PolyUCOMP-CORE_TYPED: Computing Semantic Textual Similarity using Overlapped Senses

Jian Xu **Qin Lu**
The Hong Kong Polytechnic University
Department of Computing
Hung Hom, Kowloon, Hong Kong
{csjxu, csluqin}@comp.polyu.edu.hk

Abstract

The Semantic Textual Similarity (STS) task aims to exam the degree of semantic equivalence between sentences (Agirre et al., 2012). This paper presents the work of the Hong Kong Polytechnic University (PolyUCOMP) team which has participated in the STS core and typed tasks of SemEval-2013. For the STS core task, the PolyUCOMP system disambiguates words senses using contexts and then determine sentence similarity by counting the number of senses they shared. For the STS typed task, the string kernel (Lodhi et al., 2002) is used to compute similarity between two entities to avoid string variations in entities.

1 Introduction

Sentence similarity computation plays an important role in text summarization and social network applications (Erkan et al., 2004; Jin et al., 2011). The SemEval 2012 competition initiated a task targeted at Semantic Textual Similarity (STS) between sentence pairs (Agirre et al., 2012). Given a set of sentence pairs, participants are required to assign to each sentence pair a similarity score.

Because a sentence has only a limited amount of content words, it is difficult to determine sentence similarities. To solve this problem, Hatzivassiloglou et al. (1999) proposed to use linguistic features as indicators of text similarity to address the problem of sparse representation of sentences. Mihalcea et al. (2006) measured sentence similarity using component words in sentences. Li et al. (2006) proposed to incorporate the semantic vector and word order to calculate sentence similarity. Biemann et al. (2012) applied the log-linear regression model by combining the simple string based measures, for example, word ngrams and semantic similarity measures, for example, textual entailment. Similarly, Saric et al. (2012) used a support vector regression model which incorporates features computed from sentence pairs. The features are knowledge- and corpus-based word similarity, ngram overlaps, WordNet augmented word overlap, syntactic features and so on. Xu et al. (2012) combined semantic vectors with skip bigrams to determine sentence similarity, whereas the skip bigrams take into the sequential order between words.

In our approach to the STS task, words in sentences are assigned with appropriate senses using their contexts. Sentence similarity is computed by calculating the number of shared senses in both sentences since it is reasonable to assume that similar sentences should have more overlapping senses. For the STS-TYPED task, variations might occur in author names, people involved, time expression and location. Thus, string kernel is applied to compute similarity between entities because it can capture variations between entities. Moreover, for the event similarity in STS-TYPED task, semantic relatedness between verbs is derived the WordNet.

The rest of this paper is structured as follows. Section 2 describes sentence similarity using sense overlapping and string kernel. Section 3 gives the performance evaluation. Section 4 is the conclusion.

2 Similarity between Sentences

Words are used to convey meaning in a sentence. They are tagged with appropriate senses initially and then sentence similarity is calculated based on the number of shared senses.

2.1 Sense Overlapping

When comparing word features, we did not compare their surface equality, but we first conceptualize these words and then calculate their similarities based on the hierarchial structure in WordNet. For a word in a sentence, it will be assigned a WordNet sense. In this paper, we focus on the Word Sense Disambiguation (WSD) algorithm taken by Banerjee and Pederson (2003). They measured the semantic relatedness between concepts by counting the shared words in their WordNet glosses.

In WordNet, a word sense is represented by a synset which has a gloss that defines the concept that it represents. For example, the words *walking, afoot, ambulate* constitute a single synset which has gloss representations as follows,

walking*: the act of traveling by foot*
afoot*: traveling by foot*
ambulate*: walk about*

To lift the limitations of dictionary glosses which are fairly short with insufficient vocabulary, we utilize the glosses of related senses since we assume that words co-occur in one sentence share related senses and the more glosses two senses share, the more similar they are. Therefore, we extract not only glosses of target synset, but also the glosses of the hypernym, hyponym, meronym, holonym and troponym synsets of the target synset to form a synset context. Finally, we compare the sentence contexts with different synset contexts to determine which sense should be assigned to the words.

To disambiguate word senses, a window of contexts surrounding the the target word is specified and a set of candidate word senses are extracted for the content word (noun, verb, adjective) within that window. Let the current target word index $i = 0$ that is, w_0, the window size be $2n+1$ and $-n \leq i \leq +n$. Let $|w_i|$ be the number of senses for word w_i and the j^{th} sense of w_i is $s_{i,j}$, where $1 \leq j \leq |w_i|$. Next is to assign an appropriate sense k to the target word. We achieve this by adding together the relatedness scores calculated by comparing the senses of the target word and senses of every non-target word within the window of context. The sense score for the current target word w_0 is defined as,

$$Sense_k = \sum_{i=-n}^{n} \sum_{j=1}^{|w_i|} relatedness(s_{0,k}, s_{i,j}) \tag{1}$$

The k^{th} sense which has the biggest sense score will be chosen as the right sense for the target word w_0. Now remains the question of how to define the relatedness between two synsets. It is defined as,

$$\begin{aligned} relatedness(s_{0,k}, s_{i,j}) = \\ & score(gloss(s_{0,k}), gloss(s_{i,j})) \\ & +score(hype(s_{0,k}), hype(s_{i,j})) \\ & +score(hypo(s_{0,k}), hypo(s_{i,j})) \\ & +score(hype(s_{0,k}), gloss(s_{i,j})) \\ & +score(gloss(s_{0,k}), hype(s_{i,j})) \end{aligned} \tag{2}$$

In Equation 2, the score function counts the number of overlapping words between two glosses. However, if there is a phrasal n-word overlap, then a score of n^2 will be assigned, thus encouraging the longer n-word overlap. Let $\mathbf{V}$ denote the set of n-word overlaps shared between two glosses, the score is defined as,

$$score = \sum_{w \in \mathbf{V}} \|w\|^2 \tag{3}$$

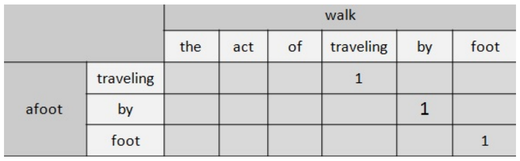
where $\|w\|$ refers to the number of words in w. In so doing, we can have corresponding senses for the sentence *Castro celebrates 86th birthday Monday* as follows,

castro/10886929-n celebrate/02490877-v birthday/15250178-n monday/15163979-n

To find the n-word overlap, we found that contiguous words in two glosses lie in the diagonal of a matrix, take the senses *walk* and *afoot* for example, their glosses are,

walking*: the act of traveling by foot*
afoot*: traveling by foot*

Place the *walking* glosses in rows and *afoot* glosses in columns, we get the matrix representation in Figure 1,



		walk					
		the	act	of	traveling	by	foot
afoot	traveling				1		
	by					1	
	foot						1

Figure 1: n-word overlap representation

Figure 1 shows that *travel by foot* is a continuous sequence of words shared by two glosses. Steps to find n-word overlapping are:

(1) Construct a matrix for two sentences;

(2) Get continuous n-word overlapping, n is greater than 1;

(3) Set the cell values to 0 if they are contained in continuous n-word.

(4) Get the words (unigrams) which are shared by two sentences.

Take *a b c d* and *b c a d* for example, we will have the matrix as follows,

	b	c	a	d
a	0	0	1	0
b	**1**	0	0	0
c	0	**1**	0	0
d	0	0	0	1

Table 1: Matrix representation for two sentences

By the step 2, we will get the *b c* and its corresponding cells *cell(1,0)* and *cell(2,1)*. We then set the two cells to zero, and obtain an updated matrix as follows,

	b	c	a	d
a	0	0	1	0
b	**0**	0	0	0
c	0	**0**	0	0
d	0	0	0	1

Table 2: Updated matrix representation for two sentences

In Table 2, we found that *cell(0,2)* and *cell(3,3)* have values greater than zero. Therefore, a and b will be extracted the common terms.

This approach can also be applied to find common n-word overlaps between sentences, for example,

s_1*: Olli Heinonen, the Head of the International Atomic Energy Agency delegation to Iran, declared yesterday that the agency has reached an agreement with Tehran on the method of conducting the negotiations pertaining to its nuclear program.*

s_2*: leader of international atomic energy agency delegation to iran , olli heinonen said yesterday , that the agency concluded a mutual understanding with tehran on the way to manage talks depending upon its atomic program .*

We will have ngrams with n ranging from 1 to 7, such as,

unigram: *of, to, its, program, yesterday*

bigram: *olli heinonen*

trigram: *that the agency*

four-gram: *with tehran on the*

seven-gram: *international atomic energy agency delegation to iran*

Similarity between two sentences is calculated by counting the number of overlapped n-words. The similarity for s_1 and s_2 is, $(1+1+1+1+1)+(2)^2+(3)^2+(4)^2+(7)^2$ = 83.

2.2 String kernel

For the STS-TYPED task, when comparing whether people or authors are similar or not, we found that some entity mentions may have tiny variations, for example,

E Vincent Harris and *E.Vincent Harris*

The difference between the entities lies in fact that the second entity has one more dot. In this case, string kernel would be a good choice in verifying they are similar or not. If we consider n=2, we obtain 79-dimensional feature space where the two entities are mapped in Table 3.

In Table 3, λ is the decay factor, in the range of [0,1], that penalizes the longer distance of a subsequence. Formally, string kernel is defined as,

$$K_n(s,t) = \sum_{u \in \sum^n} \langle \phi_u(s) \cdot \phi_u(t) \rangle \qquad (4)$$

	ev	ei	en	$\cdots$	e.	$\cdots$	rs	is
$\phi(evincentharris)$	λ^2	$\lambda^3 + \lambda^{13}$	$\lambda^2 + \lambda^4 + \lambda^7$	$\cdots$	0	$\cdots$	$\lambda^3 + \lambda^4$	$\lambda^2 + \lambda^{12}$
$\phi(e.vincentharris)$	λ^3	$\lambda^4 + \lambda^{14}$	$\lambda^2 + \lambda^5 + \lambda^8$	$\cdots$	λ^2	$\cdots$	$\lambda^3 + \lambda^4$	$\lambda^2 + \lambda^{12}$

Table 3: Feature mapping for two entities

TEAM	headlines	OnWN	FNWN	SMT	mean	rank
RUN1	0.5176	0.1517	0.2496	0.2914	0.3284	77

Table 4: Experimental results for STS-CORE

where $\sum^n$ is the set of all possible subsequences of length n. u indicates an item in the set, for example, the subsequence ev in Table 3. $\phi_u(s)$ is the feature mapping of the subsequences in s. In so doing, we can have similarity between entities in Table 3 as follows:

$K_n(s,t) = \lambda^2 \times \lambda^3 + (\lambda^3 + \lambda^{13}) \times (\lambda^4 + \lambda^{14}) + \cdots + (\lambda^3 + \lambda^4) \times (\lambda^3 + \lambda^4) + (\lambda^2 + \lambda^{12}) \times (\lambda^2 + \lambda^{12})$

To avoid enumeration of all subsequences for similarity measurement, dynamic programming, similar to the method by Lodhi et al. (2002) is used here for similarity calculation.

3 Experiments

The STS-CORE task is to quantify how similar two sentences are. We simply use the sense overlapping approach to compute the similarity. Since this approach needs to find appropriate senses for each word based on its contexts. The number of contextual words is set to 5. Experiments are conducted on four datasets. They are: headlines mined from news sources by European Media, OnWN extracted from from WordNet and OntoNotes, FNWN from WordNet and FrameNet and SMT dataset from DARPA GALE HTER and HyTER. The results of our system (PolyUCOMP-RUN1) are given in Table 4 ,

Our system achieves rather lower performance in the OnWN and FNWN datasets. This is because it is difficult to use contextual terms to find the correct senses for words in sentences of these two datasets. Take the two sentences in OnWN dataset for example,

s_1: *the act of choosing among alternatives*

s_2: *the act of changing one thing for another thing.*

The valid concepts for the two sentences are:

c_1: *06532095-n 05790944-n*

c_2: *00030358-n 00126264-v 00002452-n 00002452-n*

c_1 and c_2 have no shared senses, resulting in a zero similarity between s_1 and s_2. However, s_1 and s_2 should have the same meaning. Moreover, in the FNWN dataset, the sentence lengths are unbalanced, for example,

s_1: *there exist a number of different possible events that may happen in the future. in most cases, there is an agent involved who has to consider which of the possible events will or should occur. a salient entity which is deeply involved in the event may also be mentioned.*

s_2: *doing as one pleases or chooses;*

s_1 has 48 tokens with punctuations being excluded and s_2 has only 6 tokens. This would affect our system performance as well.

For the STS-TYPED task, data set is taken from Europeana, which provides millions of books, paintings, films, museum objects and archival records that have been digitised throughout Europe. Each item has one line per type, where the type can be the title of a record, list of subject terms, textual description of the record, creator of the record and date of the record. Participating systems are supposed to compute similarities between semi-structured items. In this task, we take the strategies in Table 5,

$Jaccard$ denotes the Jaccard similarity measure. $Stringkernel + Jaccard$ means that two types are similar if they share many terms, for example,

TEAM	general	author	people	time	location	event	subject	description	mean	rank
RUN1	0.4888	0.6940	0.3223	0.3820	0.3621	0.1625	0.3962	0.4816	0.4112	12
RUN2	0.4893	0.6940	0.3253	0.3777	0.3628	0.1968	0.3962	0.4816	0.4155	11
RUN3	0.4915	0.6940	0.3254	0.3737	0.3667	0.2207	0.3962	0.4816	0.4187	10

Table 6: Experimental results for STS-TYPED

Type	Strategy
author	String kernel
people	String kernel + Jaccard
time	String kernel + Jaccard
location	String kernel + Jaccard
event	WordNet + Jaccard
subject	Sense overlapping
description	Sense overlapping

Table 5: Strategies for computing similarity

location; and string kernel is used to determine whether two locations are similar or not. For the type of event, we extract verbs from records and count the number of shared verbs between two records. The verb similarity is obtained through WordNet. The general similarity is equal to the average of the 7 scores. Also, Stanford CoreNLP tool[1] is used to extract author, date, time, location and handle part-of-speech tagging.

In this STS-TYPED task, we use string kernel and WordNet to determine whether two terms are similar and increase the number of counts if their similarity exceeds a certain threshold. Therefore, we have chosen 0.4, 0.5 and 0.6 in a heuristic manner and obtained three different runs. Experimental results are given in Table 6.

Since the types of $author$, $subject$ and $description$ are not related to either string kernel or WordNet, their performances remain unchanged during three runs.

4 Conclusions and Future Work

In the Semantic Textual Similarity task of SemEval-2013, to capture the meaning between sentences, we proposed to disambiguate word senses using contexts and then determine sentence similarity by counting the senses they shared. First, word senses are disambiguated by means of the contextual words. When determining similarity between two senses (synsets), n-word overlapping approach is used for counting the number of shared words in two glosses. Besides, string kernel is used to capture similarity between entities to avoid variations between entities. Our approach is simple and we will apply regression models to determine sentence similarity on the basis of these features in future work.

[1]http://nlp.stanford.edu/software/corenlp.shtml

References

Daniel B., Chris Biemann, Iryna Gurevych and Torsten Zesch. 2012. UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures. *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (*SEM 2012).*

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (*SEM 2012).*

Frane Saric, Goran Glavas, Mladen Karan, Jan Snajder and Bojana Dalbelo Basia. 2012. TakeLab: Systems for Measuring Semantic Text Similarity. *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (*SEM 2012).*

Gunes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22(2004):457–479.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text Classification using String Kernels. *The Journal of Machine Learning Research*, 2(2002):419–444.

Jian Xu, Qin Lu and Zhengzhong Liu. 2012. PolyUCOMP: Combining Semantic Vectors with Skip-bigrams for Semantic Textual Similarity.

*Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*.

Ou Jin, Nathan Nan Liu, Yong Yu and Qiang Yang 2011. Transferring Topical Knowledge from Auxiliary Long Text for Short Text Understanding. *Proceedings of the 20th ACM Conference on Information and Knowledge Management (ACM CIKM 2011)*.

Rada Mihalcea and Courtney Corley. 2006. Corpusbased and Knowledge-based Measures of Text Semantic Similarity. *Proceeding of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference.*.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended Gloss Overlaps as a Measure of Semantic Relatedness. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.

Vasileios Hatzivassiloglou, Judith L. Klavans , Eleazar Eskin. 1999. Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning. *Proceeding of Empirical Methods in natural language processing and Very Large Corpora*.

Yuhua Li, David Mclean, Zuhair B, James D. O'shea and Keeley Crockett. 2006. Sentence Similarity Based on Semantic Nets and Corpus Statistics. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1138–1149.