Samsung Poland NLP Team at SemEval-2016 Task 1: Necessity for diversity; combining recursive autoencoders, WordNet and ensemble methods to measure semantic similarity.

Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak and **Piotr Andruszkiewicz**

Samsung R&D Institute Poland
Polna 11
Warsaw, Poland
{b.rychalska, k.pakulska, k.chodorowsk, w.walczak, p.andruszki2}@samsung.com

Abstract

This paper describes our proposed solutions designed for a STS core track within the SemEval 2016 English Semantic Textual Similarity (STS) task. Our method of similarity detection combines recursive autoencoders with a WordNet award-penalty system that accounts for semantic relatedness, and an SVM classifier, which produces the final score from similarity matrices. This solution is further supported by an ensemble classifier, combining an aligner with a bi-directional Gated Recurrent Neural Network and additional features, which then performs Linear Support Vector Regression to determine another set of scores.

1 Introduction

The tasks from the Semantic Textual Similarity (STS) contest have always attracted vivid interest from the NLP community. The goal is to measure the semantic similarity between two given sentences on a scale from 0 to 5, trying to emulate the idea of similarity degrees, thus replicating human language understanding.

After processing two pieces of text, semantic textual similarity software captures degrees of semantic equivalence. One of the goals of the STS task is to create a unified framework for extracting and measuring semantic similarity. Improvements achieved in the course of this task can be useful in many research areas, such as question answering (Marsi and Krahmer, 2005), machine translation (Callison-Burch, 2008), and plagiarism detection (Clough et al., 2002).

We present a solution designed to detect both the similarity between single words and longer, multi-word phrases. It employs two important components: the unfolding recursive autoencoder (RAE) (Socher et al., 2011) and the penalty-award weight system based on WordNet (Miller et al., 2002). First, RAE is used to perform unsupervised learning on parse trees, then the WordNet module adjusts the distances of RAE vectors using awards and penalties based on semantic similarities of words. The complete pipeline includes a deep net (RAE) module, a WordNet module, a normalization module and a sentence similarity matrices computing module.

Another solution that ran in parallel to the RAE pipeline, was the monolingual word aligner (in some cases we used its corrected version with additional features, including a bag-of-words). Finally an ensemble classifier was used to perform Linear Support Vector Regression (Drucker et al., 1996) over the results from all the other classifiers. This included: the base word aligner (Sultan et al., 2015), bi-directional Gated Recurrent Neural Network (Cho et al., 2014; Chung et al., 2014), the RAE with WordNet features and the corrected aligner.

2 System Overview

This section describes the modules that constitute our three runs. Detailed information about the configuration of these runs can be found in Section 3.

2.1 RAE with WordNet Features

RAE with the WordNet module is composed of two major parts: a recursive autoencoder (RAE) for unsupervised training of sentence representations and

Proceedings of SemEval-2016, pages 602–608,
San Diego, California, June 16-17, 2016.

additional WordNet-based submodule for enhancing the performance of the RAE.

The RAE takes unlabeled parse trees and word vectors as input and learns phrase features for each node in the tree. The learned features can be used to recursively reconstruct the vectors at each node in the tree. The encoding part follows the Semantic Depencency Tree Recursive Neural Network (SDT-RNN) structure described in (Socher et al., 2014). In the decoding part, the tree structure used to encode the sentence is mirrored. The reconstruction error (the total error of the network) is counted for all subtrees as the summed Euclidean distance between a subtree's decoded terminal nodes and the original word vectors. The network learns to encode representations of meaningful phrases in tree nodes. We use the word representation vectors published by (Pennington et al., 2014).

The RAE is first trained in an unsupervised way on the Corpus of Contemporary American English (Davies, 2008) combined with SemEval STS training sets released before 2015 (only sentences without labels), then a sentence similarity matrices computing module (Section 2.1.2) is used to generate similarity scores for two candidate sentences. Our first experiment involved a procedure described in (Socher et al., 2011) with unmodified word representation vectors, using Euclidean distance as a measure of word-to-word similarity.

However, we noticed that some pairs of vectors representing related concepts (e.g., 'lady' and 'woman') were located surprisingly far from each other in the Euclidean space, while others were too close. As shown in (Cho et al., 2014), words and phrases which merely belong to the same class of concepts without being exact synonyms have a low distance in Euclidean space (e.g., phrases 'a few seconds' and 'two years' are grouped together). The representation vectors returned by the RAE do not amend this problem. For this reason, we created an additional module which uses WordNet (Miller, 1995) to enhance our word similarity measures in RAE trees. In Table 1 we show the influence of individual modules.

2.1.1 WordNet awards and penalties

The WordNet module adjusts the Euclidean distance between RAE vectors with awards and penalties based on the semantic similarity of pairs of words. We combined the following ideas:

- awarding pairs of words with positive semantic similarity;
- penalizing out-of-context words and disjoint similar concepts;
- propagating scores to higher nodes of the dependency trees.

The concept of semantic similarity reflects the work of (Han et al., 2013), while out-of-context words and disjoint similar concepts reflect the ideas presented in (Han et al., 2015), but there are differences in both implementation and usage.

Awards. WordNet is used to extract semantic relations between pairs of nouns, verbs, adjectives and adverbs. Semantic distance D is measured using the following conditions for two words:

- being equal or first-sense synonyms $(D(x, y) = 0)$, e.g. *car auto*;
- sharing common sense with WordNet frequency of at least 5 $(D(x, y) = 1)$ e.g. *track chase*;
- being hypernyms or two-link hypernyms (applicable for nouns and verbs) $(D(x, y) = 2)$, e.g. *orange citrus*;
- being similar due to satellite synsets (applicable for adjectives and adverbs) $(D(x, y) = 3)$, e.g. *soggy waterlogged*;
- sharing any common sense $(D(x, y) = 3)$, e.g. *grind mash*;
- being derivationally related $(D(x, y) = 4)$, e.g. *rocket missile*;
- being enclosed in the glosses of the other word's meanings $(D(x, y) = 5)$, e.g. *Florida Fla.*

If none of the conditions are met, the semantic distance D is set to a negative value, which facilitates the counting of an award A described below. Thus, effectively, the value of D is an integer such that $D(x, y) \in \{-1, 0, 1, 2, 3, 4, 5\}$.

The semantic distance $D(x, y)$ is transformed to an award A using the formula $e^{-\alpha D(x,y)}$ introduced by (Li et al., 2003), where α is set to 0.25, as this value seemed to yield the best results:

$$A(x, y) = \begin{cases} \beta e^{-\alpha D(x,y)}, & \text{if } D(x, y) \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where β is a positive number (5 by default) used to control the level of adjustment made by the WordNet-related score. If $\beta = 5$ the maximum score for $A(x, y)$ is 5. The Euclidean distances of RAE vectors are usually in the range of $[0, 10]$, thus the parameter ensures that the WordNet-related similarity is sufficiently important.

Penalties. The out-of-context penalty for word x, $OOC(x)$, is defined as a penalty for a word not paired in the second input sentence SS (Han et al., 2015). The word is not paired if its semantic similarity (or award) $A = 0$ with all the words (referenced below by the index i) in the second sentence:

$$OOC(x) = \begin{cases} -1, & \text{if } \sum_i A(x, SS(i)) = 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

We allow three strategies for out-of-context penalization: penalize all recognizable parts of speech (nouns, verbs, adjectives and adverbs), penalize only nouns, penalize only physical objects (i.e. words which have $physical\_object$ in their WordNet's hypernyms path). The third option is used by default, since both the original research of (Han et al., 2015) and ours suggest that it usually is the best option (although, in a minority of tests, the penalization of all out-of-context nouns yields better results).

We also penalize disjoint similar concepts. Disjoint similar concepts $DSC(x, y)$ are defined as 'special care antonyms' or words of disjoint meaning (i.e. *Monday Tuesday*). In our solution they are found using WordNet's hypernyms hierarchy. If two words have a common direct hypernym, they are disjoint similar concepts (e.g. both *Monday* and *Tuesday* have *weekday* as a common hypernym in the WordNet hierarchy). By default, the $DSC(x, y)$ function returns a penalty of -2 when two words are found to be antonyms or disjoint similar concepts, and 0 otherwise. Thus, the penalty P for two words x and y is:

$$P(x, y) = OOC(x) + OOC(y) + DSC(x, y) \quad (3)$$

A complete framework of WordNet-related awards and penalties is defined by:

$$sim_{WN} = \begin{cases} A(x, y), & \text{if } D(x, y) \geq 0 \\ P(x, y), & \text{if } P(x, y) < 0 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Propagation. In the Sentence similarity matrices computing module 2.1.2, scores are calculated for all sentence subtrees, while WordNet's awards and penalties are calculated for words. Thus, to use WordNet's scores on all subtrees, the awards and penalties have to be propagated up, until they affect all nodes. To propagate WordNet's scores on a tree containing more than one word, we define a function:

$$sim_{WN}(subtree) = \sum_{i=1}^{n} \frac{sim_{WN}(i)}{depth(i)} \quad (5)$$

where i is an index for a single leaf in the subtree, and n is the total number of leaves in the subtree. $sim_{WN}(i)$ is a similarity score for a leaf i. The scores for particular leaves are divided by their depth $depth(i)$ relative to the root of the subtree to account for their importance in the more complex trees, and then the scores are summed up at the root. For example, if we have a leaf with a score of 0.75, then this score is added up at the root level with its full weight only if the leaf is a direct child of the root. If it is located deeper in the subtree, the score is divided by this leaf's depth relative to the root of the subtree. The same procedure applies to all the leaves in the current subtree. In Table 1 we compare the efficiency of the WordNet module with and without propagation.

An alternative strategy for incorporating the WordNet scores is to refine the vectors associated with particular leaves and then use these refined vectors to recompute complex tree nodes using RAE. For example, given the WordNet similarity ϵ for words 'woman' and 'lady', and the vectors A for 'woman' and B for 'lady', the vector A is refined using the following formula:

$$A_{refined} = \epsilon A + (1 - \epsilon)B. \quad (6)$$

The vector B stays the same. The tests have proven that the former strategy offers better results, so we decided to stick with it.

Table 1: The synergy of all parts of the solution (weighted mean presented).

Test set	RAE	RAE with WordNet	RAE with Propagation	Final RAE based solution
Answers forums	0.4724	0.4916	0.5404	0.6836
Answers students	0.7066	0.7185	0.7085	0.7679
Belief	0.6109	0.6002	0.6418	0.7517
Headlines	0.706	0.7115	0.7114	0.8315
Images	0.7469	0.7797	0.7952	0.8625
Weighted mean	0.6753	0.6889	0.7016	0.7949

2.1.2 Sentence Similarity Matrices Computing

As a first step in our full algorithm, the RAE computes vectors for every node in the dependency parse tree. Then the subtrees of these trees are used to create the distance matrix. The matrix is created in a number of steps: the trees are traversed in level order, the subtrees are then sorted by depth and the leaves representing stop words are removed[1]. The remaining subtrees are used to construct the distance matrix, which is then filled with Euclidean distance measures d between each pair of subtrees x and y.

$$D_{RAE} = d(x, y) - sim_{WN} \tag{7}$$

The above score is further transformed in two ways: it is made certain that the score value falls within the range 0-5, and that the distance is set to 0 if the WordNet similarity sim_{WN} has a maximum value:

$$D'_{RAE} = \begin{cases} 0, & \text{if } sim_{WN} = max \\ 0, & \text{if } D_{RAE} < 0 \\ 5, & \text{if } D_{RAE} > 5 \\ D_{RAE}, & \text{otherwise} \end{cases} \tag{8}$$

The original score D_{RAE} is replaced with the adjusted score D'_{RAE} in the distance matrix. Finally, we use dynamic pooling module as described in (Socher et al., 2011). The pooling module accounts for the varying lengths of the two trees.

[1]Stop words list contains about 60 most common words in training data set and all punctuation characters.

2.1.3 Final RAE-based solution

The final score was produced by Linear Support Vector Regression over cells from the distance matrices after pooling as well as 12 additional features:

- adjustment of roots (the Euclidean distance between WordNet-adjusted tree roots);
- cosine distance between vectors representing tree roots of sentences;
- information about the negation status of the two sentences (if both sentences contain/ do not contain negation = true, otherwise = false);
- mean out of context penalty over full tree;
- mean disjoint penalty over full tree;
- mean WordNet similarity score over full tree;
- score from aligner (Section 2.2);
- if both sentences agree on the numbers (true for no numbers or the same numbers; false otherwise);
- if both sentences have the same numbers (binary);
- the absolute difference in tokens between two sentences;
- if the numbers in one sentence contain the numbers from the second sentence (binary);
- the percentage of tokens similarity between two sentences.

A new SVM classifier was created for every test set, since for every test set a different subset of training sets was used[2]. Distance matrices for all classifiers were created and normalized independently.

$$norm_c = 0.4(\frac{\max(\min(\frac{c}{\mu}, 3\sigma), -3\sigma)}{3\sigma} + 1) + 0.1 \tag{9}$$

First the normalization process was used to calculate mean μ and standard deviation σ for the matrix, next we performed calculations according to Equation 9 for every cell c of the matrix. The equation comes from (Socher et al., 2013) and normalizes the values to range $[0.1, 0.9]$.

[2]The training data contains all previous SemEval datasets from 2012 to 2015.

Table 2: Mapping training sets to test sets.

Test set	Training sets
Answer-answer	answers-students 2015, belief 2015.
Headlines	MSRpar 2012 (training and test set), SMTnews 2012, deft-news 2014, headlines 2013, headlines 2014, headlines 2015, images 2014, images 2015.
Plagiarism	MSRpar 2012 (training and test set), answers-students 2015.
Postediting	deft-news 2014, deft-forum 2014, SMTnews 2012.
Question-question	deft-news 2014, deft-forum 2014, belief 2015.

2.2 Aligner

As a monolingual word aligner we use two algorithms: a basic aligner and a corrected aligner. Both are based on the aligner described in (Sultan et al., 2014). The basic algorithm performs the following steps for the two sentences: align identical word sequences, align named entities, align content words using dependencies and align content words using surrounding words. Scoring is calculated according to (Sultan et al., 2015):

$$score(S^1, S^2) = \frac{n^a(S^1) + n^a(S^2)}{n(S^1) + n(S^2)},$$

where $n(S^i)$ and $n^a(S^i)$ are the number of content words and aligned content words in a sentence S^i, respectively.

An aligner using only the basic algorithm could not handle negations and antonyms well, so we modified it by adding two modules. The negation module checks whether there is a negation component present in only one sentence, and if so, the module reduces the score to 0. The antonym module verifies whether the two sentences contain at least one pair of antonyms from a list based on the WordNet, and if so, also reduces score to 0.

The corrected aligner is a Linear Support Vector Regression (Drucker et al., 1996) using the following features: the modified basic aligner feature, Bag of Words features inspired by (Han et al., 2015) (element-wise absolute value difference between vectors for words and bigrams, sentences' length difference, percentage of exact lemma to lemma matches) and additional features used in (Hänig et al., 2015):

- length of the longest common subsequence of characters (some characters may be skipped),
- length of the longest common sequence of characters,
- cosine similarity between vectors of words,
- edit distance between sentences,
- WordNet word overlap (Šarić et al., 2012).

2.3 Ensemble

The ensemble classifier was actually a Linear Support Vector Regression over results from the other classifiers used for semantic similarity measurement. Each one of them returned score from 0 up to 5. The following classifiers were chosen for the ensemble approach:

- modified basic aligner, presented in Section 2.2;
- Bi-directional Gated Recurrent Neural Network (Cho et al., 2014; Chung et al., 2014) with the output neural network described in (Tai et al., 2015);
- RAE with WordNet Features, described in Section 2.1;
- corrected aligner, described in Section 2.2

The training data set was split into 75% vs 25%. All classifiers except the aligner (which does not need to be trained) were trained on the 75%. The ensemble classifier was trained on a subset of the remaining data set.

Scores returned by the above classifiers were used as features in the Linear Support Vector Regression. The final result was rescaled to get score from the $[0, 5]$ range.

3 Evaluation

In order to separately train models for each evaluation set, we created reference test sets that are similar to the evaluation sets, e.g. for headlines we used

Table 3: The results obtained over three runs in the evaluation period of SemEval 2016.

	Answer-answer	Headlines	Plagiarism	Postediting	Question-question	Weighted mean
RAE (AE)	0.6577	0.8180	0.8129	0.7885	0.5867	0.7357
Ensemble (EN1)	0.6924	0.8275	0.8414	0.8352	0.6870	0.7781
Merged (EN2)	0.6924	0.8275	0.8129	0.8352	0.5867	0.7547

Table 4: Comparison of our solutions with the 2015 winning system on SemEval 2015 evaluation datasets.

Run	Answers forums	Answers students	Belief	Headlines	Images	Weighted mean
RAE (AE)	0.6836	0.7747	0.7517	0.8363	0.8625	0.7949
Aligner (with our modifications)	0.6725	0.7715	0.7282	0.8223	0.8478	0.7854
BiGRU	0.5424	0.5925	0.5917	0.7947	0.8588	0.7032
Ensemble (EN1)	0.6489	0.7664	0.7549	0.854	0.8884	0.8027
Merged (EN2)	0.6836	0.7747	0.7549	0.8540	0.8884	**0.8091**
DLS@CU	0.7390	0.7725	0.7491	0.8250	0.8644	0.8015

30% of headlines 2015; for answer-answer we used 30% of answers-students 2015; for plagiarism 30% of MSRpar 2012 test set; for postediting 30% deft-news 2014 and for question-question 30% of deft-forum 2014. These randomly selected samples that constituted the test sets were removed from the training sets. We used these reference test sets to find the best parameters of our models and chose the best model for run EN2. The final model uses all samples from sets assigned to each evaluation set in Table 2. Our final results are presented in Table 3.

For the AE run, we used RAE with WordNet Features, as described in Section 2.1. For each test, a separate classifier was created with its own training set, as presented in Table 2. The mapping was based on the average number of words per sentence in the set.

For the EN1, run we used the ensemble model described in Section 2.3. In the EN2 model we chose either RAE or ensemble based on the results for test sets matched with evaluation sets.

As shown in Table 3, the ensemble model (EN1) yields better results than RAE (AE) for all sets. Thus, the merged model (EN2) falls between the two.

We also present the results (Table 4) of our solution for SemEval 2015 sets. Comparing them with the best run from SemEval 2015 competition (weighted mean), we concluded that bi-GRU yields the worst results. Second and third worst results came from the modified aligner and RAE respectively. The ensemble and merged model yield the best results that surpass the performance of the 2015 winning solution.

4 Conclusions and Future Work

Our solution combines a vector similarity feature derived from word embeddings without losing the information contained in lexical similarity relations. As it turned out, one of the primary limitations of our paraphrase detection system is its heavy reliance on word order, which makes the solution less universal in its application. The other drawback of converting words to word vectors is being unable to account for situations where the same information is formatted differently (for instance, units of measurement, time expressions, etc.) Thus, our future works include improving our preprocessing module, so that it would produce a unified input, e.g., all numbers written in words will be converted into numerals and all dates will be unified into one format. We will also use specifically designed training modes to prevent overfitting and create a new curriculum learning dataset to make RAE training easier.

References

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–205. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, 1406.1078.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, 1412.3555.

Paul Clough, Robert Gaizauskas, Scott SL Piao, and Yorick Wilks. 2002. Meter: Measuring text reuse. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

Mark Davies. 2008. The Corpus of Contemporary American English: 520 million words, 1990-present. http://corpus.byu.edu/coca/.

Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alexander J. Smola, and Vladimir Vapnik. 1996. Support vector regression machines. In Michael Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 155–161. MIT Press.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. Umbc ebiquity-core: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, volume 1, pages 44–52.

Lushan Han, Justin Martineau, Doreen Cheng, and Christopher Thomas. 2015. Samsung: Align-and-differentiate approach to semantic textual similarity. *SemEval-2015*, pages 172–177.

Christian Hänig, Robert Remus, and Xose de la Puente. 2015. Exb themis: Extensive feature extraction from word alignments for semantic textual similarity. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 264–268, Denver, Colorado, June. Association for Computational Linguistics.

Yuhua Li, Zuhair A Bandar, and David McLean. 2003. An approach for measuring semantic similarity between words using multiple information sources. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):871–882.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation*, pages 109–117. Citeseer.

George A Miller, C Fellbaum, R Tengi, P Wakefield, H LANGONE, and BR HASKELL. 2002. Wordnet: A lexical database for the english language. 2002. *Available from: http://wordnet.princeton.edu/*.

George A Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, nov.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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*.

Richard Socher, Chris Manning, and Yoshua Bengio. 2013. Naacl 2013 tutorial: Deep learning for nlp.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Dls@cu: Sentence similarity from word alignment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 241–246, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. Dls@cu: Sentence similarity from word alignment and semantic vector composition. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 148–153, Denver, Colorado, June. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. March.

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.