UNIBA-CORE: Combining Strategies for Semantic Textual Similarity

Annalina Caputo **Pierpaolo Basile** **Giovanni Semeraro**
Department of Computer Science
University of Bari Aldo Moro
Via E. Orabona, 4 - 70125 Bari, Italy
{annalina.caputo, pierpaolo.basile, giovanni.semeraro}@uniba.it

Abstract

This paper describes the UNIBA participation in the Semantic Textual Similarity (STS) core task 2013. We exploited three different systems for computing the similarity between two texts. A system is used as baseline, which represents the best model emerged from our previous participation in STS 2012. Such system is based on a distributional model of semantics capable of taking into account also syntactic structures that glue words together. In addition, we investigated the use of two different learning strategies exploiting both syntactic and semantic features. The former uses ensemble learning in order to combine the best machine learning techniques trained on 2012 training and test sets. The latter tries to overcome the limit of working with different datasets with varying characteristics by selecting only the more suitable dataset for the training purpose.

1 Introduction

Semantic Textual Similarity is the task of computing the similarity between any two given texts. The task, in its core formulation, aims at capturing the different kinds of similarity that emerge from texts. Machine translation, paraphrasing, synonym substitution or text entailment are some fruitful methods exploited for this purpose. These techniques, along with other methods for estimating the text similarity, were successfully employed via machine learning approaches during the 2012 task.

However, the STS 2013 core task (Agirre et al., 2013) differs from the 2012 formulation in that it provides a test set which is similar to the training, but not drawn from the same set of data. Hence, in order to generalize the machine learning models trained on a group of datasets, we investigate the use of combination strategies. The objective of combination strategies, known under the name of *ensemble learning*, is that of reducing the bias-variance decomposition through reducing the variance error. Hence, this class of methods should be more robust with respect to previously unseen data. Among the several ensemble learning alternatives, we exploit the *stacked generalization* (STACKING) algorithm (Wolpert, 1992). Moreover, we investigate the use of a two-steps learning algorithm (2STEPSML). In this method the learning algorithm is trained using only the dataset most similar to the instance to be predicted. The first step aims at predicting the dataset more similar to the given pair of texts. Then the second step makes use of the previously trained algorithm to predict the similarity value. The baseline for the evaluation is represented by our best system (DSM_PERM) resulting from our participation in the 2012 task. After introducing the general models behind our systems in Section 2, Section 3 describes the evaluation setting of our systems along with the experimental results. Then, some conclusions and remarks close the paper.

2 General Models

2.1 Dependency Encoding via Vector Permutations

Distributional models are effective methods for representing word paradigmatic relations in a simple way through vector spaces (Mitchell and Lapata, 2008). These spaces are built taking into account the word context, hence the resulting vector representation is such that the distance between vectors reflects their similarity. Although several definitions of context are possible (e.g. a sliding window of text, the word order or syntactic dependencies), in their plain definition these kinds of models account for just one type of context at a time. To overcome this limitation, we exploit a method to encode more definitions of context in the same vector exploiting the vector permutations (Caputo et al., 2012). This technique, which is based on Random Indexing as a means for computing the distributional model, is based on the idea that when the components of a highly sparse vector are shuffled, the resulting vector is nearly orthogonal to the original one. Hence, vector permutation represents a way for generating new random vectors in a predetermined manner. Different word contexts can be encoded using different types of permutations. In our distributional model system (DSM_PERM), we encode the syntactic dependencies between words rather than the mere co-occurrence information. In this way, word-vector components bear the information about both co-occurring and syntactically related words. In this distributional space, a text can be easily represented as the superposition of its words. Then, the vector representation of a text is given by adding the vector representation of its words, and the similarity between texts come through the cosine of the angle between their vector representations.

2.2 Stacking

Stacking algorithms (Wolpert, 1992) are a way of combining different types of learning algorithms reducing the variance of the system. In this model, the meta-learner tries to predict the real value of an instance combining the outputs of other machine learning methods.

Figure 1 shows how the learning process takes place. The level-0 represents the ensemble of different models to be trained on the same dataset. The level-0 outputs build up the level-1 dataset: an instance at this level is represented by the numeric values predicted by each level-0 model along with the gold standard value. Then, the objective of the level-1 learning model is to learn how to combine the level-0 outputs in order to provide the best prediction.

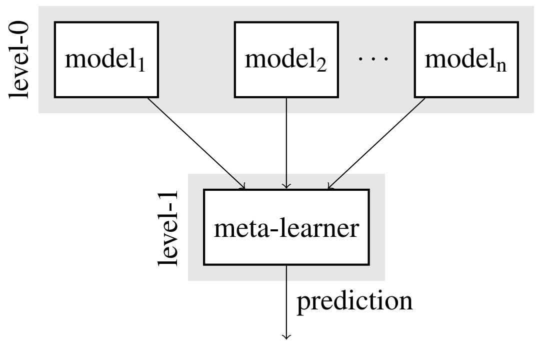

Figure 1: Stacking algorithm

2.3 Two steps learning algorithm

Given an ensemble of datasets with different characteristics, this method is based on the idea that when instances come from a specific dataset, the learning algorithm trained on that dataset outperforms the same algorithm trained on the whole ensemble.

Hence, the two steps algorithm tries to overcome the problem of dealing with different datasets having different characteristics through a classification model.

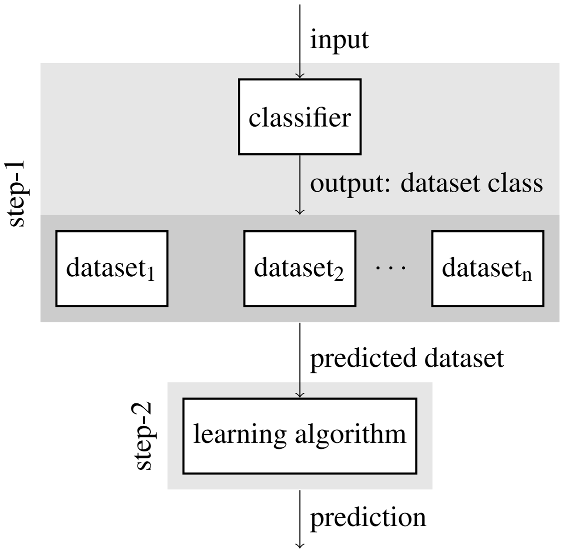

Figure 2: Two steps machine learning algorithm

In the first step (Figure 2), a different class is assigned to each dataset. The classifier is trained on a set of instances whose classes correspond to the dataset numbers. Then, given a new instance the output of this step will be the dataset to be used for training the learning algorithm in the step 2. In the second step, the learning algorithm is trained on the dataset choose in the first step. The output of this step is the predicted similarity between the two texts. Through these steps, it is possible to select the dataset with the characteristics more similar to a given instance, and exploit just this set of data for learning the algorithm.

2.4 Features

Both STACKING and 2STEPSML systems rely on several kinds of features, which vary from lexical to semantic ones. Features are grouped in seven main classes, as follows:

1. **Character/string/annotation**-based features: the length of the longest common contiguous substring between the texts; the Jaccard index of both tokens and lemmas; the Levenshtein distance between texts; the normalized number of common 2-grams, 3-grams and 4-grams; the total number of tokens and characters; the difference in tokens and characters between texts; the normalized difference with respect to the max text length in tokens and characters between texts. Exploiting other linguistic annotations extracted by Stanford CoreNLP[1], we compute the Jaccard index between PoS-tags and named entities. Using WordNet we extract the Jaccard index between the first sense and its super-sense tag.

2. **Textual Similarity**-based features: a set of features based on the textual similarity proposed by Mihalcea (Mihalcea et al., 2006). Given two texts T_1 and T_2 the similarity is computed as follows:

$$sim(T_1, T_2) = \frac{1}{2}\Big(\frac{\sum_{w \in T_1} maxSim(w, T_2)}{\sum_{w \in T_1} idf(w)} + \frac{\sum_{w \in T_2} maxSim(w, T_1)}{\sum_{w \in T_2} idf(w)}\Big) \quad (1)$$

We adopt several similarity measures using semantic distributional models (see Section 2.5), the Resnik's knowledge-based approach (Resnik, 1995) and the point-wise mutual information as suggested by Turney (Turney, 2001) computed on British National Corpus[2]. For all the features, the idf is computed relying on UKWaC corpus[3] (Baroni et al., 2009).

3. **Head similarity**-based features: this measure takes into account the maximum similarity between the roots of each text. The roots are extracted using the dependency parser provided by Stanford CoreNLP. The similarity is computed according to the distributional semantic models proposed in Section 2.5.

4. **ESA similarity**: computes the similarity between texts using the Explicit Semantic Analysis (ESA) approach (Gabrilovich and Markovitch, 2007). For each text we extract the ESA vector built using the English Wikipedia, and then we compute the similarity as the cosine similarity between the two ESA vectors.

5. **Paraphrasing** features: this is a very simple measure which counts the number of possible paraphrasings belonging to the two texts. Given two texts T_1 and T_2, for each token in T_1 a list of paraphrasings is extracted using a dictionary[4]. If T_2 contains one of the paraphrasing in the list, the score is incremented by one. The final score is divided by the number of tokens in T_1. The same score is computed taking into account T_2. Finally, the two score are added and divided by 2.

6. **Greedy Lemma Aligning Overlap** features: this measure computes the similarity between texts using the semantic alignment of lemmas as proposed by Šarić et al. (2012). In order to compute the similarity between lemmas, we exploit the distributional semantic models described in Section 2.5.

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

[2]Available at: http://www.natcorp.ox.ac.uk/

[3]Available at: http://wacky.sslmit.unibo.it/

[4]English Thesaurus for StarDict available at https://aur.archlinux.org/packages/stardict-thesaurus-ee/

7. **Compositional** features: we build several similarity features using the distributional semantic models described in Section 2.5 and a compositional operator based on sum. This approach is thoroughly explained in Section 2.6

2.5 Distributional semantic models

In several features proposed in our approaches, the similarity between words is computed using Distributional Semantic Models. These models represent word meanings through contexts: the different meanings of a word can be accounted for by looking at the different contexts wherein the word occurs. This insight can beautifully be expressed by the geometrical representation of words as vectors in a *semantic space*. Each term is represented as a vector whose components are contexts surrounding the term. In this way, the meaning of a term across a corpus is thoroughly conveyed by the contexts it appears in, where a context may typically be the set of co-occurring words in a document, in a sentence or in a window of surrounding terms.

In particular, we take into account two main classes of models: Simple Distributional Spaces and Structured Semantic Spaces. The former considers as context the co-occurring words, the latter takes into account both co-occurrence and syntactic dependency between words.

Simple Distributional Spaces rely on Latent Semantic Analysis (LSA) and Random Indexing (RI) in order to reduce the dimension of the co-occurrences matrix. Moreover, we use an approach which applies LSA to the matrix produced by RI.

Structured Semantic Spaces are based on two techniques to encode syntactic information into the vector space. The first approach uses the vector permutation of random vector in RI to encode the syntactic role (head or dependent) of a word. The second method is based on Holographic Reduced Representation, in particular using convolution between vectors, to encode syntactic information.

Adopting distributional semantic models, each word can be represented as a vector in a geometric space. The similarity between two words can be easily computed taking into account the cosine similarity between word vectors.

All models are described in Basile et al. (2012).

2.6 Compositional features

In Distributional Semantic Models, given the vector representations of two words, it is always possible to compute their similarity as the cosine of the angle between them.

However, texts are composed by several terms, so in order to compute the similarity between them we need a method to compose words occurring in these texts. It is possible to combine words through the vector addition $(+)$. This operator is similar to the superposition defined in connectionist systems (Smolensky, 1990), and corresponds to the point-wise sum of components:

$$\mathbf{p} = \mathbf{u} + \mathbf{v} \tag{2}$$

where $p_i = u_i + v_i$

The addition is a commutative operator, which means that it does not take into account any order or underlying structures existing between words. In this first study, we do not exploit more complex methods to combine word vectors. We plan to investigate them in future work.

Given a text p, we denote with $\mathbf{p}$ its vector representation obtained applying addition operator $(+)$ to the vector representation of terms it is composed of. Furthermore, it is possible to compute the similarity between two texts exploiting the cosine similarity between vectors.

Formally, if $a = a_1, a_2...a_n$ and $b = b_1, b_2...b_m$ are two texts, we build two vectors $\mathbf{a}$ and $\mathbf{b}$ which represent respectively the two texts in a semantic space. Vector representations for the two texts are built applying the addition operator to the vector representation of words belonging to them:

$$\begin{aligned} \mathbf{a} &= a_1 + a_2 + \ldots + a_n \\ \mathbf{b} &= b_1 + b_2 \ldots + b_m \end{aligned} \tag{3}$$

The similarity between $\mathbf{a}$ and $\mathbf{b}$ is computed as the cosine similarity between them.

3 Experimental evaluation

SemEval-2013 STS is the second attempt to provide a "*unified framework for the evaluation of modular semantic textual similarity and to characterize their impact on NLP applications*". The task consists in computing the similarity between pair of texts, returning a similarity score. The test set is composed by data coming from the following datasets: news headlines (headlines); mapping of lexical resources from Ontonotes to Wordnet (OnWN) and from FrameNet to WordNet (FNWN); and evaluation of machine translation (SMT).

The training data for STS-2013 is made up by training and testing data from the previous edition of STS-2012 task. During the 2012 edition, STS provided participants with three training data: MSR-Paraphrase, MSR-Video, STMeuropar; and five testing data: MSR-Paraphrase, MSR-Video, STMeuropar, SMTnews and OnWN. It is important to note that part of 2012 test sets were made up from the same sources of the training sets. On the other hand, STS-2013 training and testing are very different, making the prediction task a bit harder.

Humans rated each pair of texts with values from 0 to 5. The evaluation is performed by comparing the humans scores against system performance through Pearson's correlation with the gold standard for the four datasets.

3.1 System setup

For the evaluation, we built the distributional spaces using the WaCkypedia_EN corpus[5]. WaCkypedia_EN is based on a 2009 dump of the English Wikipedia (about 800 million tokens) and includes information about: part-of-speech, lemma and a full dependency parsing performed by MaltParser (Nivre et al., 2007). The structured spaces described in Subsections 2.1 and 2.5 are built exploiting information about term windows and dependency parsing supplied by WaCkypedia. The total number of dependencies amounts to about 200 million.

The RI system is implemented in Java and relies on some portions of code publicly available in the Semantic Vectors package (Widdows and Ferraro, 2008), while for LSA we exploited the publicly available C library SVDLIBC[6].

We restricted the vocabulary to the 50,000 most frequent terms, with stop words removal and forcing the system to include terms which occur in the dataset.

Semantic space building involves some parameters. In particular, each semantic space needs to set up the dimension k of the space. All spaces use a dimension of 500 (resulting in a 50,000$\times$500 matrix). The number of non-zero elements in the random vector is set to 10. When we apply LSA to the output space generated by the Random Indexing we hold all the 500 dimensions, since during the tuning we observed a drop in performance when a lower dimension was set. The co-occurrence distance w between terms was set up to 4.

In order to compute the similarity between the vector representations of text using UNIBA-DSM_PERM, we used the cosine similarity, and then we multiplied by 5 the obtained value.

The two supervised methods, UNIBA-2STEPML and UNIBA-STACKING, are developed in Java using Weka[7] to implement the learning algorithms. Regarding the stacking approach (UNIBA-STACKING) we used for the level-0 the following models: Gaussian Process with polynomial kernel, Gaussian Process with RBF kernel, Linear Regression, Support Vector regression with polynomial kernel, and decision tree. The level-1 model uses a Gaussian Process with RBF kernel. In the first step of UNIBA-2STEPML we adopt Support Vector Machine, while in the second one we use Support Vector Machine for regression. In both steps, the RBF-Kernel is used. Features are normalized removing non alphanumerics characters. In all the learning algorithms, we use the default parameters set by Weka. As future work, we plan to perform a tuning step in order to set the best parameters.

The choice of the learning algorithms for both UNIBA-STACKING and UNIBA-2STEPSML systems was performed after a tuning phase where only the STS-2012 training datasets were exploited. Table 1 reports the values obtained by our three systems on the STS-2012 test sets. After the tuning, we came up with the learning algorithms to employ in the level-0 and level-1 of UNIBA-STACKING and in step-1 and step-2 of UNIBA-2STEPSML. Then, the training of both UNIBA-STACKING and UNIBA-2STEPSML was performed on all STS-2012 datasets (training and test data).

[5]http://wacky.sslmit.unibo.it/doku.php?id=corpora

[6]http://tedlab.mit.edu/ dr/SVDLIBC/

	MSRpar	MSRvid	SMTeuroparl	OnWN	SMTnews	mean
UNIBA-2STEPSML	.6056	.8573	.6233	.5079	.4533	.7016
UNIBA-DSM_PERM	.4349	.7592	.5324	.6593	.4559	.6172
UNIBA-STACKING	.6473	.8727	.5344	.6646	.4604	.7714

Table 1: STS-2012 test results of Pearson's correlation.

	headlines	OnWN	FNWN	SMT	mean	rank
UNIBA- 2STEPSML	.4255	.4801	.1832	.2710	.3673	71
UNIBA- DSM_PERM	.6319	4910	.2717	.3155	.4610	54
UNIBA- STACKING	.6275	.4658	.2111	.2588	.4293	61

Table 2: Evaluation results of Pearson's correlation for individual datasets.

3.2 Evaluation results

Evaluation results on the STS-2013 data are reported in Table 2. Among the three systems, UNIBA-DSM_PERM obtained the best performances on both individual datasets and in the overall evaluation metric (mean), which computes the Pearson's correlation considering all datasets combined in a single one. The best system ranked 54 over a total of 90 submissions, while UNIBA-STACKING and UNIBA-2STEPSML ranked 61 and 71 respectively. These results are at odds with those reported in Table 1. During the test on 2012 dataset, UNIBA-STACKING gave the best result, followed by UNIBA-2STEPSML, while UNIBA-DSM_PERM gave the worst performance. The UNIBA-STACKING system corroborated our hypothesis giving also the best results on those datasets not exploited during the training phase of the system (OnWN, SMTnews). Conversely, UNIBA-2STEPSML reported a different trend showing its weakness with respect to a high variance in the data, and performing worse than UNIBA-DSM_PERM on the OnWN and SMTnews datasets.

However, the evaluation results have refuted our hypothesis, even with the use of the stacking system. The independence from a training set makes the UNIBA-DSM_PERM system more robust than other supervised algorithms, even though it is not able to give always the best performance on individual datasets, as highlighted by results in Table 1.

[7]http://www.cs.waikato.ac.nz/ml/weka/

4 Conclusions

This paper reports on UNIBA participation in Semantic Textual Similarity 2013 core task. In this task edition, we exploited both distributional models and machine learning techniques to build three systems. A distributional model, which takes into account the syntactic structure that relates words in a corpus, has been used as baseline. Moreover, we investigate the use of two machine learning techniques as a means to make our systems more independent from the training data. However, the evaluation results have highlighted the higher robustness of the distributional model with respect to these systems.

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

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In **SEM 2013: The Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2012. A study on compositional semantics of words in distributional spaces. In *Sixth IEEE International Conference on Semantic Computing, ICSC 2012, Palermo, Italy, September 19-21, 2012*, pages 154–161. IEEE Computer Society.

Annalina Caputo, Pierpaolo Basile, and Giovanni Semeraro. 2012. Uniba: Distributional semantics for textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 591–596, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on artificial intelligence*, volume 6, page 12.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the national conference on artificial intelligence*, volume 21, pages 775–780. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 236–244. The Association for Computer Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, November.

Peter Turney. 2001. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 441–448, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Dominic Widdows and Kathleen Ferraro. 2008. Semantic Vectors: A Scalable Open Source Package and Online Technology Management Application. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC2008)*, pages 1183–1190, Marrakech, Morocco. European Language Resources Association (ELRA).

David H. Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259, February.