UNIBA: Combining Distributional Semantic Models and Word Sense Disambiguation for Textual Similarity

Pierpaolo Basile and **Annalina Caputo** and **Giovanni Semeraro**
Department of Computer Science
University of Bari Aldo Moro
Via, E. Orabona, 4 - 70125 Bari (Italy)
{firstname.surname}@uniba.it

Abstract

This paper describes the UNIBA team participation in the Cross-Level Semantic Similarity task at SemEval 2014. We propose to combine the output of different semantic similarity measures which exploit Word Sense Disambiguation and Distributional Semantic Models, among other lexical features. The integration of similarity measures is performed by means of two supervised methods based on Gaussian Process and Support Vector Machine. Our systems obtained very encouraging results, with the best one ranked 6^{th} out of 38 submitted systems.

1 Introduction

Cross-Level Semantic Similarity (CLSS) is the task of computing the similarity between two text fragments of different sizes. The task focuses on the comparison between texts at different *lexical levels*, i.e. between a larger and a smaller text. The task comprises four different levels: 1) paragraph to sentence; 2) sentence to phrase; 3) phrase to word; 4) word to sense. The task objective is to provide a framework for evaluating general vs. level-specialized methods.

Our general approach consists in combining scores coming from different semantic similarity algorithms. The combination is performed by a supervised method using the training data provided by the task organizers. The data set comprises pairs of text fragments that can be rated with a score between 0 and 4, where 4 indicates the maximum level of similarity.

We select algorithms which provide similarities at different levels of semantics: surface (or string-based), lexical (word sense disambiguation), and distributional level. The idea is to combine in a unique system the semantic aspects that pertain text fragments.

The following section gives more details about the similarity measures and their combination in a unique score through supervised methods (Section 2). Section 3 describes the system set up for the evaluation and comments on the reported results, while Section 4 concludes the paper.

2 System Description

The idea behind our system is to combine the output of several similarity measures/features by means of a supervised algorithm. Those features were grouped in three main categories. The following three sub-sections describe in detail each feature exploited by the system.

2.1 Distributional Semantics Level

Distributional Semantic Models (DSM) are an easy way for building geometrical spaces of concepts, also known as *Semantic* (or *Word*) *Spaces*, by skimming through huge corpora of text in order to learn the context of word usage. In the resulting space, semantic relatedness/similarity between two words is expressed by the opposite of the distance between points that represent those words. Thus, the semantic similarity can be computed as the cosine of the angle between the two vectors that represent the words. This concept of similarity can be extended to whole sentences by combining words through vector addition ($+$), which corresponds to the point-wise sum of the vector components. Our DSM measure (**DSM**) is based on a *SemanticSpace*, represented by a co-occurrences matrix M, built by analysing the distribution of words in the British National Corpus (BNC). Then, M is reduced using the Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997). Vector addition and cosine similarity are

Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 748–753,
Dublin, Ireland, August 23-24 2014.

then used for building the vector representation of each text fragment and computing their pairwise similarity, respectively.

2.2 Lexical Semantics Level

Word Sense Disambiguation. Most of our measures rely on the output of a Word Sense Disambiguation (WSD) algorithm. Our newest approach to WSD, recently presented in Basile et al. (2014), is based on the simplified Lesk algorithm (Vasilescu et al., 2004). Each word w_i in a sequence $w_1 w_2 ... w_n$ is disambiguated individually by choosing the sense that maximizes the similarity between the gloss and the context of w_i (i.e. the whole text where w_i occurs). To boost the overlap between the context and the gloss, this last is expanded with glosses of related meanings, following the approach described in Banerjee and Pedersen (2002). As sense inventory we choose BabelNet 1.1, a huge multilingual semantic network which comprises both WordNet and Wikipedia (Navigli and Ponzetto, 2012). The algorithm consists of the following steps:

1. *Building the glosses.* We retrieve all possible word meanings for the target word w_i that are listed in BabelNet. BabelNet mixes senses in WordNet and Wikipedia. First, senses in WordNet are searched for; if no sense is found (as often happens with named entities), senses for the target word are sought in Wikipedia. We preferred that strategy rather than retrieving senses from both sources at once because this last approach produced worse results when tuning the system. Once the set of senses $S_i = \{s_{i1}, s_{i2}, ..., s_{ik}\}$ associated to the target word w_i has been retrieved, gloss expansion occurs. For each sense s_{ij} of w_i, the algorithm builds the sense extended gloss g^*_{ij} by appending the glosses of meanings related to s_{ij} to its original gloss g_{ij}. The related meanings, with the exception of "antonym" senses, are the output of the BabelNet function "getRelatedMap". Moreover, each word in g^*_{ij} is weighted by a function inversely proportional to the distance between s_{ij} and its related meaning. The distance d is computed as the number of edges linking two senses in the graph. The function takes also into account the frequencies of the words in all the glosses giving more emphasis to the most discriminative words; this can be considered as a variation of the inverse document frequency (*idf*) for retrieval that we named inverse gloss frequency (*igf*). The *igf* for a word w_k occurring gf^*_k times in the set of extended glosses for all the senses in S_i, the sense inventory of w_i, is computed as $IGF_k = 1 + log_2 \frac{|S_i|}{gf^*_k}$. The final weight for the word w_k appearing h times in the extended gloss g^*_{ij} is given by:

$$weight(w_k, g^*_{ij}) = h \times IGF_k \times \frac{1}{1+d} \quad (1)$$

2. *Building the context.* The context C for the word w_i is represented by all the words that occur in the text.

3. *Building the vector representations.* The context C and each extended gloss g^*_{ij} are represented as vectors in the *SemanticSpace* built through the DSM described in Subsection 2.1.

4. *Sense ranking.* The algorithm computes the cosine similarity between the vector representation of each extended gloss g^*_{ij} and that of the context C. Then, the cosine similarity is linearly combined with the probability $p(s_{ij}|w_i)$, which takes into account the sense distribution of s_{ij} given the word w_i. The sense distribution is computed as the number of times the word w_i was tagged with the sense s_{ij} in SemCor, a collection of 352 documents manually annotated with WordNet synsets. T he additive (Laplace) smoothing prevents zero probabilities, which can occur when some synsets do not appear in SemCor. The probability is computed as follows:

$$p(s_{ij}|w_i) = \frac{t(w_i, s_{ij}) + 1}{\#w_i + |S_i|} \quad (2)$$

   The output of this step is a ranked list of synsets.

The WSD measure (**WSD**) is computed on the top of the output of the last step. For each text fragment, we build a Bag-of-Synset (BoS) as the sum, over the whole text, of the weighted synsets associated with each word. Then, we compute the WSD similarity as the cosine similarity between the two BoS.

Graph. A sub-graph of BabelNet is built for each text fragment starting from the synsets provided by the WSD algorithm. For each word the synset with the highest score is selected, then this initial set is expanded with the related synsets in BabelNet. We apply the Personalized Page Rank (Haveliwala, 2002) to each sub-graph where the synset scores computed by the WSD algorithm are exploited as prior probabilities. The weighted rank of synsets provided by Page Rank is used to build the BoS of the two text fragments, then the Personalized Page Rank (**PPR**) is computed as the cosine similarity between them.

Synset Distributional Space. Generally, similarity measures between synsets rely on the synsets hierarchy in a semantic network (e.g. WordNet). We define a new approach that is completely different, and represents synsets as points in a geometric space that we call SDS (Synset Distributional Space). SDS is generated taking into account the synset relationships, and similarity is defined as the synsets closeness in the space. We build a symmetric matrix S which contains synsets on both rows and columns. Each cell in the matrix is set to one if a semantic relation exists between the corresponding synsets. The relationships are extracted from BabelNet limiting synsets to those occurring also in WordNet, while synsets coming from Wikipedia are removed to reduce the size of S. The method for building the matrix S relies on Reflective Random Indexing (RRI) (Cohen et al., 2010), a variation of the Random Indexing technique for matrix reduction (Kanerva, 1988). RRI retains the advantages of RI which incrementally builds a reduced space where distance between points is nearly preserved. Moreover, cyclical training, i.e. the retraining of a new space exploiting the RI output as basis vectors, makes indirect inference to emerge. Two different similarity measures can be defined by exploiting this space for representing synsets: **WSD-SDS** and **PPR-SDS**, based on **WSD** and **PPR** respectively. Each BoS is represented as the sum of the synset vectors in the SDS space. Then, the similarity is computed as the cosine similarity between the two vector representations.

2.3 Surface Level

At the surface level, we compute the following features:

EDIT The edit, or Levenshtein, distance between the two texts;

MCS The most common subsequence between the two texts;

2-gram, 3-gram For each text fragment, we build the *Bag-of-*n*-gram* (with n varying in $\{2, 3\}$); then we compute the cosine similarity between the two *Bag-of-*n*-gram* representations.

BOW For each tokenized text fragment, we build its Bag-of-Word, and then compute the cosine similarity between the two BoW.

L1 The length in characters of the first text fragment;

L2 The length in caracters of the second text fragment;

DIFF The difference between **L1** and **L2**.

2.4 Word to Sense

The *word to sense* level is different from the other ones: in this case the similarity is computed between a word and a particular word meaning. Since a word meaning is not a text fragment, this level poses a new challenge with respect to the classical text similarity task. In this case we decide to consider the word on its own as the first text fragment, while for the second text fragment we build a dummy text using the BabelNet gloss assigned to the word sense. In that way, the distributional and the lexical measures (WSD, Graph, and DSM) can be applied to both fragments. Table 1 recaps the features used for each task.

3 Evaluation

Dataset Description. The SemEval-2014 Task 3 *Cross-Level Semantic Similarity* is designed for evaluating systems on their ability to capture the semantic similarity between lexical items of different length (Jurgens et al., 2014). To this extent, the organizers provide four different levels of comparison which correspond to four different datasets: 1) Paragraph to Sentence (*Par2Sent*); 2) Sentence to Phrase (*Sent2Ph*); 3) Phrase to Word (*Ph2W*); and 4) Word to Sense (*W2Sense*).

For each dataset, the organizer released trial, training and test data. While the trial includes a few examples (approximately 40), both training and test data comprise 500 pairs of text fragments.

Run	*Par2Sent*	*Sent2Ph*	*Ph2W*	*W2Sense*	*Official Rank*	*Spearman Correlation*
bestTrain	.861	.793	.555	.420	-	-
LCS	*.527*	*.562*	*.165*	*.109*	-	-
run1	.769	.729	.229	.165	7	10
run2	.784	.734	.255	.180	6	8
run3	.769	.729	.229	.165	8	11

Table 2: Task results.

Feature	*Par2Sent* *Sent2Ph* *Ph2W*	*W2Sense*
DSM	√	√
WSD	√	√
PPR	√	√
WSD-SDS	√	√
PPR-SDS	√	√
EDIT	√	-
MCS	√	-
2-*gram*	√	-
3-*gram*	√	-
BOW	√	-
L1	√	-
L2	√	-
DIFF	√	-

Table 1: Features per task.

Each pair is associated with a human-assigned similarity score, which varies from 4 (very similar) to 0 (unrelated). Organizers provide the normalized Longest Common Substring (LCS) as baseline. The evaluation is performed through the Pearson (official rank) and the Spearman's rank correlation.

System setup. We develop our system in JAVA relying on the following resources:

- Stanford CoreNLP to pre-process the text: tokenization, lemmatization and PoS-tagging are applied to the two text fragments;
- BabelNet 1.1 as knowledge-base in the WSD algorithm;
- JAVA JUNG library for Personalized Page Rank;
- British National Corpus (tokenized text with stop word removal) and SVDLIB to build the *SemanticSpace* described in Subsection 2.1;
- A proprietary implementation of Reflective Random Indexing to build the distributional space based on synsets (SDS) extracted from BabelNet (we used two cycles of retraining);
- Weka for the supervised approach.

After a tuning step using both training and trial data provided by organizers, we selected three different supervised systems: Gaussian Process with Puk kernel ($run1$), Gaussian Process with RBF kernel ($run2$), and Support Vector Machine Regression with Puk kernel ($run3$). All the systems are implemented with the default parameters set by Weka. We trained a different model on each dataset. The DSM is built using the $100,000$ most frequent terms in the BNC, while the co-occurrences are computed on a window size of 5 words. The vector dimension is set to 400, the same value is adopted for building the SDS, where the number of seeds (no zero components) generated in the random vectors is set to 10 with one step of retraining. The total number of synset vectors in the SDS is $576,736$. In the WSD algorithm, we exploited the whole sentence as context. The linear combination between the cosine similarity and the probability $p(s_{ij}|w_i)$ is performed with a factor of 0.5. The distance for expanding a synset with its related meaning is set to one. The same depth is used for building the graph in the PPR method, where we fixed the maximum number of iterations up to 50 and the dumpling factor to 0.85.

Results. Results of our three systems for each similarity level are reported in Table 2 with the baseline provided by the organizer (LCS). Our three systems always outperform the LCS baseline. Table 2 also shows the best results ($bestTrain$) obtained on the training data by a Gaussian Process with Puk kernel and a 10-fold cross-validation. Support Vector Machine and Gaussian Process with Puk kernel, $run1$ and $run3$ respectively, produce the same results. Comparing

Task	DSM	WSD	PPR	$WSD\text{-}SDS$	$PPR\text{-}SDS$	$EDIT$	MCS	2-$gram$	3-$gram$	BOW	$L1$	$L2$	$DIFF$
$Par2Sent$	.612	.697	.580	.129	.129	.461	.44	.630	.478	.585	.002	.231	.116
$Sent2Ph$	.540	.649	.641	.110	.110	.526	.474	.376	.236	.584	.069	.357	.218
$Ph2W$	.228	.095	.094	.087	.087	.136	.120	-	-	.095	.079	.013	.071
$W2Sense$	.147	.085	-.062	.084	.062								

Table 3: Individual measures for each task.

these figures with those obtained on training data ($run1$ and $run3$ vs. $bestTrain$), we can observe that the Puk kernel tends to over-fit on training data, while RBF kernel seems to be less sensitive to this problem.

We analysed also the performance of each measure on its own; results in Table 3 are obtained by training the best supervised system ($run2$) with default parameters on each feature individually. WSD obtains the best results in the first two levels, while DSM is the best method in the last two ones. This behaviour can be ascribed to the size of the text fragments. In large text fragments the WSD algorithm can rely on wider contexts to obtain good performance; while in short texts information about context is poor. At the $W2Sense$ level, the measure based on the Personalized Page Rank obtains the worst results; however, we noticed that the ablation of that feature causes a drop in performance of the supervised systems.

After the submission deadline, we noticed that sometimes PoS-tagging produced wrong results on small texts. This incorrect behaviour influenced negatively the correct retrieval of synsets from BabelNet. Thus, we decided to exclude PoS-tagging for text fragments with less than three words. In such a case, all the synsets for a given word are retrieved. Making this adjustment, we were able to obtain the improvements ($\Delta\%$) with respect to the submitted runs reported on Table 4.

Run	$Ph2W$	$\Delta\%$	$W2Sense$	$\Delta\%$
run1	.263	+14.85	.242	+46.67
run2	.257	+ 0.78	.237	+31.67
run3	.263	+14.85	.242	+46.66

Table 4: Results after PoS-tagging removal for short text (< 3 words).

4 Conclusions

We have reported the results of our participation in the cross-level semantic similarity task of SemEval-2014. Our systems combine different similarity measures based on string-matching, word sense disambiguation and distributional semantic models. Our best system ranks $6th$ out of the 38 participants in the task with respect to the Pearson correlation, while it ranks $8th$ when Spearman was used. These results suggest that our methods are robust with respect to the evaluation measures.

Acknowledgments

This work fulfils the research objectives of the PON 02_00563_3470993 project "VINCENTE - A Virtual collective INtelligenCe ENvironment to develop sustainable Technology Entrepreneurship ecosystems" funded by the Italian Ministry of University and Research (MIUR).

References

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted lesk algorithm for word sense disambiguation using wordnet. In *Computational linguistics and intelligent text processing*, pages 136–145. Springer.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An Enhanced Lesk Word Sense Disambiguation algorithm through a Distributional Semantic Model. In *Proceedings of COLING 2014*, Dublin, Ireland, August. (http://www.coling-2014.org/accepted-papers.php, in press).

Trevor Cohen, Roger Schvaneveldt, and Dominic Widdows. 2010. Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections. *Journal of Biomedical Informatics*, 43(2):240 – 256.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference*

on World Wide Web, WWW '02, pages 517–526, New York, NY, USA. ACM.

David Jurgens, Mohammad Taher Pilehvar, and Roberto Navigli. 2014. Semeval-2014 task 3: Cross-level semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, Dublin, Ireland, August 23–24.

Pentti Kanerva. 1988. *Sparse Distributed Memory*. MIT Press.

Thomas K. Landauer and Susan T. Dumais. 1997. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104(2):211–240.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Florentina Vasilescu, Philippe Langlais, and Guy Lapalme. 2004. Evaluating variants of the lesk approach for disambiguating words. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 633–636.