A Semantics-Enhanced Language Model for Unsupervised Word Sense Disambiguation

Shou-de Lin[1] and Karin Verspoor[2]

[1] National Taiwan University
sdlin@csie.ntu.edu.tw
[2] Los Alamos National Laboratory
verspoor@lanl.gov

Abstract. An N-gram language model aims at capturing statistical word order dependency information from corpora. Although the concept of language models has been applied extensively to handle a variety of NLP problems with reasonable success, the standard model does not incorporate semantic information, and consequently limits its applicability to semantic problems such as word sense disambiguation. We propose a framework that integrates semantic information into the language model schema, allowing a system to exploit both syntactic and semantic information to address NLP problems. Furthermore, acknowledging the limited availability of semantically annotated data, we discuss how the proposed model can be learned without annotated training examples. Finally, we report on a case study showing how the semantics-enhanced language model can be applied to unsupervised word sense disambiguation with promising results.

1 Introduction

Syntax and semantics both play an important role in language use. Syntax refers to the grammatical structure of a language whereas semantics refers to the meaning of the symbols arranged with that structure. To fully comprehend a language, a human must understand its syntactic structure, the meaning each symbol represents, and the interaction between the two. In most languages, syntactic structure conveys something about the semantics of the symbols, and the semantics of symbols may constrain valid syntactic realizations. As a simple example: when we see a noun following a number in English (e.g. "one book"), we can infer that the noun is countable. Conversely, if it is known that a noun is countable, a speaker of English knows that it can plausibly be preceded by a numeral. It is therefore reasonable to assume that for a computer system to successfully process natural language, it has to be equipped with capabilities to represent and utilize both the syntactic and semantic information of the language simultaneously.

The n-gram language model (LM) is a powerful and popular framework for capturing the word order information of language, or fundamentally syntactic information. It has been applied successfully to a variety of NLP problems such as machine translation, speech recognition, and optical character recognition.

A. Gelbukh (Ed.): CICLing 2008, LNCS 4919, pp. 287–298, 2008.

As described in equation (1), an n-gram language model utilizes conditional probabilities to capture word order information, and the validity of a sentence can be approximated by the accumulated probability of the successive n-gram probabilities of its constituent words $W_1 \ldots W_k$.

$$Validity(W_1 W_2 \ldots W_k) = \prod_{i=1}^{k} P(W_i | W_{i-n+1} \ldots W_{i-1}) \quad (1)$$

As powerful as a traditional n-gram LM can be, it does not capture the semantic information of a language. Therefore it has seldom been applied to semantic problems such as word sense disambiguation (WSD). To address this limitation, in this paper we propose to expand the formulation of a LM to include not only the words in the sentences but also their semantic labels (e.g. word senses). By incorporating semantic information into a LM, the framework is applicable to problems such as WSD, semantic role labeling, and even more generally machine translation and information extraction – tasks that require both semantic and syntactic information for an effective solution.

The major advantage of our algorithm compared to conventional unsupervised WSD is that it can perform WSD without need for any sense glosses, sense-similarity measures, or other linguistic information as has been required in many other unsupervised WSD systems. We need only an unannotated corpus plus a sense dictionary for which some senses of different words have been "pooled" together into something like a WordNet synset, as we exploit the redundancy of sense sequences even where the words may differ. Therefore, our approach can be applied in the early stages of sense invention for a language or domain, where only limited lexical semantic resources are available.

2 Incorporating and Learning Semantics in a LM

The first part of this section proposes a semantics-enhanced language model framework while the second part discusses how its parameters can be learned without annotated data.

2.1 A Semantics-Enhanced Language Model

Figure 1(a) is a general finite state representation of a sentence of four words ($W_1 \ldots W_4$) connected through a bigram LM. Each word can be regarded as a state node and the transition probabilities between states can be modeled as the n-gram conditional probabilities of the involved states (here we assume the transition probabilities are bigrams). In fact each word in a sentence has a certain lexical meaning (sense or semantic label, S_i) as represented in Figure 1(c). Conceptually, for each word-based finite state representation there is a dual representation in the semantics (or sense) domain, as shown in 1(b). A Semantic Language Model (or SLM) like 1(b) records the order relations between senses. Alternatively, one can combine both representations into a hybrid language model that captures both the word order information and the word meaning, as demonstrated in 1(d). 1(d) represents a Word-Sense Language Model

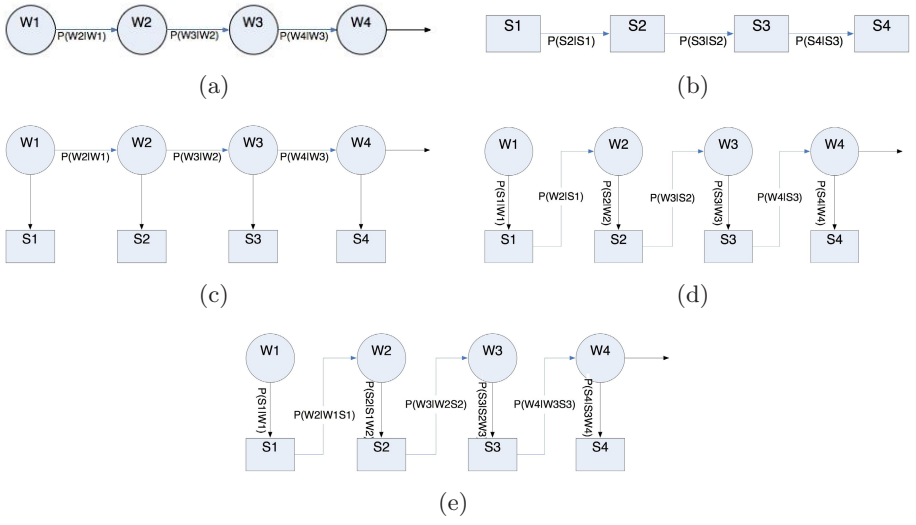

Fig. 1. (a) A standard finite-state, bigram LM representation of a sentence. (b) a Semantic Language Model. (c) each word in the sentence has a certain meaning (or semantic label). (d) a hybrid LM integrating word and sense information (WSLM). (e) like (d) except that a trigram model is used.

(or WSLM), a semantics-enhanced LM incorporating two types of states: word symbols and their semantic labels. The intuition behind WSLM is that when processing a word, people first try to recognize its meaning (i.e. $P(S_n|W_n)$), and based on that predict the next word (i.e. $P(W_{n+1}|S_n)$). Figure 1(e) is the same as 1(d) except that the bigram probabilities are replaced by trigrams. It embodies the concept that the next word to be revealed depends on the previous word together with its semantic label, and the meaning of the current word depends on not only the word itself but the meaning of the previous word.

The major reason for the success of a LM approach to NLP problems is its capability of predicting the validity of a sentence. In 1(a), we can say that a sentence $W_1W_2W_3W_4$ is valid because $P(W_2|W_1) * P(W_3|W_2) * P(W_4|W_3)$ is relatively high. Similarly, given that the semantic labels of each word in the sentence are known, the probabilities $P(S_2|S_1) * P(S_3|S_2) * P(S_4|S_3)$ can be applied to assess the semantic validity of this sentence as well. Furthermore, we can say that a word sequence together with its semantic assignment (interpretation) is valid based on a WSLM if the probability of $P(S_1|W_1) * P(W_2|S_1) * \ldots * P(W_4|S_3) * P(S_4|W_4)$ is high. We can therefore use a semantics-enhanced LM to rank possible interpretations of a word sequence.

2.2 Unsupervised Parameter Learning

The n-gram probabilities of a word-based LM such as the transition probabilities in Figure 1(a) can be easily learned through counting term frequencies and

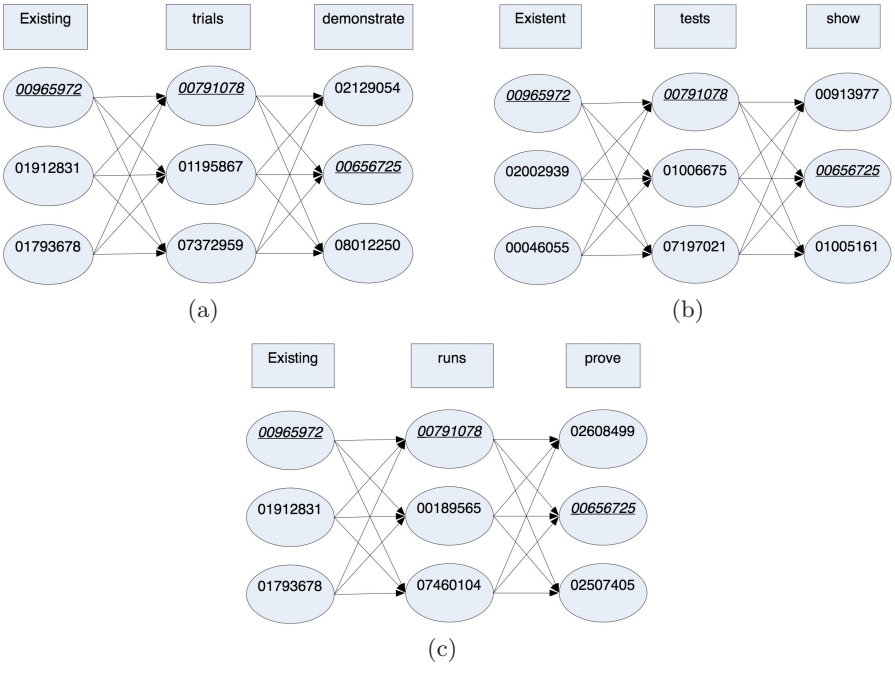

Fig. 2. Sense-based graph of word sequences (a) "Existing trials demonstrate..." (b) "Existent tests show..." (c) "Existing runs prove..."

co-occurrences from large corpora. If there were some large corpora with semantically annotated words and sentences, we could learn the semantics-enhanced LM such as 1(b) and 1(d)-1(e) directly through frequency counting as well. Unfortunately, there is no corpus containing a significant amount of semantically annotated data available. To address this problem, we discuss below an approach that allows the system to approximate the n-gram probabilities of the semantics-enhanced language models. Without loss of generality, in the following discussion we assume the transition probabilities to be learned are all bigrams.

The problem setup is as follows: the system is given a plain text, unannotated corpus together with a dictionary (assuming WordNet 2.1) that contains a list of possible semantic labels for each word. Using these resources alone, the system must learn the n-gram dependencies between semantic labels. Note that every word in the WordNet dictionary has at least one sense (or synset label), and each sense has a unique 8-digit id representing its database location. Different words can share synsets, indicating they have meanings in common. For example, the word *trial* has six senses in the dictionary and one of these (id=00791078) is shared by the word *test* and *run*. The word *demonstrate* has four meanings where one of them (id=00656725) is associated with the words *prove* and *show*. To learn a SLM, one has to learn the conditional probabilities of one sense following the other such as $P(S_k = 00656725|S_{k-1} = 00791078)$.

The first step of learning is to construct a sense-based graph representation for the plain text corpus by connecting all the senses of each word to the senses of the subsequent word. For example, Figure 2(a) is the sense-graph of the phrase "Existing trials demonstrate". For illustration purposes we display only three senses per word in the figure, though there may be more senses for the word defined in WordNet. The weights of the links in the graph, based on the concept of a LM, can be modeled by the n-gram (e.g. bigram) probabilities. If all the bigrams between senses in the graph are known, then for each path of senses (where a path contains one sense per word) we can generate its associated probability, as in equation (2). Note that if a word has no known senses in WordNet (e.g. for closed class words or proper nouns) we assign it a single "dummy" sense.

$$\begin{aligned} &Validity(existing = 00965972, trial = 00791078, demonstrate = 02129054) \\ &= Pr(00965972|start) * Pr(00791078|00965872) * Pr(02129054|00791078) \end{aligned} \tag{2}$$

This probability reflects the cumulative validity of each sense assignment for the sequence of words. One can rank all the sense paths based on their probabilities to find the optimal assignment of senses to words. If the associated probability for each path in the graph is given, we can apply a technique called *fractional counting* to determine bigram probabilities. Fractional counting counts the occurrence of each bigram in all possible paths, where the count is weighted by the associated probability of the path.

Unfortunately, without a sense-annotated corpus neither the sense bigrams nor the path probabilities can be known directly. However, since computing the likelihood for each path and generating the bigram probabilities are dual problems (i.e. one can be generated if the other is known), it is possible to apply the expectation-maximization (EM) algorithm to approximate both numbers [8]. EM is an efficient iterative procedure for computing the Maximum Likelihood (ML) estimate in the presence of missing data. It estimates the model parameters for which the observed data are most likely, using an iteration of two processes: the E-step, in which the missing data are estimated using conditional expectation given the observed data and the current estimate of the model parameters, and the M-step, in which the likelihood function is maximized under the assumption that the missing data are known (using the estimate of the missing data from the E-step).

To perform the EM learning, the first step is to initialize the probabilities of the bigrams. As will be shown in our case study, the initialization can be uniformly distributed or use certain preexisting knowledge. In the Expectation stage (E-step) of the EM algorithm, the system uses the existing bigram probabilities to generate the associated probability of each path, such as the one shown in equation 2. In the maximization stage (M-step) the system applies fractional counting to refine the bigram probabilities. It is guaranteed that the refined bigram can produce a higher probability for the observed data. The E-step and M-step continue to iterate until a local optimum is reached.

One potential problem for this approach is efficiency. The total number of paths in the graph grows exponentially with the number of words (i.e. $O(b^n)$, where n is the number of words and b is the average branching factor of nodes, i.e. the average number of senses per word). Therefore it is computationally prohibitive for the system to enumerate all paths and produce their associated probabilities one by one to perform fractional counting. Fortunately in this situation one can apply a polynomial algorithm called Baum-Welch (or *forward-backward*) algorithm for fractional counting [2]. Rather than generating all paths with their probabilities in the graph, we need to know only the total probability of all the paths that a link (bigram) occurs in. This can be generated by recording dynamically for each link the accumulated probabilities from the beginning of the graph (the alpha value) to the link and the accumulated probabilities from the link to the end (the beta value). Since in our case the alpha and beta values are independent, it is possible to generate all n-grams with polynomial time $O(nb^2)$ and space $O(nb)$. A similar approach has been applied successfully to unsupervised NLP problems such as tagging, decipherment, and machine translation ([7], [12], [13], [14]).

The simple example shown in Figure (2) describes the intuition behind the method. Imagine the system encounters the phrases "Existing trials demonstrate", "Existent tests show", "Existing runs prove" in the corpus. According to Figure (2) there is one common sense 00965972 for the words *existing* and *existent*, a single common sense 00791078 for *trial*, *test*, and *run*, and a common sense 00656725 for the words *demonstrate*, *show*, and *prove*. Based on the minimum description length principle (or Occams Razor), a reasonable hypothesis is that these three senses should have higher chance to be the right assignments (and thus should appear successively more often) compared with the other candidates, since one can then use only three senses to "explain" all the sentences.

The proposed learning algorithm captures the spirit of this idea. Assuming equal probabilities are used to initialize the bigrams and assuming all senses listed in Figure (2) do not appear elsewhere in the corpus, then after the first iteration of EM, 00791078 will have a higher chance to follow 00965972 compared with others (e.g. equation (3)). This is because the system sees 00791078 following 00965972 more times than others in the fractional counting stage.

$$Pr(00791078|00965972) > Pr(00189565|00965972) \tag{3}$$

This approach works because there are situations in which multiple words can be used to express a given meaning, and people tend not to choose the same word repeatedly. The system can take advantage of this to learn information about senses that tend to go together from the shared senses of these varied words, as formalized in the semantics-enhanced LM.

The same approach can be applied to learn the parameters in a WSLM. The only difference is that the words are included in the graph as single-sense nodes. Figure 3 is the graph presentation of a WSLM.

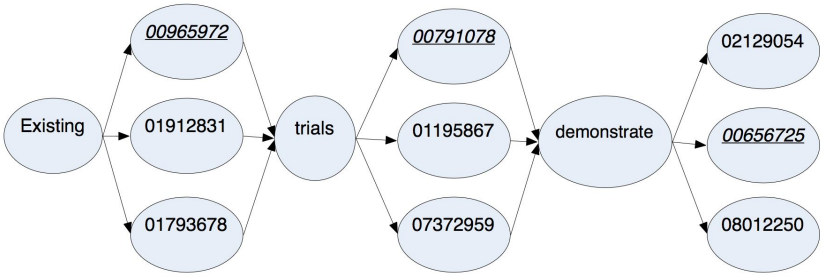

Fig. 3. The graph generated for the WSLM. Such a network has the format word1→sense1→ word2→sense2→...

3 Unsupervised WSD Using SLM and WSLM

We describe a case study on applying the SLM and WSLM to perform an all-words word sense disambiguation task. Since both language models are trained without sense-annotated data, this task is an unsupervised WSD task.

3.1 Background

Unsupervised WSD aims at determining the senses of words in a text without using a sense-annotated corpus for training. The methods employed generally fall into two categories, one for all-words, token-based WSD (i.e. assign each token a sense in its individual sentential context) and the other to find the most frequent sense of each unique token in the text as a whole (following a one sense per discourse assumption). The motivation to focus on the second type of task is that assigning the most frequent sense to every word turns out to be a simple heuristic that outperforms most approaches [11]. The following paragraphs describe the existing unsupervised WSD methods.

Banerjee and Pedersen proposed a method that exploits the concept of gloss overlap for WSD [1], where a gloss is a sentence in WordNet that characterizes the meaning of a sense (synset). It assumes the sense whose gloss definition looks most similar (overlaps strongly) with the glosses of surrounding content words is the correct one. Mihalcea's graph-based algorithm [16] first constructs a weighted sense-based graph , where weights are the similarity between senses (e.g. gloss overlap). Then it applies PageRank to identify prestigious senses as the correct interpretation. Galley and McKeown also propose a graph-based approach called lexical chains that regards a sense to be dominant if it has more strong connections with its context words [9]. The strength of connection is determined by the type of relation as well as the distance between the words in the text. Navigli and Velardi propose a conceptually similar but more knowledge-intensive approach called structural semantic interconnections (SSI) [17]. For each sense, the method first constructs semantic graphs consisting of collocation information (extracted from annotated corpora), WordNet relation information, and domain labels. Using these graphs, the algorithm iteratively chooses senses

with strong connectivity to the relevant senses in the semantic graph. McCarthy *et al* propose a method to determine the most frequent senses for words [15]. In their framework, the distributionally similar neighbors of each word are determined, and a sense of a word is regarded as dominant if it is the most similar to the senses of its distributionally similar neighbors.

Although the above methods approach the unsupervised WSD problem from different angles, they do each take advantage of semantic similarity measures derived from an existing knowledge resource (WordNet). While we are not arguing the legitimacy of this strategy, we believe there is another type of information that a system can benefit from to determine the sense of words, specifically word and sense order information. Furthermore, the strategy we propose allows the system to be deployed in environments where semantic similarity among senses cannot be determined *a priori*. The only requirement in our approach is that there exist multiple words mapped to a single concept in a sense inventory.

Based on this alternative strategy even the non-content words such as stop words (ignored in existing approaches) can be helpful. Considering the sentence "He went into the bank beside the river", most of the above approaches will likely choose the *river bank* (*bank*#2) sense for bank instead of the correct *financial institute* (*bank*#1) sense, because the former sense is semantically closer to the only other content word *river*. However, even without other context information, it is not hard for an English speaker to realize the financial bank is more likely to be the correct one, since people do not usually go into a river bank. A somewhat accurate SLM can guide the system to make this decision since it shows $P(bank\#1|into\#1the\#1) \gg P(bank\#2|into\#1the\#1)$.

Such information can be learned in an unsupervised manner if the system sees similar sentences such as "He went into a banking-company" (where *banking-company* has *bank*#1 sense in WordNet 2.1). Also consider the sentence "The tank has an empty tank". Again it would not be trivial for the previously described algorithms to realize these two *tank*s have different meanings since their frameworks (explicitly or implicitly) imply or result in one sense per sentence. However, an accurate semantics-enhanced language model can tell us that the *tank as container* sense has higher chance to follow the word *empty* while the *tank as the army tank* sense has higher chance to be followed by *has*.

3.2 System Design and Experiment Setup

We applied both bigram SLM and WSLM to perform unsupervised WSD. Our WSD system can be divided into three stages. The first stage is the initialization stage. In SLM, we need to initialize $P(S_{k+1}|S_k)$ and in WSLM there are two types of probabilities to be initialized: $P(S_k|W_k)$ and $P(W_{k+1}|S_k)$. We explore here two different ways to initialize the LMs without any *a priori* knowledge of the probability distribution of senses. The second stage is the learning stage, using the EM algorithm together with forward-backward training to learn the bigrams. The final stage is the decoding stage, in which the learned bigrams are utilized to identify the senses of words in their sentential context that optimize the total probability. Using the dynamic programming method, the overall time

Table 1. The results for all-words unsupervised WSD on SemCor using SLM and WSLM based on uniform and node-frequency initialization

Initialization	Corpus	Baseline (%)	SLM (%)	WSLM (%)
Uniform	SC	17.1	31.8	27.7
Uniform	SC+BNC	17.1	32.3	28.8
Graph Freq	SC	17.1	25.1	34.0
Graph Freq	SC+BNC	17.1	36.0	34.6

complexity for the system is only linear to the number of words and quadratic to the average number of senses per word.

We tested our system on SemCor (SC) data, which is a sense-annotated corpus that contains a total of 778K words (where 234K have sense annotations). We use SemCor and British National Corpus (BNC) sampler data (1.1 million words) for training. In the EM algorithm initializations reported on below, no external knowledge other than the unannotated corpus and the sense dictionary is exploited. The experimental setup is as follows: we first determine the baseline performance on the WSD task using only the initial knowledge (i.e. without applying language models). Then we train a semantics-enhanced LM based on the initialization and use it to perform decoding. Our model is evaluated by checking how much the learned LM can improve the accuracy over the baseline.

Initialization: Uniform N-Gram Probabilities. The baseline for this case is a random sense assignment for all-words WSD (i.e. disambiguation of all word tokens) in SemCor, resulting in 17% accuracy on the test set. The initialization simply assigns equal probability to all bigrams. As shown in Table 1, the results improve to 32.3% for SLM and 28.8% for WSLM after training on a corpus consisting of the SemCor texts plus texts from the BNC Sampler.

Initialization: Graph Frequency. The second initialization is based on the node occurrence frequency in the sense graph. That is, $Pr(S_1|S_2) = gf(S_1)$ for SLM and $Pr(S_1|W_1) = gf(S_1)$ for WSLM , where $gf(S_1)$ represents the frequency of a node S1 in the sense graph, or its graph frequency (for example, in Figure 2 00965972 appears three times). The intuition behind this initialization is that a sense should have a higher chance to appear if it occurs in multiple words that frequently occur in the text. Again, to count the node frequency we do not need any extra knowledge since the graph itself can be generated based on only the corpus and the dictionary. This initialization improves the accuracy to 36.0% for SLM and 34.6% for WSLM.

These experiments show that learned syntactic order structure can tell us much about the sense of a word in context, in the absence of external knowledge.

3.3 Discussion

The case study on applying semantics-enhanced LM to WSD reveals two important facts. The first is that syntactic order information for words and senses can

Table 2. Comparison between LM-based approaches, semantic approaches and semantics-enhanced LM approaches for all-nouns Unsupervised WSD

Initialization	Corpus	SLM (%)	WSLM (%)
Uniform	SC+BNC	35.6	32.3
gloss overlap		36.5	
Graph Freq	SC+BNC	29.6	38.0
SSI		42.7	

benefit WSD. This conclusion to some extent echoes the concept of syntactic semantics [18], which claims that semantics are embedded inside syntax. The second conclusion is that the unsupervised learning method proposed in this paper does learn a sufficient amount of meaningful semantic order information to allow the system to improve disambiguation quality. It follows from this that the framework is flexible enough to be trained on a domain-specific corpus to obtain a SLM or WSLM specifically for that domain. This has important potential applications in domains with senses not represented in resources such as WordNet.

Table 2 shows how different types of knowledge perform in WSD. We compare our system with two existing WSD systems on the all-nouns WSD task (that is, evaluating disambiguation performance only on nouns in the corpus): Banerjee and Pedersen's gloss overlap system and the SSI system (we limit ourselves to the all-nouns task as these are the results as reported in [4]). The LM-based approach without preliminary knowledge performs right in between gloss overlap and SSI approaches in predicting the nouns in SemCor. This is interesting and informative since the results demonstrate that by using only word order information and no lexical semantic information (e.g. sense similarity), we still generate competitive WSD results. Comparing Table 2 with Table 1, one can also infer that WSD on nouns is an easier task than on other parts of speech.

One advantage of our model is that it could incorporate any amount of supervised information in the initialization step. A small amount of annotated data can be used to generate the initial n-grams to be refined through EM. This would certainly result in significant improvements over the knowledge-poor experiments presented here. Given the performance of our system relative to the more knowledge-intensive approaches, that approach would also be likely to result in an overall improvement over those results since it incorporates an additional source of linguistic information.

4 Related Work

There have been previous efforts in incorporating semantics into a language model. Brown *et al* proposed a class-based language model that includes semantic classes in a LM [5]. Bellegarda proposes to exploit latent semantic analysis to map words and their relationships with documents into a vector space [3]. Chueh *et al* propose to combine semantic topic information with n-gram LM using the

maximum entropy principle [6]. Griffiths *et al* also propose to integrate topic semantic information [10] into syntax based on a Markov chain Monte Carlo method.

The major difference between our model and these is that we propose to learn semantics at the word level rather than at the document or topic level. Consequently the models are different in the parameters to be learned (in the other models, the topic usually determines words to be used while in our model the words can determine senses), preliminary knowledge incorporation (e.g. [5] used a fully connected word-class mapping during initialization) and most importantly, the applications. Other systems were evaluated on word clustering or document classification while we have made the first attempt to apply a semantics-enhanced LM to a fine-grained semantic analysis task, namely word sense disambiguation.

5 Conclusion and Future Work

There are three major contributions in this paper. First we propose a framework that enables us to incorporate semantics into a language model. Second we show how such a model can be learned efficiently ($O(nb^2)$ time) in an unsupervised manner. Third we demonstrate how this model can be used to perform the WSD task in knowledge-poor environments. Our experiments also suggest that WSD can be a suitable platform to evaluate the semantic language models, and that using only syntactic information one can still perform WSD as well as using conventional semantic (e.g. gloss) information.

There are two main future directions for this work. In terms of the model itself, we would like to integrate additional knowledge into the initialization, to take advantage of existing *a priori* knowledge, specifically sense frequency information derived from WordNet (which orders senses by frequency), as well as using the semantic hierarchy in WordNet to smooth probabilities in the language model. We would also like to investigate how much the results can be improved based on higher n-gram models (e.g. trigram). In terms of applications we would like to investigate whether the model can be applied to other natural language processing tasks that generally require both syntactic and semantic information such as information extraction, summarization, and machine translation.

References

1. Banerjee, S., Pedersen, T.: Extended gloss overlaps as a measure of semantic relatedness. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 805–810 (2003)
2. Baum, L.E.: An Inequality and Associated Maximization in Statistical Estimation for Probabilistic Functions of Markov Processes. Inequalities 627(3), 1–8 (1972)
3. Bellegarda, J.: Exploiting latent semantic information in statistical language modeling. Proceedings of IEEE 88(8), 1279–1296 (2000)
4. Brody, S., Navigli, R., Lapata, M.: Ensemble Methods for Unsupervised WSD. In: Proceedings of the ACL/COLING, pp. 97–104 (2006)

5. Brown, P.F., et al.: Class-based n-gram models of natural language. Computational Linguistics 18(4), 467–479 (1992)
6. Chueh, C.H., Wang, H.M., Chien, J.T.: A Maximum Entropy Approach for Semantic Language Modeling. Computational Linguistics and Chinese Language Processing 11(1), 37–56 (2006)
7. Cutting, D., et al.: A practical part-of-speech tagger. In: Proceedings of ANLP-1992, Trento, Italy, pp. 133–140 (1992)
8. Dempster, A.D., Laird, N.M., Rubin, D.B.: Maximum likelihood for incomplete data via the EM algorithm. Journal of Royal Statistical Society Series B 39, 1–38 (1977)
9. Galley, M., McKeown, K.: Improving Word Sense Disambiguation in Lexical Chaining. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, pp. 1486–1488 (2003)
10. Griffiths, T., et al.: Integrating Topics and Syntax. In: Proceedings of the Advances in Neural Information Processing Systems (2004)
11. Hoste, V., et al.: Parameter optimization for machine-learning of word sense disambiguation. Language Engineering 8(4), 311–325 (2002)
12. Knight, K., et al.: Unsupervised Analysis for Decipherment Problems. In: Proceedings of the ACL-COLING (2006)
13. Koehn, P., Knight, K.: Estimating word translation probabilities from unrelated monolingual corpora using the EM algorithm. In: Proceedings of the AAAI, pp. 711–715 (2000)
14. Lin, S.d., Knight, K.: Discovering the linear writing order of a two-dimensional ancient hieroglyphic script. Artificial Intelligence 170(4-5) (2006)
15. McCarthy, D., et al.: Finding predominant word senses in untagged text. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain (2004)
16. Mihalcea, R.: Unsupervised Large-Vocabulary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling. In: Proceedings of the Joint Conference on Human Language Technology / Empirical Methods in Natural Language Processing (HLT/EMNLP) (2005)
17. Navigli, R., Velardi, P.: Structural Semantic Interconnections: a Knowledge-Based Approach to Word Sense Disambiguation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27(7), 1063–1074 (2005)
18. Rapaport, W.J.: Holism, Conceptual-Role Semantics, and Syntactic Semantics. Minds and Machines 12(1), 3–59 (2002)