Constructing Text Sense Representations

Ronald Winnemöller
Regional Computer Centre
University of Hamburg
Hamburg, Germany
ronald.winnemoeller@rrz.uni-hamburg.de

Abstract

In this paper we present a novel approach to map textual entities such as words, phrases, sentences, paragraphs or arbitrary text fragments onto artificial structures which we call “Text Sense Representation Trees” (TSR trees). These TSR trees represent an abstract notion of the meaning of the respective text, subjective to an abstract “common” understanding within the World Wide Web. TSR Trees can be used to support text and language processing systems such as text categorizers, classifiers, automatic summarizers and applications of the Semantic Web. We will explain how to construct the TSR tree structures and how to use them properly; furthermore we describe some preliminary evaluation results.

1 Introduction

Many important tasks in the field of Natural Language Processing (NLP) such as text categorization, text summarization, (semi-) automatic translation and such require a certain amount of world knowledge and knowledge about text meaning and sense (Allen, 1995; R. Cole et al., 1995).

Handling the amount of textual data in the World Wide Web also increasingly requires advanced automatic text and language processing techniques: successful search engines like Google (Google, Inc., 2004) already employ text retrieval and information extraction methods based on shallow semantic information.

There are many methodologies to generate word sense representations, but efficiency and effectivity of fully automated techniques tends to be low (Diana Zaiu Inkpen and Graeme Hirst, 2003). Furthermore, formalisation and quantification of evaluation methods is difficult because in general word sense related techniques are only verifyable through theoretical examination, application on language or human judges (Alexander Budanitsky and Graeme Hirst, 2001), i.e. there is no inherent validation because there is no direct connection to the world as perceived by humans. In the case of frequency based word sense representations corpus related difficulties arise (number of tagged entities, corpus quality, etc.). In order to overcome these limitations, we developed a methodology to generate and use explicit computer-usable representations of text senses.

A common understanding of the “sense” of words is defined by the ways the word is used in context, i.e. the interpretation of the word that is consistent with the text meaning[1] - as summarized by S. G. Pulman in (R. Cole et al., 1995, Section 3.5). Extending this definition onto full texts, we introduce our notion of “Text Sense Representation” (TSR) as “the set of possible computer usable interpretations of a text without respect to a particular linguistic context”[2].

TSR Trees provide detailed answers to questions like “how close are these *n* words topically related to each other?”, “are these *m* sentences really about the same topic?” or “how much does paragraph *x* contribute to topic *y*?”. They cannot tell e.g. a telephone is a physical artifact, it’s purpose is to enable distant communication, etc.

TSR Trees are not meant to substitute meaning acquired through conceptual or linguistic analysis but are rather aimed at:

- augmenting deeper (linguistic or conceptual) methodologies by providing additional analysis clues
- standalone usage in generic shallow methods (e.g. in shallow text categorization) and specific applications (e.g. anti-spam functionality)

2 Related Work

Our notion of semantics is closely related to the notion of “naive semantics” discussed in (K. Dahlgren

[1] In this paper, we would like to extend the notion of “Word Sense” onto “Text Sense”, i.e. texts of arbitrary length

[2] Allen instead uses the term “logical form” for this kind of context-independent meaning representation, c.f. (Allen, 1995), p.14

et al., 1989). This article describes “naive semantics” as “a level of world knowledge that is general and common to many speakers of a language”, i.e. commonsense knowledge associated with words. Naive semantics identifies words with concepts which vary in type.

A discussion of fundamental corpus-related aspects of word senses is provided by Kilgariff (Adam Kilgariff, 1997). Kilgariff herein questions the use of word sense disambiguation and concludes that word senses can only be defined “relative to a set of interests” and the “basic units of word meanings” are occurrences of words in contexts. Our notion of TSR trees aims at aggregating text meaning in it’s topical context in order to construct a context independent representation.

In Literature, there are several strong directions of representing text meaning or text sense: one prominent approach uses frame-based representation languages in combination with first order logic semantics. The analyzed text is matched against the frame database in order to construct text meaning representations. An example of this approach is presented by Clark et. al. (P. Clark et al., 2003). Dahlgren et. al. present “KT”, a complex text understanding system based on naive semantics. KT also uses frames to represent semantic content.

A project that is based on a roughly similar notion of text meaning representation (TMR) concepts is the μkosmos project (Mahesh, 1996; Kavi Mahesh and Sergei Nirenburg, 1996). It is aimed at the creation of a machine translation system that uses a broad-coverage ontology and various input sources in order to translate english to spanish texts and vice versa. TMR concepts within μkosmos are handwritten frame-based data structures. Text meaning is represented by instances thereof that are derived by semantic rules from a linguistic rule database.

Frame-based meaning representations are also the basis of AutoSlog-TS, an information extraction system that automatically acquires conceptual patterns from untagged texts, using only a preprocessed training corpus (Ellen Riloff and Jay Shoen, 1995). The thusly constructed concepts can be seen as text meaning representations.

Approaches of computing text meaning similarities include using web directories for generating path-shaped data structures for text categorization (Fabrizio Sebastiani, 2003; Giuseppe Attardi et al., 1998). Sebastiani herein purports his efforts in by mining the structure of both web “catalogues” (web directories) for extracting category labels and mining web page structure for the actual classification task. This is an example for using path- and graph based methods rather than frame based structures. Another example would be the methodology described in this article.

3 TSR Trees

In this Section we will informally describe our two algorithms for constructing Text Sense Representation Trees. The first algorithm builds “initial” TSR trees of single input words or very short phrases (Section 3.1), the second generates “derived” TSR trees for arbitrary texts from pre-computed TSR trees.

3.1 Building Initial TSR Trees

The algorithm for building initial TSR trees is based on the retrieval of pages from a “web directory” A “web directory” (other sources use the term “web catalogue” (Fabrizio Sebastiani, 2003)) is a browsable taxonomy of web pages. These web pages are parsed and category descriptions and weight values are extracted from them. The extracted information is then merged into term-specific TSR trees, optionally normalized and pruned.

In the following explanations we will use the notions “input term”, “input word” and “input phrase” as follows: An *input term* is any text that is used as input to an algorithm or program. An *input word* is any singular word that is an input term. A word is defined as sequence of alphanumeric symbols not interrupted by whitespace. An *input phrase* is any sequence of input words that are separated by whitespace or other non-alphanumeric characters.

Our algorithm takes single words or very short phrases as input terms and assumes that every part of the input phrase has pragmatic and semantic relevance. Input term selection is therefore a fundamental prerequisite in this context.

The output of our algorithm consists of a tree structure of labeled and weighted nodes. The labels are short phrases that provide some meaningful context while the weights are simple integer numbers. Each tree node has exactly one label and one weight attached to it.

The following five steps will explain how to generate initial TSR Trees:

a. Retrieval The input term is redirected as input to a web directory. [3]

[3]Since our prototype was based on the “Open Directory Project” (ODP), consisting of a web directory and a search engine (Netscape Inc., 2004), we will refer to this particular service throughout this article and use it as implementation data source. Nonetheless, our algorithm is not restricted to the ODP but can use other web directories like Yahoo Inc. (Yahoo Inc., 2004) or even internet newsgroups.

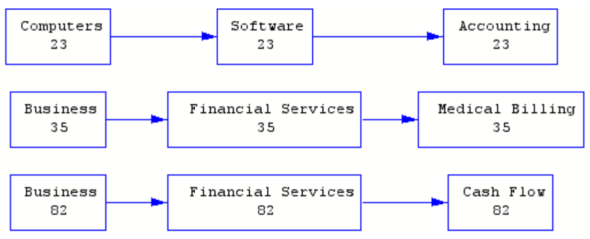

Figure 1: Category Path Listing for “account” example (excerpt)

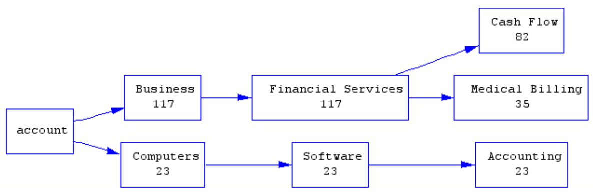

Figure 2: TSR tree for “account” example (excerpt)

The web directory to use is not assumed to meet strict requirements in terms of sensible category labels, balanced branches, etc. but can be any taxonomic structure provided it can be transformed into weighted paths and is large enough to cover a substantial subset of the target language.

Outcome of this redirection is a HTML-formatted list of categories including the number of hits for each category.

b. Tree Construction The lines of the output list returned by the web directory are then parsed and converted into a sequence of weighted category terms. Because each sequence represents a different contextual use of the word (in the symbolic sense), each sequence also represents a different sense of that word in that topical context.

Each term contains a singular category path label and the number of query hits within that category. An excerpt of the account example terms is exemplified in the Figure 1:

After that, all terms are merged into a single hierarchical tree with weighted and labeled nodes. Figure 2 provides an example hereof.

The resulting tree then reprenrepresentsts the input text phrase. Even though the uniqueness of this representation cannot be guaranteed in theory, a clash of two different terms representation is highly unlikely.

The tree generation process obviously fails if the input term cannot be found within the web directory and hence no categorical context is available for that term.

Figure 3: The complete (unpruned) “account” TSR tree

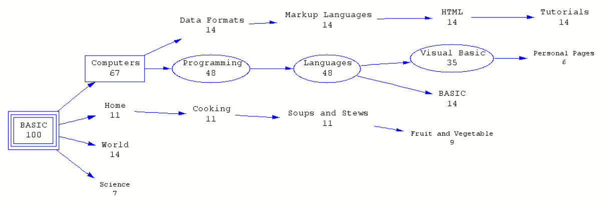

Figure 4: Pruning the “account” TSR tree by threshold 5%

c. Normalization In order to enable uniform processing of arbitrary trees, each tree has to be “normalized” by weight adjustment: The sum of all node weights is computed as 100 percent. All weights are then recalculated as percentage of this overall tree weight sum. The sum weight is attached to the TSR tree root as “tree weight”.

d. Node Pruning (optional) Due to the nature of the underlying web directory, there are sometimes false positives in wrong categories, i.e. when a term is used in a rather esoteric way (e.g. as part of a figure of speech, etc.).

In order to sort out such “semantical noise”, “insignificant” nodes can be deleted using a common heuristic. Some preliminary experiments have shown that using a certain threshold on the node weight percentage is a good heuristic. An example of a processed TSR tree is shown in Figure 4. while the corresponding unprocessed TSR tree is depicted in Figure 3[4]

e. List Transformation (optional) It is possible to transform a TSR tree into a list: by iterating the TSR tree and selecting only the nodes with the high-

[4]The labels within this figure might be printed too small to read but it is the shape of the structure that is important rather than individual node labels.

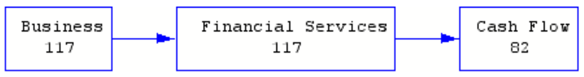

Figure 5: List representation of “account” example (from excerpt)

est weight at each respective depth, a TSR list is easily created. This list represents the *most common meaning* of the input term. Since this meaning is applicable in most cases, sufficiently robust algorithms may use these lists successfully, e.g. for simple text classification purposes. An example list, derived from the “account” example (Figure 2), is depicted in Figure 5.

f. External Representation (optional) Lastly, the tree is converted into an external representation in the RDF (O. Lassila and R. Swick, 1999) language. We chose this particular paradigm because it is an open standard and well suited for representing graph and tree based data. Furthermore, a number of tools for dealing with RDF already exist – RDF is one of the basic building blocks of the Semantic Web (O. Lassila and R. Swick, 1999) and we expect RDF based TSR trees to be of great use in that domain (e.g. for classification and information extraction).

Summary In this section we presented the construction of computer usable complex TSR trees by utilizing an underlying web directory containing explicit world knowledge. The generated trees are our basic building blocks to represent the “sense” of the input term in a programmatically usable way.

The construction of a TSR tree can therefore be seen as the result of a (shallow) text “understanding” process as defined in (Allen, 1995).

3.2 Constructing Derived TSR Trees

TSR trees can also be constructed by merging existent TSR trees. This process provides means of dealing with complex phrases: through adding TSR trees (applying the set union operation) it is possibly to acquire TSR trees of arbitrary text fragments, i.e. to build TSR trees by merging the TSR trees of its constituents.

By using the derivation process, TSR trees can be built for arbitrary input texts while maintaining comparability through the respective tree features (see 4.1).

Since TSR trees consist of weighted paths, out-of-context senses of single terms will be eliminated in the merging process. This makes using TSR trees in large texts a very robust algorithm (preliminary experiments have shown that virtually all errors occur in preprocessing steps such as language identification, etc.). Superficial investigation showed that TSR trees generated from complex descriptions are of higher quality than TSR trees from single terms (less “semantic noise”, features are more expressive).

On the other hand, the derivation process (in conjunction with a dictionary) can also be used to build TSR trees of *descriptions* of words that cannot be found in the web directory as a substitute for the word itself.

It is a matter of current research whether TSR trees derived from dictionary-like descriptions of terms are in general preferrable to the use of initial TSR trees (see the discussion of the “distance” feature in 4.2).

4 Using Text Sense Representation Trees

In this Section, the term “feature” will be used quite synonymous to the term “operation”: a feature is a bit of information that is retrieved by application of the corresponding operation on a TSR tree.

It is important to note that even though the TSR trees themselves are very subjective to the underlying web directory, the resulting features do not show this weakness. Any NLP application implementing algorithms that are based on TSR trees should not rely on the tree representations themselves (in terms of tree structure or node labels), but rather on the operational TSR tree features discussed in this section.

4.1 Simple TSR Tree Features

At first, we define a set of four features that can be computed for single TSR trees:

1. **Tree Weight.** The individual tree’s weight can be interpreted as quantitative measure of the input term within the web directory. By comparing the weight of individual trees, it is possible to determine which term occurs more often in written language (as defined by the web directory itself).

2. **Generality.** The tree’s breadth factor is an indicator for the “generality” of the input term, i.e. the broader the tree, the more general the use of the word in written language and the more textual contexts the word can be used in.

   General terms tend to be not specific to particular web pages, hence will show up in a number of pages throughout the web taxonomy. In contrast, less general terms tend to occur only on pages specific to a particular category in the web taxonomy.

3. **Domain Dependency.** The tree's depth factor can be interpreted as "domain dependency indicator" of the input term. Deep structures only occur when the input term is most often located at deep subcategory levels in the web directory which is an indicator for restricted use of that term within a particular domain of interest.

4. **Category Label.** Usually the node labels themselves provide clues to its respective terms meaning. Even though these clues may be quite subjective and in some cases misleading or incomplete, in most cases they can serve as hints for human skimming of the categories involved. Since these labels are provided as english words or short phrases, they might themselves be subject to initial TSR tree building (see Section 4.3).

4.2 Advanced TSR Tree Features and Operations

While operations on single TSR trees provide simple text processing clues, operations on sets of trees are much more informative:

1. **Difference.** A number of "difference" features are available that can be used to compare individual features of a number of trees:

   - Tree weight difference
   - Breadth difference
   - Depth difference

   These difference features arise from comparisons of the simple TSR tree features, hence they describe numerical differences between such values. For example, a high weight difference shows a high difference between the respective terms' general use in language.
   It is important to note that the difference features are not only usable in respect to complete trees but can be applied to tree branches as well, e.g. in order to analyze tree behaviour in certain contexts.

2. **Distance.** The "distance" feature is computed by counting the number of "edit" operations (add node, delete node, transform node) it takes to transform one tree into another. This feature is designed much after the "Levenshtein-distance feature" in the field of text-processing (D. S. Hirschberg, 1997).

   In general, this feature describes a notion of "semantic relatedness" between two input terms, i.e. a high distance value is expected between largely unrelated terms such as "air" and "bezier curve" while a low value is expected between closely related terms such as "account" and "cash".

   The distance feature can be implemented by applying the set difference operation: the subtracting of TSR tree from one another results in a number of remaining nodes, i.e. the actual distance value.

   Recent findings have shown though that this simple procedure is only applicable on trees of roughly the same number of nodes: obviously, the computed distance of two words can achieve a high value when one word is much more common in language than the other (and is thusly represented by a much larger tree). This is true even when these two words are actually synonyms of each other, or just two different lexical representations of the *same* word, like "foto" and "photo". In fact, because the co-occurence of different lexical representations of the same word in the same text is quite seldom, is is very likely that in these situations a high distance will show.

   It can be reasoned that these difficulties will prominently lead to the use of TSR trees derived from term descriptions rather than initial trees (see 3.2).

4.3 TSR Tree Translation and Corpus Related Features

In some cases, a need for "higher-level" operations will occur, e.g. when two agents cooperate, who use different web taxonomies. Our approach is able to deal with these situations through translation of TSR trees of category labels (this can be interpreted as a simple classification task).

Sometimes, information about TSR tree features of a corpus as a whole is important. In these cases, the individual TSR trees of all items that constitute the respective corpus are merged into one "corpus" TSR tree. Afterwards, the corpus tree can be analyzed using the features described in Section 4.1 and Section 4.2.

5 Preliminary Evaluation Results

For testing, we set up some preliminary experiments[5]:We built a prototype system based on a Tomcat application server[6] that was able to generate TSR trees for lists of input terms and store these

[5]Exhaustive evaluation is the goal of our current research efforts

[6]c.f. http://jakarta.apache.org

trees along with their width, weight and depth features in an SQL database. From this database we extracted the data used in the evaluation process.

We applied each feature explained in Section 4 on a set of words taken from 4 corpora. These corpora were constructed as follows:

The *Basic* corpus: The 100 first terms from a dictionary of "basic english words" like account, brother, crush, building, cry, etc. The *Med* corpus: The 100 first terms from a specialized medical dictionary. The *Comp* corpus: The 100 first terms from a specialized dictionary of computer science. The *Top*' corpus: The 100 terms that were ranked as "top 100" by the Wortschatz engine (Uwe Quasthoff, 2000).

We expected terms of the Basic and Top corpora to show high weight and breadth and low depth values. We also expected terms from the Med and Comp corpora to be of high depth but differing in weight and breadth.

These Expectations were supported by our results from generating and comparing the respective corpus TSR trees (see below).

For brevity, we will only present a summary of our findings here.

Single Tree Features Comparing the outcome of applying single TSR tree features onto the four corpora showed some interesting results:

1. **Tree Weight.** Terms from the *Med* corpus are often not represented within the web directory which means that a TSR tree cannot be built for these terms. In general, terms from the *Med* corpus have a very low tree weight value (in most cases < 10). Strangely, some words such as "by", "and", "because" etc. from the *Top* corpus also have low ratings. Examining the actual web directory pages exhibits that these terms seldom contributed to a web pages semantic context and thusly were seldom represented in the web directory. It appears that all input terms were interpreted by the ODP search engine as being semantically relevant, e.g. the word "about" only generated hits in categories about movie titles, e.g. `Arts:Movies:Titles:A:About a Boy`, `Arts:Movies:Titles:A:About Adam`, etc.

   This strongly indicates that the input to the algorithm should be a noun or a common noun phrase.

   Terms from the *Basic* corpus and the *Comp* corpus are rated comparably high, e.g. some common words from the Basic corpus such as "air", "animal", etc. were assigned very high weight values (weight > 100).

2. **Generality.** The generality values listing exhibits that indeed mostly general terms are identified by this feature. Surprisingly, some terms such as "software" and "design" were also attributed high generality. Further investigation shows that "generality" is a context dependent feature, e.g. the term "software" is very general for the computer domain. Only at the first tree level, a domain independent generality factor can be attributed to this feature.

   We also found that **pruning** has its greatest effect on this feature; this leads to the conclusion that the generality feature should be applied on TSR trees that are not pruned according to some threshold.

3. **Domain Dependency.** Except a very few cases, all top rated terms are in the *Comp* or in the *Med* corpus i.e. the two specialized corpora. These terms are apparently more specific in context than the lower rating terms.

Advanced Tree Features Even though we tested the Multi Tree features on only a few test cases (about 30), we are confident that future evaluation will confirm our preliminary results.

1. **Difference.** Computing the difference of two or three single TSR trees turned out to be less informative than the distance value between these trees but a small number of experiments lead us to the conclusion that TSR trees of large text fragments can be compared by difference features with a conclusive outcome.

2. **Distance.** Using node labels and weights for comparison in any case resulted in a 100% distance. This effect derived from the fact that even though some trees were similar in structure, their respective weights differed in every case. The distance feature therefore is applicable to node labels only or has to introduce arithmetical means for adjusting weights. After correcting the distance algorithm, it worked as expected on trees with about the same node number (High distance between e.g. "blood" and "air", low distance between "account" and "credit"). We also achieved reasonable results on trees differing in node number when applying a methodology of filtering homonymous aspects of the respective larger TSR tree (i.e. by using the node number of the smaller tree as upper bound and filtering first level tree nodes).

Nonetheless we did not yet manage to find an *absolute* numerical expression that describes the distance feature appropriately.

TSR Tree Translation and Corpus Features

1. **Corpus Tree Features.**

   We have merged all of the terms of each respective corpus in order to generate a "corpus representation tree". These corpus representations can be used to demonstrate certain properties of the chosen corpora. Our experiments exhibit that terms from the "general" corpora (*Basic* and *Top*) had a higher generality value than terms from the more specialized corpora (*Comp* and *Med*). The same results also confirm our hypothesis of the WWW occurrence property of the computer corpus, since it is also well represented in the web dictionary.

6 Conclusions

In this paper, we have introduced a novel concept of representing text senses as defined in Section 1.

According to the results of our preliminary evaluation, our approach shows the following advantages:

TSR trees can be used to unambiguously represent text senses. There is a fundamental semantic relationship between TSR trees and their respective input terms. Their use is *efficient*: TSR trees can – once retrieved and computed – be re-used without necessary modifications. In that sense they can be used "stand-alone". Application of TSR tree features is very fast (one SELECT SQL statement within our prototype system). Meaning representation within TSR trees is *robust*: generating trees of text fragments[7] by merging the TSR trees of its constituents reduces potential errors.

TSR trees are in close interaction with the semantic context of the input terms, it is therefore possible to determine topical relationships between textual fragments.

Nonetheless, our findings also exhibit some weaknesses and dependencies:

If an input term cannot be found within the web directory in use, a corresponding initial TSR tree cannot be built. This is a big problem for languages that are not well represented in the web directory (there is a strong bias towards the english language). Very specialized domains (e.g. medical topics) are also underrepresented in the web directory and hence problematic for the same reason. Observations show that there is a strong bias towards computer and business related topics. One approach to solving these problems would be to use derived TSR trees in place of directly acquired TSR trees. It is yet a matter of current research to which degree intial TSR trees should be substituted by derived TSR trees.

TSR tree usage usually depends on the output quality of a number of preprocessing steps, e.g. language identification, noun phrase identification, morphological analysis, etc.

[7]e.g. sentences, paragraphs or static size text windows

7 Future Work

We will continue research on the TSR tree topic. In particular, we will investigate the relationship between derived and initial TSR trees and in turn we will find a more appropriate "distance" feature. We are also evaluating a new feature based on comparing tree labels.

We will also thoroughly evaluate our approach against application based testing methodologies, e.g. on text classification. We will also implement a number of example applications in the fields of text classification and text summarization.

References

Adam Kilgariff. 1997. I don't believe in word senses. *Computers and the Humanities*, 31 (2):91–113.

Alexander Budanitsky and Graeme Hirst. 2001. Semantic Distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, second meeting of the North American Chapter of the Association for Computational Linguists*, Pittsburgh.

James Allen. 1995. *Natural Language Understanding*. Benjaming/Cummings Publish. Corp, CA, 2 edition.

D. S. Hirschberg. 1997. Serial computations of Levenshtein distances. In A. Apostolico and Z. Galil, editors, *Pattern matching algorithms*, pages 123–141. Oxford University Press.

Diana Zaiu Inkpen and Graeme Hirst. 2003. Automatic sense disambiguation of the near-synonyms in a dictionary entry. In *Proceedings of the 4th Conference on Intelligent Text Processing and Computational Linguistics*, pages 258–267, Mexico City.

Ellen Riloff and Jay Shoen. 1995. Automatically Acquiring Conceptual Patterns Without an Annotated Corpus. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 148–161.

Fabrizio Sebastiani. 2003. Text categorization. In Alessandro Zanasi, editor, *Text Mining and its*

Applications. WIT Press, Southampton, UK. Invited chapter. Forthcoming.

Giuseppe Attardi, Sergio Di Marco, David F. Salvi, and Fabrizio Sebastiani. 1998. Categorization by context. In David Schwartz, Monica Divitini, and Terje Brasethvik, editors, *Proceedings of IIIS-98, First International Workshop on Innovative Internet Information Systems*, pages 1–13, Pisa, IT.

Google, Inc. 2004. Google. http://www.google.com.

K. Dahlgren, J. McDowell, and E. Stabler. 1989. Knoweldge representation for commonsense reasoning with text. *Computational Linguistics*, 15(3).

Kavi Mahesh and Sergei Nirenburg. 1996. Meaning representation for knowpedge sharing in practical machine translation. In *Florida Artificial Intelligence Research Symposium, FLAIRSÂ96, Special Track on Information Interchange*, pages 19–22, Key West, FL.

Kavi Mahesh. 1996. Ontology development for machine translation: Ideology and methodology. Technical Report MCCSÂ96Â292, Computing Research Laboratory New Mexico State University.

Netscape Inc. 2004. Open directory project. http://dmoz.org.

O. Lassila and R. Swick. 1999. *Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium.

P. Clark, P. Harrison, and J. Thompson. 2003. A Knowledge-Driven Approach to Text Meaning Processing. In *Proceedings of the HLT Workshop on Text Meaning Processing*, pages 1–6. ACL Press.

R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue. 1995. Survey of the state of the art in human language technology. http://www.coli.uni-sb.de/ hansu/publ.html.

Uwe Quasthoff. 2000. Deutscher wortschatz online. http://wortschatz.uni-leipzig.de.

Yahoo Inc. 2004. Yahoo. http://www.yahoo.com.