LMSim : Computing Domain-specific Semantic Word Similarities Using a Language Modeling Approach

Sachin Pawar[1,2] **Swapnil Hingmire[1,3]** **Girish K. Palshikar[1]**

{sachin7.p, swapnil.hingmire, gk.palshikar}@tcs.com
[1]Systems Research Lab, Tata Consultancy Services Ltd., Pune, India
[2]Department of CSE, IIT Bombay, Mumbai, India
[3]Department of CSE, IIT Madras, Chennai, India

Abstract

We propose a method to compute domain-specific semantic similarity between words. Prior approaches for finding word similarity that use linguistic resources (like WordNet) are not suitable because words may have very specific and rare sense in some particular domain. For example, in customer support domain, the word `escalation` is used in the sense of "problem raised by a customer" and therefore in this domain, the words `escalation` and `complaint` are semantically related. In our approach, domain-specific word similarity is captured through language modeling. We represent context of a word in the form of a set of word sequences containing the word in the domain corpus. We define a similarity function which computes weighted Jaccard similarity between the set representations of two words and propose a dynamic programming based approach to compute it efficiently. We demonstrate effectiveness of our approach on domain-specific corpora of Software Engineering and Agriculture domains.

1 Introduction

In text clustering, we expect to cluster these two sentences together - `Ricestar HT is a good product for sprangletop control.` and `Barnyardgrass can be controlled by usage of Clincher.` We observe that though these two sentences do not share any content word, still they are semantically similar to each other. The similarity between these two sentences will not be high unless the clustering algorithm takes into account that the words within the pairs `Ricestar-Clincher`[1] and `sprangletop-Barnyardgrass`[2] are semantically similar. In addition to text clustering, discovering semantically similar words has rich applications in the fields of Information Retrieval, Question Answering, Machine Translation, Spelling correction, etc.

We propose an algorithm which assigns high similarity for words, which are highly *replaceable* by each other in their context without affecting its syntax and "meaning". In above example, the words `sprangletop` and `Barnyardgrass` can be safely replaced by each other while preserving the syntax and "meaning" of the original sentences.

Approaches for computing similarity between words can be broadly classified into two types - i) Approaches using linguistic resources like WordNet (Miller, 1995) or thesaurus (Budanitsky and Hirst, 2006) and ii) Approaches using statistical properties of words in a corpus (Blei et al., 2003; Halawi et al., 2012; Brown et al., 1992).

In this paper, our focus is to develop a word similarity algorithm which discovers similar words for a specific domain. We mainly focus on two domains - Software Engineering (SE) and Agriculture. Our motivation behind choosing these domains is that these are relatively unexplored and as per our knowledge they lack domain-specific lexical resources.

WordNet-based approaches are not suitable for some domains because words may have very specific and rare sense in those domains. Moreover, creating lexical resources like WordNet for a certain domain is a quite challenging task as it requires extensive human efforts, expertise, time and cost. In SE domain, we can say that the words `Oracle` and `DB2` are simi-

[1]both are names of herbicides
[2]both are names of weeds

D S Sharma, R Sangal and J D Pawar. Proc. of the 11th Intl. Conference on Natural Language Processing, pages 101–106, Goa, India. December 2014.

lar because they both are Database Management Softwares. However, the word `DB2` is not present in WordNet and the sense of the word `Oracle` as a "database management software" is not covered in WordNet.

Corpus-based approaches for finding word similarity using topic models (Blei et al., 2003) or Distributional Similarity (Lee, 1999) make use of word context to capture its meaning. Though these approaches use higher order word co-occurrence, they do not consider order of other words within the context of a word. Our hypothesis is that considering word order results in better word similarities and this is evident from our experimental results. Brown Word clustering algorithm (Brown et al., 1992) considers order of words and learns a language model based on word clusters. However, this algorithm is sensitive to number of clusters. In this paper, we use word-based Language Model to capture the context of words in the form of word sequences (preserving word order) and to assign weights for each of the word sequences by computing their probabilities. The major contributions of this paper are - i) a new algorithm (LMSim) to compute domain-specific word similarity and ii) an efficient dynamic programming based algorithm for fast computation of LMSim.

2 Related Work

Various WordNet-based approaches are proposed for computing similarity between two words which are surveyed by Budanitsky and Hirst (2006). Pedersen et al. (2007) adapted WordNet-based word similarity measures to the biomedical domain using SNOMED-CT, which is an ontology of medical concepts.

In order to overcome several limitations of WordNet, Gabrilovich and Markovitch (2007) proposed Explicit Semantic Analysis (ESA), that represents and compares the meaning of texts in a high-dimensional space of concepts derived from Wikipedia.

Halawi et al. (2012) proposed an approach for learning word-word relatedness, where known pairs of related words can be provided as an input to impose constraints on the learning process. For both the domains that we are focusing on - SE and Agriculture domain, we do not have any such prior knowledge about related word pairs.

Notation	Details
D	Domain-specific text corpus
K	Length of context window
$C(w_i, K)$	Set of context words for the word w_i
ws_i^j	j^{th} sequence of context words for w_i
$\lambda(ws_i^j)$	Weight of the word sequence ws_i^j
$n(w_i, w_j)$	Frequency of bigram $\langle w_i, w_j \rangle$ in D
$n(w_i)$	Frequency of word w_i in D
$P_f(w_i\|w_j)$	Prob. of observing w_i after w_j
$P_b(w_i\|w_j)$	Prob. of observing w_i before w_j
$C_{seq}(w_i, K)$	Set of all possible K length word sequences (formed using words in $C(w_i, K)$) that start or end in w_i

Table 1: Details of notations used in this paper

3 Computing Word Similarity

We take help of a *statistical language model* to represent the context of each of the words.

3.1 Language Model

A statistical language model (Jurafsky and Martin, 2000) assigns a probability to a sequence of words using n-gram statistics estimated from a text corpus. Using bigram statistics, the probability of the word sequence w_1, w_2, w_3, w_4 is computed as follows,

$$P(w_1, w_2, w_3, w_4) = P(w_1)P(w_2|w_1)P(w_3|w_2)P(w_4|w_3)$$

The main motivation behind using a Language Model is that it not only provides a way to capture the context information for words but also retains the word order information of context words. Moreover, using a Language Model, we can weigh each word sequence in the context of a particular word by computing its probability as explained above.

In our work, we have used a bigram ($L_0 = 2$) Language Model along with Laplace smoothing. Our algorithm is currently designed for bigram model and in future we plan to extend it to higher order ($L_0 > 2$) models.

3.2 LMSim : Algorithm to Compute Language Model based Similarity

For finding semantic similarity between two any words w_1 and w_2 (refer Table 1 for notations), sets of context words ($C(w_1, K)$ and $C(w_2, K)$) are created by collecting all the words falling in the window of $\pm K$ in the given text corpus. Let I be the set of words which appear in the context of both w_1 and w_2, i.e. $I = C(w_1, K) \cap C(w_2, K)$.

For w_1		For w_2	
Forward	Backward	Forward	Backward
w_1, c_1, c_2	c_1, c_2, w_1	w_2, c_1, c_2	c_1, c_2, w_2
w_1, c_1, c_3	c_1, c_3, w_1	w_2, c_1, c_3	c_1, c_3, w_2
w_1, c_2, c_1	c_2, c_1, w_1	w_2, c_2, c_1	c_2, c_1, w_2
w_1, c_2, c_3	c_2, c_3, w_1	w_2, c_2, c_3	c_2, c_3, w_2
w_1, c_3, c_1	c_3, c_1, w_1	w_2, c_3, c_1	c_3, c_1, w_2
w_1, c_3, c_2	c_3, c_2, w_1	w_2, c_3, c_2	c_3, c_2, w_2

Table 2: Example word sequences considered

The context of w_1 and w_2 is then defined as set of word sequences of length K starting at or ending at either w_1 or w_2. The word sequences are formed by considering all possible combinations of the words in the set I. If $|I| = M$, then M^K word sequences starting at w_1 or w_2 are considered and M^K word sequences ending at w_1 or w_2 are considered. If $I = \{c_1, c_2, c_3\}$ and $K = 2$, then the word sequences created for w_1 and w_2 are as shown in the table 2. Each word sequence is also associated with a weight which corresponds to its Language Model probability. For a forward sequence w_1, c_1, c_2, its weight is set as follows,

$$\lambda_{w_1,c_1,c_2} = P_{forward}(c_1, c_2|w_1) = P_f(c_1|w_1)P_f(c_2|c_1)$$

$P_f(c_1|w_1)$ is computed by using the bigram statistics learnt from the text corpus, i.e. it is the ratio of number of times the word c_1 succeeds w_1 in the corpus to the number of times w_1 occurs in the corpus.

$$P_f(c_1|w_1) = \frac{n(w_1, c_1)}{n(w_1)}$$

Similarly, for a backward sequence c_1, c_2, w_1, we set its weight as,

$$\lambda_{c_1,c_2,w_1} = P_{backward}(c_1, c_2|w_1) = P_b(c_2|w_1)P_b(c_1|c_2)$$

$P_b(c_2|w_1)$ is computed as the ratio of number of times the word c_2 precedes w_1 in the corpus to the number of times w_1 occurs in the corpus.

$$P_b(c_2|w_1) = \frac{n(c_2, w_1)}{n(w_1)}$$

The context of a word is a set of word sequences (ws's) and their corresponding weights (λ's) as follows,

$$C_{seq}(w_1, K) = \{\langle ws_1^1, \lambda(ws_1^1)\rangle, \cdots, \langle ws_N^1, \lambda(ws_N^1)\rangle\}$$

$$C_{seq}(w_2, K) = \{\langle ws_1^2, \lambda(ws_1^2)\rangle, \cdots, \langle ws_N^2, \lambda(ws_N^2)\rangle\}$$

It is to be noted that the word sequence ws_i^1 (in $C_{seq}(w_1, K)$) corresponds to ws_i^2 (in $C_{seq}(w_2, K)$) and they both are equal except that ws_i^1 starts or ends in w_1 whereas ws_i^2 starts or ends in w_2.

Word pair	$\mathbf{sim_{jaccard}}$	$\mathbf{sim_{context}}$	LMSim
`client, complaint`	0.1545	0.754	0.1165
`succession, plan`	0.2019	0.9651	0.1949
`collaboration, realization`	0.4259	0.2643	0.1126
`mapping, publication`	0.4672	0.2951	0.1379

Table 3: Word pairs having low LMSim

We define similarity based on Language Model between w_1 and w_2 as follows,

$$sim_{jaccard}(w_1, w_2) = \frac{\sum_{i=1}^{N} \min(\lambda(ws_i^1), \lambda(ws_i^2))}{\sum_{i=1}^{N} \max(\lambda(ws_i^1), \lambda(ws_i^2))} \quad (1)$$

The similarity $sim_{Jaccard}$ between w_1 and w_2 is nothing but the weighted Jaccard similarity (Surgey, 2010) between $C_{seq}(w_1, K)$ and $C_{seq}(w_2, K)$. Here, we treat ws_i^1 and ws_i^2 as equal while computing Jaccard similarity as noted earlier.

As we are constructing the word sequences by only using the words in the intersection of the contexts of the two words, we weigh down the similarity by multiplying it with the common context factor, $sim_{context}$ defined as,

$$sim_{context}(w_1, w_2) = \max\left(\frac{|I|}{|C(w_1, K)|}, \frac{|I|}{|C(w_2, K)|}\right)$$

LMSim, the similarity between two words w_1 and w_2 is computed as,

$$LMSim(w_1, w_2) = sim_{jaccard}(w_1, w_2) * sim_{context}(w_1, w_2)$$

3.2.1 Why Are Both $sim_{jaccard}$ and $sim_{context}$ Important?

Considering only $sim_{context}$ can be misleading, as many co-occurring word pairs will have a high $sim_{context}$. This is evident for the top 2 word pairs in Table 3 which have got high $sim_{context}$ even though they are not similar. As $sim_{jaccard}$ for these pairs is very low, LMSim is reduced as desired.

On the other hand, due to smaller common context, considering only $sim_{jaccard}$ can be misleading. In Table 3, we can observe that the bottom 2 word pairs have relatively high $sim_{jaccard}$, but due to their smaller common context, their $sim_{context}$ is low which results in low LMSim as desired.

3.3 Efficient Computation of LMSim

For computing $sim_{jaccard}$ (Eq. 1), the sum of N terms is required to be computed. Here,

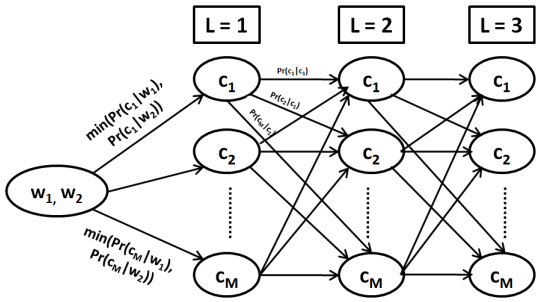

Figure 1: All possible forward word sequences (intersection scenario)

$N = 2 \cdot M^K$ where K is the context window considered and M is the number of words in the context of both the words in consideration for computing similarity. Even for a moderate size corpus, the value of M can be in thousands and even for a small window of size 3, N will be 1000^3 which is a huge number. Hence, it is infeasible to actually enumerate all the N distinct word sequences. We propose an efficient, dynamic programming based approach similar to Viterbi Algorithm (Forney, 1973) for fast computation of Language Model similarity.

Figure 1 shows all possible forward paths for words w_1 and w_2 formed using words in the set $I = \{c_1, c_2, \cdots c_M\}$. It depicts the scenario where the numerator (intersection of two sets with weighted members) of Eq. 1 is being computed. All possible backward paths can be constructed in similar way.

At each level (L in the figure 1 which varies from 1 to K), for each context word, we store sum of probabilities of all the word sequences ending in that particular word. This is similar to the Viterbi algorithm used for finding most probable state sequence for HMMs. The only difference is that, at each level for each state, Viterbi algorithm stores *maximum* of probabilities of all state sequences ending in that particular state whereas our algorithm stores the sum of probabilities of all word sequences ending in a particular word. This algorithm to compute $sim_{jaccard}$ reduces the number of computations from $O(M^K)$ to $O(KM^2)$.

4 Experimental Analysis

In this section, we describe details about experimental evaluation of our algorithm.

4.1 Datasets

We use text corpora from following domains:
1. SE domain: A collection of 138159 task descriptions/expectations for various roles performed in Software services industry containing 1276514 words.
2. Agriculture domain: A set of 30533 documents in English containing 999168 sentences and approximately 19 million words. It was collected using crawler4j[3] by crawling various agriculture news sites.

There are no publicly available datasets of similar words for the SE and Agriculture domains. We selected some domain-specific word pairs for these domains and obtained human assigned similarity scores for these pairs. For the SE domain, we had 500 word pairs annotated for similarity scores (0:No similarity, 1:Weak, 2:High) by 4 human annotators familiar with the domain terminologies. The average of the similarity scores assigned by all the annotators is then considered as the gold-standard similarities for these word pairs. Similarly, we created a standard dataset[4] of 200 word pairs in Agriculture domain.

4.2 Comparison with Other Approaches

1. Distributional Similarity (DistSim): For each word, a TF-IDF vector is constructed which encodes how the other words co-occur with that particular word. The similarity between any two words is then computed as the cosine similarity between their corresponding TF-IDF vectors.
2. Latent Dirichlet Allocation (LDA): LDA (Blei et al., 2003) is a probabilistic generative model that can be used to uncover the underlying semantic structure of a document collection. LDA gives per-word per-document topic assignments that can be used to find a likely set of topics and represent each document in the collection in the form of topic proportions. We find probability distribution of a word over topics using the number of times the word is assigned to a topic. We compute similarity between words w_i and w_j as the Jensen-Shannon (JS) divergence between their respective probability distributions over

[3] code.google.com/p/crawler4j/
[4] contact authors for the datasets

topics. We experimented with different number of topics and reported the results of best performing number of topics.

3. WordNet and Wikipedia based similarities: We used WordNet-based approach "Lin similarity" (Lin, 1998) for computing word similarities. We used its NLTK implementation[5] with Brown corpus for IC statistics. This algorithm computes similarity between two synsets (senses). As we are not disambiguating sense of the words in a given word pair, we compute Lin similarity between all possible combinations of senses of the two words and consider the maximum of these similarities as overall similarity between words. We consider that the words not present in WordNet have similarity score of 0 with any other word. We also compare LMSim with Wikipedia based ESA[6].

4. Brown Word Clustering: We used the Brown word clustering implementation[7] by Percy Liang. Each word cluster is represented by a binary (0/1) string indicating the path taken from the root to the word in consideration in the hierarchical word clustering output. The algorithm does not explicitly compute the word similarities, hence for any two words w_1 and w_2 having binary string cluster representations s_1 and s_2, we compute similarity as,

$$sim_{brown}(w_1, w_2) = \frac{|CommonPrefix(s_1, s_2)|}{Average(|s_1|, |s_2|)}$$

Here, $|s_i|$ indicates the length of the binary string s_i. The only parameter required for Brown word clustering algorithm is the number of clusters to form. We got the best results with 1000 clusters for SE domain and 500 clusters for Agriculture domain.

4.3 Results

We use each algorithm discussed above to assign similarity scores to each word pair in our gold-standard dataset. Performance of each algorithm (see Table 4) is judged by computing correlation between an algorithm assigned word similarities with the gold-standard word similarities. We could not compare LMSim with LDA for Agriculture dataset due to large size of the corpus. We can observe here that for both the datasets, LMSim performs better than WordNet and Wikipedia based approaches because of absence of many domain-specific words/senses in these resources. At the same time, LMSim performs better than DistSim, LDA and Brown word clustering algorithm, so we can say that LMSim encodes the context information in better way through forward and backward context word sequences. Table 5 and Table 6 show some of the highly similar word pairs discovered for both the domains. Following are some key observations of the experimental results.

Algorithm	Correlation with Gold-standard	
	SE	Agriculture
Lin Similarity	0.1675	0.2303
ESA	0.1527	0.3940
DistSim	0.3278	0.3645
LDA	0.4738	NA
Brown	0.479	0.5945
LMSim	**0.5639**	**0.6229**

Table 4: Relative performance of algorithms

1. Importance of Word Order : LMSim scores over other context based word similarity approaches like DistSim and LDA because it encodes word order of the context words using Language Model. For example, a word pair `asset - wiki` incorrectly assigned high similarity scores by both DistSim and LDA, because these two words co-occur quite frequently. Some example text fragments in which they usually co-occur are as follows:

```
-Number of Assets to be logged into wiki ...
-No.  of assets created in team Wiki..
```

At least one word not present in WordNet

Word Pair	Comment
`clarity, epm`	Clarity is a tool for EPM (Enterprise Project Management)
`sit ,uat`	SIT:System Integration Testing, UAT:User Acceptance Testing
`solution, POCs`	POC : "Proof of concept"
`.Net, Java`	Types of softwares

Both words present in WordNet

Word Pair	Comment
`people, resource`	"person" sense of `resource` not present in WN
`complaint, escalation`	"complaint" sense of `escalation` absent in WN
`test, regression`	Regression is a kind of testing

Table 5: Examples from SE domain

[5] `www.nltk.org`

[6] `http://treo.deri.ie/easyesa/`

[7] `http://github.com/percyliang/brown-cluster`

At least one word not present in WordNet

Word Pair	Comment
cruiser, gaucho	Both are insecticides
clincher, regiment	Both are herbicides
ethanol, biodiesel	ethanol is used in producing biodiesel
fusarium, verticillium	Names of fungi
glyphosate, herbicide	Glyphosate is a herbicide

Both words present in WordNet

Word Pair	Comment
farmer, producer	"farmer" sense of producer is absent in WN
subsoil, topsoil	Different layers of soils

Table 6: Examples from Agriculture domain

-Number of assets posted to Wiki..

For this word pair, LMSim assigns a low similarity score as they do not tend to share similar context word sequences.

2. Limitations of LMSim: We observed that LMSim assigns high similarity for synonyms (client-customer), antonyms (dry-wet) and siblings (spring-summer, .Net-Java), but for some hypernymy relations (like consultant-associate) it assigns a low similarity score because of strict replaceability constraint. In future, we will revise our algorithm to overcome this limitation.

5 Conclusions and Future Work

We proposed a new algorithm LMSim using a Language Modeling approach for computing domain-specific word similarities. We demonstrated the performance of LMSim on two different domains - Software Engineering (SE) domain and Agriculture domain. To the best of our knowledge this is the first attempt for discovering similar words in these domains.

The important advantages of LMSim over previous approaches are - i) it does not require any linguistic resources (like WordNet) hence it saves cost, time and human efforts involved in creating such resources, ii) LMSim incorporates the word order information within context of a word resulting in better estimates of word similarity compared to other corpus-based approaches that ignore word order. We proposed an efficient dynamic programming based algorithm for computing LMSim. Our experiments show that LMSim better correlates with human assigned word similarities as compared to other approaches.

In future, we plan to revise LMSim by extending it to higher order language models, trying better smoothing techniques and using other information like POS tags for having better estimates of word probabilities. We also plan to extend our algorithm to work for Indian languages as lexical resources for these languages are not widely available. We would like to experiment with other domains like Mechanical Engineering, Legal domain etc. We believe that our algorithm can facilitate construction of domain-specific ontologies.

References

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3:993–1022, March 2003.

A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47, 2006.

G. D. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren. Large-scale learning of word relatedness with constraints. In *SIGKDD*, pages 1406–1414. ACM, 2012.

S. Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. *Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 2010*, pages 246–255, 2010.

D. Jurafsky and J. H. Martin. *Speech & Language Processing*. Pearson Education India, 2000.

L. Lee. Measures of distributional similarity. In *ACL* 1999, pages 25–32.

D. Lin. An information-theoretic definition of similarity. In *ICML*, vol. 98, pages 296–304, 1998.

G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

T. Pedersen, S.V. Pakhomov, S. Patwardhan and C.G. Chute. Measures of semantic similarity and relatedness in the biomedical domain. *Journal of biomedical informatics*, 40(3):288–299, 2007.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. Class-based n-gram models of natural language. In *Computational linguistics*, 18(4), pages 467–479.