Automatically Selecting Complementary Vector Representations for Semantic Textual Similarity

Julien Hay, Tim Van de Cruys, Philippe Muller, Bich-Liên Doan, Fabrice Popineau and Ouassim Ait-Elhara

Abstract The goal of the Semantic Textual Similarity task is to automatically quantify the semantic similarity of two text snippets. Since 2012, the task has been organized on a yearly basis as a part of the *SemEval* evaluation campaign. This paper presents a method that aims to combine different sentence-based vector representations in order to improve the computation of semantic similarity values. Our hypothesis is that such a combination of different representations allows us to pinpoint different semantic aspects, which improves the accuracy of similarity computations. The method's main difficulty lies in the selection of the most complementary representations, for which we present an optimization method. Our final system is based on the winning system of the 2015 evaluation campaign, augmented with the complementary vector representations selected by our optimization method. We also present evaluation results on the dataset of the 2016 campaign, which confirms the benefit of our method.

J. Hay (✉) · O. Ait-Elhara
Octopeek, 22 Rue du Général de Gaulle, 95880 Paris, Enghien-les-Bains, France
e-mail: julien.hay@lri.fr; julien.hay@octopeek.com

O. Ait-Elhara
e-mail: ouassim.aitelhara@octopeek.com

J. Hay · B.-L. Doan · F. Popineau
LRI, Bat 650, Rue Noetzlin, 91190 Paris, Gif-sur-Yvette, France
e-mail: bich-lien.doan@centralesupelec.fr

F. Popineau
e-mail: fabrice.popineau@centralesupelec.fr

T. Van de Cruys · P. Muller
IRIT, Université Toulouse III Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse, France
e-mail: tim.vandecruys@irit.fr

P. Muller
e-mail: muller@irit.fr

B. Pinaud et al. (eds.), *Advances in Knowledge Discovery and Management*, Studies in Computational Intelligence 834,
https://doi.org/10.1007/978-3-030-18129-1_3

1 Introduction

Many recent work focus on semantic similarity, either between words or between groups of words, that is syntagms, sentences or even complete documents. By gathering words or phrases by their meanings, we can use semantic features in models while avoiding the dispersion inherent to the vocabulary size or to the space of possible sentences. Most works in this direction calculate similarities between representations built on distributional bases, i.e. where the similarity of meaning derives from the similarity of contexts in which the word appear, an assumption stated by Harris (1954). The representations take the form of vectors, matrices and distribution tensors (Turney and Pantel 2010), where the dimensions correspond to lexical (Curran 2004), syntactic co-occurrences (Baroni and Lenci 2010) or transformations of these contexts, either using dimensionality reduction (Pennington et al. 2014) or by learning vectors with neural networks (Mikolov et al. 2013a, b).

The evaluation of these vector space models relies either on external tasks where they are involved, or on intrinsic measurements, based on samples of similar words, or groups of similar words.

More recently, the notion of semantic textual similarity motivates vector representations beyond words alone. The document/sentence level representation can be made by the composition of lexical representations (Mitchell and Lapata 2008; Van de Cruys et al. 2013) or by vectors learned via an ad-hoc neural network (Le and Mikolov 2014; Baroni and Zamparelli 2010).

Semantic similarity works can rely on annotated data of pairs of sentences categorized in paraphrasing/not paraphrasing, since (Dolan and Brockett 2005). More gradual data, with different levels of similarity, emerged with the organization of the first *STS* (Semantic Textual Similarity) task during the 2012 *SemEval* evaluation campaign.

The methods applied to paraphrases recognition or measurement of similarities are therefore based on vector representations and the different ways of combining them, and also use matching of the sentence elements (Sultan et al. 2015). The main issue with the vector representation is that learning them and combining them for textual similarity relies on numerous hyperparameters. Furthermore, there is little work that attempts to combine different representations to take advantage of possible complementarities of the hyperparameters choices.

We hereby present our study on the combination of vector representations to explore the potential of associating these representations at different scales. In the next section, we will delve deeper into the *STS* task with a brief state of the art about the *SemEval* campaign. Section 3 will present the hypotheses that motivated the search for complementarity of vector representations. We will then detail the method we used to find complementarity vectors through two algorithms optimizing two criteria in Sect. 4. Finally, we will discuss the results obtained by comparing them to the results of the 2016 *SemEval* campaign.

2 Semantic Textual Similarity

We briefly go deeper into vector representation models of words and documents to discuss their interest in the specific task of measuring the textual similarity.

2.1 Vector Representation Models

The representation of text in vector space is a technique that has been used increasingly often through recent years in several natural language processing (*NLP*) tasks, such as sentiment analysis and machine translation (Le and Mikolov 2014). This representation makes it possible to bring together words, sentences, and, in a general way, text snippets, without going through an accurate representation of all the elements which compose the general meaning. The vector representation thus makes it possible to build a shape that can position those elements relatively to each other.

Each word being reduced to a sequence of numbers, it is therefore possible to calculate a similarity with simple measures such as cosine similarity. The construction of vector representations can be achieved through unsupervised learning on large corpora. The various techniques used in the construction of a vector take into account, for each word, their neighbors in the text, i.e. words that are close in a sentence: namely the "context".

For the high level representation, vectors can be combined using numerous methods. First we can merge vectors of a text snippet by averaging each word vectors. Unsupervised methods (Le and Mikolov 2014) directly tackle the representation of sentences without going through the composition of word vectors. Supervised approaches also allow the representation of sentences through the use of deep neural networks such as recursive networks, recurrent networks, and convolutional networks. These methods are particularly useful as they consider the order of the words and the structure of the sentence.

2.2 The STS Task

The *STS* task consists of the quantitative measure of the semantic similarity of two sentences on a scale of 0–5, the value of 5 implies that the sentences are strictly identical in their meaning. Each pair of phrases made available since 2012 (Agirre et al. 2012) has been evaluated by human annotators via the *Amazon Mechanical Turk* crowdsourcing marketplace, eliminating unreliable annotators and sentences showing a high statistical variance. Each pair of sentences has an average score, which can take a real values in the interval [0, 5].

Over 14, 000 pairs of text snippets have been made available to this task from 2012 to 2016. Pairs of text snippets are divided into different groups, and task participants

can train their models and set up their systems on a sub-part of each of these groups. System evaluation is performed on test data made available without annotations before the results' publishing date. The score of a system is calculated, for each group, by the Pearson's Correlation between their results and the annotated test data. The ranking of teams is then established according to the average of the correlations of all the data groups, weighted by the number of pairs of each group. Each team can send 3 results (for instance from 3 systems or different settings of the same) and train each system on all labeled datasets from previous years.

In general, teams tackle the semantic similarity as a supervised regression problem, and use standard descriptors such as the number of words in common, sequences of different lengths in common, and so on. Similarity measures based on word alignment and measurements from the field of machine translation are also used. In 2016, the 3 winning participants exploited new vector representation techniques, all based on deep neural network architectures like (Rychalska et al. 2016) that used a *Recursive Neural Network*. Another example is the use of a *DSSM* (*Deep Structured Semantic Model*) which is a deep neural network based on a Siamese architecture (Huang et al. 2013). This *DNN* takes 2 text snippets and is trained to predict their similarity. Afzal et al. (2016) trained a *DSSM* to learn a similarity function on labeled *SemEval* data prior to 2016 and predicted similarities of the test dataset from the 2016 campaign.

3 Motivations

3.1 Hypothesis

A basic comparison between two sentences would consist in calculating their topical and lexical similarity. Other methods from the distributional semantics allow for the representation of the whole sentence in a semantic space common to all the sentences of a given corpus. These methods require the preprocessing of a corpus and to set parameters of the algorithm that learns to represent each sentence. In this paper, we propose a method which can represent sentences more accurately without constraining it to specific parameters.

In this case, we believe that the humans are able to target various semantic aspects in order to compare pieces of text on several levels, each having their own weight. These aspects may include, for example, the topic, the action in the sentence, the movement, the entities involved, the spatio-temporal information, etc. In the NLP field, one generally either seeks to find the best algorithm that generates the vector representations, or tries to optimize parameters of algorithms. To our knowledge, no work attempts to automatically detect semantic aspects that would allow an optimal semantic comparison of a human level. In this paper we suggest doing so by varying several text preprocessing and vector building parameters. We trained our models on data made available between the 2012 *STS* task (Agirre et al. 2012) and the 2015

STS task (Agirre et al. 2015), only on english text snippets. We then evaluate them on the 2016 *SemEval* campaign gold standard.

Furthermore, in order to automatically capture semantic aspects allowing for an optimal comparison, we optimize the selection of different vector representations on the criterion of complementarity. This variation offers the possibility of targeting various semantic aspects of the text, thus making a similarity judgment closer to the human one. In our paper, we define a series of the most complementary vector representations as a sequence of representations, which, combined, make it possible to obtain the best results on the *STS* task by exploiting a diversity in the assignment of text preprocessing and vector building parameters.

We used an extension of *Word2Vec* (Mikolov et al. 2013a), often called *Doc2Vec* which can learn document vectors in a common semantic vector space. We used its implementation in *Gensim*[1] (Řehůřek and Sojka 2010).

3.2 Parameters Variation

Our contribution is based on the assumption that the combination of different vector representations can improve the quality of the calculated similarity score, if these representations are sufficiently complementary. More specifically, we believe that a variation of the parameters of text preprocessing in the corpus and the parameters involved in the vectors learning can diversify the semantic representation, and that the most complementary sequence is able to direct the measurement of similarity towards the adequate semantic aspects. The parameters taken into account in our system are as follows:

size (between 2 and 10,000) indicates the size of generated vectors. The same sentence will carry different semantic meanings in its granularity (topical aspects or finer semantic aspects) if represented on 50 dimensions or 3000 dimensions.

removePunct (boolean) indicates the deletion (or not) of the punctuation for each sentence. The structure of the sentence will vary depending on the value of this parameter. A comma will indicate a separation between two parts of a sentence. A question mark will indicate that the sentence is a question or a request, and that the information is probably not factual. On the other hand, removing punctuation will produce vectors that focus only on words and their neighbors.

window (between 1 and 20) defines the size of the context window for each word. The larger the window, the more the context of a word will consist of distant neighbors in the sentence, on the left and on the right. For example, in the sentence "*Bob plays the piano*", a *window* of 1 (which will correspond to a window of 3 words, except at the edge of the sentences) centered on *Bob* will take into account

[1] *Gensim* is a tool integrating different methods from distributional semantics, including methods to perform similarity measurements. https://radimrehurek.com/gensim/.

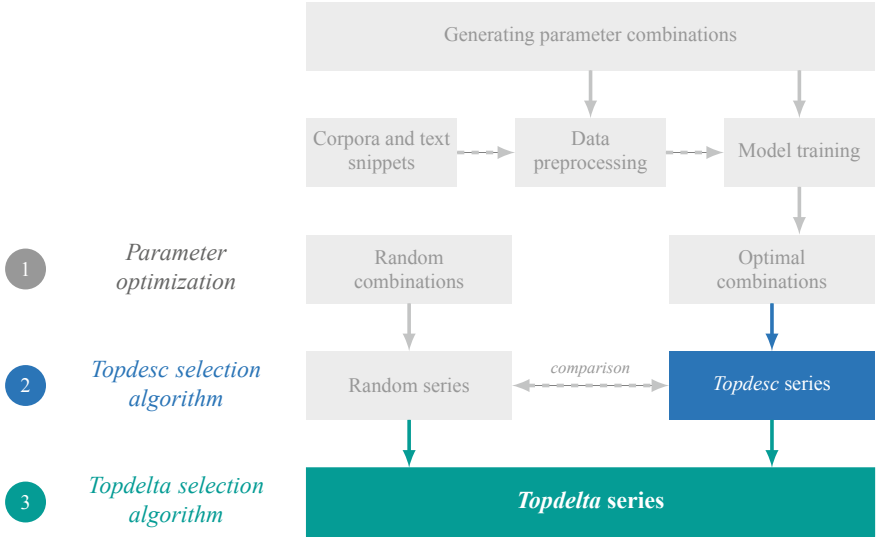

Fig. 1 Diagram illustrating the 3 steps: optimization, *topdesc* and *topdelta* selection algorithms

the action (the verb *play*) while a *window* of 4 will also consider what the action is on (*the piano*).

toLowerCase (boolean) indicates whether the capital letters in the sentence are kept or not. For example, capital letters can be used to differentiate a noun from a proper noun. Their removal allows for the reduction of personal names to their common form if it exists and for the normalization of any word at the start of each sentence or not.

removeStopWords (boolean) indicates whether stopwords are deleted from each sentence or not. Stopwords are words that are not very informative but which make it possible to link the informative words and to structure the sentence. Deleting them will focus the analysis on semantically rich words. On the contrary, preserving them will better represent the sequence of semantically rich words.

lemma (boolean) indicates whether the sentence is lemmatized or not. A lemmatized sentence loses semantic informations since, for example, by removing the conjugation, the link between a verb and its subject is weakened. A lemmatization will reduce the vocabulary size and sentences will be closer, which will accentuate their topical similarity.

Other parameters, directly related to the generation of vectors by the algorithm, will also vary: *alpha*, *iter*, *sample*, *negative* and *min_count*. The link between those hyperparameters and the final semantic representation is not easy to determine. We propose an optimization method that looks for the most complementary combinations of all parameters without having to look at the intuitive relevance of certain parameters.

Figure 1 schematizes our method in 3 basic steps. First of all, the optimization of parameters which makes it possible to obtain a set of models ranked according to their score with the variation of parameters described in Sect. 3.2. In conjunction with this optimization, we generate models with randomly chosen parameters. Then, the *topdesc* selection algorithm associates sufficiently different models by iterating over the ranking of parameters combination generated during the step 1. Finally, our last step consists of the *topdelta* selection algorithm which analyzes all series obtained in step 2 and combines the most complementary models, it offer a strong added value in the series selection as it relies on the criterion of complementarity. The Python implementation of our method has been made available.[2]

4 Combination of Complementary Models

4.1 Parameters Optimization

We start by generating models[3] by searching for optimal parameters using a local search optimization method. The 11 parameters listed in Sect. 3.2 were used for each model. A model is generated using *Doc2Vec*.

Doc2Vec takes a corpus (a set of documents/sentences) preprocessed by some of the parameters, and also takes parameters specific to the library. *Doc2Vec* gives, for each document/sentence, a representative vector. The similarity of each pair of sentences can therefore be calculated using these vectors. Other models are generated with random parameter assignments. Over 50, 000 models were generated during this first step: one half by the optimization procedure and the other using a random assignment of the 11 parameters. Each model makes it possible to obtain different vector representations of all *STS* text snippets. The vector representations are learned by *Doc2Vec* on the data of the *STS* task as well as on the *Brown Corpus* for a total of about 85, 000 sentences (around 14, 000 from the task).

In order to obtain a score for each model reflecting its performances on the *STS* task, we integrate each model as a third feature in the *DLS 2015* system proposed by (Sultan et al. 2015). The semantic similarity computation is learned by a *Ridge* linear regression implemented in *Scikit-learn*.[4] The *DLS 2015* system consists of two descriptors:

[2] https://github.com/hayj/STSSeries/.

[3] We define a model as a set of parameter assignments for the corpus preprocessing (e.g. lemmatization, stopwords removal) and vectors building (e.g. dimension size, window size) in the corresponding intervals (mentioned in Sect. 3.2) which subsequently produce a set of vector representations for each text snippet in the corpus.

[4] *Scikit-learn* is a tool written in Python integrating several machine learning methods http://scikit-learn.org/ (Pedregosa et al. 2011).

1. the first is an alignment score between two text snippet. This score is obtained from the number of words that the aligner has successfully connected between the two sentences using different metrics (thesaurus, Levenshtein distance, etc.);
2. the second feature corresponds to a cosine similarity calculated using word vectors of (Baroni et al. 2014).

When we integrate a model, it means that we take each vector of each sentence (i.e. the vector representations obtained from a model with a certain combination of parameters), and then we calculate the cosine similarities between each pair of sentences. These similarities are added as new features in the *DLS 2015* system. Subsequently, the use of several models will produce several features sets.

For our first optimization phase, we use the similarity score from a single model as an additional feature of the *DLS 2015* system. We evaluate a model (defined by its parameter assignment) by the extended system performance with, in addition, the new feature from the model. Finally, each model is ranked according to its performance by a cross-validation on all data prior to 2016.

4.2 Topdesc **Selection Algorithm**

Once this ranking is obtained, we use the *topdesc* algorithm to select a series of models. A model is defined in Sect. 4.1. We define a series of models as an ordered set of models having different parameters (text preprocessing and vector building). The *topdesc* algorithm takes a description as input, iterates over all models generated during the optimization step (from most to least efficient) and either selects or ignores models. It returns a series of models corresponding to the given description. The algorithm selects models based on 2 criteria:

- the selected model must be sufficiently different from the others;
- it must be effective when used alone.

The description given to the *topdesc* algorithm as input will guide how *topdesc* will select models according to each parameter assignment. The description is a vector D of tuples $D_i = (P_i, K_i, m_i)$ where:

P_i is a set of parameters to take into account in the differentiation with previous models. For example $[size, window]$.

K_i is a list of minimum differences with previously selected models on parameters in P which allows for the selection of the current model. For example $[100, 2]$ says the algorithm has to select the current model if it differs by at least 100 and 2 for $size$ and $window$ from all previously selected models.

m_i is the number of models to select for the current tuple.

Algorithm 2 shows how *topdesc* selects all models in the returned list $selection$ according to top and D (description) given. Note that $|selection| = \sum_{i=1}^{|D|} m_i$.

Algorithm 2 *topdesc* selection

```
procedure TOPDESC(top, D)
    selection ← [ ]
    for i in 1 to |D| do
        P_i, K_i, m_i ← D_i
        added ← 0
        for model in top do
            if model not in selection then
                add ← true
                for u in 1 to |P_i| do
                    p, k ← P_iu, K_iu
                    for alreadySelectedModel in selection do
                        val1 ← model_p
                        val2 ← alreadySelectedModel_p
                        if |val1 − val2| < k then
                            add ← false
                if add is true then
                    selection ← selection + [model]
                    added ← added + 1
                if added ⩾ m_i then break
    return selection
end procedure
```

Figure 2 shows all scores for multiple descriptions in blue and the randomly generated series in grey. Each point corresponds to the score of the series according to a certain number of models used in the series. Points on the right thus associate more models than points on the left, which makes it possible to observe performance curves of series.

The x coordinate therefore corresponds to the number of selected models and the y coordinate to the score of the *DLS 2015* system including, in additional features, cosine similarities of models in the current series sub-part until the x^{th} model. Points at $x + 1$ are scores of the extended *DLS 2015* system including another similarity from the $x + 1th$ model of the current series.

The objective of this second step was to look at models and their improvement capability in multiple series. About 10 series were tested by varying descriptions. For readability reasons, only five of them are shown.

Topdesc offers a gain of approximately 1.8% on the basis of the state of the art system in 2015 (the *DLS 2015* system) and the best model always present at the beginning of each series. However, it should be noted that randomly generated series have a higher score if enough models are used. To counterbalance this effect, we introduce the *topdelta* algorithm.

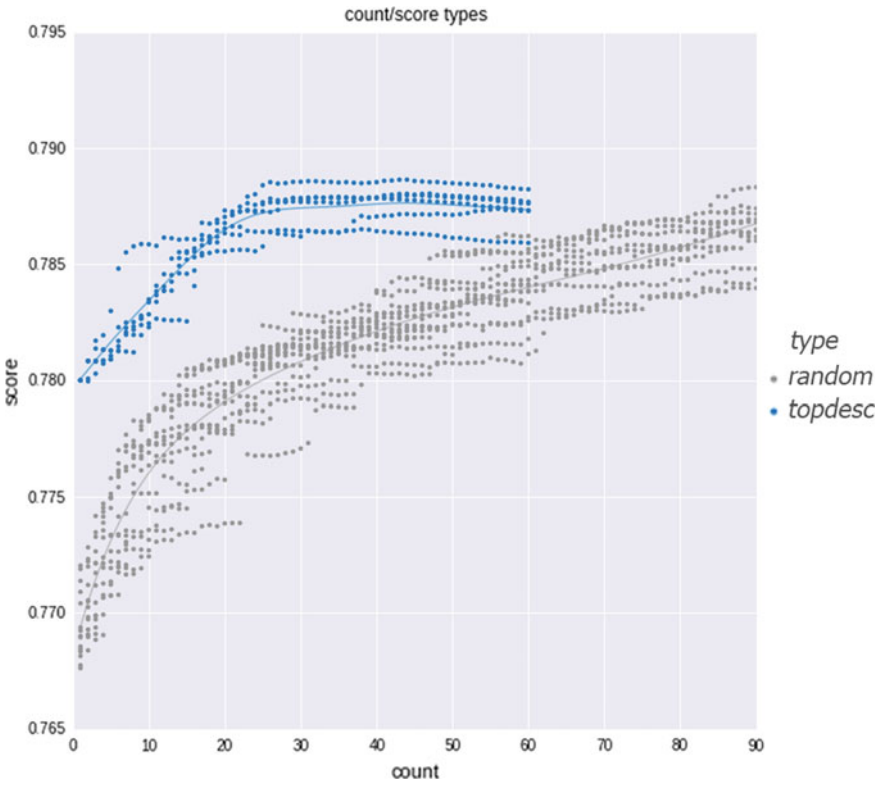

Fig. 2 Graphic gathering *topdesc* and random generated series

4.3 topdelta Selection Algorithm

The *topdesc* selection algorithm offers the possibility to create series of relevant models. But as we have seen, it does not result in a significant gain in performance. An algorithm looking for the most complementary models must be capable of automatically discriminating the parameters that have the least influence on the diversity of semantic representations. For example, we can assume that certain parameters such as the number of iterations when learning vectors are directly related to the performance of the model and a variation of these parameters will not necessarily direct the representation towards varied semantic aspects. The selection was made only on the performance criterion and did not take into account the concrete contribution of the similarity calculation derived from the model among all others features. The *topdelta* algorithm overcomes this by combining models by their complementary power. The underlying assumption is that the most complementary models are those that most improve a series. This algorithm consists of assigning a complementarity power score to each model used in the *topdesc* series and randomly generated series.

A *topdelta* series will therefore correspond to the series of models having the best scores. Multiple series can be generated depending on the parameter assignments in the equation calculating the complementarity power score that we will detail.

More specifically, we can intuitively consider that the most complementary models that can be part of a series take 2 factors into account:

1. on the one hand the performance of the model alone, i.e. its score outside a series;
2. on the other hand its "complementarity power", i.e. the average difference in performance improvement between this model and all possible previous models (ancestors, i.e. models that appear before in a series).

The score factor alone corresponds to the scores obtained during our first step. The second factor is more complicated to measure. Indeed, the average complementary power of a model should ideally take into account all possible associations of the model and its ancestors since a model can improve a series only through the poor performance of its ancestors. Each model is used on average 3 times and thus corresponds on average to 3 points on the Fig. 2. It was therefore possible to average the differences between the series score and the series score with the current model in additional feature.

The *topdelta* score (which is the complementary power) is defined as follows:

$$tdscore(x) = (1-\alpha)s + \alpha \frac{\sum_{i=1}^{n} \frac{\tau_i(x)+\Delta_i}{1+\beta}}{n} \quad (1)$$

α is used to adjust the trade-off between the influence of the score alone (s) and the complementarity on the right part of Eq. 1. This right part averages all Δ_i, which are differences between the current model x and the set of all existing ancestors (from 1 to n) in the set of *topdesc*/random series already generated. n is the number of ancestors of the current model x. All Δ_i are normalized in [0, 1] according to their minimum and maximum values amongst all series. For greater readability, this normalization is not included in the equations.

The further a model is in a series, the lower the probability that it will improve that series. This is a general observation in most machine learning tasks: any feature can be independently powerful, but the associated performances of multiple powerful features will not correspond to the sum of their individual performances. It is thus relevant to introduce a bonus which can increase the *delta* according to the position of the model in the series. So the τ bonus was introduced into the Eq. 1 and normalized by its upper bound β. This bonus can be defined according to 2 informations:

1. the number of ancestors, since the more ancestors the model has, the less chances it has to improve the series;
2. the series score at this model, since the higher the series score compared to other series, the less likely the model is to improve the whole series. In order to simplify the equations, this information will be considered normalized as Δ and s are.

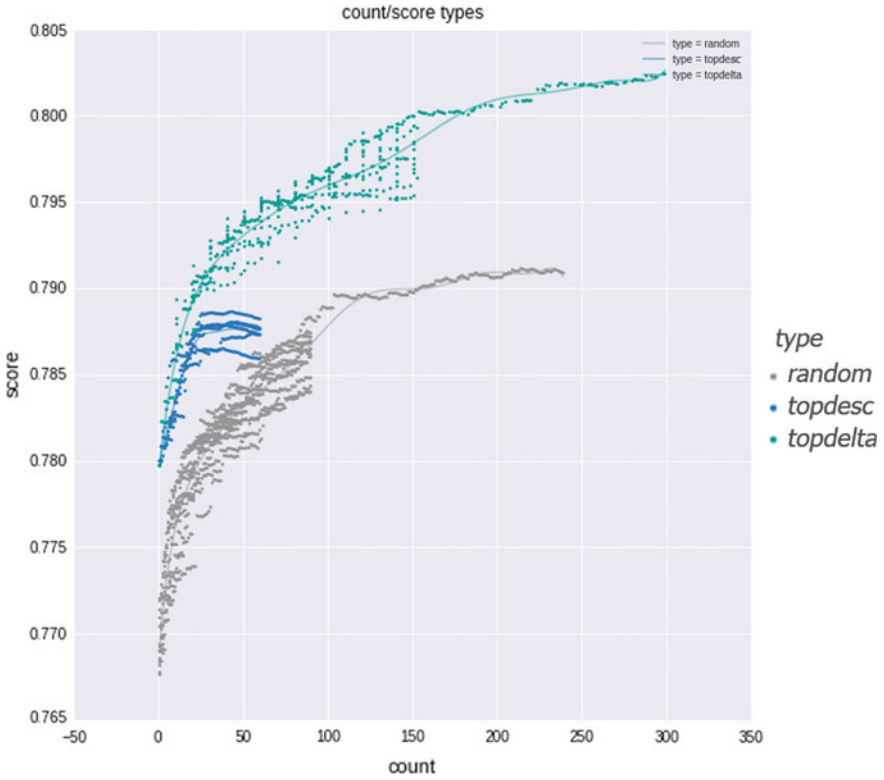

Fig. 3 Graphic gathering the 3 types of series

The τ bonus of the x model for the current ith series therefore corresponds to Eq. 2. With $nbAncestors$ the number of ancestors, σ adjusting the trade-off between both information and β weighting τ (i.e. the higher β is, the higher the bonus will be):

$$\tau_i(x) = \sigma\left(\beta \times \frac{nbAncestors(x,i)}{nbAncestorsMax}\right) + (1-\sigma)(\beta \times s_i) \tag{2}$$

Figure 3 shows *topdelta* series generated and optimized (by a local search method) on parameters α, β and σ. The best assignments, represented by the highest *topdelta* curve on the graph, are $\alpha = 0.9$, $\beta = 1.0$ and $\sigma = 0.5$. We will notice that the complementarity factor is more important than the score of the model alone as shown by the α assignment. The β assignment shows that the bonus is as important as the *delta* itself. Finally, the assignment of σ does not prioritize the information of the number of ancestors against the series score.

The first models selected by *topdelta* obtained uniformly distributed *size* and *window* parameter affectations. For instance, top 3 $(size, window)$ affectations are $(150, 2)$, $(4000, 20)$ and $(1100, 1)$. The punctuation removal and the stop words

Table 1 Score and comparison

System	Score
Baseline	0.51
Median	0.69
DLS 2015	0.69
DLS 2015 + topdelta	**0.73**

removal parameter affectations are also quite uniformly distributed, however lemmatizing words was the most common affectation among the top models.

On Fig. 3, we continued only one *topdelta* series and one randomly generated series because of memory space issues, choosing the best among those already generated. If we consider the baseline score as the *DLS 2015* system score with a randomly generated model in additional feature, then the *topdelta* series provides an increase of 4%. The overall complexity of our method from the first step to the *topdelta* algorithm is linear in the size of the corpus which is dedicated to train embeddings, i.e. linear in the number of sentences used. The running complexity of our method is linear in the number of text snippets pairs to assess. The next part is intended to experimentally evaluate the best *topdelta* series by testing our system on the *Gold Standard* dataset made available in 2016.

5 Experiments

During *SemEval* 2016, 43 teams participated in the task for a total of 119 systems. Overall scores ranged from 0.4 to 0.77 with a median of 0.69. The campaign baseline corresponds to a similarity based on the cosine similarity of the bag-of-words representations and obtained a score of 0.51.

Table 1 shows that the best *topdelta* series significantly improved the performance of the *DLS 2015* system. Through the automatic selection of complementary models, our system was able to obtain a score above the median.

6 Conclusion and Perspectives

Through our experimental work on the search for complementarity, we were able to show that it is possible to select vector representations complementary enough to guide the calculation of similarity on various semantic aspects.

More recently, a new STS campaign has been organized and a new test set was released (Cer et al. 2017). There were 31 teams participating in this task which included the assessment of cross-lingual similarities. The best performing system was

the ECNU team's system (Tian et al. 2017). The authors implemented an ensemble of 3 machine learning algorithms and 4 deep learning models. The machine learning algorithms were Random Forest, Gradient Boosting and XGBoost. NLP features used for these algorithms were quite diverse: n-gram overlap, edit distance, machine translation features, word alignments from *DLS 2015* (Sultan et al. 2015) etc. Deep Learning models were word embeddings-based models, a deep averaging network and a LSTM network. These models produced pair representations. Representations were feeded to a last fully-connected neural network which learned similarity scores. 7 similarity scores were generated for each pair and the average gave the final similarity score.

Our work does not exploit the most recent data from the 2017 *SemEval* task (Cer et al. 2017). Training our model on these new annotated text snippet pairs can be part of further work. However, our experiment shows that a complementarity search is an improvement track in semantic similarity while research teams propose methods which focus on finding the best performing model and meaningful features for this task.

We proposed a method to choose complementary vectors in an ad-hoc and efficient way which can later be compared to well-studied optimization algorithms in the machine learning field such as boosting (Schapire 2003) and genetic algorithms (Goldberg and Holland 1988). Furthermore, due to high calculation time, we only trained our models on a limited corpus composed of the *Brown Corpus* and the dataset from *SemEval* prior to 2016.

Subsequently, we are thinking of improving our methodology for larger corpora. This would make it possible to take into account other parameter variations, such as the targeting named entities, which can be retained or not in sentences. Named entities represent a large vocabulary space and we believe they are playing a specific role in the *STS* task.

The supervised methods that have shown better results (Rychalska et al. 2016; Afzal et al. 2016) on the *STS* task can also be used in our search for complementarity. In addition, other vector representation methods can be combined with *Doc2Vec* in order to capture various and potentially complementary semantic aspects such as *fastText* (Bojanowski et al. 2017) or *Sent2Vec* (Pagliardini et al. 2017).

References

Afzal, N., Wang, Y., & Liu, H. (2016). MayoNLP at SemEval-2016 task 1: Semantic textual similarity based on lexical semantic net and deep learning semantic model. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 674–679), San Diego, California: Association for Computational Linguistics.

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., et al. (2015). SemEval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 252–263). Association for Computational Linguistics.

Agirre, E., Cer, D., Diab, M., & Gonzalez-Agirre, A. (2012). SemEval-2012 task 6: A pilot on semantic 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)* (pp. 385–393). Montréal, Canada: Association for Computational Linguistics.

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors (pp. 238–247). ACL.

Baroni, M., & Lenci, A. (2010). Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, *36*(4), 673–721.

Baroni, M., & Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 1183–1193). Cambridge, MA: Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., & Specia, L. (2017). SemEval-2017 task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 1–14). Association for Computational Linguistics.

Curran, J. R. (2004). *From distributional to semantic similarity*. Ph.D. thesis, University of Edinburgh, UK.

Dolan, W. B., & Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, *3*(2), 95–99.

Harris, Z. (1954). Distributional structure. *Word*, *10*(23), 146–162.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, CIKM 2013* (pp. 2333–2338), New York, NY, USA: ACM.

Le, Q. V. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014* (pp. 1188–1196).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. In *In Proceedings of Workshop at ICLR*.

Mikolov, T., Yih, W., & Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9–14, 2013* (pp. 746–751). Atlanta, Georgia, USA: Westin Peachtree Plaza Hotel.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15–20, 2008, Columbus, Ohio, USA* (pp. 236–244).

Pagliardini, M., Gupta, P., & Jaggi, M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. *CoRR*. abs/1703.02507.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25–29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (pp. 1532–1543).

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50). Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Rychalska, B., Pakulska, K., Chodorowska, K., Walczak, W., & Andruszkiewicz, P. (2016). Samsung poland NLP team at SemEval-2016 task 1: Necessity for diversity; combining recursive autoencoders, wordnet and ensemble methods to measure semantic similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 602–608). San Diego, California: Association for Computational Linguistics.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification* (pp. 149–171). Springer.

Sultan, M. A., Bethard, S., & Sumner, T. (2015). Dls@cu: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 148–153). Denver, Colorado: Association for Computational Linguistics.

Tian, J., Zhou, Z., Lan, M., & Wu, Y. (2017). ECNU at SemEval-2017 task 1: Leverage kernel-based traditional NLP features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)* (pp. 191–197). Association for Computational Linguistics.

Turney, P., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, *37*(1), 141–188.

Van de Cruys, T., Poibeau, T., & Korhonen, A. (2013). A tensor-based factorization model of semantic compositionality. In *Conference of the North American Chapter of the Association of Computational Linguistics (HTL-NAACL)* (pp. 1142–1151).