UBC: Cubes for English Semantic Textual Similarity and Supervised Approaches for Interpretable STS

Eneko Agirre, Aitor Gonzalez-Agirre, Iñigo Lopez-Gazpio,
Montse Maritxalar, German Rigau, Larraitz Uria
University of the Basque Country
Donostia, 20018, Basque Country
{e.agirre, aitor.gonzalez-agirre, inigo.lopez,
montse.maritxalar, german.rigau, larraitz.uria}@ehu.eus

Abstract

In Semantic Textual Similarity, systems rate the degree of semantic equivalence on a graded scale from 0 to 5, with 5 being the most similar. For the English subtask, we present a system which relies on several resources for token-to-token and phrase-to-phrase similarity to build a data-structure which holds all the information, and then combine the information to get a similarity score. We also participated in the pilot on Interpretable STS, where we apply a pipeline which first aligns tokens, then chunks, and finally uses supervised systems to label and score each chunk alignment.

1 Introduction

In Semantic Textual Similarity (STS), systems rate the degree of semantic equivalence on a graded scale from 0 to 5, with 5 being the most similar. We participated in two of the subtask for STS in 2015 (Agirre et al., 2015). For the English subtask, we present a system which relies on several resources for token-to-token and phrase-to-phrase similarity to build a data-structure which holds all the information, and then combine the information to get a similarity score. We also participated in the pilot on Interpretable STS, where we apply a pipeline which first aligns tokens, then chunks, and finally uses supervised systems to label and score each chunk alignment.

Note that some of the authors participated in the organization of the task. We scrupulously separated the tasks in such a way that the developers of the systems did not have access to the test sets, and that they only had access to the same training data as the rest of the participants.

2 Cubes for English STS

In this section we describe a novel approach to compute similarity scores between two sentences using a cube where each layer contains token-to-token and phrase-to-phrase similarity scores from a different method and/or resource. Our assumption is that we can obtain better results using this similarity scores together than independently.

2.1 Building Cubes

The first step is to produce parse trees for the sentences using the Stanford Parser (Toutanova et al., 2003). After parsing the sentences each pair of sentences can be represented by a NxM matrix, being N is the number of nodes of the parse tree of the first sentence, and M the number of nodes of the parse tree of the second sentence. Note that some nodes (terminals) correspond to words, while others (non-terminals) represent phrases. We can have as many matrices as we wish, and fill them with different similarity scores, forming a cube.

In this first attempt we used three layers:

1. Euclidean distance between Collobert and Weston Word Vector (Collobert and Weston, 2008). The vector representations for each non-terminal node in the tree were learnt using Recursive Autoencoder (RAE) (Socher et al., 2011).

2. Euclidean distance between Mikolov Word Vectors (Mikolov et al., 2013a; Mikolov et al., 2013b). To compute the vector representations for each non-terminal node in the tree, we summed the vectors and normalize them dividing by the number of words in the phrase.

3. PPDB Paraphrase database values (Ganitkevitch et al., 2013). We used the XXXL version. In this case both words and some phrases

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 178–183,
Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistics

are contained in the resource. This resource yields conditional probabilities. As our scores are undirected, in case the database contains values for both directions, we average.

The first two produce a dense layer, where most of the cells have a value. The third one produces a sparse layer, where only the pairs occurring in the resource have a value. Note that some of the phrases in PPDB do not correspond to a node in the tree. In this case, we add extra columns and rows.

2.2 Producing STS Score

Before computing a similarity score we flatten our cube into a single layer, where each of the element in the new NxM matrix is the maximum between the values for that position across all the layers. We do that because we studied the different resources and we think that these resources have less False Positives (FP) than False Negatives (FN). In other words, if one of the resources says that something is very similar we trust on it and take that similarity score instead of the other (even if they are very low). Moreover, our assumption is that the final similarity score is specially based in similarities between tokens/phrases in the sentences, and not on dissimilarities.

Once we have this matrix, we compute the final STS score using the scoring function seen in (Mihalcea et al., 2006).

$$sim(S_1, S_2) = \frac{1}{2}\left(\frac{\sum_{w \in S_1}(maxSim(w_{\in}S_2) * idf(w))}{\sum_{w \in S_1} idf(w)} + \left(\frac{\sum_{w \in S_2}(maxSim(w_{\in}S_1) * idf(w))}{\sum_{w \in S_2} idf(w)}\right)\right)$$

2.3 Results

Due to time constraints we submitted a single run, which ranked 54 among 74 runs. We expect to improve this results adding more layers and combining them using more sophisticated aggregation methods.

3 Participating on the Interpretable STS Pilot Subtask

The SemEval 2015 STS task offered a new *pilot subtask on interpretable STS*[1]. Given a sentence pair, the objective of the subtask is to align segments pertaining to one sentence with the segments pertaining to the other sentence. The whole subtask is in deep described in (Agirre et al., 2015).

In sum, every alignment may consist of a **similarity score** and a **relatedness tag**. The similarity score is a real number bounded by [0,5] where 0 means no relation at all and 5 means complete equivalence. As regards the relatedness tag, there exists a set of categorical values to choose on, such as: *equivalence*, *opposition*, *specialization* (direction is relevant), *similarity* and one more tag for *other kind of relatedness*.

For the case of unaligned segments there are another two possible categorical values. The one for declaring segments unaligned (*not aligned*); and the other to declare that the segment related to the current segment has already been aligned (*context alignment*). Notice that due to the limitations of the current pilot the only way to align segments is making 1:1 alignments. Thus, 1:N alignments are simulated making an 1:1 alignment and several *context alignments*. This concept is relevant to the work done in section 3.1.2 when we extend the *Hungarian-Munkres* (Clapper, 2009) algorithm to identify *already aligned chunks*.

In addition, *factuality* or *polarity* connotations can be added as requested to the previously mentioned tags. Two different scenarios are provided in the pilot subtask, the first one makes gold standard segments available for participants (***Gold Chunks*** *or GS scenario*); and, the second one, only provides sentence raw text (***System Chunks*** *or SYS scenario*).

In conclusion, the first pilot on interpretable STS seems challenging because participating systems must not only discover and score the relatedness between segments, but also identify the inner relation between them.

3.1 System Description

This section describes the principal algorithm and the distinct modules it uses, modules are further described in the following subsections (3.1.1, 3.1.2, 3.1.3 and 3.1.4). System configurations (***runs***) used to submit results are described in section 3.1.5.

The system makes use of **several modules** to identify segments over sentence pairs, and then, make alignments between them. First of all, the *input handling and chunking module* is responsible

[1] http://alt.qcri.org/semeval2015/task2/index.php?id=proba

for linguistically processing the given input, and for creating the internal representation of the sentences. Once the input is processed the *alignment module* identifies related and unrelated segments among sentences. Finally, by using segment pair based features the *classification module* and the *scoring module* produce respectively the final relatedness tag and the similarity score.

3.1.1 Input Handling and Chunking Module

We use the **Stanford NLP parser** (Klein and Manning, 2003) to linguistically process input sentences and register lowercased token information (lemma, part of speech analysis and dependency structure is also needed for the following module). The next step consists of determining segments or token regions. This information is gathered according to the specified scenario (GS or SYS). In the case of the GS scenario the baseline obviously uses gold standard input; and, in the SYS scenario the baseline uses the *ixa-pipes-chunker* (Agerri et al., 2014).

Ixa-pipes-chunk has been trained using the Apache OpenNLP API (OpenNLP, 2011), which is a maximum entropy chunker. Nevertheless, the chunker's output has been improved using simple regular expressions to fit to our task proposal. Actually, we developed four rules to optimize how conjunctions, punctuations and prepositions are handled. In brief, the developed rules try to join consequent chunks forming new chunks consisting of the previous ones, for instance, we found significant improvement if prepositional phrases followed by a nominal phrase were unified as a single chunk. We also developed some rules to unify nominal phrases separated by punctuations or conjunctions, or a combination of those.

3.1.2 Alignment Module

The alignment module mainly focuses on the work done by the monolingual word aligner described in (Sultan et al., 2014), and *Hungarian-Munkres* algorithm.

The **monolingual word aligner** is a simple and ready-to-use system that has demonstrated state-of-the-art performance. To begin with we start by constructing the *token to token link matrix* in which each element at position (i,j) determines that there exists a link between token i (from sentence 1) and token j (from sentence 2). A link exists in the matrix if and only if the monolingual word aligner has determined that both tokens are related.

Then, the system uses token regions to group individual tokens into segments, and calculates the weight between every segment in the sentence pair. The weight among two segments is proportional to the number of links that interconnect tokens inside those segments. In other words, by summing regions we collapse the token to token link matrix onto a *chunk to chunk link matrix*. After that, we use the mentioned **Hungarian-Munkres** algorithm to discover which are the segments (x,y) which score the highest weight (link ratio); but also, we extend it to discover which are the segments that are linked to either segment x or segment y, but not with a maximum alignment ratio. This processing to find not-maximal weights is essential to effectively assign the *context alignment* tag for 1:N relations. In addition, the system is also aware of chunks that have been left unaligned.

3.1.3 Classification Module

The system can use one of the following approaches to assign relatedness tags to segment pairs: the *naive approach* and the *machine learning approach*. The **naive approach** directly assigns the tag as a majority classifier would do, that is: for the segments with highest weight it always assigns the equivalence tag, for the segments that are linked with lower weights it always assigns the *context alignment* tag, and for the not aligned segments it always assigns the not aligned tag.

The **machine learning approach** makes use of the segment-pair to calculate a total of 21 features to improve the tag assignment. The objective of the induced model is to refine the output given by the naive approach only for segment pairs tagged as equivalent. The features used to induce the model can be classified in the following groups: Jaccard overlap related features, segment length related features, WordNet similarity related features among segment heads, WordNet depth related features, and other kind of features obtained by means of the cube described in section 2.

To induce the model we use the *Support Vector Machine* (SVM) implementation described in (Chang and Lin, 2011) under the latest experimental version of *Weka* (Hall et al., 2009) using randomly shuffled 5-fold cross validation. We indistinctly join

the available datasets and grid search to optimize the cost and gamma parameters.

3.1.4 Scoring Module

To assign segment pair similarity scores the system can also use two distinct approaches: the *naive approach* and the *cube based regression approach*. The **naive scorer** directly assigns a certain score to each one of the tags, which has been previously assigned using the naive tagger: for equivalence tags it assigns a score of 5 and for not aligned and context aligned tags it assigns 'NIL'. (as requested by the guidelines). The **regression approach** uses the cube described in section 2 to improve the score given to segment pairs tagged by the machine learning tagger. Its returning value is used directly as the value for the pair similarity score.

3.1.5 Submitted Runs

Even the subtask allows the submission of up to three runs, we only submitted two distinct configurations, named *run1* and *run2*. run1 and run2 are mainly the same system, but **run1** makes use of the naive approaches for both classification and scoring tasks; whereas **run2** makes use of the machine learning approach for the tag assignment and the cube based regression approach for the scoring assignment.

3.2 Result Analysis

Participating runs were evaluated using the official scorer provided by task organizers, which computes four distinct metrics: *F1 ALI* (segment pair alignment correctness regardless of the tag), *F1 Type* (segment pair alignment correctness taking tag into account), *F1 Score* (segment pair alignment correctness taking score into account) and *F1 Type + Score* (segment pair alignment correctness taking tag and score into account).

3.2.1 Development

We developed two runs as above described using the training data provided by task organizers. Training data consists of two datasets (images dataset and headlines dataset) with 750 sentence pairs each. We built and evaluated our system using 5-fold cross validation and using a grid search optimization to tune the SVM parameters. Results for both runs are shown in table 1.

I GS	Ali	Type	Score	Type + Score
Run1	0.8942	0.5115	0.7776	0.5115
Run2	0.8942	0.7408	0.8175	0.6934
I SYS	**Ali**	**Type**	**Score**	**Type + Score**
Run1	0.8379	0.4734	0.7271	0.4734
Run2	0.8379	0.6499	0.7627	0.6106
H GS	**Ali**	**Type**	**Score**	**Type + Score**
Run1	0.8920	0.5740	0.7869	0.5738
Run2	0.8920	0.6908	0.8133	0.6544
H SYS	**Ali**	**Type**	**Score**	**Type + Score**
Run1	0.7650	0.4808	0.6707	0.4808
Run2	0.7650	0.5210	0.6862	0.4902

Table 1: Development results for both datasets in the two scenarios. 'I' stands for the images dataset, and 'H' stands for the headlines dataset.

The table shows that run2 outperforms run1 in all of the scenarios, which was expected as run1 is using the naive approach for both: the relatedness tag and the similarity score assignment. Notice that both runs obtain the same F1 Alignment score as both runs are using the same input handling and chunking module. Without the shadow of a doubt, we can observe that for both datasets the **F1 alignment is noticeable higher** in the *GS scenario* than in the *SYS scenario*. Moreover, as evaluation measures are incremental, F1 Type, F1 Score and F1 Type + Score are also lower for the SYS scenario.

It is also important to mention that the difference in performance (F Type+Score) between run1 and run2 is **more noticeable in the images dataset**, actually, for the headlines dataset in the SYS scenario, the difference between both runs is under 0.01. This difference increases up to 0.08 for the headlines dataset in the GS scenario.

3.2.2 Test

The test dataset was composed of 378 sentence pairs for the headlines dataset and of 375 sentence pairs for the images dataset. Table 2 illustrates the results obtained by run1 and run2. The results obtained for the test datasets follow in general the same tendency as the one seen for the development. In fact, **run2 most of the times outperforms run1**; being this difference in performance more noticeable in the images dataset than in the headlines dataset. It might be necessary to further analyze the

I GS	Ali	Type	Score	Type + Score
Baseline	0.8388	0.4328	0.721	0.4326
Run1	0.8846	0.4749	0.7709	0.4746
Run2	0.8846	0.6557	0.8085	0.6159
MAX Par	0.887	0.6143	0.7968	0.5964
AVG Par	0.8193	0.5004	0.7197	0.4748
I SYS	**Ali**	**Type**	**Score**	**Type + Score**
Baseline	0.706	0.3696	0.6092	0.3693
Run1	0.8388	0.445	0.728	0.4447
Run2	0.8388	0.6019	0.7634	0.5643
MAX Par	0.8336	0.5759	0.7511	0.5634
AVG Par	0.67	0.4086	0.5892	0.3912
H GS	**Ali**	**Type**	**Score**	**Type + Score**
Baseline	0.8448	0.5556	0.7551	0.5556
Run1	0.8991	0.5882	0.8031	0.5882
Run2	0.8991	0.6402	0.8211	0.6185
MAX Par	0.8984	0.6666	0.8263	0.6426
AVG Par	0.8365	0.5576	0.7468	0.5381
H SYS	**Ali**	**Type**	**Score**	**Type + Score**
Baseline	0.6701	0.4571	0.6066	0.4571
Run1	0.7709	0.5019	0.6892	0.5019
Run2	0.7709	0.4865	0.7014	0.4705
MAX Par	0.782	0.5154	0.7024	0.5098
AVG Par	0.6870	0.4498	0.6094	0.4335

Table 2: Test results for both datasets in the two scenarios. 'I' stands for the images dataset, 'H' stands for the headlines dataset and 'Par' stands for participants.

unique scenario in which run1 obtains higher accuracy than run2 (Headlines SYS), but actually, results have been also very close at development in this context.

The baseline seems to be not that trivial as it sometimes outperforms participants average performance; but as we can see both of our runs obtain higher accuracy than the baseline, in both cases by large margin. For example, in the images dataset the difference between the baseline and the second run is 0.18 and 0.19 respectively for the GS and the SYS scenario. Regarding other participants, we can conclude that our runs obtain quite good results, specially for the images dataset where run2 obtains the highest score.

4 Conclusions and Future Work

Through this paper we have described the systems that participated in the Semantic Textual Similarity task 2A (English STS) and 2C (Interpretable STS). Our main focus in the English subtask was on deploying our idea of building a cube with similarity information from several sources. We are currently working on more layers, including Random Walks over WordNet and Wikipedia, string similarity (Ferrone and Zanzotto, 2014), and also a special layer to deal with numbers. Additionally, we are considering the idea of dissimilarity layers, for instance, adding information about negation and antonymy. We are also developing new methods to combine these knowledge to generate the final STS score.

Regarding the interpretable STS system, this was the first time a pilot was put in place. We obtained excellent results, even if we had very little time to develop the system. Future work will focus on further improvements. For instance, our experiments showed that grouping chunks lead to a considerable improvement for the F1 Type evaluation score. We would also like to incorparate factuality or polarity information.

Although our original idea was to combine the cube and the interpretable system, we did not have time for that. In one direction, we would like to incorparate some of the semantic similarity information in the cube into our system, including similarity between chunks. On the other direction, the information from the similarity module might be a good feature to improve the overall STS score.

Acknowledgements

This work was partially funded by MINECO (CHIST-ERA READERS project – PCIN-2013-002-C02-01, SKaTeR project – TIN2012-38584-C06-02), and the European Commission (QTLEAP – FP7-ICT-2013.4.1-610516). Aitor Gonzalez-Agirre and Iñigo Lopez-Gazpio are supported by doctoral grants from MINECO. The IXA group is funded by the Basque Government (A type Research Group).

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. 2014. Ixa pipeline: Efficient and ready to use multi-

lingual nlp tools. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC2014), Reykjavik, Iceland, May*, pages 26–31.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 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)*, Denver, CO, June.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

BM Clapper. 2009. munkres: a python module implementing the "hungarian method" described by munkres (1957). version 1.0. 5.3.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA.

Lorenzo Ferrone and Massimo Fabio Zanzotto. 2014. Towards syntax-aware compositional distributional semantic models. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 721–730.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the American Association for Artificial Intelligence (AAAI 2006)*, Boston, Massachusetts, July.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Apache OpenNLP. 2011. Apache software foundation. *URL http://opennlp. apache. org*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, pages 219–230.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.