Two Birds with One Stone: Learning Semantic Models for Text Categorization and Word Sense Disambiguation

Roberto Navigli, Stefano Faralli
Dipartimento di Informatica
Sapienza Università di Roma
Roma, Italy
{navigli,faralli}@di.uniroma1.it

Aitor Soroa, Oier de Lacalle, Eneko Agirre
IXA NLP Group
University of the Basque Country
Donostia, Spain
{a.soroa,oier.lopezdelacalle,e.agirre}@ehu.es

ABSTRACT

In this paper we present a novel approach to learning semantic models for multiple domains, which we use to categorize Wikipedia pages and to perform domain Word Sense Disambiguation (WSD). In order to learn a semantic model for each domain we first extract relevant terms from the texts in the domain and then use these terms to initialize a random walk over the WordNet graph. Given an input text, we check the semantic models, choose the appropriate domain for that text and use the best-matching model to perform WSD. Our results show considerable improvements on text categorization and domain WSD tasks.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text analysis

General Terms

Algorithms

Keywords

Word Sense Disambiguation, Text Classification

1. INTRODUCTION

In everyday life the use of natural language is often confined to specific fields of knowledge. Examples include newspaper sections (crime, sports, finance, etc.), blogs and Web sites on specific topics, business documents, etc. Performing a semantic analysis of domain text is thus a crucial Natural Language Processing (NLP) task that can boost applications such as Question Answering, Information Retrieval and Information Extraction.

The first step in determining the content of an input text is to assign it a domain or topic label, a task referred to as Text Classification (TC). Text classification is a widely studied topic in machine learning and NLP, and has attracted the use of a wide range of machine learning algorithms and text processing approaches. More recently the use of encyclopedic knowledge resources such as Wikipedia for classification purposes has received considerable attention in the scientific community. In fact, Wikipedia contains a vast amount of information covering virtually all domains of knowledge. Wikipedia has been used to enrich shallow text representations with concepts in order to improve supervised text classification [16] and concept-based kernels have been developed to perform semantically-informed supervised text categorization [15].

In the last few years attention has also been paid to the semantic analysis of domain texts at the lexical level, a task called Domain Word Sense Disambiguation (WSD). The automated association of meanings with words occurring in a domain context can benefit from domain cues such as topic words. Current approaches either exploit supervised techniques [2] or are based on unsupervised models of domains [5] and word sense dominance [10, 9]. However, the former rely on sense-annotated corpora for many domains, whereas the latter do not achieve high performance (unless a very large knowledge resource is available [12]). Although intermediate solutions have recently been proposed [7], sense-tagged data are still needed for accurate domain WSD.

In this paper we present a novel approach to the semantic analysis of domain texts. For each domain of interest we automatically learn a so-called semantic model vector for that domain. To this end we extract relevant terms from selected texts in the domain, determine their meaning and then use this information to initialize a random walk over the WordNet [3] graph. This yields a semantic model for each domain, represented as a vector of probabilities over WordNet nodes. Given an input text we identify the best-matching model, i.e., domain, for that text. As a result, the corresponding semantic model is used to disambiguate the words in the text, assigning the senses that are best fit to the context of occurrence **and** the domain of the text. The strength of our work lies in the use of the same semantic model to perform two different tasks, namely Text Classification and Word Sense Disambiguation. In fact, our experiments show that our approach achieves high performance in both tasks.

2. SEMANTIC MODEL VECTORS

We first learn a semantic representation for each domain in the form of Semantic Model Vectors. To this end we perform the following steps:

1. Analyse the training documents in each domain and produce for each **a weighted list of terms**, where the weight is higher for those terms which are most important in that domain.
2. Use these lists of terms to obtain **a weighted list of synsets for each domain**, where the weight is higher for those synsets which are most important in that domain.
3. For each domain perform a random walk over the WordNet graph, thus spreading domain information across the graph. We use **Personalized PageRank** [1] for the random walk and the above weighted synset list for initialization.
4. Reweight synsets according to their prominence **across domains**, demoting synsets with similar prominence in all domains, and promoting synsets which are only prominent in few domains.

5. We thus obtain a semantic model for each domain in the form of a weighted list of synsets, which we call the **Semantic Model Vector (**SMV**)** for that domain.

2.1 Weighting domain terms

Given a domain d, we first extract candidate domain terms from the training document collection for that domain, and weight them according to TFIDF:

$$\mathrm{TFIDF}_d(t) = \mathrm{TF}_d(t) \cdot log_2 \frac{N_d}{n_{d,t}} \tag{1}$$

where $\mathrm{TF}_d(t)$ is the frequency of term t in the document collection for domain d, N_d is the number of documents in the domain collection, and $n_{d,t}$ is the number of domain documents in which t occurs.

2.2 Weighting domain synsets

Given the weighted list of domain terms, we produce a weighted list of synsets in two different ways.

The **uniform** method is to distribute the TFIDF weight of each word across all synsets that are denoted by the word (i.e., all its senses) uniformly. Let s be a synset, $Synonyms(s)$ the set of words (i.e., synonyms) in the synset s, $\mathrm{TFIDF}_d(w)$ the TFIDF value of a word w in domain d and $Synsets(w)$ the set of synsets (i.e., senses) for the word w. Then, the uniform method assigns the following score to synset s:

$$UniformScore_d(s) = \sum_{w \in Synonyms(s)} \frac{\mathrm{TFIDF}_d(w)}{|Synsets(w)|}. \tag{2}$$

We also tried a more sophisticated method based on **related words**. In this method we use the information about related words to compute the synset relevance to the domain. We first define a set of related words for each synset s as follows:

$$RW_d(s) = \bigcup_{s' \in R(s)} Synonyms(s') \cup GlossWords(s') \tag{3}$$

where $R(s)$ is the set of WordNet synsets directly related to s by any lexical or semantic relation, also including s itself, and $Synonyms(s)$ and $GlossWords(s)$ are the set of content words which denote s and appear in the gloss of s, respectively. Then, we define a score for synset s given by the normalized sum of the domain TFIDF values of the words in the set of related words of s:

$$RelWordsScore_d(s) = \frac{\sum_{w \in RW_d(s)} \mathrm{TFIDF}_d(w)}{|RW_d(s)|} \tag{4}$$

Finally, we normalize the domain scores to range between zero and one, and produce a weighted list of synsets for each domain d.

2.3 Applying Personalized PageRank

Given the graph-based representation of WordNet, we apply Personalized PageRank to each domain, obtaining a probability distribution over WordNet synsets for each domain d, the Semantic Model Vector SMV_d. For initializing the random walk we set the so-called reset distribution, which defines which synsets are most relevant for the domain of interest (cf. [1] for further details on Personalized PageRank). We tried two different reset distributions: the weighted list of synsets for each domain, as produced by the **uniform** and the **related words** method in Section 2.2.

2.4 Cross-domain reweighting (CDR)

In the previous step we obtained an SMV for each domain. Ideally, domain-specific synsets will rank high in the corresponding domain SMV, and low in the others.

However, some synsets are very central in the WordNet graph (i.e., they are linked to synsets which also have a high number of relations) and receive high scores on most domains. Examples include synsets such as $\mathrm{individual}^1_n$ or $\mathrm{britain}^1_n$. Thus, we decided to reweight the SMVs obtained in the previous step, trying to promote domain specific synsets and demote general-purpose synsets (i.e., those concepts with similar scores in all domains).

Let SMV_d be the SMV vector for domain $d \in$ Domains, so that $\mathrm{SMV}_d(s)$ is the score of synset s in the domain d. We calculate the entropy for s using the formula:

$$\mathrm{ent}(s) = -\sum_{d \in \mathrm{Domains}} \mathrm{SMV}_d(s) \cdot \log(\mathrm{SMV}_d(s)) \tag{5}$$

Senses with high scores in some domains but low scores in other domains will receive a low entropy. In contrast, senses with high (or low) values on all domains will receive high entropy.

After calculating the entropy, we obtain new SMV'_d vectors by dividing each synset score by the synset's entropy:

$$\mathrm{SMV}'_d(s) = \mathrm{SMV}_d(s) \cdot \frac{1}{\mathrm{ent}(s)}. \tag{6}$$

2.5 Text Classification

Given a test document and the SMVs for each domain (either SMV_d or SMV'_d, see Sections 2.3 and 2.4), we combine the information coming from the TFIDF from each domain (cf. Section 2.1) and the SMVs. Given the fact that the synsets with lowest weights might confuse the classification algorithm, we decided to keep the synsets with words having high TFIDF, and assign a zero weight to the rest. For a domain d, we first select the set top_K of K top-ranking words according to their TFIDF_d values. We then keep the synsets in each SMV_d that contain at least one word in top_K, zeroing the rest and normalizing, yielding a new vector SMV^K_d. This vector is very close to SMV_d, but synsets with low TFIDF words are assigned a zero weight. Finally we select the best domain d_{BEST} for document doc as follows:

$$d_{\mathrm{BEST}} = \underset{d \in Domains}{\mathrm{argmax}} \sum_{w \in doc} \max_{s \in Synsets(w)} \mathrm{SMV}^K_d(s) \tag{7}$$

where $Synsets(w)$ returns the synsets denoted by w.

2.6 Domain Word Sense Disambiguation

In order to perform Word Sense Disambiguation of the words in a test document, we first classify the document as described above. For each content word to be disambiguated in the document we then combine the information in the chosen SMV and the context(s) of the target word. We thus create a large context by joining all the sentences where the word occurs in the document, set the reset distribution with the content words in the context, and run Personalized PageRank, yielding a Personalized Pagerank vector for the word (PPV_w).

We then select the best sense of the word as follows:

$$sense_{\mathrm{BEST}}(w) = \underset{s \in Synsets(w)}{\mathrm{argmax}} \mathrm{SMV}_d(s) \cdot \mathrm{PPV}_w(s) \tag{8}$$

where w is the word to be disambiguated, $Synsets(w)$ are the synsets of word w, d is the chosen domain, SMV_d is the SMV of domain d and PPV_w is the PPV of the word's context.

3. EXPERIMENTAL SETUP AND RESULTS

We first describe our general experimental setup, which we used to perform experiments on text classification (Section 3.3) and WSD (Section 3.4).

3.1 Target Domains and Training Set

We first chose the set of target domains. We selected the 29 domain labels used in featured Wikipedia articles – a list of the best articles in Wikipedia, as determined collectively by its editors.[1] The domain labels cover a wide variety of topics, including business, arts, computing, sport, etc.

For each domain, we randomly picked up 30 featured articles, thus totalizing $30 \cdot 29 = 870$ articles. Each article in the dataset has a single correct domain label. Note that we use Wikipedia only to define the training and test dataset (i.e., a fixed set of domains and the corresponding articles), but that our semantic model is solely based on WordNet.

3.2 Semantic Model Construction

To build our semantic models we extracted the terms from the training set and calculated their TFIDF scores for each domain. For each domain d, we then used the TFIDF domain scores to weight synsets and built the SMV_d and the corresponding model with cross-domain reweighting.

3.3 Text Classification

3.3.1 Test set

To perform text classification, we prepared a dataset of 290 Wikipedia articles (10 for each of the 29 domains defined for the training set, cf. Section 3.1) that were collaboratively classified as "good articles"[2], i.e., of good quality but that are not yet, or are unlikely to reach, featured article quality. Similarly to the training set, the test set is single-labeled.

3.3.2 Systems

We experimented with four variants of our SMVs, each obtained by selecting a specific combination of the **reset distribution** (either the **Uniform** or the **Related Words** strategy) and **cross-domain reweighting (CDR)** (entropy-based reweighting either enabled or disabled).

3.3.3 Parameters

Our approach has only one parameter, that is, the K constant used for selecting the top domain terms in each SMV_d for text classification. In order to choose the best value of K, we created a tuning dataset by randomly sampling a set of 58 "good" Wikipedia articles (2 articles per domain), with no overlap with our training and test sets. We then ran our classification algorithm on this tuning set and found the optimal values for each variant of our method (1250 and 750 respectively for Uniform SMVs with and without cross-domain reweighting; similarly, 400 and 1350, respectively, for RelWords SMVs).

3.3.4 Baselines and skyline

As a first baseline we calculated the **random baseline**, which selects a random domain label. We also constructed a baseline TFIDF-based classification system. Given the test document, we select the domain that maximizes the sum of the TFIDF scores for the terms in the document:

$$d_{\text{BEST}} = \underset{d \in Domains}{\text{argmax}} \sum_{w \in doc} \text{TFIDF}_d(w). \qquad (9)$$

In order to give a reference for the results of a state-of-the-art machine learning technique, we chose Support Vector Machines (SVM) [6] as the skyline. Given that we are working on many domains, we adopt the *one versus all* paradigm.

We used a linear kernel, whose C parameter was set to 0.5, after using the tuning dataset for optimization (cf. Section 3.3.3). The features were composed of word stems and their normalized TFIDF weights, following the widely used bag-of-words approach.

3.3.5 Results

In Table 1 we report the results of our systems in the classification task. We show recall@k, i.e., the ratio of examples where the correct answer is found among the system's top k domain labels ($k = 1, 2, 3$). The main evaluation measure is recall@1, which is equivalent to recall, precision, F1 and Break-Even Point (BEP) in this dataset.

Table 1: Classification results on Wikipedia (†: signif. diff. from RelWords SMV+CDR, $p < 0.001$).

System	Recall@k		
	k=1	k=2	k=3
Uniform SMV	58.0†	72.9	79.9
Uniform SMV + CDR	64.6†	76.3	80.5
RelWords SMV	64.2†	75.7	81.9
RelWords SMV + CDR	**67.0**	**76.7**	**83.7**
Random baseline	3.5†	-	-
TFIDF baseline	60.4†	71.9	79.5
SVM skyline	68.2†	76.8	81.7

Three of our best SMV variants surpass the TFIDF baseline by several points for all three values of k, showing that our classification performance is generally good. The results show that CDR is beneficial in all cases.

We observe that a 67% recall@1 obtained with our RelWords+CDR SMV on this dataset is a significant result, considering that we have 29 classes with identical prior probabilities (the performance of the random baseline is extremely low, a mere 3.5%).

3.3.6 Analysis

Note that our method is basically the TFIDF baseline with WordNet information added on top of it. The improvement of our systems over the TFIDF baseline could be caused by the use of cutoffs alone (cf. Section 3.3.3). We thus compared our systems against a TFIDF baseline with cutoff on the top-ranking K words (the best value for K was selected on the tuning dataset). The result for this TFIDF top K baseline is 61.1 recall@1, only one point above the unthresholded baseline, showing that semantic information from WordNet can improve results beyond the keyword level.

In a related experiment, we used the result of the "weighting domain synsets" step (cf. Section 2.2) directly. This would shed some light on the usefulness of the Personalized PageRank step when constructing the SMVs. The synset weights provided by the related words approach, when used by our TC algorithm without CDR, yields a recall@1 of 60.8 when used directly, well below the 64.2 attained by our SMV with no CDR ("RelWords SMV" row in Table 1). The only difference between the two runs is the use of PPR, beneficial when building SMVs. We attribute such positive difference to the semantic spreading effect of PPR.

3.4 Word Sense Disambiguation

3.4.1 Test set

For our domain WSD experiment we used the gold standard dataset released by Koeling *et al.* [8]. This dataset comprises examples of 41 words retrieved from the Sports and Finance sections of the Reuters corpus, and also from the balanced British National Corpus (BNC). The selected words are quite polysemous and thus difficult to disambiguate, with an average polysemy of 6.7 senses, ranging from 2 to 13 senses. We created a pseudo-document for each word containing all its contexts, classified this pseudo-document into one of the domains, and proceeded as in Section 2.6.

3.4.2 Baselines and skyline

As baselines we adopted two well-known strategies, i.e., the **random baseline** – which selects a random sense for each word item to disambiguate, and the **SemCor Most Frequent Sense (MFS) baseline** – based on sense occurrence counts within the largest sense-tagged corpus. As a skyline we applied the **test MFS**, i.e., we assigned to each

[1] http://en.wikipedia.org/wiki/Wikipedia:Featured_articles

[2] http://en.wikipedia.org/wiki/Wikipedia:Good_articles

Table 2: **WSD results on Finance and Sports (†: signif. diff. from RelWords SMV + CDR, $p < 0.001$).**

System	Sports	Finance
Uniform SMV	29.4†	51.2†
Uniform SMV + CDR	41.5†	54.2†
RelWords SMV	27.7†	52.6†
RelWords SMV + CDR	**52.7**	**58.2**
UKB	40.3†	49.9†
Random baseline	19.2†	19.5†
SemCor MFS	19.6†	37.1†
Test MFS skyline	77.8†	82.3†

word its most frequent sense as taken from the test set. This skyline is hard to beat, even for supervised systems [8].

3.4.3 Systems

We experimented our **Uniform** and **RelWords SMV** methods with and without cross-domain reweighting (CDR). Note that no additional parameter tuning is needed for WSD. We compared our systems with **UKB** [1], a state-of-the-art WSD system based on Personalized PageRank.

3.4.4 Results

We calculated recall on the two domain test sets (see Table 2). Our RelWords SMV + CDR outperforms all baselines and the UKB method by 12 points on Sports and 8 points on Finance (relative improvement of 31.5% and 16.6% respectively), and achieves state-of-the-art performance.

We remark that the same configuration, i.e., RelWords SMV + CDR, attains the best performance in both categorization and WSD tasks, which provides an added value to our system. Interestingly, Uniform SMV + CDR also outperforms UKB – by a small margin on Sports and many points on Finance – but other variants obtain lower performance.

4. RELATED WORK

Text Classification. The state of the art in text classification uses word-based features and sophisticated machine learning algorithms such as Support Vector Machines or logistic regression among others. Much work tries to improve TC by enriching the documents with semantic information, e.g., with limited information from WordNet [14]. In contrast, our approach uses the rich structure of WordNet, and yields improvement over the text-only baseline using just the semantic model.

Subsequent work focused on Wikipedia [4, 16], which has been used to augment the text representation with semantic features with a larger coverage of named entities. In contrast to the above approaches, in which Machine Learning is combined with bags of words, we show that a semantics-only representation improves over a bag-of-words representation.

Domain Word Sense Disambiguation. Domain WSD is often performed in a type-based fashion, i.e., by assigning a single sense per word. Distributionally similar neighbors in raw text can be used as cues to determine the predominant sense of a target word using a semantic similarity measure [8, 9]. Other distributional methods use word-category matrices and synonym occurrence counts (see [11] for a survey).

Knowledge-based approaches – which exploit the semantic structure of resources such as WordNet – take the middle ground between domain-driven [13] and supervised methods [7], and have been shown to perform equally well on domain texts [1, 12]. In this paper we go one step further: we propose an approach with little human intervention which first classifies a word context and then uses the selected SMV for WSD. In contrast to state-of-the-art WSD [12], we do not use the MFS as a backoff strategy, thanks to our effective combination of the classification and disambiguation steps.

5. CONCLUSIONS

In this paper we presented a unified approach for categorizing Wikipedia pages and performing Domain Word Sense Disambiguation. Key to our approach is the automatic acquisition of semantic models for multiple domains which we then use to perform semantic analysis of input texts.

The crucial fact that we provide a unified approach with remarkably good results in **both tasks** opens an exciting direction of research, thus enabling deeper understanding and hopefully boosting performance on applications such as Question Answering and Information Retrieval.

Acknowledgments

The first author gratefully acknowledges the support of the ERC Starting Grant MultiJEDI No. 259234.

6. REFERENCES

[1] E. Agirre, O. L. de Lacalle, and A. Soroa. Knowledge-based WSD on specific domains: performing better than generic supervised WSD. In *Proc. of IJCAI 2009*, pages 1501–1506, Pasadena, California, USA, 2009.

[2] E. Agirre and O. Lopez de Lacalle. Supervised domain adaption for WSD. In *Proc. of EACL 2009*, pages 42–50, Athens, Greece, 2009.

[3] C. Fellbaum, editor. *WordNet: An Electronic Database.* MIT Press, Cambridge, MA, 1998.

[4] E. Gabrilovich and S. Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34:443–498, March 2009.

[5] A. Gliozzo, C. Strapparava, and I. Dagan. Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech and Language*, 18(3):275–299, 2004.

[6] T. Joachims. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms.* Kluwer/Springer, 2002.

[7] M. Khapra, A. Kulkarni, S. Sohoney, and P. Bhattacharyya. All words domain adapted WSD: Finding a middle ground between supervision and unsupervision. In *Proc. of ACL 2010*, pages 1532–1541, Uppsala, Sweden, July 2010.

[8] R. Koeling, D. McCarthy, and J. Carroll. Domain-specific sense distributions and predominant sense acquisition. In *Proc. of HLT-EMNLP 2005*, pages 419–426, Vancouver, Canada, 2005.

[9] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. Unsupervised acquisition of predominant word senses. *Computational Linguistics*, 33(4):553–590, 2007.

[10] S. Mohammad and G. Hirst. Determining word sense dominance using a thesaurus. In *Proc. of EACL 2006*, pages 121–128, Trento, Italy, 2006.

[11] R. Navigli. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69, 2009.

[12] S. P. Ponzetto and R. Navigli. Knowledge-rich Word Sense Disambiguation rivaling supervised system. In *Proc. of ACL 2010*, pages 1522–1531, Sweden, 2010.

[13] C. Strapparava, A. Gliozzo, and C. Giuliano. Pattern abstraction and term similarity for word sense disambiguation: Irst at senseval-3. In *Proc. of Senseval-3*, pages 229–234, Barcelona, Spain, 2004.

[14] L. Ureña-López, M. de Buenaga Rodríguez, and J. Gómez. Integrating linguistic resources in TC through WSD. *Computers and the Humanities*, 35(2):215–230, 2001.

[15] P. Wang and C. Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proc. of KDD 2008*, pages 713–721, Nevada, 2008.

[16] P. Wang, J. Hu, H.-J. Zeng, and Z. Chen. Using wikipedia knowledge to improve text classification. *Knowledge Information Systems*, 19(3):265–281, 2009.