRs that was introduced in Experiment 6. This didn't have nearly the desired impact, only barely modifying the score to the preferred parents, and only changing one proposed parent. Figure 8.5 shows a comparison of similarity scores to the preferred parent for each word sense from Experiments 6 and 7, and Table 8.16 shows the proposed parents for both experiments, highlighting the only change:

The final set of small changes to the learner occurred in Experiment 8: more filtering of undesired metadata from the TMRs was introduced, along with a few branches of the ontological subtree being intentionally blacklisted from the tree-crawling algorithm (these

| Word | Preferred Parent | EXP5 Prop. Parent | EXP6 Prop. Parent |
|---|---|---|---|
| rubberneck | VOLUNTARY-VISUAL-EVENT | MENTAL-EVENT | MENTAL-EVENT |
| bottleneck | SPACE | CAPITAL-CITY | OBJECT |
| jackknife | KNIFE | KNIFE | KNIFE |
| jackknife | ACCIDENT | ACCIDENT and JUMP | **KNIFE** |
| hitchhike | TRAVEL-EVENT | ERR | MENTAL-EVENT |
| dragster | AUTOMOBILE | HIGHWAY | **ERR** |
| dragster | SPORTS-ROLE | BUS-DRIVER | **OBJECT** |
| chicken | CHICKEN | HUMAN | **TEST-RESULT-AS-ENTITY** |
| chicken | SPORTS-COMPETITION | ACCUSE | **TEST-RESULT-AS-ENTITY** |
| curb | ROAD-SYSTEM-ARTIFACT | PAINT-BRUSH | OBJECT |
| curb | RESTRAIN | VISUAL-EVENT | **ERR** |
| skid | SLIDE | FIND | **MENTAL-EVENT** |
| skid | WHEELED-VEHICLE | BASE-OF-OBJECT | **ANALYZE** |
| freeway | HIGHWAY | ALLEY | OBJECT |
| truck | AUTOMOBILE | SPORT-UTILITY-VEHICLE | **MENTAL-EVENT** |
| truck | WHEELBARROW | CAMPER | **MENTAL-EVENT** |
| truck | TRANSFER-OBJECT | CARRY | **MENTAL-EVENT** |

EXP5 Prop. Parent = The proposed parent for the target word in experiment 5.
EXP6 Prop. Parent = The proposed parent for the target word in experiment 6.

Table 8.15. Comparison of proposed parents for Experiments 5 and 6.

| Word | Preferred Parent | EXP6 Prop. Parent | EXP7 Prop. Parent |
|---|---|---|---|
| rubberneck | VOLUNTARY-VISUAL-EVENT | MENTAL-EVENT | MENTAL-EVENT |
| bottleneck | SPACE | OBJECT | OBJECT |
| jackknife | KNIFE | KNIFE | KNIFE |
| jackknife | ACCIDENT | KNIFE | KNIFE |
| hitchhike | TRAVEL-EVENT | MENTAL-EVENT | MENTAL-EVENT |
| dragster | AUTOMOBILE | ERR | ERR |
| dragster | SPORTS-ROLE | OBJECT | OBJECT |
| chicken | CHICKEN | TEST-RESULT-AS-ENTITY | TEST-RESULT-AS-ENTITY |
| chicken | SPORTS-COMPETITION | TEST-RESULT-AS-ENTITY | TEST-RESULT-AS-ENTITY |
| curb | ROAD-SYSTEM-ARTIFACT | OBJECT | OBJECT |
| curb | RESTRAIN | ERR | **ACTIVE-COGNITIVE-EVENT** |
| skid | SLIDE | MENTAL-EVENT | MENTAL-EVENT |
| skid | WHEELED-VEHICLE | ANALYZE | ANALYZE |
| freeway | HIGHWAY | OBJECT | OBJECT |
| truck | AUTOMOBILE | MENTAL-EVENT | MENTAL-EVENT |
| truck | WHEELBARROW | MENTAL-EVENT | MENTAL-EVENT |
| truck | TRANSFER-OBJECT | MENTAL-EVENT | MENTAL-EVENT |

EXP6 Prop. Parent = The proposed parent for the target word in experiment 6.
EXP7 Prop. Parent = The proposed parent for the target word in experiment 7.

Table 8.16. Only one change in proposed parents between Experiments 6 and 7.

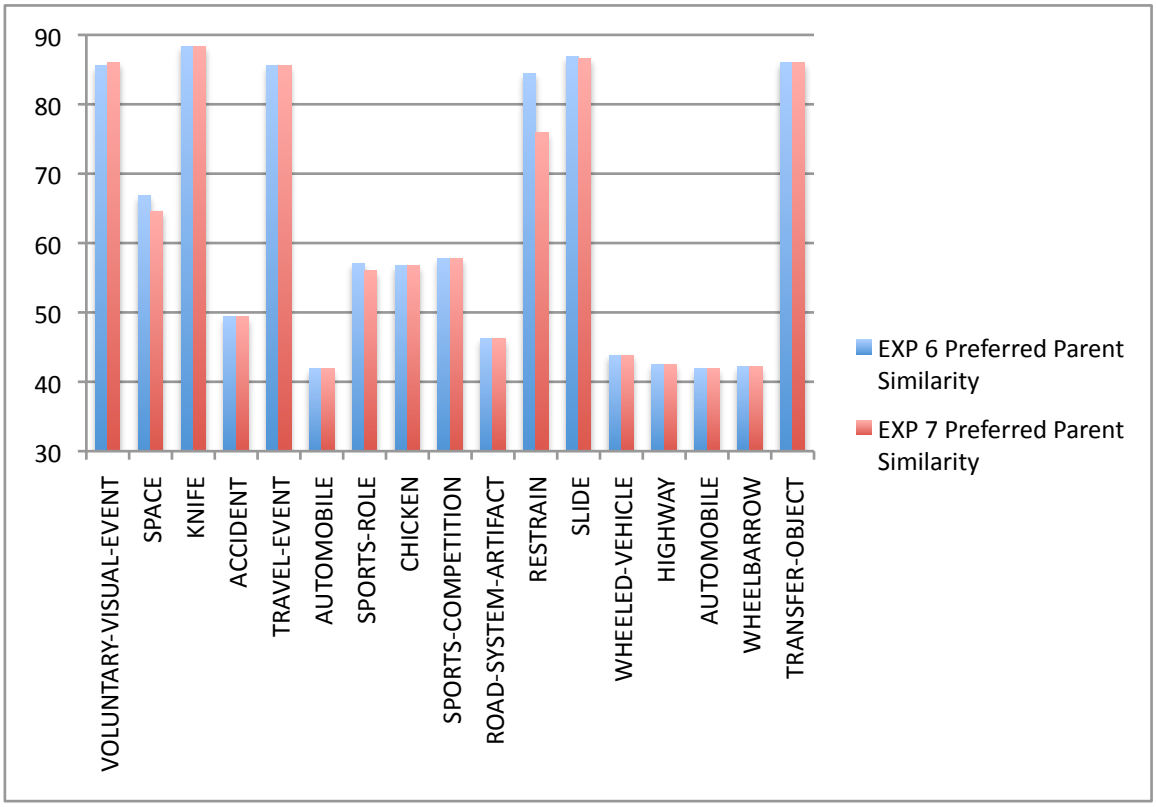


FIG. 8.5. Comparison of similarity scores to preferred parents for Experiments 6 & 7.

included debugging and metadata-style branches, that do not represent static knowledge). Once again, the changes to the results were minimal, with the similarity scores to the preferred parents shifting slightly (Figure 8.6 compares Experiments 6, 7 and 8) and only a few of the proposed parents changing, as highlighted in Table 8.17:

The minor changes introduced throughout Experiments 5, 6, 7 and 8, along with the corpus and target words, set the stage for major changes to the learner and evaluation pipeline for the following three experiments. It became clear that a few fundamental changes to the tree-crawling algorithm, similarity metric, and corpus construction would be needed in order to substantially improve the output of the learner.

| Word | Preferred Parent | EXP7 Prop. Parent | EXP8 Prop. Parent |
|---|---|---|---|
| rubberneck | VOLUNTARY-VISUAL-EVENT | MENTAL-EVENT | MENTAL-EVENT |
| bottleneck | SPACE | OBJECT | OBJECT |
| jackknife | KNIFE | KNIFE | KNIFE |
| jackknife | ACCIDENT | KNIFE | KNIFE |
| hitchhike | TRAVEL-EVENT | MENTAL-EVENT | MENTAL-EVENT |
| dragster | AUTOMOBILE | ERR | **QUANTIFY** |
| dragster | SPORTS-ROLE | OBJECT | OBJECT |
| chicken | CHICKEN | TEST-RESULT-AS-ENTITY | TEST-RESULT-AS-ENTITY |
| chicken | SPORTS-COMPETITION | TEST-RESULT-AS-ENTITY | TEST-RESULT-AS-ENTITY |
| curb | ROAD-SYSTEM-ARTIFACT | OBJECT | OBJECT |
| curb | RESTRAIN | ACTIVE-COGNITIVE-EVENT | **DIFFERENTIATE** |
| skid | SLIDE | MENTAL-EVENT | MENTAL-EVENT |
| skid | WHEELED-VEHICLE | ANALYZE | ANALYZE |
| freeway | HIGHWAY | OBJECT | OBJECT |
| truck | AUTOMOBILE | MENTAL-EVENT | MENTAL-EVENT |
| truck | WHEELBARROW | MENTAL-EVENT | MENTAL-EVENT |
| truck | TRANSFER-OBJECT | MENTAL-EVENT | MENTAL-EVENT |

EXP7 Prop. Parent = The proposed parent for the target word in experiment 7.
EXP8 Prop. Parent = The proposed parent for the target word in experiment 8.

Table 8.17. A pair of changes in proposed parents between Experiments 7 and 8.

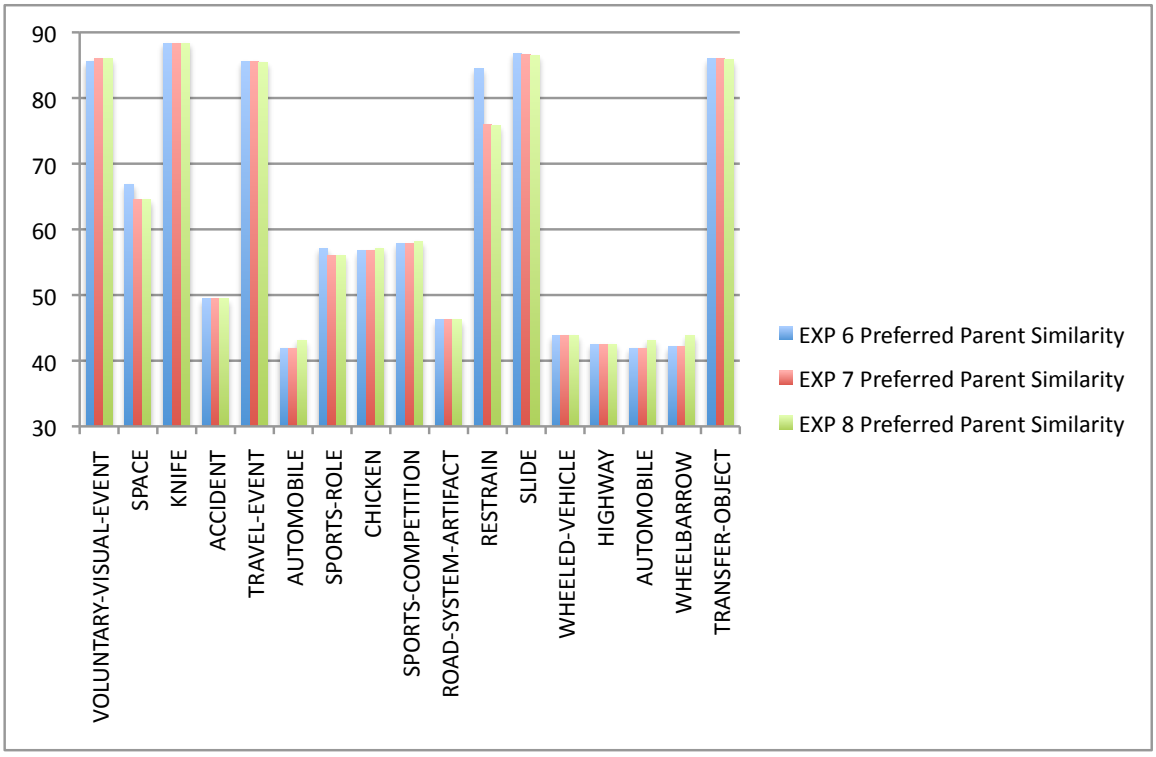


FIG. 8.6. Comparison of similarity scores to preferred parents for Experiments 6 - 8.

## 8.6 Experiments 9 - 11

Beginning with the ninth experiment, the custom similarity metric was changed to no longer consider inherited values in the candidate frame at all (previously, they were fully considered, and then in the sixth experiment, their influence was reduced). This decision was made as a result of a detailed inspection of the TMRs that were produced as part of the preprocessing phase. An abundance of property / filler pairs led to the conclusion that the extracted contents of a candidate frame (the knowledge gleaned directly from the TMRs) was sufficient enough to stand on its own, and should be considered absolute in regards to how the candidate's similarity score to an existing frame was calculated. As a result, the values from Experiment 9 on are once again reduced, but now reflect exactly how much of the existing frame's knowledge (including its inherited property / filler pairs) was found in

the candidate. These numbers are much more telling, providing a more stable grounds for evaluation. Figure 8.7 shows the similarity scores to the preferred parents for Experiment 9:

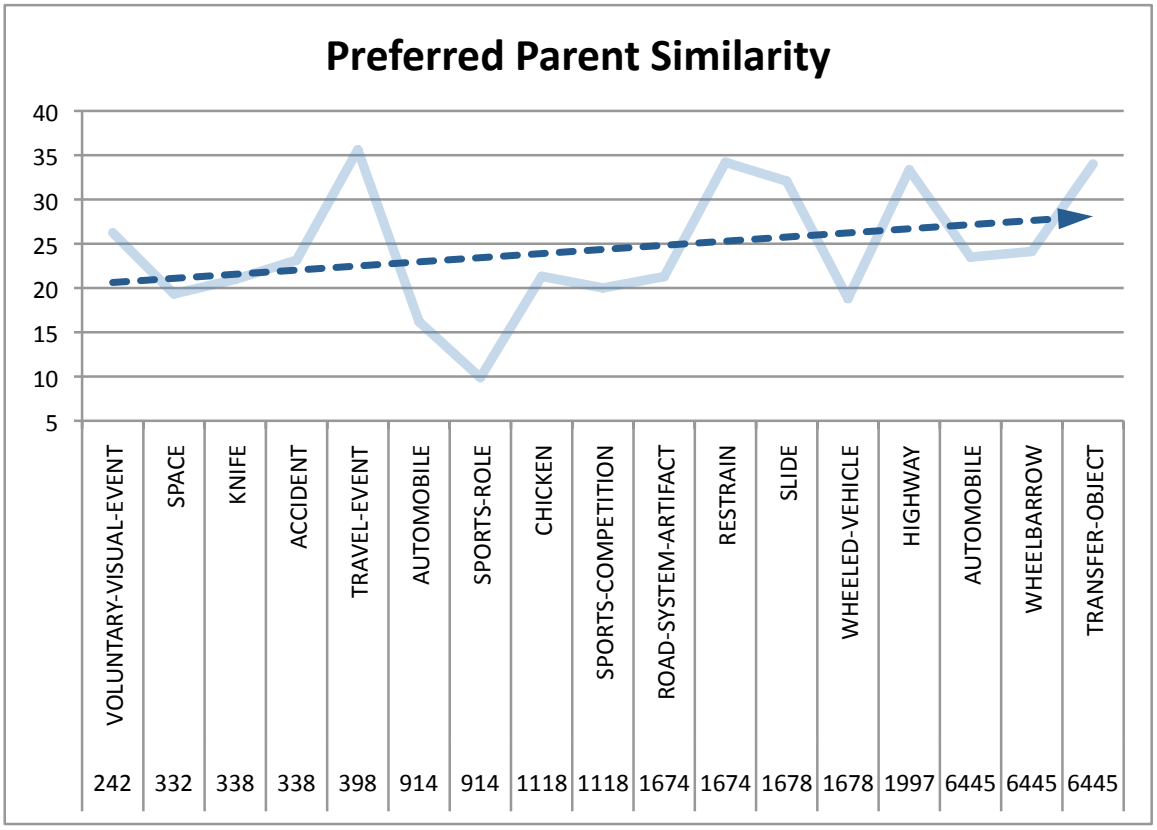


FIG. 8.7. Preferred parent similarity scores for Experiment 9.

The chart also reflects an increasing score trend: as the number of sentences made available to the learner for a target word increased, so too did the similarity score. Essentially, more raw input resulted in broader coverage.

In addition to the change in the similarity metric, the tree-crawling algorithm was also modified heavily to allow for multiple starting points (in an attempt to avoid local maxima) as well as a look-forward algorithm that prevented the tree-crawler from prematurely terminating (see Chapter 7: Experimental Setup and Evaluation Processes). Unfortunately,

| Word | # Sen | Preferred Parent | Proposed Parent |
|---|---|---|---|
| rubberneck | 242 | VOLUNTARY-VISUAL-EVENT | ENERGY-EVENT |
| bottleneck | 332 | SPACE | GEOPOLITICAL-ENTITY |
| jackknife | 338 | KNIFE | SOCIAL-OBJECT |
| jackknife | 338 | ACCIDENT | SOCIAL-OBJECT |
| hitchhike | 398 | TRAVEL-EVENT | OBJECT |
| dragster | 914 | AUTOMOBILE | BUILDING |
| dragster | 914 | SPORTS-ROLE | PHYSICAL-EVENT |
| chicken | 1118 | CHICKEN | WHEEL |
| chicken | 1118 | SPORTS-COMPETITION | WHEEL |
| curb | 1674 | ROAD-SYSTEM-ARTIFACT | OPEN |
| curb | 1674 | RESTRAIN | OPEN |
| skid | 1678 | SLIDE | SOCIAL-OBJECT |
| skid | 1678 | WHEELED-VEHICLE | SOCIAL-OBJECT |
| freeway | 1997 | HIGHWAY | ANIMATE |
| truck | 6445 | AUTOMOBILE | SOCIAL-OBJECT |
| truck | 6445 | WHEELBARROW | SOCIAL-OBJECT |
| truck | 6445 | TRANSFER-OBJECT | SOCIAL-OBJECT |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Proposed Parent = The proposed parent for the word sense.

Table 8.18. Proposed parents for Experiment 9.

although the similarity scores to the preferred parents were encouraging, the proposed parents selected by the tree-crawling algorithm left much to be desired. Table 8.18 shows the proposed parents for each target sense, in many cases only one word sense was suggested by the learner, and thus was mapped to each target sense:

Subjectively, none of the proposed parents were considered a ”good” match, so the modifications to the tree-crawling algorithm, although theoretically sound, would need to be readdressed.

Experiment 10 focused on reducing noise in the input corpus by implementing two filters at the raw text level to eliminate texts that had a low probability of being an English sentence, were deemed too simple, or were deemed too complex (using a count of syntac-

tic parse possibilities as a metric for complexity). The result was a corpus substantially reduced in size, down from 15,188 to 4,768 sentences. Figure 8.8 shows the number of sentences each target word contributed to the new corpus, compared with the number of sentences for each target word in Experiment 9. Not unexpectedly, the words with high sentence counts in the previous experiments still contributed the most sentences to the slimmer corpus:

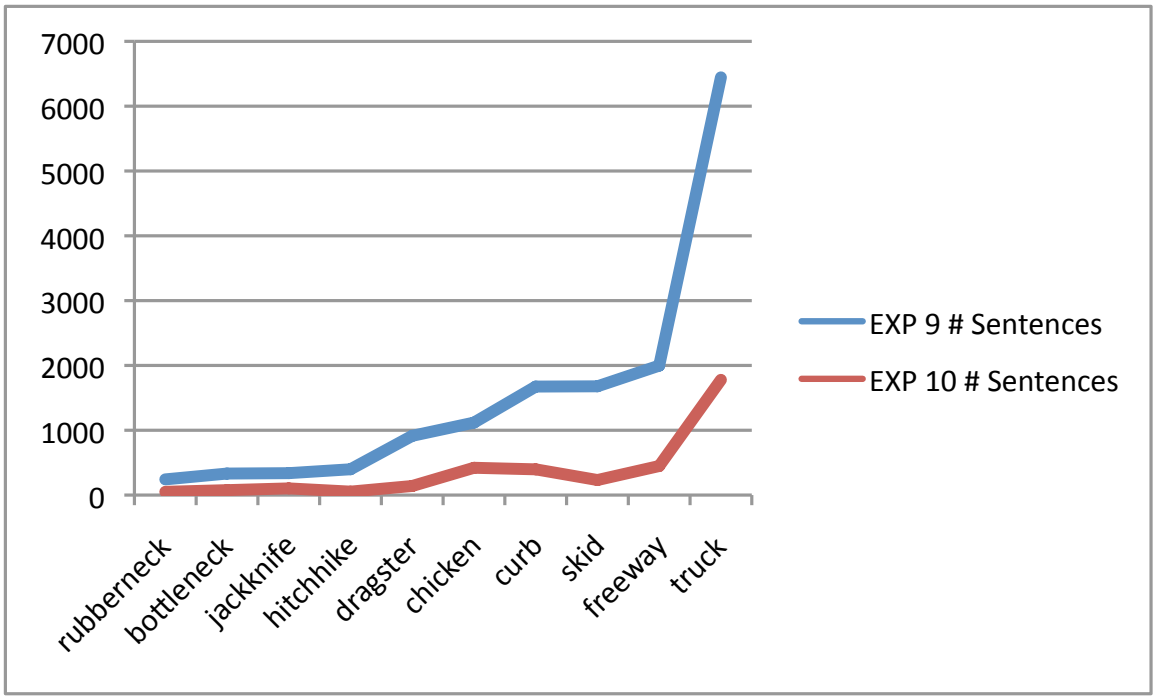


FIG. 8.8. Contributed sentences to Experiments 9 and 10 per target word.

Although the corpus was now less than one third of its original size, the contributions made by the TMRs to the learner were still significant, indicating that the filtering mostly removed redundant, erroneous or noisy texts, leaving the contentful sentences for the learner. Experiment 9 averaged 24.35% similarity between candidate frames and preferred parents, while Experiment 10 averaged 20.14%, a loss of only 4.21% similarity using a corpus 68.61% smaller. Further, the trend of increasing similarity given a large input set

continued; Figure 8.9 shows the preferred parent similarities of Experiments 9 and 10, emphasizing the increasing trend given a larger set of input texts per target word:

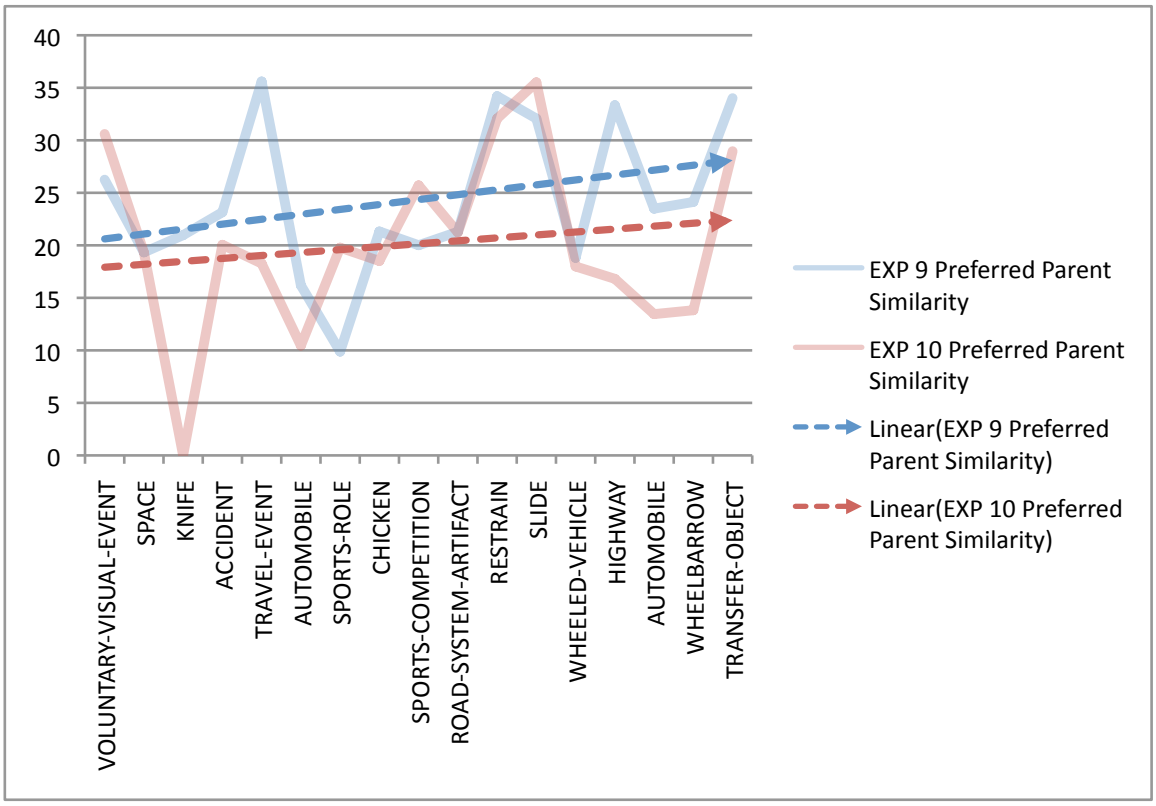


FIG. 8.9. Preferred parent similarity scores for Experiments 9 and 10.

Once again, however, the tree-crawling algorithm resulted in a set of undesirable results, similar to those presented in Experiment 9. The algorithm did recurse further into the ontological structure than in the previous experiment (the ninth experiment had 9 proposed parents that were no more than 3 levels deep in the ontology, whereas the tenth experiment had only 6), but still produced a set of subjectively wrong proposed parents. Table 8.19 shows the proposed parents for Experiment 10:

In order to address the poor proposed parents found in Experiments 9 and 10, the tree-crawling algorithm was once again modified by reducing the rigidity of its decisions

| Word | # Sen | Preferred Parent | Proposed Parent |
|---|---|---|---|
| rubberneck | 50 | VOLUNTARY-VISUAL-EVENT | PHYSICAL-EVENT |
| hitchhike | 73 | TRAVEL-EVENT | FILL |
| bottleneck | 76 | SPACE | GEOPOLITICAL-ENTITY |
| jackknife | 103 | KNIFE | DEVICE-EVENT |
| jackknife | 103 | ACCIDENT | DEVICE-EVENT |
| dragster | 140 | AUTOMOBILE | SOCIAL-OBJECT |
| dragster | 140 | SPORTS-ROLE | SOCIAL-OBJECT |
| skid | 232 | SLIDE | SOCIAL-OBJECT |
| skid | 232 | WHEELED-VEHICLE | SOCIAL-OBJECT |
| curb | 397 | ROAD-SYSTEM-ARTIFACT | CLOSE |
| curb | 397 | RESTRAIN | CLOSE |
| chicken | 421 | CHICKEN | GEOPOLITICAL-ENTITY |
| chicken | 421 | SPORTS-COMPETITION | GEOPOLITICAL-ENTITY |
| freeway | 448 | HIGHWAY | SOCIAL-OBJECT |
| truck | 1778 | AUTOMOBILE | BURY |
| truck | 1778 | WHEELBARROW | BURY |
| truck | 1778 | TRANSFER-OBJECT | BURY |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Proposed Parent = The proposed parent for the word sense.

Table 8.19. Proposed parents for Experiment 10.

with the intention of allowing the system to more easily avoid local maxima. Additionally, the comparison metric was expanded to also include inverse properties in the comparison, which it had previously ignored. Although the inverse properties found in the TMRs (and thus included in the candidate concept) would no longer be penalized (unless they were erroneous), the candidate itself would now be penalized for not having the vast array of inverses available to an existing concept. Over all, the results from the eleventh experiment were more positive than the previous two, with the preferred parent similarity scores being almost universally better than those in Experiment 10, and an improvement over those found in Experiment 9 as well. Figure 8.10 shows the results of all three experiments:

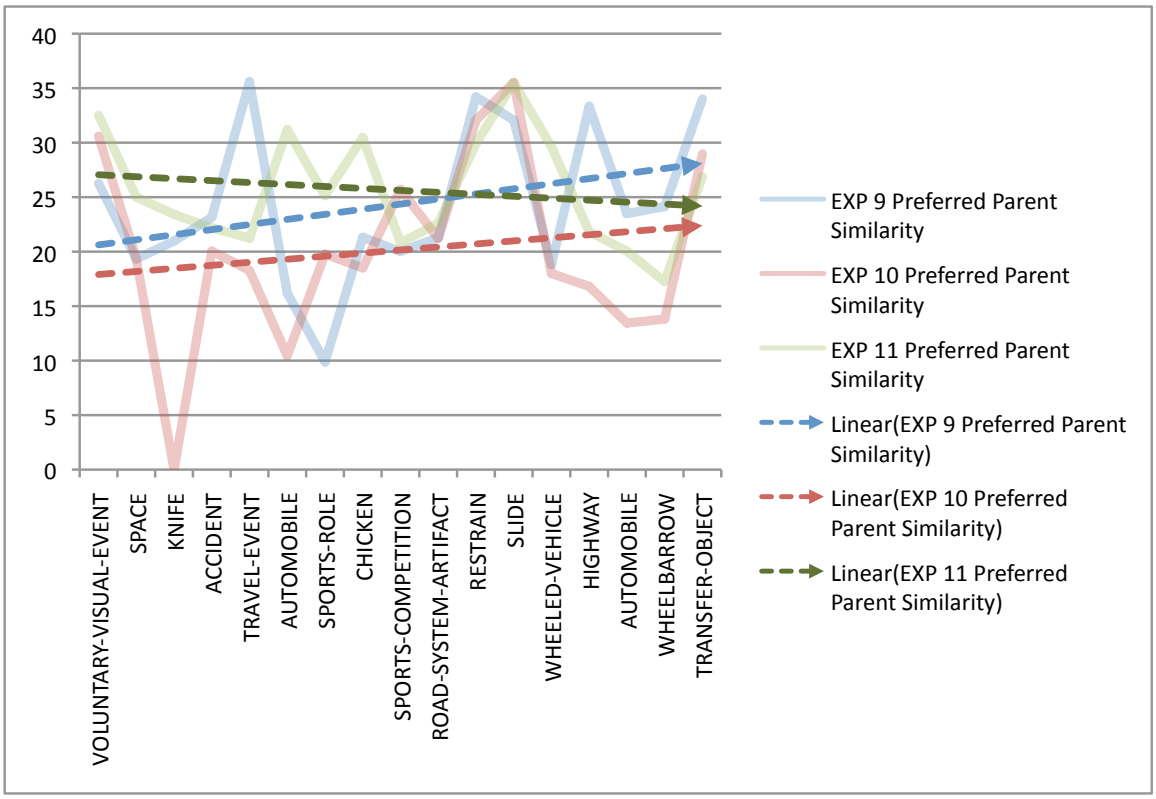


FIG. 8.10. Preferred parent similarity scores for Experiments 9, 10, and 11.

The preferred parent similarity scores of Experiment 11 trended downward (slightly) as the corpus size increased, initially beating out most of the early scores of Experiment 9, but eventually falling low, resulting in a very slight average decline. This is likely a result of the inclusion of inverse properties into the similarity metric: as the corpus size increases, so too will the number of relations (including inverses) included in a target word's set of TMR frames. The noise caused by a large volume of inverse relations would begin to penalize the similarity score more than help. Overall, however, the similarity scores in Experiment 11 were almost universally better than those found in Experiment 10, suggesting the inclusion of inverse relations into the similarity metric was a positive change.

Once again, the set of proposed parents still left much to be desired, although one target word sense was correctly mapped (KNIFE). However, the modifications to the similarity metric and tree-crawling algorithm served to once more reduce the number of shallow proposed parents (concepts in the ontology no more than three levels deep). The count of shallow proposed parents was reduced from 6, in Experiment 10, to 3 in Experiment 11. Table 8.20 shows all of the proposed parents for the eleventh experiment:

The results presented for Experiments 9, 10 and 11 showed a great deal of promise when comparing the contents of the target concept to its preferred parent, but could be vastly improved. On the other hand, the tree-crawling algorithm tended to fixate on local maxima, or deviate down an entirely incorrect path. The final experiment, presented in the next section, was designed not to maximize or improve results, but to isolate each major component of the learner, manually correct the results, and investigate the changes to the results that occurred. Effectively, Experiment 12 is an evaluation of the process, not of the results.

| Word | # Sen | Preferred Parent | Proposed Parent |
|---|---|---|---|
| rubberneck | 50 | VOLUNTARY-VISUAL-EVENT | RAFFLE |
| hitchhike | 73 | TRAVEL-EVENT | RAFFLE |
| bottleneck | 76 | SPACE | SOCIAL-OBJECT |
| jackknife | 103 | KNIFE | KNIFE |
| jackknife | 103 | ACCIDENT | KNIFE |
| dragster | 140 | AUTOMOBILE | SOCIAL-OBJECT |
| dragster | 140 | SPORTS-ROLE | SOCIAL-OBJECT |
| skid | 232 | SLIDE | SOCIAL-ENVIRONMENT |
| skid | 232 | WHEELED-VEHICLE | SOCIAL-ENVIRONMENT |
| curb | 397 | ROAD-SYSTEM-ARTIFACT | ENTERTAIN-EVENT |
| curb | 397 | RESTRAIN | ENTERTAIN-EVENT |
| chicken | 421 | CHICKEN | ORGANIZATION |
| chicken | 421 | SPORTS-COMPETITION | ORGANIZATION |
| freeway | 448 | HIGHWAY | CREATIVE-WORK |
| truck | 1778 | AUTOMOBILE | CREATIVE-WORK |
| truck | 1778 | WHEELBARROW | CREATIVE-WORK |
| truck | 1778 | TRANSFER-OBJECT | CREATIVE-WORK |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Proposed Parent = The proposed parent for the word sense.

Table 8.20. Proposed parents for Experiment 11.

## 8.7 Experiment 12

Experiment 12, as detailed in Chapter: Experimental Setup and Evaluation Processes aimed to isolate the major components of the learner, and identify how much influence each had over the final results by substituting manually entered values in their place. As such, Experiment 12 was divided into four sub-experiments: Baseline, the fully automatic learner (nearly identical to the learner used in Experiment 11); AS-IS, the fully automatic learner provided with a hand-selected small set of input sentences; Edited, the learner with the hand-selected sentences and hand-annotated TMRs; and Lexicalized, the learner with the hand-selected sentences, hand-annotated TMRs, and full lexical coverage (excepting the target word sense).

In order to more carefully evaluate the impact of each phase, the custom similarity metric was modified to report precision, recall and f-measure, rather than just recall, as it had previously done. The calculations used for each of these scores is described fully in 7.4.2. Further, golden ontological concepts were specifically acquired (manually) for each target word, and will be used in place of the preferred parents in the similarity metric, providing a truer evaluation of the quality of the learned senses.

### 8.7.1 Baseline

The setup for the baseline of Experiment 12 was taken directly from the previous experiment. Using the same 4768 input sentences, as well as the same target word senses, the learner processed the corpus normally, producing a set of similarity scores to the preferred parents, as well as a set of proposed parents (derived from the tree-crawling algorithm). Figure 8.11 shows the each similarity score for the target word senses ordered from least to most input texts:

The recall metric used is identical to the recall metric used throughout the previous

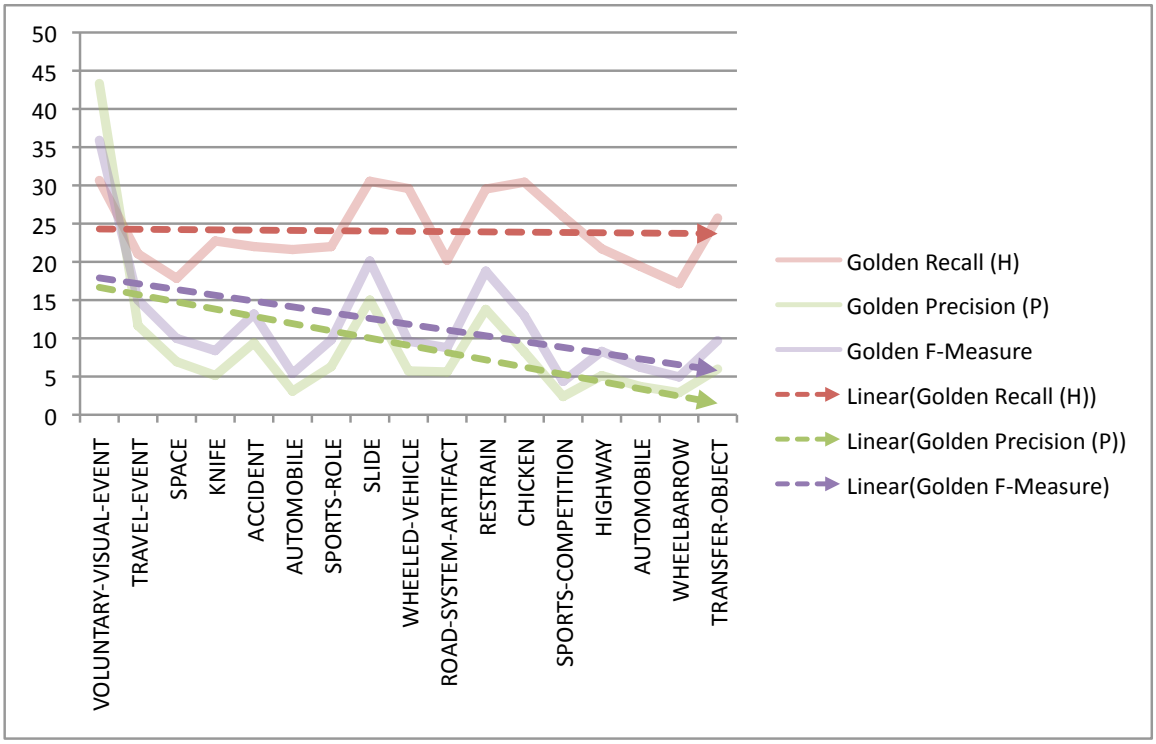


FIG. 8.11. Experiment 12 baseline similarity scores.

experiments as the main similarity calculation. Unsurprisingly, the values reported maintain just under 25%, directly on target with the results from Experiment 11. The precision metric was expected to be lower than recall, and averages just over 9%: we had originally developed our evaluation to target recall rather than precision due to the nature of the experiment. Given enough volume of data, we should be able to maximize recall, but at the expense of precision (by introducing too much noise). Further, this explains the relatively steep downward trend of the precision value (which greatly affects the f-measure): as the size of the input corpus increased, so too did the noise, introducing erroneous data into the target frames which was later penalized by the similarity metric.

The parents proposed by the tree-crawling algorithm were just as subjectively wrong as the previous experiment. Although KNIFE was properly identified, the remainder were

| Word | # Sen | Preferred Parent | Proposed Parent |
|---|---|---|---|
| rubberneck | 50 | VOLUNTARY-VISUAL-EVENT | RAFFLE |
| hitchhike | 73 | TRAVEL-EVENT | RAFFLE |
| bottleneck | 76 | SPACE | SOCIAL-OBJECT |
| jackknife | 103 | KNIFE | KNIFE |
| jackknife | 103 | ACCIDENT | KNIFE |
| dragster | 140 | AUTOMOBILE | SOCIAL-OBJECT |
| dragster | 140 | SPORTS-ROLE | SOCIAL-OBJECT |
| skid | 232 | SLIDE | LEAK-OUT |
| skid | 232 | WHEELED-VEHICLE | SOCIAL-ENVIRONMENT |
| curb | 397 | ROAD-SYSTEM-ARTIFACT | ENTERTAIN-EVENT |
| curb | 397 | RESTRAIN | ENTERTAIN-EVENT |
| chicken | 421 | CHICKEN | ORGANIZATION |
| chicken | 421 | SPORTS-COMPETITION | ORGANIZATION |
| freeway | 448 | HIGHWAY | CREATIVE-WORK |
| truck | 1778 | AUTOMOBILE | CREATIVE-WORK |
| truck | 1778 | WHEELBARROW | CREATIVE-WORK |
| truck | 1778 | TRANSFER-OBJECT | CREATIVE-WORK |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Proposed Parent = The proposed parent for the word sense.

Table 8.21. Proposed parents for Experiment 12's Baseline phase.

found to have too much distance between the proposed parent and the preferred parent to be considered a hit. Table 8.21 shows the proposed parents for Experiment 12's Baseline:

Notice that nearly all of the proposed parents were also chosen in Experiment 11, suggesting that the few tweaks to the learner and the static knowledge between the two experimental runs had very minor impact on the final results, which was not unexpected. The vast changes that would be made in the next phase of the experiment, would however, change the list of proposed parents.

| Word | Baseline # Sentences | AS-IS # Sentences |
|---|---|---|
| rubberneck | 50 | 9 |
| hitchhike | 73 | 7 |
| bottleneck | 76 | 9 |
| jackknife | 103 | 11 |
| dragster | 140 | 10 |
| skid | 232 | 10 |
| crub | 397 | 9 |
| chicken | 421 | 8 |
| freeway | 448 | 15 |
| truck | 1778 | 18 |

Table 8.22. Corpus size comparison: Baseline (automatic) and AS-IS (manual).

### 8.7.2 AS-IS

The AS-IS phase of the experiment aimed to reduce the noise coming from the corpus by manually trimming over 98% of the corpus, hand-selecting only 92 input sentences to be used by the learner. Sentences were selected from the original corpus based on their structure and content (see 7.3.5). Table 8.22 shows the size of the input corpus for each target word from the baseline, compared to the size of the manually selected corpus:

The goal was to maintain approximately ten input texts per target word; however the last two words (*freeway* and *truck*) were intentionally inflated by selecting texts for other words that also included one of the two higher-count words. This was done to preserve the intention of the original corpus: to have domain interdependence between words in the corpus (see 8.5).

An immediate impact of the reduced corpus size was, unsurprisingly, a reduction in the similarity scores, some of which were severe. Although the results, subjectively, appear worse than ever, this was an expected outcome: the quality of the analysis and learning phases has not been improved, only the corpus size has changed. With only 2% of the texts available, recall (and consequently the f-measure) should be expected to suffer. Table 8.23

| Preferred Parent | # Sen | Golden Recall | Golden Precision | Golden f-Measure |
|---|---|---|---|---|
| TRAVEL-EVENT | 7 | 40.54 | 20.00 | 26.78 |
| CHICKEN | 8 | 18.13 | 13.99 | 15.80 |
| SPORTS-COMPETITION | 8 | 0.00 | 0.00 | 0.00 |
| VOLUNTARY-VISUAL-EVENT | 9 | 0.00 | 0.00 | 0.00 |
| SPACE | 9 | 2.91 | 1.66 | 2.12 |
| ROAD-SYSTEM-ARTIFACT | 9 | 0.00 | 0.00 | 0.00 |
| RESTRAIN | 9 | 18.64 | 17.50 | 18.05 |
| AUTOMOBILE | 10 | 10.47 | 6.66 | 8.14 |
| SPORTS-ROLE | 10 | 7.70 | 4.99 | 6.06 |
| SLIDE | 10 | 0.00 | 0.00 | 0.00 |
| WHEELED-VEHICLE | 10 | 5.48 | 3.33 | 4.14 |
| KNIFE | 11 | 0.00 | 0.00 | 0.00 |
| ACCIDENT | 11 | 0.00 | 0.00 | 0.00 |
| HIGHWAY | 15 | 5.40 | 1.99 | 2.92 |
| AUTOMOBILE | 18 | 17.13 | 4.64 | 7.30 |
| WHEELBARROW | 18 | 25.44 | 3.57 | 6.26 |
| TRANSFER-OBJECT | 18 | 15.49 | 2.14 | 3.76 |

# Sen = The number of input sentences for the target word.

Table 8.23. Similarity scores for Experiment 12's AS-IS phase.

shows the new set of similarity scores for each of the target word senses:

Immediately, it should stand out that several of the target word senses have no score at all. Investigation showed that this is due, simply, to a sufficient lack of valid knowledge. In the cases of the word senses for *jackknife* (KNIFE and ACCIDENT), no knowledge was gleaned from the input corpus at all, and thus no senses were proposed by the clusterer to the tree-crawling algorithm, resulting in no scores. The remainder of the zero scores were a result of very sparsely populated candidate frames that simply had no match to the golden frame at all. As an example, the target word *rubberneck* (VOLUNTARY-VISUAL-EVENT) resulted in only one candidate concept, which contained only one property / filler pair: DESTINATION-OF / VELOCITY, which is not only nonsensical, but also has no match at all to the manually created golden concept for *rubberneck*, resulting in a zero score. Figure

8.12 shows that the trends of precision and recall are similar (although universally lower) to those presented in the baseline phase:

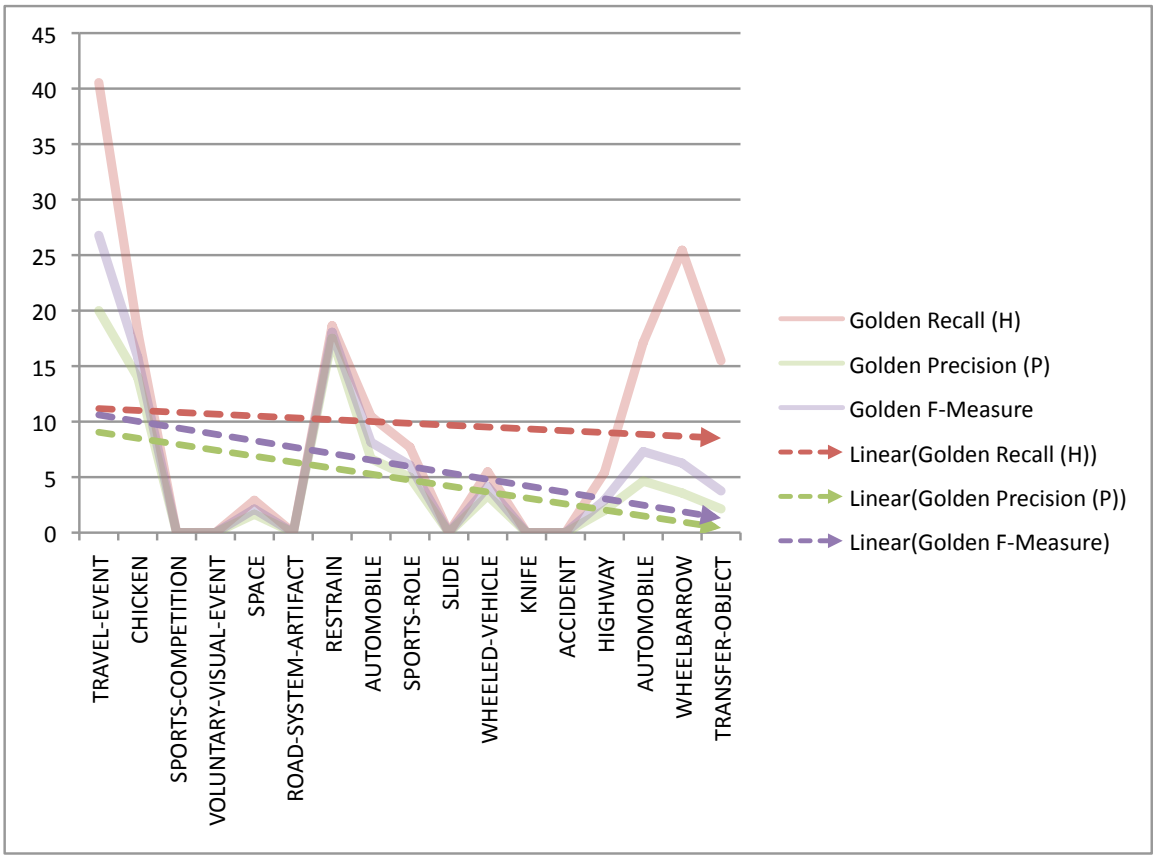


FIG. 8.12. Experiment 12 AS-IS similarity scores.

Further, the parents proposed by the tree-crawling algorithm still fell short of expectations. However, most of the proposed parents differed greatly from those proposed in the Baseline phase, indicating that the changed values as a result of the reduced corpus size unsurprisingly weighed heavily on the decisions made by the tree-crawling algorithm. Table 8.24 shows the proposed parent for the AS-IS phase of Experiment 12:

The AS-IS phase showed that in order to overcome the inherent weaknesses of the semantic analyzer and the learner itself, a minimum amount of input text is required to pro-

| Word | # Sen | Preferred Parent | Proposed Parent |
|---|---|---|---|
| hitchhike | 7 | TRAVEL-EVENT | FOLD |
| chicken | 8 | CHICKEN | PLAY-THEATRE |
| chicken | 8 | SPORTS-COMPETITION | PLAY-THEATRE |
| rubberneck | 9 | VOLUNTARY-VISUAL-EVENT | PHYSICAL-EVENT |
| bottleneck | 9 | SPACE | ENTERTAIN-EVENT |
| curb | 9 | ROAD-SYSTEM-ARTIFACT | RAFFLE |
| curb | 9 | RESTRAIN | RAFFLE |
| dragster | 10 | AUTOMOBILE | ANIMAL |
| dragster | 10 | SPORTS-ROLE | ANIMAL |
| skid | 10 | SLIDE | WAR-ACTIVITY |
| skid | 10 | WHEELED-VEHICLE | WAR-ACTIVITY |
| jackknife | 11 | KNIFE | - |
| jackknife | 11 | ACCIDENT | - |
| freeway | 15 | HIGHWAY | COLLABORATE |
| truck | 18 | AUTOMOBILE | PRIMATE |
| truck | 18 | WHEELBARROW | PRIMATE |
| truck | 18 | TRANSFER-OBJECT | PRIMATE |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Proposed Parent = The proposed parent for the word sense.

Table 8.24. Proposed parents for Experiment 12's AS-IS phase.

duce meaningful results. The downward trend shown in the baseline suggests, combined with the results presented here, that a sweet spot may exist where enough data is present to produce meaningful results without overwhelming the candidate with noise. This hypothesis was further strengthened in the next phase, where the automatically produced TMRs were manually cleaned and corrected, producing higher quality inputs for the learner.

### 8.7.3 Edited

The edited phase of the twelfth experiment maintained the manually selected corpus of 92 input sentences, but replaced the fully automatically created TMRs with a set of manually corrected and annotated TMRs to demonstrate the impact of proper semantic analyses on the results produced by the learner. Recall from the previous section that six of the target word senses resulted in a zero score due to a complete lack of correct semantic information in the TMRs. Table 8.25 shows the similarity scores for the Edited phase, which now include only four target word senses with a zero score:

In addition to an elimination of a few zero scores, many of the non-zero scores increased as a result of higher-quality input TMRs. This suggests that even with a very small input corpus, useful knowledge can still be extracted given a better semantic analysis. Intuitively, this makes sense: the learner is better to able to assimilate the meaning of a word given its context, if the context is clear (properly analyzed). If the context is ambiguous or confusing, the learner should be expected to have a harder time deciphering the meaning of an unknown word. Figure 8.13 shows the trends of the similarity scores reported for the Edited phase; once again, the recall score is relatively stable, while the precision follows a sharp decline:

Along with improvements to the similarity scores, the tree-crawling algorithm showed very minor (subjective) improvements, as a few of the proposed parents edged closer (in the ontological hierarchy) to the preferred parents. In many cases, unlike the previous phases,

| Preferred Parent | # Sen | Golden Recall | Golden Precision | Golden f-Measure |
|---|---|---|---|---|
| TRAVEL-EVENT | 7 | 24.99 | 19.99 | 22.22 |
| CHICKEN | 8 | 25.32 | 25.00 | 25.15 |
| SPORTS-COMPETITION | 8 | 0.00 | 0.00 | 0.00 |
| VOLUNTARY-VISUAL-EVENT | 9 | 40.58 | 40.00 | 40.29 |
| SPACE | 9 | 12.08 | 12.99 | 12.52 |
| ROAD-SYSTEM-ARTIFACT | 9 | 15.41 | 9.99 | 12.13 |
| RESTRAIN | 9 | 9.50 | 11.11 | 10.24 |
| AUTOMOBILE | 10 | 10.47 | 6.66 | 8.14 |
| SPORTS-ROLE | 10 | 7.70 | 4.99 | 6.06 |
| SLIDE | 10 | 0.00 | 0.00 | 0.00 |
| WHEELED-VEHICLE | 10 | 10.47 | 3.33 | 5.05 |
| KNIFE | 11 | 0.00 | 0.00 | 0.00 |
| ACCIDENT | 11 | 0.00 | 0.00 | 0.00 |
| HIGHWAY | 15 | 5.40 | 1.66 | 2.54 |
| AUTOMOBILE | 18 | 35.53 | 18.63 | 24.44 |
| WHEELBARROW | 18 | 35.83 | 18.78 | 24.65 |
| TRANSFER-OBJECT | 18 | 20.51 | 3.63 | 6.17 |

# Sen = The number of input sentences for the target word.

Table 8.25. Similarity scores for Experiment 12's Edited phase.

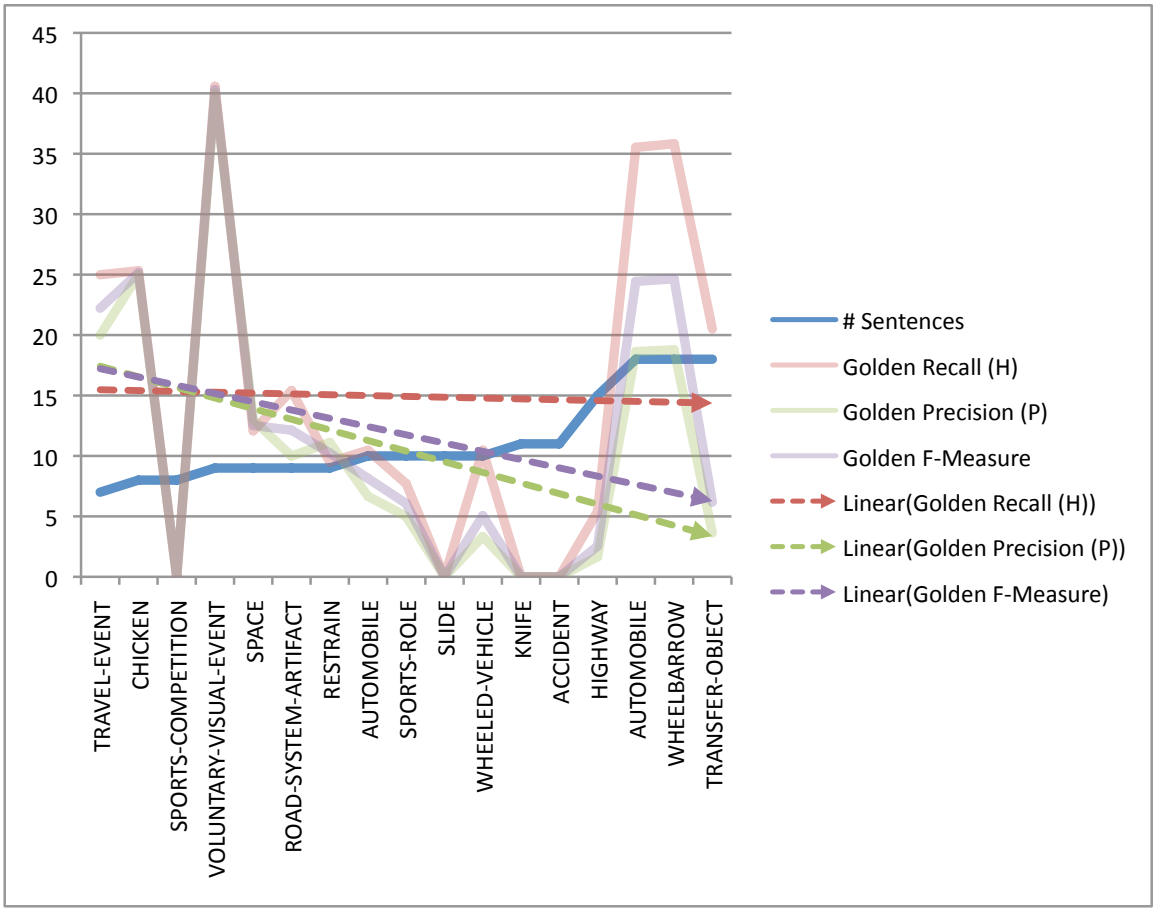


FIG. 8.13. Experiment 12 Edited similarity scores.

at least the main subtree of the ontology chosen (OBJECT or EVENT) was correct, and in a few cases a fairly accurate parent was proposed (such as ROAD-SYSTEM-ARTIFACT, for *curb*, which was mapped to ENGINEERED-ARTIFACT). Table 8.26 shows the proposed parents for the Edited phase of Experiment 12:

The Edited phase of the twelfth experiment demonstrated that even with a minimal amount of input, if the input is of high enough quality, a significant improvement in results when compared to a gold-standard can be gained. Although reducing noise in the input corpus may intuitively seem solid, the AS-IS phase showed that too much reduction results in the learner not being able to statistically overcome the inherent weaknesses in the se-

| Word | # Sen | Preferred Parent | Proposed Parent |
|---|---|---|---|
| hitchhike | 7 | TRAVEL-EVENT | CIRCUS |
| chicken | 8 | CHICKEN | PRIMATE |
| chicken | 8 | SPORTS-COMPETITION | PRIMATE |
| rubberneck | 9 | VOLUNTARY-VISUAL-EVENT | SEND |
| bottleneck | 9 | SPACE | PHYSICAL-EVENT |
| curb | 9 | ROAD-SYSTEM-ARTIFACT | ENGINEERED-ARTIFACT |
| curb | 9 | RESTRAIN | WAR-ACTIVIY |
| dragster | 10 | AUTOMOBILE | ANIMAL |
| dragster | 10 | SPORTS-ROLE | ANIMAL |
| skid | 10 | SLIDE | ENTERTAIN-EVENT |
| skid | 10 | WHEELED-VEHICLE | MODALITY |
| jackknife | 11 | KNIFE | - |
| jackknife | 11 | ACCIDENT | - |
| freeway | 15 | HIGHWAY | ENTERTAIN-EVENT |
| truck | 18 | AUTOMOBILE | GUN-BARREL |
| truck | 18 | WHEELBARROW | GUN-BARREL |
| truck | 18 | TRANSFER-OBJECT | GUN-BARREL |

# Sen = The number of input sentences for the target word.
Preferred Parent = The preferred parent for the word-sense.
Proposed Parent = The proposed parent for the word sense.

Table 8.26. Proposed parents for Experiment 12's Edited phase.

mantic analyzer; this was corroborated in the Edited phase with the manual elimination of those inherent weaknesses, leaving only a small, manually annotated corpus from which the learner was able to extract an impressive amount of data.

### 8.7.4 Lexicalized

The Edited phase manually corrected the input TMRs to the best quality given the state of the static knowledge at the time. This reflected the best possible set of TMRs OntoSem could have produced. The lexicalized stage took this one step further, by manually acquiring all lexical senses that were present in the input corpus but not already covered by the static knowledge (excluding the target word senses). The TMRs were then further corrected, using the new knowledge to a near-golden state, representing the very best quality TMR that could be produced given all the required static knowledge. The purpose of this phase was to show an improvement in the learner's results when provided with the highest quality input corpus that could be expected from OntoSem. This phase continued to use the reduced corpus (of 92 sentences) and the same target word senses. As a result, one of the four remaining zero-score word senses was eliminated, leaving only three word senses that did not have enough information to score against the gold-standard. Table 8.27 shows the precision and recall scores for the Lexicalized phase:

When compared with the results in the Edited phase, there is a notable increase in recall, while the precision values show a high degree of fluctuation. The resultant f-measures are a near split, with six of the values being slightly less than the previous phase, and seven being equal to or more than the previous phase. Figure 8.14 plots the f-measure scores over each of the phases:

From Figure 8.14, we can observe the trends of each f-measure: while the AS-IS trend never tops the Baseline, the Edited trend is approximately bisected by the Baseline, implying that the difference made by the manual annotation of the TMRs approximates

| Preferred Parent | # Sen | Golden Recall | Golden Precision | Golden f-Measure |
|---|---|---|---|---|
| TRAVEL-EVENT | 7 | 40.54 | 16.00 | 22.94 |
| CHICKEN | 8 | 25.32 | 25.00 | 25.15 |
| SPORTS-COMPETITION | 8 | 0.00 | 0.00 | 0.00 |
| VOLUNTARY-VISUAL-EVENT | 9 | 40.58 | 40.00 | 40.29 |
| SPACE | 9 | 27.92 | 27.50 | 27.70 |
| ROAD-SYSTEM-ARTIFACT | 9 | 15.82 | 11.99 | 13.65 |
| RESTRAIN | 9 | 9.50 | 11.11 | 10.24 |
| AUTOMOBILE | 10 | 10.46 | 3.99 | 5.78 |
| SPORTS-ROLE | 10 | 7.69 | 2.99 | 4.31 |
| SLIDE | 10 | 25.54 | 16.66 | 20.17 |
| WHEELED-VEHICLE | 10 | 20.48 | 13.33 | 16.15 |
| KNIFE | 11 | 0.00 | 0.00 | 0.00 |
| ACCIDENT | 11 | 0.00 | 0.00 | 0.00 |
| HIGHWAY | 15 | 5.40 | 1.24 | 2.03 |
| AUTOMOBILE | 18 | 35.51 | 14.64 | 20.73 |
| WHEELBARROW | 18 | 35.82 | 14.76 | 20.90 |
| TRANSFER-OBJECT | 18 | 20.49 | 2.85 | 5.01 |

# Sen = The number of input sentences for the target word.

Table 8.27. Similarity scores for Experiment 12's Lexicalized phase.

the difference an additional 4,700 input sentences makes. The Lexicalized trend, importantly, tops the Baseline entirely: bluntly, this means that 92 input texts that were manually annotated, including acquisition of required static knowledge, proved better than 4,768 automatically annotated texts. Intuitively, this demonstrates the hypothesis that the learner performs better given higher quality inputs.

## 8.8 Lessons Learned

As each set of experiments was concluded, certain lessons could be drawn from the results: patterns emerged in the output that suggested a trend that should be explored, certain processes were shown to have a less than desirable effect and would need to be reviewed, and some hypotheses were shown to hold, providing a springboard for future research.

The first three experiments taught two valuable lessons: first, an inappropriate metric should not be used, regardless of convenience. The use of OntoSearch was a poor choice due to a contradiction in its fundamental design (OntoSearch assumes two existing concepts in the ontology, of which our candidates were not). Replacing OntoSearch with a custom metric proved to be a good move, allowing us a finer grain of control over the evaluation for all the following experiments. Second, we should not ignore polysemy in words; commonly, words have multiple senses, and the nature of our research would easily mix data, which should otherwise be divided, without direct action. Additionally, the effect of noise was also felt in the early experiments, however it was not properly identified until several experiments later; nonetheless, properly handling noise in both the raw input corpus, and from the shortcomings of the semantic analyzer is a challenging task that cannot be taken lightly.

The fourth experiment showed that the unique resources at our disposal can allow

us to outperform commonly accepted methods of text-based learning: our custom word sense clusterer was capable of producing improved results over the standard bag-of-words technique, using property / filler pairs and an implementation of EM. The importance of a strong, self-contained evaluation emerged from the fourth experiment as well. By capturing the quality of the EM-based word sense clusterer in absolute terms, we were able to safely isolate that functionality, allowing it to work as a black-box from which we could expect reasonably high-quality results. Unfortunately, this lesson was not applied often enough, and should be considered a priority feature in future research.

Experiments 5, 6, 7 and 8 taught the importance of a strong and well-conceived set of target words. By definition, the quality of the corpus is tied to the target words. Selecting a set of independent target words, and a set of dependent target words (words whose corpus texts must also include one of the independent target words) allowed us to explore domain interdependence, and showed that more generic texts often produced higher quality results. Having a large, stable corpus for the remainder of the experiments also provided us with a solid basis for comparison; as our experiments are too independent from other work in the area, it can be difficult to provide grounded evaluation. However, the use of a stable corpus and set of target words provided the next best baseline for evaluative purposes.

The ninth, tenth and eleventh experiments helped to prove one of our long-standing hypotheses that increased input size would lead to increased coverage, but higher quality input could also bridge the coverage gap. By filtering a large portion of the input corpus using a pair of heuristics, we were able to gather much of the knowledge covered by the full corpus, but with less noise to reduce precision. We learned, however, that each component of the process had a hold on the component immediately following, and as we discovered in the fourth experiment, should be strongly self-evaluated.

Although the results presented in the final experiment were never subjectively amazing, and despite the poor showing of the tree-crawling algorithm, the results were highly

encouraging, and the twelfth experiment assisted in spotlighting the influences that various components of the learner's pipeline had on the final results. Unsurprisingly, more input texts leads to reduced precision, but increased recall, while any manual intervention in the TMR creation phase improves the overall output. These observations, the processes that worked, and those that failed provide a solid baseline for extending this research into new directions. In the next chapter, Future Research Directions, we'll discuss possible alternatives and enhancements to each of the learner's stages and to the process as a whole.

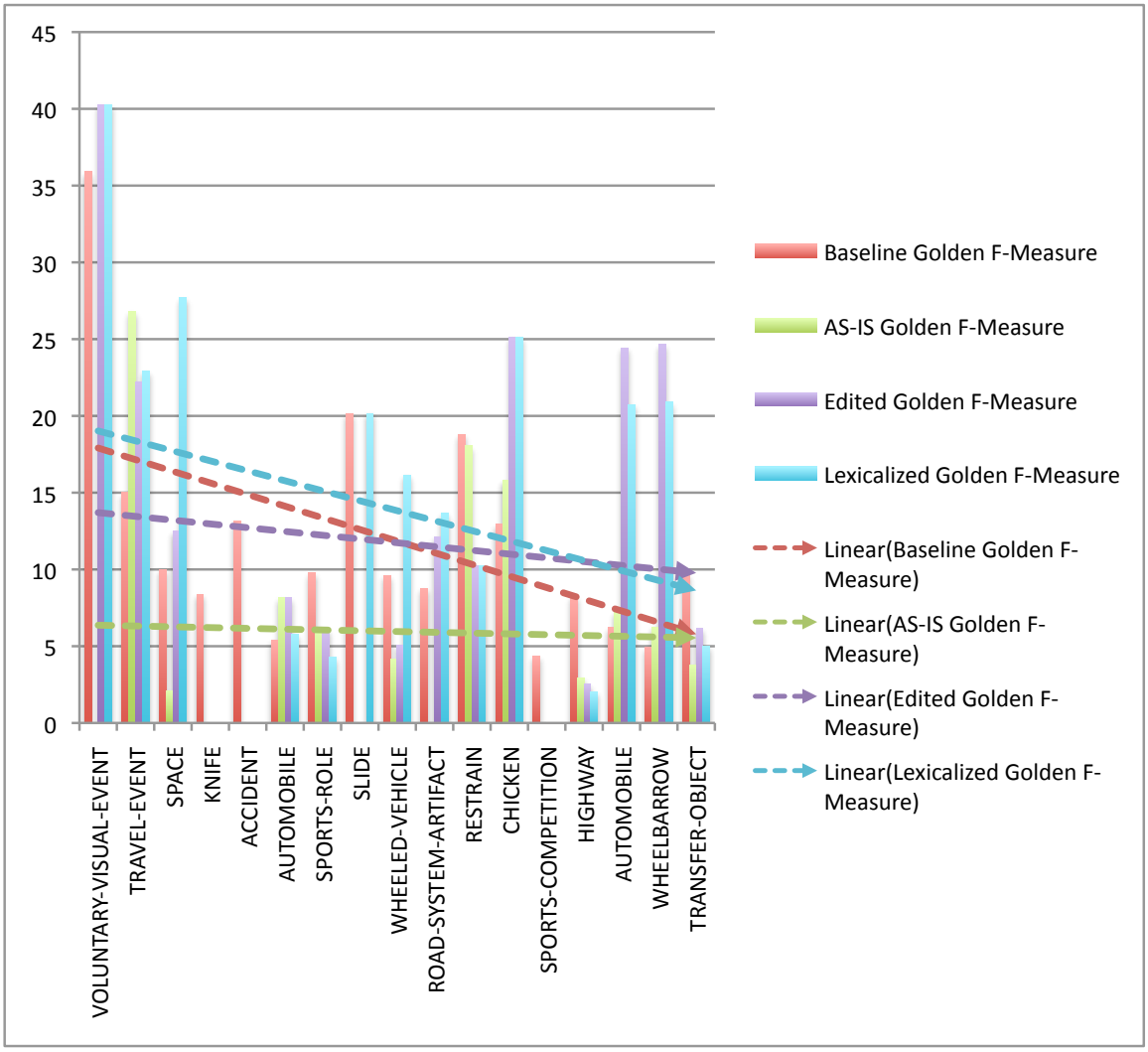


FIG. 8.14. Experiment 12's f-measure scores over all phases.

Chapter 9

# FUTURE RESEARCH DIRECTIONS

## 9.1 Introduction

The results presented in the previous chapter demonstrated a baseline of potential for a lexical and ontological learner, using a semantically analyzed corpus as input. In many ways, the results could be considered positive: the learner was often able to extract a portion of the correct semantic information for an unknown word sense, far better than a true random baseline. Subjectively, however, the results could be improved. The output from the learner appeared in two forms, each of which was evaluated: first, the contents of the candidate frame, and second the chosen location of the candidate frame within the ontology (the proposed parent). The latter of these outputs had, subjectively, the worst showing of the results, and is a prime candidate to be reevaluated and worked on. However, as the twelfth experiment showed, each component of the learning pipeline impacted the subsequent processes and the final results, so it is prudent to address each of these components in terms of potential improvements.

In this chapter, we will present some possibilities for future research and compare a hypothesis for each with the standing results (where applicable). We will start, in the next section, with a breakdown of each of the main existing processes (raw corpus construction, TMR corpus construction, sense clustering, the similarity metric, the tree-crawling

algorithm, and evaluation) and describe possible directions for research in those areas. In the third section, Methodological Shifts, we will describe broader possibilities for future research that break the existing mould we have created and described. Throughout this chapter, we will speculate on the effects that the proposed changes may have on the process and results as a whole, with the understanding that none of these changes have been implemented or tested on our current system (although similar experiments may have been performed by other researchers in the field).

## 9.2 Existing Processes

Each component in the learner's pipeline can be considered an independent process: there is a system for extracting raw text from the web that includes the target word, a system for converting these raw texts into a set of TMRs, and a system to cluster the property / filler pairs found in the TMR frames into word senses. In addition, there are several post-processing systems, including a custom similarity metric, and the tree-crawling algorithm, both of which are used as part of the learning pipeline, and as part of evaluation. As each of these components can be considered a black-box, with specified inputs and outputs, we can consider each independently and discuss how they can be improved. For all future work, an emphasis must be given to properly isolating and attesting these components: for example, we need to be reasonably sure that the quality of the raw texts produced from the web via our querying component will meet a determined threshold.

In the case of our custom EM-based word-sense clusterer, we did isolate the system, and independently attest its results against a well-known existing method of word-sense clustering (bag-of-words, see 8.4). Each existing component should have a similar treatment applied, and all new components (and modifications) proposed in the remainder of this chapter should also be similarly charged. The unique nature and position of this re-

search makes presenting truly meaningful evaluation of results challenging, so all extraneous complications should be removed wherever possible. If nothing else, an obvious direction of continued research would be to simply attest the existing components in isolation, providing a solid baseline of quality that should be expected from the results; no doubt, doing so would also expose many minor upgrades to the components themselves.

Aside from a general push towards attested higher-quality components to use in the pipeline, we can also propose changes and improvements (at a broader scale) to each component. Beginning with the construction of the raw text corpus, followed by the conversion of raw text to TMRs, the word-sense clustering, the similarity calculation, the tree-crawling algorithm, and evaluation of the results themselves.

### 9.2.1 Raw Text Corpus Construction

As described fully in Chapter 6: Experiment Preparations: Corpus Building and Preprocessing, the raw text corpus used as input to the learner is derived from a series of queries made to the web via common search engines (such as Google or Yahoo!). The resultant web pages are stripped of their HTML, the relevant sentences are extracted and then are filtered by a pair of processes designed to reduce web junk (by checking for a likelihood that the proposed text is an English sentence) and noise (by eliminating texts that are too syntactically simplistic or complex). Once these phases are complete, the system presents the collection of raw text to be semantically analyzed.

Most of these processes should be examined, scrutinized, and improved. In (Kilgarriff & Grefenstette 2003), using the web as a corpus for language research was investigated. The conclusion, as we've also discovered in our research, is the benefits of using a free, open, and enormous corpus come at the cost of noise and precision in searching. Although popular search engines produce reasonably good results, they are limited in some areas which can be frustrating for our line of work: often search results are derived from headings

or titles, which may be misleading in context and when used automatically without any further filtering, can lead to a poor input corpus for language learning. It is therefore essential to improve on the results that are returned from the web directly. This can be done in a number of ways, including issuing more refined searches, searching through multiple engines (to reduce the impact of the nuances of an individual one) or by more carefully filtering the results prior to full semantic analysis.

Ensuring that the text that is presented as the final corpus is the text that the search engine intended is a challenging task itself. Removing markup, such as HTML, without regard to the semantics of the tags themselves, can result in erroneous ”texts” being submitted to the analyzer. Often, web pages include menus, toolbars, or hidden search term lists, all of which result in a meaningless jumble of raw text when the markup is removed. Taking care to identify, when possible, these (as well as other) phenomena and completely removing them, rather than simply stripping markup, could result in improved performance down the line. Currently, a few heuristics are used to accomplish this task, with mild success; however, this process could be improved, and attested to perform to a satisfactory level.

Texts that are returned after HTML stripping must eventually pass through a series of filters designed to further attest that each is a valid English sentence. These filters could be further improved, including adding entirely new ones. Although the pair of filters that are currently used, subjectively, perform well at removing texts that would serve only to confuse the analyzer (and eventually the learner), they must be properly attested to verify that a suitable level of texts that pass them are of the desired quality.

Presuming the process has presented a quality set of texts without markup, the next task is to improve the extraction mechanic that pulls out only the texts containing the target word. At present, a simple string search is used, which results in many valid texts being discarded: search engines often perform stemming and return texts that match many

derivates of the input word. These matches should also be extracted, and could be done so by using a lexical preprocessor to stem all of the texts, and selecting any whose root word matches the root of the target word. To further broaden coverage, rather than discarding any sentences which do not contain the target word, sentences surrounding the target word could be selected as one text, to be processed through a reference resolution engine, providing more varied and deep context to the target word, which would improve the output of the learner. A parallel line of our research involves improving and deploying a full-scale reference resolution engine for OntoSem, which could be used by the learner in future experiments.

The foundation step of the learner's entire process is that of corpus construction. As Experiment 12 showed (see Chapter 8: Results and Discussion), a high-quality input corpus can have a dramatic effect on the final results. Therefore it is imperative, in future research, to address this process with as much attention as any other. Provably improving the final set of raw texts will demonstrate both better results, and allow the research to continue to focus on other components of the system with the knowledge that the basic input is of a high standard.

### 9.2.2 TMR Corpus Construction

Following the construction of the raw text corpus, a collection of TMRs must be created for the learner to work with. Although the direct improvement of OntoSem and its static knowledge resources is outside the scope of this research, there are several improvements that could be made by the learner in order to improve the results presented following semantic analysis.

Although the raw texts will have passed several filters of quality, such filters can only use string-matching and heuristics; following semantic analysis, a new set of filters could be devised to eliminate TMRs that either show poor, or confusing results, or simply do not

have enough useful semantic information. A filter could be constructed to remove TMRs whose frame for the target word does not contain a certain threshold of information (not enough property / filler pairs, or even not enough well-specified property / filler pairs). The threshold could demand a quantity of genuinely useful information, and could discount very generalized data, such as THEME-OF / EVENT.

Further, the process of semantic analysis has, to now, been used as a black-box. The texts have simply been presented to OntoSem, and TMRs have been returned. Without delving into the development of the semantic analyzer itself, many options and settings that already exist could be further explored to more finely tune the process to the task of language learning. OntoSem has been designed to handle unexpected input by relaxing restrictions within the text until there is a valid match: perhaps this technique could be modified when the task, specifically, is to learn unknown words.

Although the majority of the improvements that can be made during this phase of the learner are parallel research for the general semantic analyzer and its static knowledge resources, the inclusion of post-TMR filtering and a deeper investigation into all of the tools that OntoSem already exposes should result in a higher quality corpus of TMRs containing the target word.

### 9.2.3 Sense Clustering

Once a corpus of TMRs has been created, data in the form of property / filler pairs are extracted from the frames matching the target word, and are clustered into word-senses. Both the extraction phase and the clustering phase are targets for improvement. Currently, all property / filler pairs that are found in the matching TMR frames are extracted (although a few are blacklisted), and then are run through the EM-based clusterer to produce a set of proposed word-senses. A typical TMR frame does not include inverse relations explicitly. An inverse relation is simply a mirror relation pointing back from the destination frame to

the origination frame. As an example, imagine the frame EVENT-1, has the property / filler pair AGENT / HUMAN-1. The inverse relation, for this example, would be found in HUMAN-1, and would be AGENT-OF / EVENT-1. Generally, TMRs do not explicitly include these, rather they are assumed. It may be very relevant to the learner to have this information at its disposal. If the target frame was primarily used as a filler in other frames, it may appear relatively empty (and may even be filtered out for being too vague) when it actually has many links to other frames in the TMR. Dynamically populating and including inverse properties may assist in broadening coverage, but could also introduce noise. Investigating the impact of such a change should be considered for future work, and would be necessary if reference resolution is incorporated as suggested in the previous section.

After the relevant property / filler pairs have been extracted, the word-sense clustering system is used. Although the existing system has been attested, much work exists in the field of word-sense disambiguation. Investigating other algorithms and systems may yield better results. Importantly, due to the large collection of techniques available, a voting system may be appropriate: allowing the custom EM-based clusterer, bag-of-words, and a few other high-performing algorithms to all suggest word-sense clusters, with the highest scoring amongst the collection being considered the correct answer could prove to be valuable. There are not many sub-systems throughout the learner that are appropriate for heavily machine-learning inspired tactics, however, in the case of word-sense clustering, it is important to allow other techniques to weigh in.

Further, the word-sense clusterer could be trained using a combination of the voting system (assembled of many independent clusterers) and a human validation step. Providing such a system with feedback in the form of corrections would allow the clusterer to learn from its mistakes, and may provide higher quality output after the training phase was completed. With the property / filler pairs clustered into word-senses, the similarity metric, tree-crawling algorithm and evaluation processes should now be addressed.

### 9.2.4 Similarity Metric

The custom similarity metric was devised to replace OntoSearch, and uses a pair-wise comparison approach, attempting to match each filler in a property in the candidate frame to the closest filler in the property in the existing (parent) frame. Although this approach is sound, the actual comparison values calculated have room for improvement, and again, the entire process should be thoroughly attested and then black-boxed.

Currently, all properties have the same weight in the similarity metric; in reality, we might prefer that some properties have a higher degree of importance. Introducing a weighted scale to each property as meta-data to the ontology would allow us to shift similarity calculations in a desired way. Often, we would prefer that the case roles (such as AGENT, and THEME) have a higher say in the similarity calculation than less-used relations, such as HAS-MEMBER (a property used mostly for ORGANIZATIONs). The idea of creating a set of weights can be extended further: multiple sets could be created to emphasize different learning tasks, and could be manually or automatically selected (automatic selection could be based on a presumed location in the ontology, see 9.3.2).

A second modification to the calculations that may enhance accuracy in similarity reporting would be to modify the process that calculates subsumption distance in the ontology to weight its result based on the density of the local subtree. This process is used throughout the similarity metric to determine ”nearness” of fillers. However, some subtrees have had particular attention drawn to them, and have been heavily developed, whilst others are stubs or placeholders. The distance reported between a leaf and a subtree root in a heavily developed subtree may appear to be much greater than the distance reported in an underdeveloped subtree, when, ontologically, not much difference exists. Weighting the cost of one subsumption step by the similarity difference between the parent and child would help to mitigate this problem: if a parent and child are practically identical, then

moving up the tree from the latter to the former should not count as heavily as moving between two concepts that are drastically different. This, of course, presumes the existence of a similarity metric capable of determining these values, which presents, theoretically, a chicken and egg scenario.

Regardless, the custom similarity metric was shown to be a necessary replacement to OntoSearch, and should be scrutinized heavily, and improved to an acceptable level, as it provides the backbone for both the tree-crawling algorithm, and general evaluation.

### 9.2.5 Tree-crawling Algorithm

The system responsible for identifying a location for a candidate in the ontology was changed from a brute-force search using OntoSearch to a system emulating a decision tree using the custom similarity metric. The change was developed due to time-constraints introduced with the custom similarity metric, which made the brute-force option no longer feasible. Unfortunately, as the previous chapter showed, the tree-crawling algorithm performed relatively poorly, even after extensive modifications. Most commonly, the algorithm would start down an incorrect subtree, or would simply halt in a local maxima (or sometimes both). The results were discouraging, so the tree-crawling algorithm will need an extensive reboot.

The algorithm could simply be replaced with another variant, but in order to truly improve accuracy, the only real solution is to return to the brute-force system. This means resources that could be used to improve and test the tree-crawling algorithm itself might be better used to re-engineer the custom similarity metric in an effort to improve its runtime. Alternatively, throwing more hardware at the problem may provide a solution of sorts. The learner, to now, has been run on a single mid-level server system, however the brute-force task is perfectly suited to a distributed computing solution. Simply dividing the task of comparison amongst several machines, as well as improving the efficiency of the similarity

metric itself may provide enough of a speed gain to eliminate the need for clumsy searches and allow for higher accuracy when placing the candidate frame in the ontology.

Alternatively, section 9.3.3. proposes a learning methodology that eliminates the need for post-processing searches through the ontology: in the proposed method, the location of the candidate frame is known prior to any property / filler pairs being populated, so the need for either an improved tree-crawling algorithm, or an efficient brute-force search is negated entirely.

### 9.2.6 Evaluation

One of the fundamental problems this research faces in evaluation is the uniqueness of the goal and results: it is not possible to compare the output objectively to research done by others as even the format of the output is proprietary to the OntoSem system. Therefore, maintaining a solid baseline is the best strategy for self-evaluation, as seen in the final experiments presented in the previous chapter. Selecting a set of target words carefully, will allow future experiments to evaluate against each other. The corpus could vary between experiments, but if the goal (a collection of target word-senses) remains constant, then a baseline will be developed that can then be used to compare against. Although this mentality was used for the majority of the existing experiments, it is important enough to mention again here: all future experiments must also be capable of evaluating against a baseline.

Beyond this methodology, the task of evaluation itself could be modified depending on the experiment's goal. The learned words could be bootstrapped in to the static knowledge, and a new round of learning could commence using the automatically acquired knowledge to improve the second round of learned knowledge. An experiment of this nature would require, similar to the corpus in the final experiments, a set of independent and dependent target words. After learning all of the words once, the independent words could be added

to the static knowledge, and the dependent words could be re-learned. The difference in quality between the first round of learned dependent words and the second round would produce an evaluation of the process as a whole and an evaluation of the quality of the learned independent words.

Evaluation could also be performed in a completely different way from the current method: the results could be used in another application (or task) that has its own evaluation. As an example, automatic (or semi-automatic) production of TMRs is a task that would benefit from automatically learned knowledge. If the task of TMR production itself has an evaluation method, then the task could be run twice, once without the learned knowledge, and the second time with the new knowledge included. The delta of evaluation of the TMR task would be indicative of the quality of the learned knowledge. And the task is not limited to TMR production, but any knowledge-based task that intends to use the ontology and has a pre-defined method of evaluation would suffice.

Finally, a more subjective evaluation could be performed by ”judging” how much post-editing is required for a human acquirer to correct the learned knowledge, versus the effort required in creating the knowledge completely by hand. This evaluation would need to be performed with multiple acquirers (to avoid bias) and is hard to objectively label effort (it is not necessarily a time- or action-saving evaluation, but more of a subjective grading scale). Although this method is not as solid as other forms of evaluation, it does provide a tangibility to the results.

In this section, we’ve discussed possible changes and improvements to each element of the existing learner, from corpus construction, to word-sense clustering, to evaluation. However, the entire learning paradigm can also be revisited, rather than simply improving on existing segments, entire processes can be removed or reorganized. In the next section we’ll discuss a series of large-scale changes that could be implemented, as well as even broader uses than ontology learning for the learner.

## 9.3 Methodological Shifts

While tuning each component of the existing process could lead to a series of large-scale research projects in their own right, it is also worth discussing how the process as a whole could be modified. Although some of the components may see reuse, and would benefit from the changes proposed in the previous section, addressing the entire pipeline as a single process could prove fruitful. Several known practices in machine learning could be integrated into the pipeline, including evidence patterns (verifying previously learned data, or finding contradictions), as well as human validation (correcting the system's output in a way that assists the learner in performing better in subsequent tasks) should be considered. Further, changing the "direction" of the learning process, which currently extracts the property / fillers, clusters them and then finds locations for the word-senses in the ontology. The flow could be reversed, by finding locations for word-senses in the ontology, and then populating them with property / fillers. Finally, alternate uses for the learner could be investigated, including targeting higher-level semantic knowledge, such as scripts (see Chapter 3: OntoSem). In this section, we'll discuss each of these possibilities with a coarse-grained level of detail.

### 9.3.1 Evidence Patterns

Using evidence after learning to reinforce or reverse a decision is not unheard of, and has been used in language learning as far back as (Mooney 1987). Often however, as is the case in the research, the learning process is considered unidirectional: once knowledge has been learned, it is firm and can be used for future tasks, including further learning. This introduces the possibility of cascading errors: as Experiment 12 showed, each component of the learning process is intrinsically dependent on the prior components, including the existing static knowledge. A never-ending language learner, with the intent to bootstrap

itself perpetually can generate a supply of increasingly erroneous knowledge based on the use of incorrect learned knowledge at an early stage.

By introducing a degree of flexibility to the never-ending learning process, cascading errors can be avoided. Rather than considering anything learned to be absolute, a degree of confidence could be applied that could be adjusted up or down during subsequent learning phases. As more evidence is found to corroborate the learned knowledge, the confidence would rise until some threshold was met where it could be considered absolute; conversely, as contradictory evidence is discovered, the confidence would fall until some threshold was met where it was simply discarded as incorrect. Learned knowledge still in this ”limbo” could be considered by the semantic analyzer, but with less weight than ”official” knowledge. Manipulating the thresholds, the amount of evidence required, and the flexible use of the unofficial knowledge in OntoSem presents a complete set of research goals that live entirely independent from how the knowledge is actually learned.

This proposal could be taken yet further, as evidence could be used during the learning process itself: rather than selecting a corpus of a single size, and extracting all values available, the learner could instead continue to query the web seeking to corroborate or disprove the evidence it discovers for each property / filler pair, only allowing one to be added to the candidate after a threshold of positive evidence is encountered. This method plays nicely with the proposed method of reversed learning direction discussed ahead (see 9.3.3). Using a system of this nature, the quality of each candidate put forth would likely be improved; as importantly, a history, or trail of evidence, could be assembled from the learner’s choices, which would assist in evaluation and future improvements.

### 9.3.2 Human Validation

Converting the learner into a supervised learning system could prove a challenging, yet worthwhile endeavor: by allowing the results (either the final candidate, or at each

stage) to be validated manually, the system will be provided with a set of positive and negative examples that can be used in future learning exercises. The goal would still be to produce a fully automatic learning system, however the bootstrapping phase prior to life-long learning would involve manual intervention. Once the set of examples has been provided, no further manual intervention would be required.

In order for the system to fully make use of this supervised method, it must be able to recall each decision it made, and ideally attest the correctness of the end result to one or more attributes along the pipeline. This may involve further manual intervention, wherein the human validator both selects a correctness for the result, as well as rating some (or all) of the results from the individual components.

An alternative method to directly ranking the outputs of each component manually, a human acquirer could create, by hand, the output each stage should produce, allowing for a phase of automatic evaluation which would result in training samples. This effort could be used as part of the larger effort to attest and black-box each component, where evaluation of individual steps will be required regardless. Although integrating supervised learning into the system as it stands will be an enormous effort (from both an engineering and an acquisition stance), the results may speak for themselves.

### 9.3.3 Per-feature Queries

The strongest paradigm shift proposed is a move from per-word queries to per-feature queries, essentially reversing the order of the learner's process. Currently, an unknown word is used to trigger the learner, which queries the web for texts containing that word, which are used to extract a collection of property / filler pairs that are clustered and then placed as candidates in the ontology. By focusing on the potential gain of using a higher quantity of more specified queries, the flow of control can be reversed: a candidate can be selected and placed into the ontology first. From there, the ontology describes which

properties it should inherit, and each can then be filled by performing directed web searches for the candidate word, and words descriptive to the property. The results from the web would still be semantically analyzed, and the fillers extracted would be used as fillers for the individual property in the candidate frame.

A method similar to this has been investigated in (Dimitroff 2008), where fillers for existing properties in a concept were attested using the web as an encyclopedia (with a focus on attributes, rather than relations). In effect, we intend to change the goal from ”what is an XYZ?” to ”what does an XYZ look like? and what does an XYZ do?” Previously, the answers to the latter questions were found by chance, and were used to find the answer to the former. Assuming that an answer to the former already exists, we can instead direct our attention to carefully finding answers to the latter, which is a process that lends itself well to many of the previously described areas of future research, such as the integration of evidence patterns, supervised learning, and the removal of the tree-crawling algorithm.

Of note, the big concern is how to already know the location of the target word in the ontology. In a semi-automatic system, this can simply be manually provided as part of the input, and may be a good method for evaluating aspects of this new learner. However, in our proposed fully automatic system, the location of the candidate must be derived entirely without human intervention. One proposal is to use web-based dictionary style definitions to attempt to immediately locate the position in the ontology for the candidate. Texts of the nature ”XYZ is a type of ABC” can allow the learner to instantly plot the candidate into the ontology, and have a wealth of presumed properties to begin filling one at a time using the method proposed above.

A variety of resources are openly available and provide the structured data required. Recently, the use of wikipedia (www.wikipedia.org ) as a corpus has become increasingly popular (e.g. (Gleim, Mehler, & Dehmer 2006)) due to its vast breadth and natural reading text. Further, resources are also available such as mirror pages in alternate

languages, including a simplified English. Beyond wikipedia, a variety of open- and closed-collaboration dictionary efforts exist, including wiktionary (www.wiktionary.org ) and dictionary.com (www.dictionary.com ), as well as marked-up resources such as WordNet (Miller 1995). By using heuristics to target ”is-a” structures in raw text, and then verifying with a full semantic analysis, the system may be able to learn the place of a target word-sense in the ontology with no further knowledge about its properties. The prospects of such a system are fresh and exciting, and should be considered a strong direction to forge in as much work is being done in this area both by language researchers and in the ML and IR communities.

### 9.3.4 Script Learning

In addition to manipulating the components to perform better, or even the flow of control itself, the goal of the learner can be reordered to target more than just ontological concepts and their associated lexical entries. Learning ontological scripts is a natural extension of the current research that would benefit greatly from the introduction of the per-feature queries proposed in the previous section. Scripts are essentially stories used to describe an event. For example, the lexical phrase ”take a flight” implies that one is going to board an airplane to change locations. While this can simply be mapped to the appropriate ontological concept, it could also be mapped to a more complex script that implies the following and more:

1. book the flight
2. pack luggage
3. drive to the airport
4. check luggage

5. pass security

6. embark on airplane

The script also describes all of the agents involved, including the passenger, as well as other passengers, pilots, security guards, and even the ticketing system. Scripts can be highly elaborate and conveniently fill a hole in deep semantic descriptions of complex strings of events that are often labeled as a single event (with implied sub-events).

The existing learner could be modified to learn scripts in two ways: either as a sub-system to another automatic or manual system of script compilation, wherein the learner fills in the details of one individual component of the script at a time, or as the original creator of the script, wherein the learner identifies that the phrase ”take a flight” implies a series of events that must each be learned one at a time, and strung together. Currently, the system would be better utilized in the former situation: learning an element of a script is fundamentally similar to learning an ontological concept, and fits well with the per-feature query system. The script building system (a hypothetical, which may or may not be fully automatic) would provide the learner with the target: for example, to learn the script element for ”packing luggage”. As part of the input, the learner would be provided with at least one agent, and possibly more properties and fillers. The resultant ”concept” would be plugged back in to the script, and the fillers for the connecting properties (those that string the script together) could be learned in a targeted manner.

### 9.3.5 Closing Remarks

One of the major hurdles overcome during the course of experimentation was the absence of the necessary tools and infrastructure for the task at hand. Although OntoSem was used as a black-box, as were a few other systems (e.g., WEKA), the framework required to facilitate experiments in language learning was largely missing. Throughout the

experimentation, this framework was developed and tested in a real research environment (see Appendix B). The absence of these systems introduced an extra challenge to the entire process, resulting, however, in a series of well designed and functional tools keyed towards exploration of NLP and language learning. The existence of these systems moving forward in this research will be a benefit to any such future endeavors.

In this chapter we've discussed a series of directions for future research stemming from the experiments presented earlier. These tasks range from individual component improvements to sweeping methodological changes, and even hint at targeting knowledge in new ways. We've emphasized the need to strongly attest each component to provide for easier and more accurate evaluation. We've discussed strengthening the most important and irreplaceable components by highlighting troubling areas as represented by the actual results, and in some cases we've suggested completely removing poor-performing elements. An emphasis on stronger integration with classic machine learning, as well as experimentation with the latest craze in online collaborative documents shows the flexibility of the research to both look to prior lessons as well as those ahead. The possibilities for further exploration in this area are vast and exciting, and the low-hanging fruit that has been sampled throughout the existing research serves only to entice.

**Chapter 10**

# CONCLUSION

The knowledge acquisition bottleneck describes the assumed difficulty, or impossibility, of manually acquiring the broad and accurate coverage of knowledge needed to support language-based processors, such as semantic analysis engines. Fully automatic replacements for manual knowledge acquisition, historically, have focused on narrow domains or specific knowledge structures, and rarely on general purpose semantic knowledge.

In this work, we've presented a method of general purpose language learning using an existing semantic analysis engine bootstrapped with a database of manually acquired knowledge in the form of a language-independent world-view ontology and semantic lexicon. We demonstrated the feasibility of such a system through a series of experiments aimed to automatically learn new ontological concepts from open text available on the web. The system automatically constructed a corpus through web queries, produced a set of TMRs from the text, and used the semantic information from context presented to construct a set of candidate concepts for the target word, each of which was placed into the ontology using a custom-built concept similarity metric.

During the course of experimentation, we demonstrated the key hypothesis that results could be improved by using far fewer inputs presuming the quality of those inputs was significantly above the average incoming noise from an open corpus. Often, language learning

takes a machine-learning oriented approach, using statistical heuristics rather than semantic analysis to improve language resources; in these cases, volume of input is necessary to attest the results of any heuristic, as well as to smooth out noise in the corpus. Although we make no claims on the use of quantity over quality when working with statistical learners, our experimentation shows that when working with semantic analysis capabilities, less can be more if noise is reduced.

In addition to general purpose automatic language learning, we have also demonstrated ways to integrate manual assistant into an automatic learner, methods for automatically constructing a large-scale raw text corpus, heuristics for calculating similarity between ontological concepts and an EM-based word-sense clusterer using property / filler pairs from TMRs as input.

Finally, we've introduced an array of possible expansions on the work presented here, ranging from improvements on existing component, to complete methodological shifts. Importantly, we've taken the first major step towards fully automatic general purpose language learning using a semantic analysis system such as OntoSem. We've identified mistakes made and have rounded out several hypotheses and goals of this research. Work in this field has only just begun, and with much anticipation we prepare to take what we've learned during the course of this research and apply it to the next step along the way.

**Appendix A**

# PROOF OF CONCEPT: AUTOMATIC TARGET WORD SELECTION

In Chapter 7: Experiment Preparations: Corpus Building and Preprocessing, we touched on the concept of automatically selecting target words for the learner, rather than manually (as we did for each of the experiments). In this appendix, we demonstrate how we achieved automatic selection of target words, and include snippets of a log file showing this system, and comparing two generated TMRs (one with the learner's inclusion, and one without).

Our approach to automated target word selection was straightforward: if the analyzer encountered a lexical token that it had no entry for, the learner was triggered automatically, using that token as the input target word. In order to test the functionality developed, we manually created a test sentence which would trigger the desired target word only (in other words, the sentence consisted of only one unknown word). The system responsible for triggering the learner upon detection of the unknown word was designed to process the TMR both with and without the learned knowledge, highlighting the change in output resulting from the inclusion of the learner.

As a proof of concept, this process was run one time, manually, on the sentence "The truck drove down the street." We set up the experiment by loading OntoSem normally, and

then manually removing all meanings of the word ”truck” from the lexical and ontological resources. Further, for the sake of time, we pre-cached the results from the preprocessing phases (up through the creation of the TMR corpus) using results from the experiments. Although the web querying and semantic analyzing were not done in real time, the results were directly copied from an experiment that was, so are a valid replacement. The learner's phases, following TMR creation were run live inside the triggered environment, and the resultant candidate concept was returned to the analyzer to be used in the processing of the original input sentence.

Below is a collection of annotated snippets from the generated log file. These clips will demonstrate the usefulness of the automatic target word selection technique:

```
[7]> (setf (gethash 'TRUCK *temp-ontology-hash*) 'no-entry)
NO-ENTRY
[8]> (setf (gethash 'truck *temp-lex-hash*) 'no-entry)
NO-ENTRY
[9]> (analyze-sentence-with-thp-harness
            "The truck drove down the street.")
```

**Log 1:** The existing lexical and ontological entries are removed in [7] and [8] and the analyzer is turned on with the learner enabled (harnessed) in [9].

Following the learner, the analyzer loads in the candidate concept to its static knowledge, and continues to process the TMR as normal. The TMRs produced by the analyzer for the simple input sentence are nearly identical: one includes a frame based on *UNKNOWN-OBJECT*, the default concept assigned to an unknown word (that satisfies the criteria of an OBJECT, rather than an EVENT). The TMR produced after the automatic triggering of the learner includes a frame based on OBJECT-CANDIDATE, the name of the candidate concept produced by the learner. The contents of both of these frames are unsurprisingly identical (as the word is still used in the same way in the input text), however, the lexical map-

```
--begin pre-cached output--
Querying web for texts...
...
We sell used trucks as well as a very large variety of truck parts.
Learn more about the cause of your truck accident.
Keep your distance or pull over and let the truck pass.
We carry the right parts for your Heavy Duty Trucks.
It's on my bike in the back of the truck.
Commercial trucks are all over the road.
...
Found 92 texts...
Analyzing texts...
Semantic analysis finished...
--end pre-cached output--
```

**Log 2:** The pre-cached phase of the learner is invoked, rather than the fully dynamic one. The texts loaded (a sample is shown) came from a previous run of the system; the texts are then analyzed (cached analyses are loaded) and the learner's preprocessing phase is completed.

ping chosen for the word "drove" in the sentence changes based on the new information provided by the candidate concept (see Examples A.1 and A.2).

These results demonstrate that even on a very small and simple input text, automatically triggering the learner when an unknown token appears can impact the resultant TMR. Although this may not be the common or desired way to interact with the learner, the above logs and examples do demonstrate a proof of concept, and at least suggest that target words could be selected automatically via the above process, and queued in an external system to be processed manually by an acquirer.

```
TMR Frame Extractor Started!
...Using setting [textpointer=TRUCK]
...Using setting [fileName=.../TRUCK.xml]
...Using setting [stage=AS-IS]
...Using setting [filterProperties=true]
...Using setting [filterParametrics=true]
...Using setting [filterCorefers=true]
...logging in user
...issuing query for frames
...adding frame *UNKNOWN-OBJECT*-10852
.....filling frame *UNKNOWN-OBJECT*-10852
...adding frame *UNKNOWN-OBJECT*-9234
.....filling frame *UNKNOWN-OBJECT*-9234
...adding frame *UNKNOWN-OBJECT*-11055
.....filling frame *UNKNOWN-OBJECT*-11055
...adding frame *UNKNOWN-OBJECT*-15965
.....filling frame *UNKNOWN-OBJECT*-15965
...adding frame *UNKNOWN-OBJECT*-17308
.....filling frame *UNKNOWN-OBJECT*-17308
...adding frame *UNKNOWN-OBJECT*-17150
.....filling frame *UNKNOWN-OBJECT*-17150
...adding frame *UNKNOWN-OBJECT*-17459
.....filling frame *UNKNOWN-OBJECT*-17459
...adding frame *UNKNOWN-OBJECT*-15635
.....filling frame *UNKNOWN-OBJECT*-15635
...
...logging out user
.....exporting frames to xml
DONE!
```

**Log 3:** The learner extracts the TMR frames (abbreviated).

```
TMR Frame Clusterer Started!
...Using setting [inputFileName=.../TRUCK.xml]
...Using setting [outputFileName=.../TRUCK.xml]
...loading frames from file
...loading frames complete: 21 frames loaded
...logging in user
...performing basic clustering
...basic clustering complete: 1 clusters formed
...clustering on ontological depth
.....cluster OBJECT is too shallow
TMR Property Clusterer Started!
...Using setting [head=OBJECT]
...calculated total property count is 113
...creating attributes
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-10852
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-9234
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-11055
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-15965
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-17308
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-17150
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-17459
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-15635
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-16402
.....attribute=IN-FRAME-*UNKNOWN-OBJECT*-10118
...
...creating instances object
...creating unique instance names
...creating instances
...clustering with EM
DONE!
...logging out user
...writing to file
DONE!
```

**Log 4:** The learner clusters the property / fillers (abbreviated).

```
Candidate Constructor Started!
...Using setting [inputFileName=.../TRUCK.xml]
...Using setting [outputFileName=.../TRUCK.xml]
...logging in user
...importing tmr frame clusters
.....reading from file
.....parsing xml
.....extracting clusters
.....constructing candidate OBJECT-CANDIDATE
.....constructing candidate OBJECT-CANDIDATE
.....inheriting from parent
.......inheriting [age / (=>=< 0 1)]
.......inheriting [age / (> 0)]
.......inheriting [location / object]
.......inheriting [object-relation / object]
.......inheriting [property / all]
.......inheriting [scalar-financial-object-attribute / any-number]
.......inheriting [represents / object]
.......inheriting [represents / event]
.......inheriting [precondition / object]
.......inheriting [precondition / event]
.......inheriting [genuineness / yes]
.......inheriting [genuineness / no]
.......inheriting [fr-property / object]
.......inheriting [fr-property / event]
.......inheriting [extra-ontological / object]
.......inheriting [extra-ontological / event]
.......inheriting [corefer / object]
.......inheriting [enablement / object]
.......inheriting [enablement / event]
.......inheriting [area-studied-by-field / field-of-study]
.......inheriting [location / place]
...
total: 63
.....exporting to file
.......exporting candidate 0 (63)
.......done with candidate 0
...logging out user
DONE!
Harness DONE!
```

**Log 5:** The learner constructs the candidate (abbreviated).

```
<concept>
     <name>DRIVE-142</name>
     <instance-of>DRIVE</instance-of>
     <attributes>
          <attribute>
               <type>WORD-NUM</type>
               <facet>VALUE</facet>
               <value>2</value>
          </attribute>
          <attribute>
               <type>TEXTPOINTER</type>
               <facet>VALUE</facet>
               <value>DRIVE</value>
          </attribute>
          <attribute>
               <type>FROM-SENSE</type>
               <facet>VALUE</facet>
               <value>DRIVE-V1</value>
          </attribute>
     </attributes>
     <relations>
     </relations>
</concept>
```

**Example A.1:** The sense of ”drove” selected by the analyzer without help from the learner: drive-v1.

```
<concept>
     <name>DRIVE-63</name>
     <instance-of>DRIVE</instance-of>
     <attributes>
          <attribute>
               <type>WORD-NUM</type>
               <facet>VALUE</facet>
               <value>2</value>
          </attribute>
          <attribute>
               <type>TEXTPOINTER</type>
               <facet>VALUE</facet>
               <value>DRIVE</value>
          </attribute>
          <attribute>
               <type>FROM-SENSE</type>
               <facet>VALUE</facet>
               <value>DRIVE-V2</value>
          </attribute>
     </attributes>
     <relations>
     </relations>
</concept>
```

**Example A.2:** The sense of ”drove” selected by the analyzer when the learned knowledge was available: drive-v2.

**Appendix B**

# TOOLS AND INTERFACES

## B.1 Introduction

During the course of the experimentation presented throughout this work, various support systems had to be devised in order to facilitate the task at hand. A large collection of tools and services was needed as a foundation to make accessing OntoSem more convenient. In order to assist in analyzing the results, several visual interfaces would also need to be developed. The remainder of this appendix contains a light introduction to the supporting systems created during the course of experimentation that have since been used by other researchers and as a matter of daily routine for manual acquisition and development of OntoSem. Section B.2 describes the DekadeAPI, a researcher-friendly services front for accessing the majority of OntoSem's systems and resources. Section B.3 describes DekadeAtHome, a graphical user interface constructed on top of the DekadeAPI to facilitate manual acquisition and scrutinization of OntoSem. Section B.4 discusses the THPProcesor, a set of services built on top of the DekadeAPI specifically for the research presented in this work. Finally, the last section will discuss the THPInterface, a second graphical user interface designed to facilitate this research by organizing the various results of each experiment in a logical and accessible way.

## B.2 DekadeAPI

OntoSem has been developed primarily in CLISP but also has modules and components in several other languages (e.g., C, C++, Perl, and Java). Additionally, the knowledge resources (both static, such as the lexicon and ontology, as well as dynamic, e.g. TMRs) are internally defined in CLISP's s-expression format. In order to provide a more human-friendly interface to the system for experimental development purposes, the DekadeAPI was developed. DEKADE, an acronym for "Development, Evaluation, Knowledge Acquisition and Demonstration Environment" originally encompassed a user interface to assist in lexical and ontological acquisition, as well as TMR inspection. Crucial elements of this system were converted into a fully-fledged API, written completely in Java to allow access to the most commonly required functions in OntoSem, as well as to provide a more accessible format for knowledge resources by implementing high-level objects as wrappers.

Essentially, every knowledge structure was converted into a convenient high-level object, with an array of methods (for querying and access) as well as a standardized XML format defined. These objects became the conversation-medium for the DekadeAPI, which exposed, at a service level, all of the critical processors made available by OntoSem (such as the sentence breaker, preprocessor, syntactic and semantic analyzers, and reference engine). By blurring the communication barrier between these systems using a standardized object set, the DekadeAPI allows research programmers to quickly and conveniently access the majority of OntoSem's resources.

Developing the DekadeAPI was a necessary step towards furthering the experiments presented in this work. The system enabled rapid-prototyping of experimental projects, making it easy to try "what if" scenarios and facilitated the use of a variety of other openly available projects written in Java (such as the WEKA toolkit for machine learning). Further,

the development of the API has facilitated a variety of other research projects (e.g., (Dimitroff 2008), (Stone 2007)) and is commonly used as a support system for development of OntoSem (via DekadeAtHome as well as other test suites and user interfaces).

## B.3 DekadeAtHome

To further enhance the usability of OntoSem, as well as to facilitate development and acquisition of the system, a graphical user interface was constructed around the DekadeAPI. DekadeAtHome (DAH) (English 2006), a desktop client relying on a remote connection to communicate with the latest version of OntoSem allows the user to navigate and edit the static and dynamic knowledge resources, and to use and test the analyzer in a convenient way. DAH contains a full browser and editor for each knowledge resource, integrated with each other (where appropriate) to maximize ease-of-use for the acquirer. In this section, we'll provide an example use of three of the editors (the lexicon, the ontology, and TMRs).

### B.3.1 Lexical Acquisition (The Lexicon Editor)

Lexical acquisition involves populating a series of fields for each word-sense acquired. The editor found in DAH allows for advanced searching of the existing static knowledge. Creating a new word-sense or modifying an existing one presents the user with a screen similar to the one found in Figure B.1:

The acquirer can populate each field as desired, using the predefined syntax. Usability improvements, such as parenthesis highlighting assist in the acquisition process. After completion, the user can validate the work (an automatic process which checks for syntactic or semantic errors, highlighting any problem areas) and then store the changes into the static knowledge. Once this is done, OntoSem has immediate access to the new data.

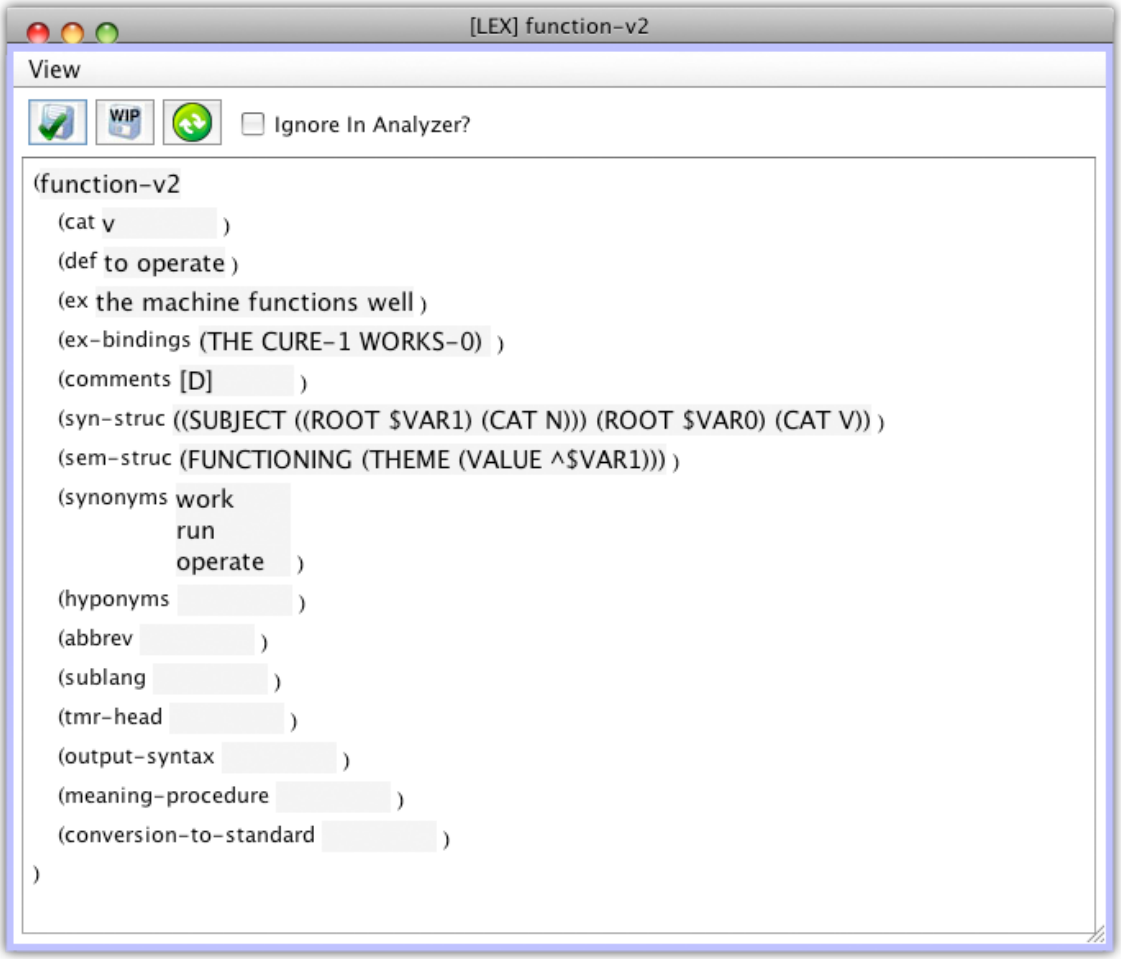


FIG. B.1. A typical lexicon entry in the DAH editor.

### B.3.2 Ontological Acquisition (The Ontology Editor)

The ontology editor is significantly more complex (due to the nature of the data) than the lexical editor. Essentially, the ontology editor consists of two main interfaces: the primary interface allows for editing of the property / fillers of an individual concept. A secondary interface allows for browsing and manipulation of the subsumption links of the ontology in a tree-view format. The primary editor is shown in Figure B.2:

The editor allows the user to select any of the properties defined for the concept (or

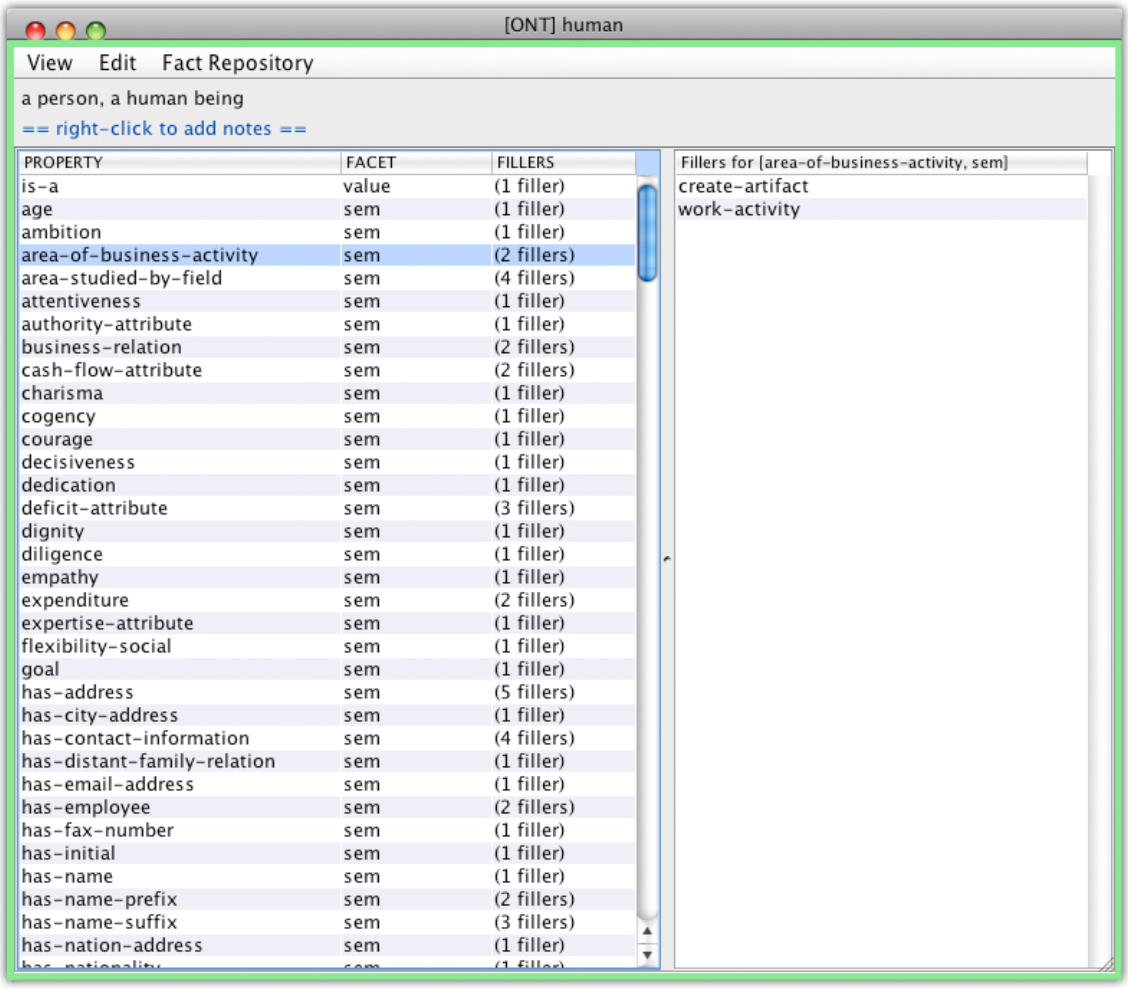


FIG. B.2. An example of ontology editing.

to define new ones is required), and then to modify the fillers of that property as desired. As each change is made to the concept, the validator is automatically invoked to ensure the new data is acceptable. Any invalid (syntactically or semantically) fillers are rejected immediately; and valid changes are stored and are accessible to OntoSem at that point.

To view the the subsumption links of the ontology, the secondary interface presents the hierarchy in a tree structure, and allows for navigation and rearrangement of the tree (which results in changes to the inherited values of the affected concepts). The tree-view

editor is shown in Figure B.3:

Each of the editors has been designed with a variety of convenience features to facilitate the arduous task of ontological acquisition. The primary editor integrates with the lexical resources and fact repository resources (the user can load connecting resources with a simple action, e.g., lexical entries that reference the selected ontological concept).

### B.3.3 TMR Editor

The TMR editor, one of many editors used to manipulate the results of analysis, has a deep level of integration with both the lexicon and ontology. After analysis, the user can investigate the results, modify the outputs and even-reprocess after manipulating the necessary resources. The TMR editor allows the user to assign conceptual instances to each word (token) in the input text, and then to assign relational links between the words (e.g., AGENT and THEME). The editor uses the knowledge resources to intelligently limit the options presented to the user while allowing an override if desired. An example of the TMR editor in use is shown in Figure B.4:

## B.4 THP Processor

To further facilitate the ease of experimentation, an additional services layer was constructed specifically to support the research presented here. Constructed on top of the DekadeAPI, the THP Processor (an acronym from the first days of the experiment: The Hobbit Project referring to the goal of automating the learning of texts written by J.R.R. Tolkien) added a series of wrappers around the atomic functions exposed by the DekadeAPI that targeted the research on language learning. Many of the functions created were conveniences for automating a task over a corpus, rather than a single instance of an object (such as a TMR). However, the THP Processor also defined a database of results that were

organized by individual experiment are were independently queryable.

The THP Processor's main contribution was the further abstraction of the programmatic details required to organize a large suite of experiments in language learning. As such, in addition to a complete storage solution (the user could simply select which experiment they wished to access as a parameter), the following services were made available at a very high-level:

1. A web-crawler: this system allows the user to select a set of target words, a search engine, a minimum and maximum (optional) count of returned sentences, target word dependencies, and an array of other parameters; the results were collection of raw texts with the matching target words, completely stripped of HTML and markup, and having passed through a variety of quality filters.

2. A batch analyzer: this system allows the user to provide a collection of raw texts, as well any number of optional parameters for the semantic analyzer, and would result in a corpus of TMRs.

3. A candidate constructor: a provided list of TMRs and a target word would results in a candidate ontological concept constructed from the collation of the property / filler pairs contained within the matching TMR frames.

4. A similarity decision tree: this system used the provided candidate concept, a similarity metric and an input ontology to locate a suitable home for the concept by finding the highest similarity parent.

5. A variety of smaller systems: e.g., the EM-based word-sense clusterer, a collection of similarity metrics, a bag-of-words implementation, and a variety of analysis scripts for assisting in evaluation.

By stringing these services together (and modifying them as needed) a distinction between ”research” and ”engineering” was easily drawn, allowing this work to progress in a more organized fashion.

## B.5 THP Interface

Much like DekadeAtHome, the THP Interface is a graphical user interface wrapping the THP Processor (as well as the DekadeAPI). The THP Interface was designed to simplify the task of running the experiments by providing an easy click-button interface and visually simplified results. The system allows the user to select the current experiment (or a past one), and commence processing at any stage of the pipeline. The interface automatically polls any related analysis scripts to simultaneously provide both the actual results and a glimpse into automatic evaluation of the outcome of processing. An example of the THP Interface in use is shown in Figure B.5:

## B.6 Conclusion

The need to implement a suite of supporting software to facilitate experimentation on the learning system was clear from the early stages. The amount of work spent on these supporting systems was not trivial, and although the THP Processor and THP Interface have limited use outside the scope of this research, developing those tools was a fraction of the work required to develop the DEKADE systems (both of which are heavily used in other research and for general OntoSem development). Any future research on language learning will continue to use each of the tools described here, and will enhance them as needed to support the next batch of experiments.

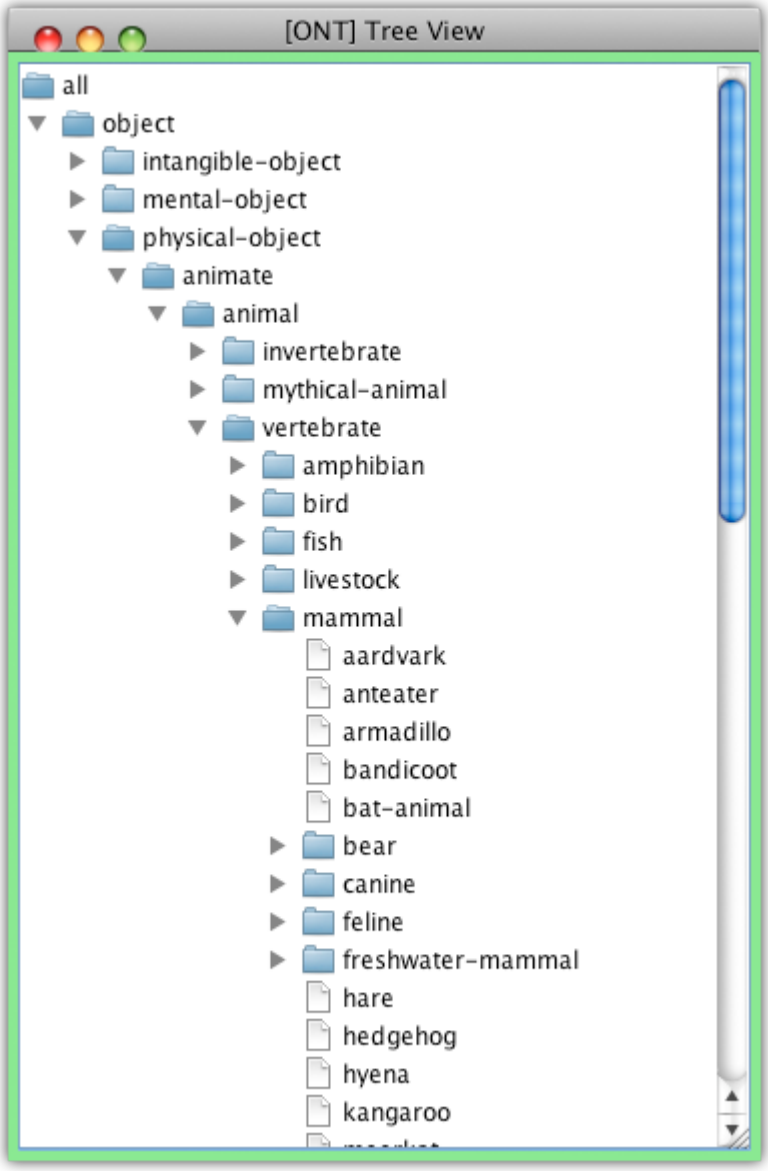


FIG. B.3. The ontology tree-view editor.

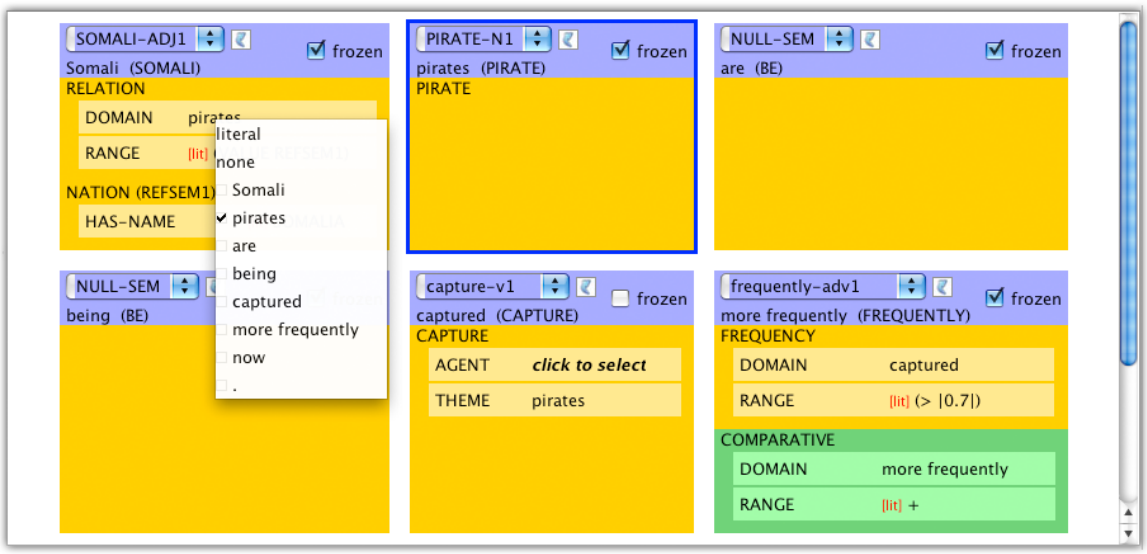


FIG. B.4. An example of the TMR editor in DAH being used on the input text ”Somali pirates are being captured more frequently now.”

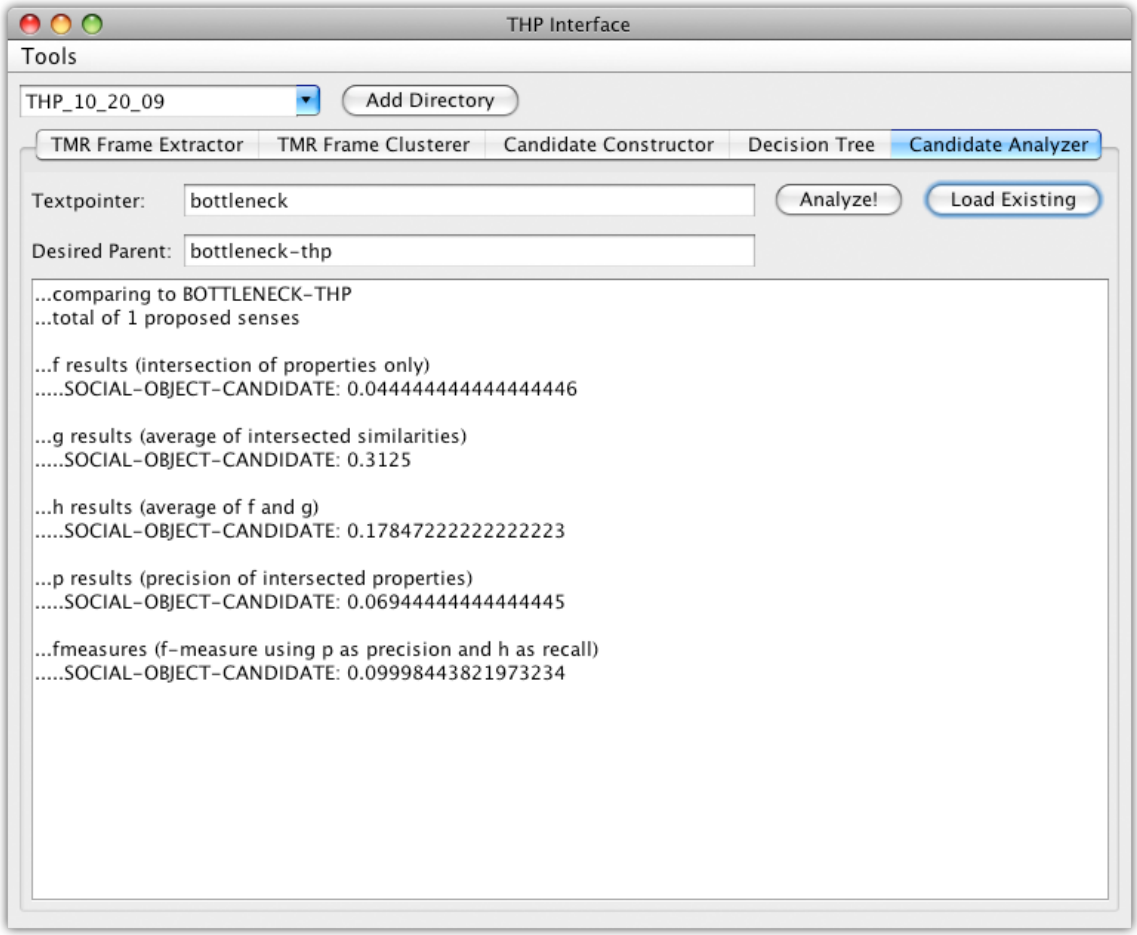


FIG. B.5. The THP Interface displaying interim results of analysis.

## REFERENCES


[1] Ankolekar, A. 2003. Investigating semantic knowledge for text learning. In *In Proceedings of ACM SIGIR Workshop on Semantic Web*.

[2] Aussenac-Gilles, N.; Bibow, B.; and Szulman, S. 2000. Corpus Analysis for Conceptual Modelling.

[3] Basili, R.; Pazienza, M.; and Vindigni, M. 1999. Lexical Learning for Improving Syntactic Analysis. In *ACAI99 Workshop on Machine Learning in Human Language Technology*.

[4] Betteridge, J.; Carlson, A.; Hong, S. A.; Hruschka, E. R.; Law, E. L. M.; Mitchell, T. M.; and Wang, S. H. 2009. Toward Never Ending Language Learning.

[5] Bobrow, D.; Condoravdi, C.; Karttunen, L.; and Zaenen, A. 2009. Learning by reading: normalizing complex linguistic structures onto a knowledge representations. In *AAAI Symposium 2009: Learning by Reading and Learning to Read*.

[6] Brewster, C.; Iria, J.; Ciravegna, F.; and Wilks, Y. 2005. The Ontology: Chimaera or Pegasus. In *In Proc. Dagstuhl Seminar on Machine Learning for the Semantic Web*, 13–18.

[7] Brill, E. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics* 21:543–565.

[8] Cardie, C. 1996. Embedded Machine Learning Systems for Natural Language Processing: A General Framework. In *Lecture Notes in Artificial Intelligence Series*, 315–328. Springer.

[9] Cimiano, P.; Staab, S.; and Tane, J. 2003. Automatic Acquisition of Taxonomies from Text: FCA meets NLP. In *Proceedings of the ECML/PKDD Workshop on Adaptive Text Extraction and Mining, Cavtat-Dubrovnik, Croatia*, 10–17.

[10] Clark, P., and Harrison, P. 2009. Large-scale extraction and use of knowledge from text. In *K-CAP '09: Proceedings of the fifth international conference on Knowledge capture*, 153–160. New York, NY, USA: ACM.

[11] Clark, P., and Porter, B. 1997. Building Concept Representations from Reusable Components. In *In AAAI-97*, 369–376. AAAI Press.

[12] Clark, P.; P.Harrison; THompson, J.; Wojcik, R.; Jenkins, T.; and Israel, D. 2007. Reading to Learn: An Investigation into Language Understanding. In *Proceedings of AAAI 2007 Spring Symposium on Machine Reading*.

[13] Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S. 2000. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence* 118(1-2):69–113.

[14] Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society* 39(1):1–38.

[15] Dimitroff, D. 2008. A Semi-Automatic Advisory System for an Ontology to Self-Update via Web Browsing with Shallow Parsing and Clustering.

[16] Duda, R. O.; Hart, P. E.; and Stork, D. G. 2001. *Pattern Classification*. John Wiley & Sons.

[17] English, J., and Nirenburg, S. 2007a. Calculating Concept Similarity Heuristics for Ontology Learning from Text. Technical report, University of Maryland, Baltimore County.

[18] English, J., and Nirenburg, S. 2007b. Ontology Learning from Text Using Automatic Ontological-Semantic Text Annotation and the Web as the Corpus. In *Proceedings of the AAAI 2007 Spring Symposium Series on Machine Reading*.

[19] English, J. 2006. DEKADE II: An Environment for Development and Demonstration in Natural Language Processing. Master's thesis, University of Maryland, Baltimore County.

[20] Esposito, F.; Ferilli, S.; Fanizzi, N.; and Semeraro, G. 2000. Learning from parsed sentences with INTHELEX. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, 194–198. Morristown, NJ, USA: Association for Computational Linguistics.

[21] Faure, D., and Nedellec, C. 1999. Knowledge Acquisition of Predicate Argument Structures from Technical Texts Using Machine Learning: The System ASIUM. In *EKAW '99: Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management*, 329–334. London, UK: Springer-Verlag.

[22] Fleischman, M., and Roy, D. 2005. Intentional context in situated natural language learning. In *CONLL '05: Proceedings of the Ninth Conference on Computational Natural Language Learning*, 104–111. Morristown, NJ, USA: Association for Computational Linguistics.

[23] Frank, G.; Farquhar, A.; and Fikes, R. 1999. Building a Large Knowledge Base from a Structured Source: The CIA World Fact Book. *IEEE Intelligent Systems* 14:14–1.

[24] Futrelle, R., and Gauch, S. 1993. Experiments in syntactic and semantic classification and disambiguation using boostrapping. In *Acquisition of Lexical Knowledge from Text*.

[25] Gennari, J. H.; Musen, M. A.; Fergerson, R. W.; Grosso, W. E.; Crubézy, M.; Eriksson, H.; Noy, N. F.; and Tu, S. W. 2003. The evolution of Protégé: an environment for knowledge-based systems development. *Int. J. Hum.-Comput. Stud.* 58(1):89–123.

[26] Gleim, R.; Mehler, A.; and Dehmer, M. 2006. Web corpus mining by instance of Wikipedia. In *WAC '06: Proceedings of the 2nd International Workshop on Web as Corpus*, 67–74. Morristown, NJ, USA: Association for Computational Linguistics.

[27] Gomez, F.; Hull, R.; and Segami, C. 1994. Acquiring knowledge from encyclopedic texts. In *Proceedings of the fourth conference on Applied natural language processing*, 84–90. Morristown, NJ, USA: Association for Computational Linguistics.

[28] Gupta, A. 2005. Using Ontologies and the Web to Learn Lexical Semantics. Master's thesis, University of Maryland.

[29] Hahn, U., and Marko, K. G. 2002. Ontology and Lexicon Evolution by Text Understanding. In *In Proceedings of the ECAI 2002 Workshop on Machine Learning and Natural Language Processing for Ontology Engineering*.

[30] Hahn, U., and Schnattinger, K. 1997. Knowledge mining from textual sources. In *CIKM '97: Proceedings of the sixth international conference on Information and knowledge management*, 83–90. New York, NY, USA: ACM.

[31] 2006. HTML Parser.

[32] Iria, J.; Brewster, C.; Ciravegna, F.; and Wilks, Y. 2006. An Incremental Tri-partite Approach to Ontology Learning. In *Proceedings of LREC2006*.

[33] Iwańska, L. M.; Mata, N.; and Kruger, K. 2000. Fully automatic acquisition of taxonomic knowledge from large corpora of texts: limited-syntax knowledge representation system based on natural language. 335–345.

[34] Java, A.; Kolari, P.; Finin, T.; Mayfield, J.; Joshi, A.; ; and Martineau, J. 2007. BlogVox: Separating Blog Wheat from Blog Chaff. In *Proceedings of the Workshop on Analytics for Noisy Unstructured Text Data, 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*.

[35] Kilgarriff, A., and Grefenstette, G. 2003. Introduction to the special issue on the web as corpus. *Comput. Linguist.* 29(3):333–347.

[36] Kohomban, U. S., and Lee, W. S. 2005. Learning semantic classes for word sense disambiguation. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, 34–41. Morristown, NJ, USA: Association for Computational Linguistics.

[37] Krymolowski, Y., and Roth, D. 1998. Incorporating Knowledge in Natural Language Learning: A Case Study. In *In COLING-ACL'98 workshop on the Usage of WordNet in Natural Language Processing Systems*, 121–127.

[38] Landauer, T., and Dumais, S. 1997. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review* 104:221–240.

[39] Lavelli, A.; Magnini, B.; and Sebastiani, F. 2002. Building Thematic Lexical Resources by Bootstrapping and Machine Learning. In *Proceedings of the LREC-2002 Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data*.

[40] Lewis, D. D. 1998. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, 4–15. London, UK: Springer-Verlag.

[41] Maedche, A.; Maedche, E.; and Volz, R. 2001. The Ontology Extraction Maintenance Framework Text-To-Onto. In *In Proceedings of the ICDM01 Workshop on Integrating Data Mining and Knowledge Management*.

[42] Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM* 38(11):39–41.

[43] Mooney, R. 1987. Integrated Learning of Words and Their Underlying Concepts. Technical report.

[44] Nirenburg, S.; Beale, S.; and McShane, M. 2004. Evaluating the performance of the OntoSem semantic analyzer. In *TextMean '04: Proceedings of the 2nd Workshop on Text Meaning and Interpretation*, 33–40. Morristown, NJ, USA: Association for Computational Linguistics.

[45] Nirenburg, S.; Dimitroff, D.; English, J.; and Pfeifer), C. 2007. Three Experiments on Mining the Web for Ontology and Lexicon Learning. Technical report, University of Maryland, Baltimore County.

[46] Ogata, N., and Collier, N. 2004. Ontology Express: Statistical and Non-Monotonic Learning of Domain Ontologies from Text. In *ECAI 2004 Workshop on Ontology Learning and Population*.

[47] Onyshkevych, B. 1997. *Ontosearch: Using an ontology as a search space for knowledge based text processing*. Ph.D. Dissertation, Carnegie Mellon University.

[48] Palmer, S. 2001. The Semantic Web: An Introduction.

[49] Reinberger, M. 2004. Discovering Knowledge in Texts for the Learning of DOGMA-Inspired Ontologies. In *ECAI 2004 Workshop on Ontology Learning and Population*.

[50] Riloff, E. 1996. An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains. *Artificial Intelligence* 85:101–134.

[51] Seo, Y.; Ankolekar, A.; and Sycara, K. 2004. Feature Selection for Extracting Semantically Rich Words. Technical report.

[52] Stone, A. 2007. EBIDS-SENLP: An Email-Based Intrusion Detection System to find Social Engineering using Natural Language Processing. Master's thesis, University of Maryland, Baltimore County.

[53] Thompson, C. A., and Mooney, R. J. 1998. Semantic Lexicon Acquisition for Learning Natural Language Interfaces. Technical report, Austin, TX, USA.

[54] Will, T. 1999. Singular Value Decomposition.

[55] www.askoxford.com. AskOxford.com How many words are there in the English language?

[56] www.dictionary.com. Dictionary.com.

[57] www.wikipedia.org. Wikipedia, the Free Encyclopedia.

[58] www.wiktionary.org. Wiktionary, the Free Dictionary.

[59] Zernik, U., and Dyer, M. G. 1985. Towards a self-extending lexicon. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, 284–292. Morristown, NJ, USA: Association for Computational Linguistics.