Natural Language Engineering **5** (2): 147–156. Printed in the United Kingdom
© 1999 Cambridge University Press

A corpus-based bootstrapping algorithm for Semi-Automated semantic lexicon construction†

ELLEN RILOFF and JESSICA SHEPHERD
Department of Computer Science, University of Utah,
Salt Lake City, UT 84112, USA
e-mail: riloff@cs.utah.edu

(*Received 5 November 1998; revised 15 April 1999*)

Abstract

Many applications need a lexicon that represents semantic information but acquiring lexical information is time consuming. We present a corpus-based bootstrapping algorithm that assists users in creating domain-specific semantic lexicons quickly. Our algorithm uses a representative text corpus for the domain and a small set of 'seed words' that belong to a semantic class of interest. The algorithm hypothesizes new words that are also likely to belong to the semantic class because they occur in the same contexts as the seed words. The best hypotheses are added to the seed word list dynamically, and the process iterates in a bootstrapping fashion. When the bootstrapping process halts, a ranked list of hypothesized category words is presented to a user for review. We used this algorithm to generate a semantic lexicon for eleven semantic classes associated with the MUC-4 terrorism domain.

1 Introduction

Natural language understanding requires both syntactic and semantic knowledge, yet there are surprisingly few resources available for lexical semantic information. In contrast, a variety of dictionaries and computational tools are available for acquiring syntactic information (e.g. Brill 1994, Church 1989, Marcus, Santorini and Marcinkiewicz 1993, Weischedel, Meteer, Schwartz, Ramshaw and Palmucci 1993). Ideally, one would like to have a semantic knowledge base that contains semantic representations of all words, phrases and concepts in the language. Given the vast scope of human knowledge and the practical limitations of manual knowledge engineering, it is unrealistic to expect a complete semantic knowledge base any time soon. Nevertheless, there have been two noteworthy efforts to build general-purpose semantic knowledge bases, WordNet (Miller 1990) and Cyc (Lenat, Prakash and Shepherd 1986).

† This research is supported in part by the National Science Foundation under grants IRI-9509820 and IRI-9704240.

While general-purpose semantic information may be sufficient for some tasks, it is unlikely to be sufficient for domain-specific applications. For example, consider a Natural Language Processing (NLP) application for extracting information from medical texts. A medical information extraction system needs a lexicon of syntactic and semantic information about medical terms and concepts. A general-purpose semantic lexicon will not contain the specialized medical jargon and terminology that are required for medical information processing. Even within the context of a single domain, different applications may require different levels of specialization. For example, an NLP system for cardiology texts would probably need a different dictionary than an NLP system for podiatry texts.

One possible solution is to develop automated methods for generating domain-specific semantic lexicons, which can be used to replace or supplement broad-coverage dictionaries. We define a domain-specific lexicon as a dictionary that contains lexical information pertaining to a specific subject matter. For example, a domain-specific lexicon for cardiology would contain the words, phrases, and concepts that are most important for understanding cardiology texts.

Creating a domain-specific lexicon has several benefits. First, by definition, a domain-specific lexicon contains the specialized terminology that is required for in-depth understanding of the subject matter. Second, many ambiguity problems in natural language can be simplified by taking advantage of the limited domain. For example, the word ‘monitor’ has several noun word senses, including one that refers to a computer screen and one that refers to a lizard. Within the context of a specific domain, one of the word senses will dominate. The noun ‘monitor’ usually refers to a computer screen in computer science texts, but usually refers to a lizard in zoology texts. If an NLP system is designed for a limited domain, including only relevant word senses in the lexicon can substantially reduce ambiguity resolution problems.

Many NLP systems do rely on domain-specific lexicons, but these dictionaries are usually constructed by hand. Manual dictionary construction is time consuming and prone to errors of omission. To address these problems, we have adopted a semi-automated approach to semantic lexicon construction. Given a few sample words that belong to a semantic class, our algorithm automatically hypothesizes new words that are also likely to belong to the semantic class. The most confident predictions are added to the lexicon automatically and the process repeats in a bootstrapping fashion. When automatic bootstrapping is finished, a human reviews the list of hypothesized category members and decides which ones to add to the lexicon. This corpus-based approach has several advantages over manual dictionary construction: a semantic lexicon can be built very quickly, the lexicon will be tailored for the domain represented by the text corpus, and important domain words are less likely to be omitted from the dictionary.

First, we present the corpus-based bootstrapping algorithm that automatically generates candidate words for a semantic category. Next, we present an experiment to evaluate the effectiveness of this algorithm empirically by generating a semantic lexicon for eleven semantic classes associated with terrorism. Finally, we compare our approach to related work and summarize our conclusions.

2 A corpus-based bootstrapping algorithm

The goal of our approach is to begin with just a handful of *seed words* that are known to belong to a semantic class, and then leverage those seed words to find new words that also belong to the semantic class. Our algorithm is based on the observation that members of a semantic category often appear near other members of the category in natural language text. Four common syntactic constructions often group together members of the same semantic class:

Conjunctions	*lions and tigers and bears*
Lists	*lions, tigers, bears*
Appositives	*the horse, a black stallion*
Compound Nouns	*tuna fish* ; *oak tree*

These syntactic constructions will not always contain members of the same semantic class, but they have a tendency to do so. Conjunctions and lists often group together similar items. Appositives and compound nouns frequently represent subclass/superclass relationships. For example, the appositive phrase "the horse, a black stallion" is a superclass followed by its subclass, and the phrase "the stallion, an impressive horse" is a subclass followed by its superclass. Compound nouns usually have the superclass as head noun with the subclass as a modifier (e.g. "tuna fish").

Since these four types of syntactic constructions are very common in natural language text, their affinity for semantically similar words can be exploited by a corpus-based bootstrapping approach. The general idea of our algorithm is to begin with a small number of known category words and then identify other words that are collocated near the known category words with unusual regularity.

2.1 Overview of the algorithm

The input to our algorithm is a text corpus that is representative of the domain, and a small set of seed words for the semantic category of interest. The output is a list of new words that are hypothesized to belong to the same semantic class, in ranked order based upon their strength of association with the class.

During the first iteration, the algorithm generates a ranked list of new words that are hypothesized to belong to the same semantic category as the seed words. The top N words in the ranked list are assumed to be correct, and are dynamically added to the seed word list as new seed words. The enhanced seed word list represents a larger context in which to look for additional category members during the next iteration. The process repeats for a fixed number of iterations, or until no new seed words can be found. When the bootstrapping process halts, the final ranked list of hypothesized category members is reviewed by a person to confirm (or disconfirm) whether each word should be added to the lexicon.

2.2 Bootstrapping algorithm

The general idea behind our approach is motivated by Yarowsky's (1992) word sense disambiguation algorithm, and the notion of statistical salience. Our algorithm uses a somewhat different statistical measure, but it is based on the same general principle of looking for words that are more frequent within a category context than in the corpus as a whole. The bootstrapping algorithm for hypothesizing words that belong to a semantic category can be broken down into five steps.

1. Identify all sentences in the corpus that contain a seed word. Run each sentence though a parser to segment the sentence into simple noun phrases, verb phrases, and prepositional phrases. (A partial parser is sufficient.)
2. Collect a narrow context window around each seed word that occurs as a head noun in a noun phrase. Restricting the seed words to be head nouns ensures that the seed word is the main concept of the noun phrase. Also, this reduces the chance of finding different word senses of the seed word (although multiple noun senses can still be a problem). The context window consists of only two words: the closest noun to the left of the word, and the closest noun to the right of the word. Only nouns are collected under the assumption that most, if not all, true category members should be nouns. The context windows do not cross sentence boundaries. Note that our context window is much narrower than those used by other researchers (e.g. (Yarowsky 1992)). We experimented with larger context windows and found that the narrow window more consistently includes words of the same semantic class, presumably because it focuses on local syntactic constructions. For simplicity, we will refer to the set of nouns gathered during this step as the *category context* for a semantic class.
3. Compute a *category score* for each word in the category context. The category score of a word W for category C is defined as:

   $$Score(W,C) = \frac{frequency\ of\ W\ in\ C's\ context\ windows}{frequency\ of\ W\ in\ corpus}$$

   This score is essentially the conditional probability that the word appears in a category context. Note that this is not exactly a conditional probability because a single word occurrence can belong to more than one context window. For example, consider the sentence: *They seized one M-16, a rifle, and a .45-caliber pistol.* The word *rifle* is in the context windows for both *M-16* and *pistol* even though there is just one occurrence of it in the sentence. Consequently, the category score for a word can be greater than 1.
4. Remove stopwords, numbers and words with a total frequency ≤ 5 (these words are assumed to be statistically unreliable). We used a stopword list containing about 30 general nouns, mostly pronouns (e.g. *I, he, she, they*) and determiners (e.g. *this, that, those*). Finally, sort the remaining nouns by category score so that the nouns most strongly associated with the category appear at the top.
5. Add the top five nouns (that are not already seed words) to the seed word list, go back to Step 1, and repeat the process. The process halts after a fixed

number of iterations or if no new seed words are found. This bootstrapping mechanism dynamically grows the seed word list so that each iteration produces a larger category context. In our experiments, the top five nouns were added automatically without any human intervention, but this sometimes allows non-category words to dilute the seed word list. A few inappropriate words are not likely to have much impact, but many inappropriate words or a few high frequency words can weaken the feedback process. One alternative is to have a person verify that each word belongs to the target category before adding it to the seed word list, but this would require human intervention at each iteration of the feedback cycle. We chose to avoid this additional human interaction.

3 Experimental results

To determine the effectiveness of our bootstrapping algorithm, we built a semantic lexicon for a real application. Our experiments focused on the domain of Latin American terrorism, which was the topic of the information extraction task for the Fourth Message Understanding Conference (MUC-4) (MUC-4 Proceedings 1992). We used the 1700 texts in the MUC-4 data set as our text corpus, and chose eleven semantic classes representing items that needed to be extracted for the MUC-4 task: *building*, *civilian*, *energy*, *financial*, *government official*, *location*, *military*, *terrorist*, *time*, *vehicle*, *weapon*. Some of these classes represent people who are common perpetrators or victims of terrorism: *civilian*, *government official*, *military*, *terrorist*. A few of the classes represent objects that are frequent targets of terrorism: *building*, *energy*, *financial*, *vehicle*. The other classes represents dates (*time*), locations (*location*), and types of weapons (*weapon*).

As input to our algorithm, we also need a set of seed words for each semantic class. We used two general criteria as guidance when selecting seed words:

1. The word should be frequent in the domain. This is necessary to ensure that there will be many occurrences of the word in the corpus.
2. The word should be (relatively) unambiguous. This minimizes the risk of finding inappropriate contexts around the word.

Using this criteria, we defined five seed words for each of the eleven semantic classes. They are shown in Table 1. We do not claim that this is the best possible set of seed words, but they worked well in our experiments. More experimentation is needed to determine the effect of the initial seed words on the final output. We experimented with different numbers of seed words and found that using five initial seed words worked about as well as using more.

We ran the bootstrapping algorithm for eight iterations, adding five new words to the seed word list after each iteration. When the bootstrapping process finished, we had a ranked list of hypothesized words for each of the eleven semantic classes. The bootstrapping algorithm found many new members of each class, although the quality of the ranked lists varied depending on the semantic class. In the next section, we evaluate the results of this experiment quantitatively.

Table 1. *Seed word lists for the terrorism domain*

Building:	*building buildings hotel homes office*
Civilian:	*civilians peasants businessmen passengers residents*
Energy:	*fuel gas gasoline oil power*
Financial:	*bank banking currency dollar money*
Gov't Official:	*governor mayor ambassador president senator*
Location:	*city town region district neighborhood*
Military:	*army commander infantry soldier troop*
Terrorist:	*terrorists terrorist guerrillas guerrilla rebels*
Time:	*hour day week month year*
Vehicle:	*airplane car jeep plane truck*
Weapon:	*bomb dynamite explosives gun rifle*

3.1 Evaluating the results

To evaluate the effectiveness of the bootstrapping algorithm, we manually reviewed the top 500 words on the ranked lists for each semantic class. We judged a word to belong to a semantic category if:

1. The word is a member of the category. For example, *TNT* and *rifle* are members of the weapon category. Or,
2. The word refers to a part of a member of the category. For example, *cartridge* and *clips* are parts of a weapon. The rationale behind judging subparts to be category members is that the subpart belongs to the same semantic class as the whole object.

The manual review process took about 30 minutes for each semantic class, except for a few categories that required dictionary lookup (e.g. the human reviewer did not recognize many of the weapon names). In general, however, reviewing the ranked lists is quite fast for someone who has knowledge of the domain. Figure 1 shows the words that were judged to be legitimate category members for the building, civilian, and weapon classes.

When the manual review process was finished, the semantic lexicon for terrorism contained 494 words. Since 500 words were reviewed for each of the eleven classes (except the financial and weapon classes which produced fewer than 500 words), this result implies that only about 10% of the words on the ranked lists were members of their respective semantic categories. This viewpoint is misleading though, because we should only expect the most highly ranked items to be true category members. The density of true category members should decrease as one progresses down the list.

We reviewed 500 words for each category because we did not know how quickly the density of true category members would drop off, or how far down the lists we would need to go. We answered these questions empirically by analyzing the density of true category members at various points in the ranked lists. Table 2 shows the results of this analysis. We walked down each ranked list and measured the percentage of true category members after each set of 50 words. The actual number of true category members appears in parentheses. For example, Table 2 shows that

Building: ministry hospital garrison house bank prison embassy home stores offices palace schools doors residence gas_station room houses tower ministries establishments hospitals housing registry entrance restaurant entry gate walls office buildings building windows floor homes mansions hotel apartment properties Sheraton

Civilian: Salvadoran Salvadorans jesuits person police Colombian priests personnel students Msgr Panamanian citizens people advisers families civilian judges body persons brothers Ecuadoran family peoples employees peasant journalists woman men owners victims crew policeman bishops witnesses bodies policemen foreigners newsmen women nuns crowd Americans activists children clergymen faces manager workers friend child corpses leg residents professionals farmers brother Nicaraguans industrialists lady Indian survivors businessmen passengers civilians peasants intellectuals pedestrians drivers

Weapon: rockets bombs car_bomb missile missiles tanks arms bullets rocket bullet weapons car_bombs artillery firearms guns machinegun pistol cannon submachinegun gun bomb mortars explosives ammunition submachineguns cartridges pistols fuse machineguns grenades rifles dynamite AR-15 M-60 clips AK-47 M-16 rifle cartridge mortar grenade TNT M-79

Fig. 1. The semantic lexicon for three categories.

13 of the top 50 hypothesized building words (26%) were actually buildings. Only four additional building words were found among the next 50 words, so the density of true category members drops to 17% after reviewing the top 100 words.

Table 2 confirms our hypothesis that the density of true category members is highest at the top, which suggests that our scoring metric is doing a good job of promoting the most likely category members. The effectiveness of the ranking scheme yields a law of diminishing returns: the payoff of finding additional category members steadily decreases as you walk down the list.

Of the eleven semantic classes in our experiment, none of them added many new category members after the 400th item. And only a few categories were still adding new members after the 300th item, which suggests that reviewing only the top 300 words is probably sufficient. Table 2 also suggests a mechanism for automatically deciding when to stop the manual review process: if the number of new category members does not change much after reviewing a fixed number of words, then it is probably safe to assume that there are not many new category members lying ahead. For example, 43 government officials were found among the top 150 words, but no new government officials were found thereafter. If we had noticed this trend, we could have stopped the review process after reviewing 200 words.

Another interesting aspect of our results is the varying levels of effectiveness for different semantic classes. The location class performed best: the bootstrapping algorithm found 164 locations, with a 74% hit rate for the top 50 words and nearly a 50% hit rate even after 250 words. The energy class was the most problematic: the bootstrapping algorithm found only six energy words. The MUC-4 texts primarily describe terrorist and military incidents, so energy sources are mentioned only when they are the target of an attack. The corpus does contain a few descriptions of attacks

Table 2. *Density of category members in the ranked lists*

	Building	Civilian	Energy	Financial	Govt	Location
After 50	0.26 (13)	0.36 (18)	0.12 (6)	0.28 (14)	0.38 (19)	0.74 (37)
After 100	0.17 (17)	0.22 (22)	0.06 (6)	0.19 (19)	0.35 (35)	0.57 (57)
After 150	0.15 (22)	0.18 (27)	0.04 (6)	0.15 (22)	0.29 (43)	0.51 (77)
After 200	0.13 (26)	0.19 (38)	0.03 (6)	0.11 (22)	0.21 (43)	0.50 (100)
After 250	0.11 (28)	0.17 (42)	0.02 (6)	0.09 (23)	0.17 (43)	0.46 (114)
After 300	0.11 (32)	0.17 (51)	0.02 (6)	0.08 (23)	0.14 (43)	0.43 (129)
After 350	0.10 (36)	0.16 (55)	0.02 (6)	0.07 (23)	0.12 (43)	0.40 (139)
After 400	0.09 (37)	0.14 (58)	0.01 (6)	0.06 (23)	0.11 (43)	0.37 (150)
After 450	0.09 (39)	0.14 (62)	0.01 (6)	–(–)	0.10 (43)	0.35 (158)
After 500	0.08 (39)	0.14 (68)	0.01 (6)	–(–)	0.09 (43)	0.33 (164)

	Military	Terrorist	Time	Vehicle	Weapon
After 50	0.28 (14)	0.34 (17)	0.06 (3)	0.30 (15)	0.58 (29)
After 100	0.19 (19)	0.23 (23)	0.05 (5)	0.19 (19)	0.34 (34)
After 150	0.17 (26)	0.18 (27)	0.04 (6)	0.15 (22)	0.25 (38)
After 200	0.15 (30)	0.17 (35)	0.04 (8)	0.13 (26)	0.20 (40)
After 250	0.14 (34)	0.15 (38)	0.03 (8)	0.10 (26)	0.16 (40)
After 300	0.13 (40)	0.13 (39)	0.05 (14)	0.09 (27)	0.14 (43)
After 350	0.11 (40)	0.12 (43)	0.05 (19)	0.08 (28)	0.12 (43)
After 400	0.10 (41)	0.11 (45)	0.06 (23)	0.07 (29)	0.11 (43)
After 450	0.09 (42)	0.10 (47)	0.05 (23)	0.06 (29)	0.10 (43)
After 500	0.09 (43)	0.10 (48)	0.05 (24)	0.06 (29)	–(–)

on oil pipelines and electricity substations, but it is doubtful that a wide variety of energy words are present in the corpus. Therefore the energy category illustrates an important point: it is probably overkill to use this algorithm for a semantic class that will not be well-represented in the domain. For most of the eleven semantic classes, however, a substantial number of new category members were found. Recently, we used the terrorism semantic lexicon as part of another algorithm to generate conceptual case frames automatically (Riloff and Schmelzenbach 1998). The semantic lexicon allows the algorithm to infer the conceptual roles and semantic constraints for case frame slots.

4 Related work

Semantic lexicons are built by hand for most NLP applications, but several techniques have been developed to learn lexical semantic information automatically. Most of these methods learn the meanings of an unknown word by using contextual expectations from the definitions of surrounding words (e.g. Granger 1977, Carbonell 1979, Jacobs and Zernik 1988, Cardie 1993, Hastings and Lytinen 1994). An alternative approach is to derive knowledge automatically from on-line dictionaries (Dolan, Vanderwende and Richardson 1993). All of these approaches rely on an existing dictionary or knowledge base as a starting point. Our system is the first aimed at building semantic lexicons from scratch using only a representative text corpus and a handful of predefined seed words. The only additional knowledge used by our system is a part-of-speech dictionary for syntactic segmentation. We used a hand-crafted part-of-speech dictionary for these experiments, but statistical and corpus-based taggers are widely available (e.g. Brill 1994, Church 1989, Weischedel 1993).

One other relevant piece of related research is Roark and Charniak's work (Roark and Charniak 1998), which improves upon preliminary results that we reported in (Riloff and Shepherd 1997). Roark and Charniak confirmed our intuition that conjunctions, appositives, lists, and compound nouns can help identify items of the same semantic class by adapting our algorithm to look exactly for those syntactic constructions.

5 Conclusions

Semantic lexicons are essential for many NLP applications, but creating lexical resources by hand is extremely time consuming. Furthermore, hand-built lexicons are often incomplete because humans can easily overlook words that are important for the domain. The corpus-based bootstrapping algorithm that we presented can assist humans in building semantic lexicons quickly and in providing assurance that important words are not overlooked. This algorithm requires no pre-existing semantic knowledge or specialized resources. To use this technique for a new application domain, the user only needs to supply a representative text corpus and a handful of seed words for each semantic category of interest. The corpus-based bootstrapping algorithm illustrates how text corpora can be exploited to acquire semantic information semi-automatically, without the need for special resources.

References

Brill, E. (1994) Some Advances in Rule-based Part of Speech Tagging. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 722–727. AAAI Press/The MIT Press.

Carbonell, J. G. (1979) Towards a Self-Extending Parser. *Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics*, 3–7.

Cardie, C. (1993) A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 798–803. AAAI Press/The MIT Press.

Church, K. (1989) A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *Proceedings of the Second Conference on Applied Natural Language Processing*.

Dolan, W., Vanderwende, L. and Richardson, S. (1993) Automatically Deriving Structured Knowledge Bases from On-Line Dictionaries. *Proceedings of the First Conference of the Pacific Association for Computational Linguistics*, 5–14.

Granger, R. H. (1977) FOUL-UP: A Program that Figures Out Meanings of Words from Context. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 172–178.

Hastings, P. and Lytinen, S. (1994) The Ups and Downs of Lexical Acquisition. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 754–759. AAAI Press/The MIT Press.

Jacobs, P., and Zernik, U. (1988) Acquiring Lexical Knowledge from Text: A Case Study. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 739–744.

Lenat, D. B., Prakash, M. and Shepherd, M. (11986) CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge-Acquisition Bottlenecks. *AI Magazine* **6**: 65–85.

Marcus, M., Santorini, B. and Marcinkiewicz, M. (1993) Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* **19**(2): 313–330.

Miller, G. (1990) Wordnet: An On-line Lexical Database. *Int. J. Lexicography* **3**(4).

MUC-4 Proceedings. (1992) *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. San Mateo, CA: Morgan Kaufmann.

Riloff, E. and Schmelzenbach, M. (1998) An Empirical Approach to Conceptual Case Frame Acquisition. *Proceedings of the Sixth Workshop on Very Large Corpora.*

Riloff, E. and Shepherd, J. (1997) A Corpus-Based Approach for Building Semantic Lexicons. *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, 117–124.

Roark, B. and Charniak, E. (1998) Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, 1110–1116.

Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L. and Palmucci, J. (1993) Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics* **19**(2): 359–382.

Yarowsky, D. (1992) Word sense disambiguation using statistical models of Roget's categories trained on large corpora. *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*, 454–460.