Semantic Textual Similarity

Madhura Parikh
Department of Computer Science
University of Texas, Austin
mparikh@cs.utexas.edu

Abstract

Through this project, we will explore Semantic Textual Similarity - a shared task in Sem Eval 2012. The problem involves comparing a pair of sentences for semantic equivalence, reporting the equivalence on a scale of 0 5 with 0 meaning no equivalence and 5 meaning semantically equivalent. This task is more challenging than determining if the two sentences are paraphrases (a simple yes/no answer), and thus is likely to have much wider applications for a host of NLP tasks like Machine Translation, Question-Answering and Information Retrieval. We build upon the winning 2012 Sem Eval entries, and in addition use a tree-kernel based Support Vector Regression that compares the top-k syntactic parses of the the two sentences as an added similarity measure. Our results seem promising, beating the 2012 winning entry overall, and even more importantly doing significantly better on the surprise test data. We also discuss future extensions that are likely to strengthen our approach.

1 Introduction

1.1 Motivation

In this project we look at the problem of Semantic Textual Similarity(STS). Given two sentences, we would like to judge how *semantically* similar they are. This is a problem of great interest to the natural language community, since most approaches to sentence similarity will aim at only *lexical* and *syntactic* similarity. There have been other efforts that have looked at more semantically focussed tasks like resolving metaphors or idiomatic phrases. However most of these efforts have never been scaled up to entire sentences. This was one of the major motivation for introducing this task as a Sem Eval shared task.

A vast majority of NLP systems would benefit hugely if it were possible to evaluate the semantic similarity of sentences in a pair. For instance Information Retrieval systems could use this to retrieve results that were the most semantically similar to the user's query. Similarly Question-Answering systems could use this to return question-answer pairs that were semantically the most similar. Obvious applications can also be imagined for Machine Translation systems as well as several others.

There have been other approaches [8] that have tackled a similar but simpler problem - Sentence paraphrases. However this task differs from that - rather than returning a simple yes/no, a graded equivalence measure is the output in this case. Such a graded measure is much more informative and will serve much better for NLP tasks. With its introduction as a Sem Eval task, there have been several systems that have been submitted for STS. In the next paragraph, we discuss briefly the most common approaches used by the teams.

1.2 Existing approaches

A good summary of the most commonly used NLP tools and packages by the Sem Eval participating teams is provided in [2]. One major advantage of working on the STS task is that there have been 80 odd submissions for the official Sem Eval task in both 2012-13, mean- ing that we can draw from these experiences to find out which approaches and features worked the best and how we can leverage off them to get improved results. A majority of the teams have used a supervised approach. For lexical similarity most teams have used the edit distance and other metrics of string similarity such the Needleman- Wunsch distance, the Smith-Waterman distance, etc. Most of the teams have also worked with the lemmatized sentences rather than considering the actual words. Most teams have also used POS- tags as a feature and some have also used chunk- ing and chunk-overlap amongst the two sentences as well as n-gram and skip-gram overlap as features. Another often used feature is based on the dependency parse of the sentence, which enables comparing the S-V-O (subject-verb-object) structure of sentences. For the n-gram modeling, several approaches have used not just the training data but also external corpora such as the Google n-gram 1T corpus, Wikipedia, etc. Another simple but highly useful metric was to find the number overlap , i.e if the numbers/dates/percentages that appear in one sentence match with those in the other.

A much harder aspect of the problem is judging the semantic similarity of the two sentences. Most approaches first picked out the content words of each sentence and then used WordNet to find the similarity between the words pairwise, aggregating them in different ways to come up with a final similarity score. Interestingly while some teams used the least common ancestor or the shortest path metric, there really was no well defined metric that could be deemed helpful for the task - which is something we could look at. The source used for similarity also played a very important role and one team mentions that using the Rogets thesaurus gave better results than WordNet. The BLEU measure was also a frequently used metric. LSA and Random Indexing were also commonly used as were several different distributional vector models. WSD was performed using the Lesk algorithm in conjunction with WordNet for most cases. A few teams also used Named Entity Recognition(NER) for judging similarity, while shallow semantic parsing(SRL) was used just by a couple of teams.

A few other interesting approaches used LDA based models to assign topics to a sentence, and used IR engines like Lucene as well as TF-IDF for comparing similarity. Most of the teams used Support Vector Regression(SVR) for predicting the final outcome and a couple of teams used an ensemble-based approach such as boosting.

1.3 Our contribution

Our primary contribution to this project is that we introduce a tree-kernel [5] based approach to compare the top-k syntactic parses of the sentence pair, and use this as a measure of semantic similarity. We use Support Vector Regression with a third degree polynomial kernel to combine our computed tree similarity with existing features from Takelab [6] - one of the top-5 entries in Sem Eval 2012 - to arrive at the final score. We find that using parse tree similarity helps in improving the overall results by a good measure. Even more importantly, using tree kernels gives a significant boost to our performance on the surprise test data, where our improvement is much higher than in the overall case. This we believe motivates a strong case for using tree kernel based features in STS systems, to make them much more robust even in unsupervised/transfer-learning based settings.

In the rest of this report we elaborate on our approach. Section 2 defines the problem precisely and in Section 3 we describe our experimental setup and results, also analyzing our performance. Finally in Section 4 we evaluate our work against the existing approaches and in Section 5 we describe further improvements that can be made to our current approach. We conclude in Section 6.

2 Problem definition and algorithm

As we mentioned earlier, our project deals with the Sem Eval 2012 pilot task 6 -comparing the semantic textual similarity of a pair of sentences [2].

2.1 Problem statement

Given 2 sentences $s1$ and $s2$ return a score of how similar they are on a scale of $0 - 5$. A score of 0 indicates that the sentences are totally dis-similar whereas a score of 5 indicates semantic equivalence. The quality of the scores is judged by calculating their correlation (Pearson) with manual gold standard scores for the same data.

2.2 Our approach

First, for each sentence in the input, we obtain its top-k syntactic parse trees using lexicalized English PCFG grammar. Next, we compare the similarity between the parse trees of the two sentences $s1$ and $s2$ in a pair by using Partial Tree Kernels [5]. We use tree kernels to compute the similarity between every possible pair of the top-k trees of $s1$ and $s2$, i.e. a total k^2 tree-pairs will be compared. In [5] they describe three type of tree kernels: Sub-tree kernels(ST), Subset-tree kernels(SST) and Partial-tree kernels(PT). Below we give a very high level description of these tree kernels.

Each tree kernel represents the tree by its subparts and uses the kernel to compute the number of such subparts that are common to both the trees. The three kernels we mentioned above differ in how they define the subparts of the tree:

Sub-tree: It is the tree that is obtained by picking up any node and considering the entire tree that lies below it, including all its descendants, right upto the leaves.

Subset-tree: It considers all valid subsets of trees from the current tree. Thus while it requires that the tree is still valid under the given grammar, it may not necessarily include the leaf nodes.

Partial-tree: This is similar to subset tree kernels, except that it is no longer required that the tree be valid under the given grammar.

In figure 1 we represent an example of partial trees for the parse tree of the phrase 'brought a cat'.

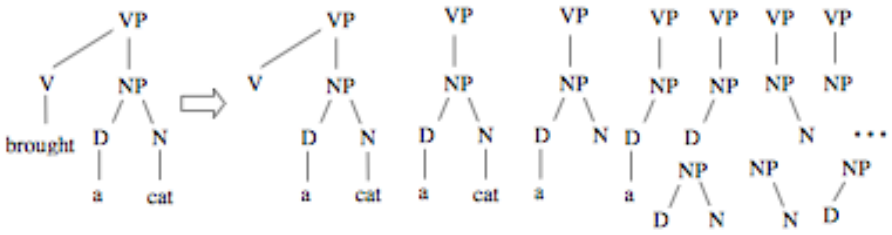

Figure 1: Partial trees for the phrase 'brought a cat' [5]

We choose the PTK because it has a fast and efficient implementation and also has more flexible sub-structures that seemed more suitable for the semantic equivalence task. We combine tree kernel based similarity with other features that are computed by Takelab to compute the resulting similarity using SVR with 3 degree polykernel. Since we consider the top-k parse trees, and further compare k^2 trees between the two sentences, we end up with k^2 scores for each sentence pair. We then consider the maximum[1] of these k^2 scores and return this as our final score for the given sentence pair. Below we present the pseudocode that presents these steps more concisely:

[1]We tried other summary statistics like minimum and mean, but max returned the best correlation with the manual scores in all cases.

Algorithm 1 STS-Algorithm

procedure EVALUATE_SIMILARITY(s_1, s_2)
    $wordlist1 \leftarrow$ preprocessAndTokenize(s_1)
    $wordlist2 \leftarrow$ preprocessAndTokenize(s_2)
    $parseforest1 \leftarrow$ getTop-kParses($wordlist1$)
    $parseforest2 \leftarrow$ getTop-kParses($wordlist1$)
    $featurevector \leftarrow$ getTakelabFeatures(s_1, s_2)
    $scorelist \leftarrow \{\}$
    for i in 1..k **do**:
        for j in 1..k **do**:
            $score \leftarrow$ PTK-similarity($parseforest1[i], parseforest2[j]$) + $Poly3DKernel(featurevector)$
            $scorelist \leftarrow scorelist \cup score$
    return max(*scorelist*)

3 Exerimental Evaluation

3.1 Data

We use the same data that was released as a part of the Sem Eval STS 2012 task. Specifically the following datasets were released by the organizers:

MSRpar : The Microsoft Paraphrase corpus was developed at Microsoft Research using manually annotated paraphrase datasets and consists of 5801 sentences. The organizers picked 1500 sentences from this that gave an even distribution of all levels of similarity $(0 - 5)$ and randomly split these into $50\% - 50\%$ train-test sets.

MSRvid: The MSR Video Paraphrase corpus was generated by Microsoft asking Amazon Mechanical Turkers to generate descriptions of short video clips. More than a million descriptions for 2000 videos were collected. Again the organizers picked up 1500 sentences from this corpus and randomly split these into $50\% - 50\%$ train-test sets.

SMTeuroparl: This data was picked up from the ACL workshops on Machine translation(WMT) and consisted of pairs of machine translated text paired with its reference human translation. In this case 1468 sentences were picked for the french-english translation systems and randomly split into $50\% - 50\%$ train-test sets.

Out-of-domain data: In addition to the above datasets, the systems were evaluated on two 'surprise' datasets, that were not provided as a part of the training data. One of the surprise datasets comprised of human ranked outputs of the WMT for the french-english system for news conversations (**SMTnews**) and comprised of 399 sentence pairs. The second dataset was quite unlike most of the training data and comprised of glosses from OntoNotes and WordNet (**OnWN**). It had a total of 750 sentence pairs.

Henceforth we will use the corresponding boldface name to refer to a particular dataset. It is clear that the data provides a varied distribution that is challenging for the STS task. What is especially of interest is the out-of-domain data and how the trained models perform on them.

3.2 Tools and methodology

We use several available open source tools for our project. The code for this project is written in Java and in Python. We use the Stanford PCFG parser [4] trained on the Wall Street Journal(WSJ) sections $2 - 21$ from the Penn Linguistic Data Consortium(LDC) to get the top-k parses for each of our sentences. The code we wrote for preprocessing the sentences so that they were tokenized and in the format expected by the parser was written in Java. We also used Java to write the code that uses the Stanford Parser API to interact with the parser and get the top-k parses.

Next we used the openly released system from Takelab [6] to generate features in addition to the tree kernels that we were using. This code was released in Python. Briefly the system generates a total of 21 features. Only there *simple* version is released for use, which is what we utilize. The 21 features are generated by first of all using standard pre=processing techniques like lemmatization

and stopword-removal. Next they compute features based on different length n-grams, as well as WordNet and vector-space similarity based features using LSA. They use the New York times Corpus and Wikipedia for generating the vectors. They also define and use some features related to n-gram overlap. More details can be found in their paper [6]

We use the SVM-Light[2] framework augmented with tree kernels for performing both our tree similarity and SVR. This code is available in C. For our project, we use the following parameters for running the SVR to combine our tree kernels and the feature vectors from the Takelab.:

- `-z r` use regression rather than classification.
- `-t 5` use a combination of tree kernels and feature vectors
- `-W A` Apply the tree kernel to all tree pairs from the tree pairs and sum.
- `-F 3` Use a 3 degree polykernel for the feature vector from Takelab
- `-C +` Sum the contribution from both the trees and the feature vectors.

  We performed our experiment first using the tree kernel with just the topmost returned parse and next by using the top 3 parses[3].

3.3 Results

Here in table 1 we present the results for our system. These are the Pearson correlations of our scores with the gold standard scores that were human-generated and released as a part of the official competition. We use the perl script that was released by the competition organizers to calculate our results. The column *#Parse Trees* indicates the number of top-k parses that we considered. All these results are for the test data released as a part of Sem Eval 2012. The row *Individual* indicates the results that were obtained when a model trained on the training dataset was tested on its corresponding test dataset (e.g. MSRpar trained model $\rightarrow$ MSRpar test data). *All* indicates the model that was trained by combining training data from all the three datasets (MSRpar, MSRvid and SMTeuroparl) and then running it on each individual test data. The row *Best-2012* indicates the results that were obtained by the winning entry of 2012 - UKP [3]. The *Mean* column indicates the score that was obtained by using a weighted mean of the performance across different test sets based on the dataset size. Note that since SMTnews and OnWN are surprise test sets, they are blank in the Individual row, as there were no models trained corresponding to them.

#Parse trees	Model	MSRpar	MSRvid	SMTeuroparl	OnWN	SMTnews	Mean
top-1	Individual	0.692	0.863	0.504	-	-	-
	All	0.685	0.864	0.477	**0.715**	**0.552**	**0.687**
top-3	Individual	**0.699**	**0.877**	**0.547**	-	-	-
-	Best−2012	0.683	0.873	0.528	0.664	0.493	0.677

Table 1: Pearson Correlation results for our system

A more visual representation comparing our results with the best performer is presented in the bar char (fig 2)

3.4 Discussion

We see that including parse tree similarity via tree kernels, definitely improves our performance, as compared to the winning entries from 2012. Especially when we considered the top-k parse trees as compared to just the topmost parse tree, the results were even better. We note that including the tree kernel based approach seems to serve much better when testing on out of domain data. For instance, our results improve by $7 - 8\%$ in the best case. We believe that it is this result that especially puts forth a strong case for using tree kernel based features for STS. One of the major reasons we believe that tree kernels are more resilient to out-of-domain testing is that they do not

[2] `http://disi.unitn.it/moschitti/Tree-Kernel.htm`

[3] We also conducted the experiment with the top 10 parses but the SVR did not reach convergence even after over 20 hours

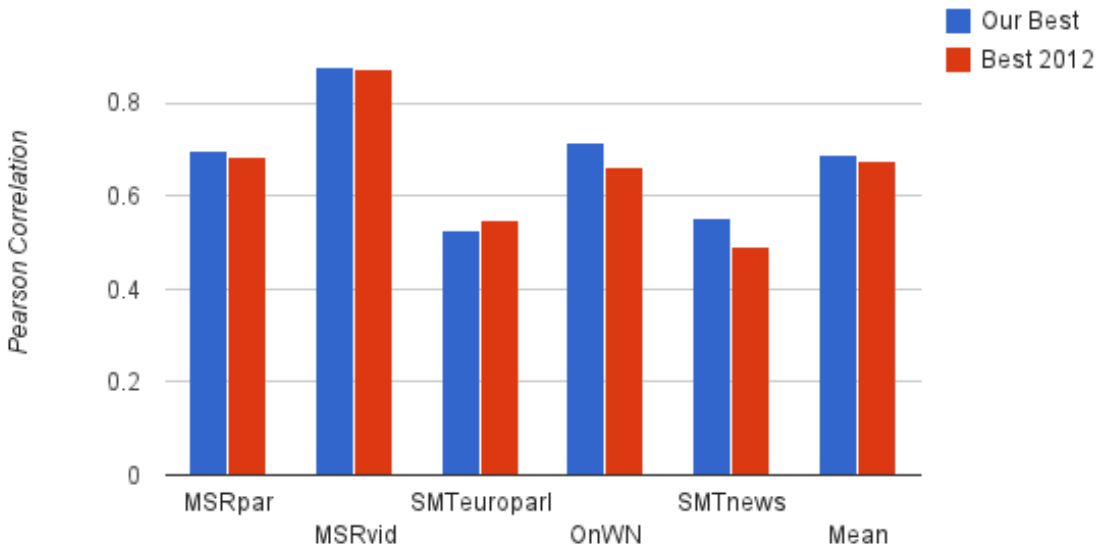

Figure 2: Our system vs the best entry from Sem Eval 2012

rely on the individual words in the sentence per se, rather they consider the syntactic structures of the two sentences. These are much better at capturing the similarity properties of the language at large because similar sentences are likely to have a similar structure even if they are in a different domain - since the grammar underlying them is the same. Thus using such structures serves as a much more informative representation for the purposes of semantic similarity.

4 Future work

There are a couple of further extensions to our current approach we would like to explore.

- First we would like to find more efficient ways of comparing top-k trees between the two sentences. With our current approach, the number of trees to compare blows up quickly, and a more efficient way would therefore be very useful. Our results seem to indicate that increasing k seems to improve the results. Thus we think that it would be interesting to see the performance when the top-10 or top-20 parse trees are compared for every sentence pair.

- We would especially like to explore how well the tree kernel approach is able to do on out-of -domain data, by experimenting with more diverse training and testing sets. For instance Sem Eval 2014 also includes multilingual STS and STS at sentence vs paragraph levels. It would be interesting to see how our approach does int hose cases.

- . Finally we would like to generate our feature vectors more selectively. Currently we did not use the features that were used by the winning team form 2012. We believe that this could further boost our results

5 Conclusion

In this report, we have presented our ideas on augmenting tree kernels based syntactic parse similarities with existing features used by systems [4] for the STS task. We see that using such a feature definitely results in a better performance when compared to the best systems using the same data. Further our results seem to indicate that this approach is much more resilient and better equipped when tested on out-of-domain data. This we see to be the USP of this approach. In the future, we would like to further improve our approach and make it more efficient and scalable for comparing several of the top parses.

Acknowledgements

We would like to thank Prof Mooney for teaching us practical nuances in the course that were immensely useful for the project. We also would like to thank the UTCS condor system without which it would not have been possible for us t conduct our experiments.

References

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

[2] Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of 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*, pages 385–393. Association for Computational Linguistics, 2012.

[3] Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of 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 '12, pages 435–440, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

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

[5] Alessandro Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329. Springer, 2006.

[6] Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. Takelab: Systems for measuring semantic text similarity. In *Proceedings of 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*, pages 441–448. Association for Computational Linguistics, 2012.

[7] Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. ikernels-core: Tree kernel learning for textual similarity. *Atlanta, Georgia, USA*, page 53, 2013.

[8] Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, pages 801–809, 2011.

[4] We came across another approach that also appears to use tree kernels [7]. However we did not have time to compare their performance with ours.