UniMelb at SemEval-2016 Task 3: Identifying Similar Questions by Combining a CNN with String Similarity Measures

Doris Hoogeveen[1,2]
dhoogeveen@student.unimelb.edu.au

Yitong Li[2]
yitong14@student.unimelb.edu.au

Huizhi Liang[2]
oklianghuizi@gmail.com

Bahar Salehi[1,2]
bsalehi@student.unimelb.edu.au

Long Duong[1,2]
lduong@student.unimelb.edu.au

Timothy Baldwin[2]
tb@ldwin.net

[1]NICTA
[2]Department of Computing and Information Systems
The University of Melbourne
VIC, Australia

Abstract

This paper describes the results of the participation of The University of Melbourne in the community question-answering (CQA) task of SemEval 2016 (Task 3-B). We obtained a MAP score of 70.2% on the test set, by combining three classifiers: a NaiveBayes classifier and a support vector machine (SVM) each trained over lexical similarity features, and a convolutional neural network (CNN). The CNN uses word embeddings and machine translation evaluation scores as features.

1 Introduction

In this paper we present the system we submitted for the community question-answering (CQA) task of the SemEval 2016 workshop (Task 3-B: Nakov et al. (2016)). By finding an automatic way to answer new questions based on existing ones, we unlock an enormous wealth of information stored in online CQA archives. In the task as specified for the SemEval workshop, we were given 70 query questions. Each question had at most ten candidate questions, which we were to re-rank according to their similarity to the query question. Each question consisted of a title and a description. The data was taken from the Qatar Living forum.[1] The training data set was small: 267 queries, with ten candidate duplicate questions each. The candidate questions were originally labelled according to the three classes: RELEVANT, PERFECTMATCH and IRRELEVANT. Subsequently, however, RELEVANT and PERFECTMATCH were merged into a single class. In an ideal ranking, documents with relevant labels were to be ranked higher than the IRRELEVANT documents.

The system we submitted combines the predictions of three different classifiers through simple voting. The first two classifiers (naive Bayes and SVM) made use of semantic similarity measures as features, and the third one was a convolutional neural network (CNN) that used word embeddings and machine translation evaluation scores as input. The combined system achieved a MAP score of 70.2% on the test set.

2 Approach

Our system combined the scores of three different classifiers based on simple voting. If at least two of the three classifiers considered a candidate question relevant to a query question, it was considered to be relevant. The candidate questions were then ranked according to this judgement, with the relevant ones on top (in any order, since this was not taken into account in the official evaluation). In this section we will describe the details of the three classifiers.

2.1 String Similarity Features (SS1)

Our first set of features consisted of string similarity measures, which we selected based on our recent success in applying these features to measure the compositionality of multiword expressions (Salehi and Cook, 2013), to estimate semantic textual similarity (Gella et al., 2013), and to detect cross-lingual textual entailment (Graham et al., 2013).

To measure the string similarity between two questions, the titles and the descriptions of the questions were lemmatized using NLTK (Bird, 2006).

[1] http://www.qatarliving.com/forum

Hoogeveen, Doris, Huizhi Liang, Long Duong, Yitong Li, Bahar Salehi and Timothy Baldwin (to appear) UniMelb at SemEval-2016 Task 3: Identifying Similar Questions by Combining a CNN with String Similarity Measures. In Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016), San Diego, USA.

We used two string similarity measures in this study: longest common substring and the Smith-Waterman algorithm.[2] The output of each measure was normalized to the range of $[0, 1]$, where 0 indicated that the questions were completely different, while 1 showed that they were identical. More details on the string similarity measures and how we normalise the scores are described in Salehi and Cook (2013).

Our primary experiments showed that measuring string similarity between the titles of questions led to a higher accuracy than using the question descriptions. Therefore, we only considered the titles in our final run. Ultimately, in order to combine all string similarity measures into one score, we used the linear kernel SVM as implemented in the scikit-learn package[3], using the default parameters.

2.2 String Similarity Features 2 (SS2)

For our second model, we used five more lexical similarity measures as features: the Jaccard similarity, cosine similarity calculated over binary term vectors, the overlap coefficient, de Sørensen-Dice coefficient, and Kullback-Leibler (KL) divergence. With these features we obtained the best results by using both the title and the description of the question. In contrast to our first classifier, no stemming or lemmatization was applied because it was found not to make any difference. The classifier we used was naive Bayes.[4]

2.3 Convolutional Neural Network (CNN)

The third classifier we used was a convolutional neural network ("CNN"). CNN structures have been shown to be very successful in speech recognition and computer vision tasks (Graves et al., 2013; Krizhevsky et al., 2012). Recently, they have also been applied to natural language tasks, and again, have achieved good results (Collobert and Weston, 2008; Yin and Schütze, 2015).

Kalchbrenner et al. (2014) developed a CNN-based model that can be used for sentence modelling problems. With several combinations of convolutional filters and dynamic k-max pooling filters, the model is very good at capturing features on both the local word level and the global sentence level. The word-level features are combined in several stages to model sentences. This characteristic of capturing meaning at different levels is particularly attractive for the target domain, as two questions with similar meaning may have a very different surface form. A neural network that can model meaning at the sentence level may recognise two questions as being similar even though they have very little lexical overlap. This is the reason we decided to use Kalchbrenner's CNN for our task, enhanced with some aspects of Yin and Schütze (2015)'s model, who used a CNN for Paraphrase Identification (PI).

In our approach, we used the CNN to compare two sentences at different levels in the model. The similarity scores obtained in this way were combined with several machine translation evaluation scores and fed into a multi-Layer perceptron (MLP) classifier to get a final similarity score. We decided to make this final classifier a neural network too so that we could get a non-linear output for the newly added features. The expectation is that this will improve the classification.

In the following subsections we explain the details of the CNN: how to model sentences on different levels and how to generate the features using sentence embeddings. We also explain some other features that we added, and how we trained the model.

2.3.1 Model Overview

Figure 1 show the CNN model. Each tokenised input sentence $\mathbf{S}$ consists of n words $\{w_1, w_2, \cdots, w_n\}$, where n denotes the length of the sentence. Each word has a word vector $e_n \in \mathbb{R}^d$, where d is the dimension of the word vector. All the word vectors combined form a sentence matrix embedding $\mathbf{E} \in \mathbb{R}^{d \times n}$, which is used as the input to our CNN model. Different input sentences will have different lengths, but this is not a problem at this stage. We deal with this issue when comparing the sentences to obtain features, as explained in Section 2.3.2.

For each convolutional level l, we convolved the input matrix with a wide one-dimensional convolution filter, and generated a convolutional matrix

[2] We also experimented with Levenshtein and modified Levenshtein, but we did not observe any improvement when using these features.

[3] http://scikit-learn.org

[4] We used the version implemented in the Weka toolkit: `http://www.cs.waikato.ac.nz/ml/weka/`, using the default settings.

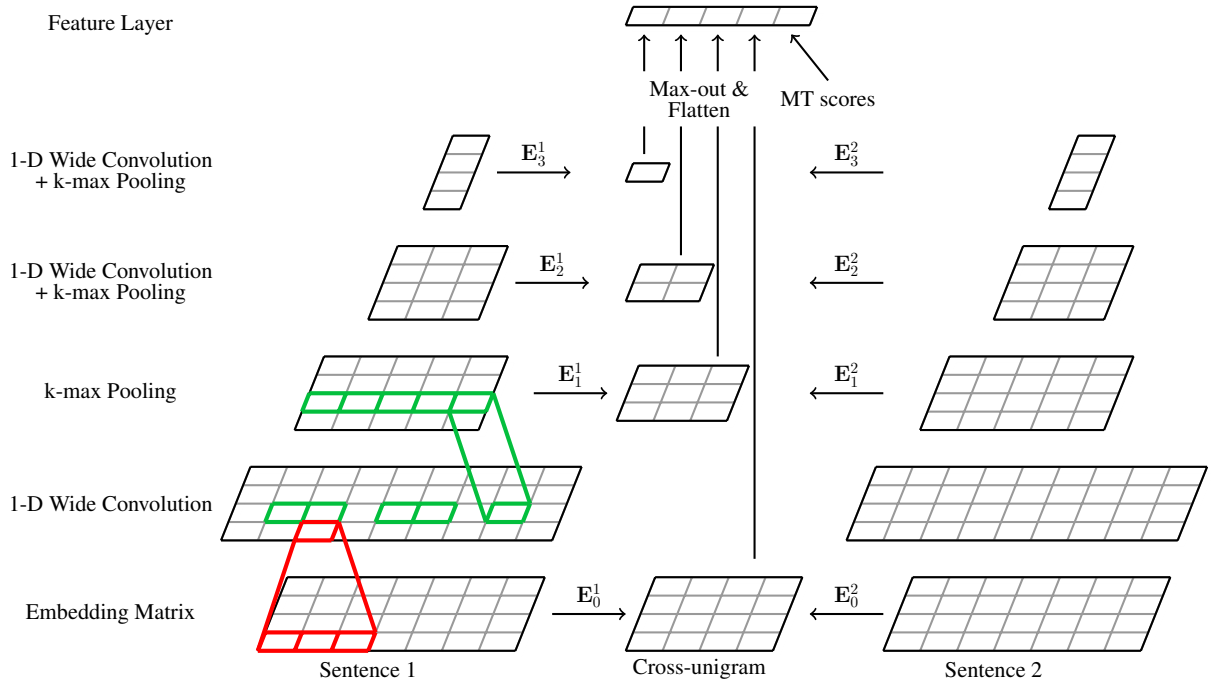

Figure 1: The Convolutional Neural Network.

$\mathbf{C} \in \mathbb{R}^{d\times(n+m-1)}$, where m is the filter width, which is set to 3 (see the red highlighting in Figure 1). We then applied a non-linear function (rectified linear units (ReLU)) to get the convolutional layer $\mathbf{C\prime} = ReLU(\mathbf{C})$. C and $C\prime$ are combined in Figure 1 and shown as "1-D Wide Convolution". Next, we applied Kalchbrenner et al. (2014)'s dynamic k-max pooling approach on $\mathbf{C}'$. For each dimension of $\mathbf{C}'$, we extracted a maximum of k features and calculated the pooling layer. The output of the pooling layer is $\mathbf{E}_l \in \mathbb{R}^{d\times k_l}$. $\mathbf{E}_l$ is the sentence embedding of level l and is used as the input for the next level $(l+1)$ of the CNN.

After a series of such convolution operations, we get a deep CNN structure as a representation of each question.

2.3.2 Features

As explained in the previous section, the input features to the CNN consisted of word embeddings. We used constant word embeddings directly from Mikolov et al. (2013)'s pre-trained word embeddings model, with dimension $d = 300$.

After applying the convolution operations on a question pair $(\mathbf{S}_1, \mathbf{S}_2)$, we obtained an embedding set for each question: $\{\mathbf{E}_0^1, \mathbf{E}_1^1, \cdots, \mathbf{E}_l^1\}$ and $\{\mathbf{E}_0^2, \mathbf{E}_1^2, \cdots, \mathbf{E}_l^2\}$, where l is the number of levels in our CNN structure. Different embeddings represent the semantics of the question with a different granularity. $\mathbf{E}_0$ corresponds to the input word embeddings, which represent the semantics of the question at the word level. Each increase in subscript represents a convolutional level in the model. The higher up we get, the more convolved the features will be, until we end up with $\mathbf{E}_l$, which represents the document-level meaning of the question.

To obtain the features for the final classifier, we determined the similarity of the embedding matrices $\mathbf{E}_l^1$ and $\mathbf{E}_l^2$ for each level, by comparing each vector in $\mathbf{E}_l^1$ to each vector in $\mathbf{E}_l^2$; a process known on the word level as cross-unigram comparison. The similarity of the vectors was calculated using both cosine similarity and Euclidean distance. This resulted in two matrices per question pair per level, which we concatenated. To reduce the size of these low-level matrices we applied a two-dimensional maxout filter on them. The height and width of the filter were adjusted for different sentence lengths of the questions $\mathbf{S}_1$ and $\mathbf{S}_2$, to ensure that the output always had the same length. Next we flattened the matrix into a vector and used this as our sentence embed-

ding similarity features that formed part of the input to our final classifier.

Apart from the sentence embedding similarity features, we also used several machine translation evaluation measures as extra features before applying the final classifier. Machine translation evaluation measures are designed to detect whether two sentences have a similar meaning or not. They have been shown to be useful features for paraphrase identification (Madnani et al., 2012), a task very similar to ours. The measures we used were BLEU (Papineni et al., 2002), NIST (Doddington, 2002), METEOR (Banerjee and Lavie, 2005), Ter (Snover et al., 2006), Ter-Plus (Snover et al., 2009), and MaxSim (Chan and Ng, 2008). After adding these additional features to the sentence embedding similarity features, we fed our feature vectors into a Multiple Layer Perceptron (MLP) classifier to get a final similarity score.

2.3.3 Training the model

In this section we will explain how we trained our model. Our CNN consisted of three convolutional layers, with a MLP classifier at the top. The MLP combined three fully-connected hidden layers, which contained 512 nodes each and ReLU as its activation function, with a softmax layer on the top.

For the network training, we used AdaDelta (Zeiler, 2012) to update the weights of the model, and set the initial learning rate to $\alpha = 10^{-4}$. Dropout (Srivastava et al., 2014) was added to the input layer of the MLP, with L_2-regularization. The dropout rate was set to 0.5.

3 Experiments

In this section, we will describe the experimental setup and the results of our experiments.

3.1 Dataset

Table 1 presents basic statistics for the SemEval-2016 Task 3-B dataset. Each query question was paired up with at most ten archived questions, which we had to re-rank according to relevance. The dataset was partitioned into three components: (1) a training set, (2) a development set, and (3) a test set.

	Training Set	Development Set	Test Set
query questions	267	50	70
archived questions	2,669	500	700

Table 1: The number of query and archived question pairs in the SemEval-2016 dataset

3.2 Results

To evaluate the effectiveness of the different feature sets, we report on the results of both the combined model and the separate classifiers in Tables 2 and 3. Baselines 1 and 2 are the official baselines as given by the SemEval organisers. The information retrieval (IR) baseline was produced using Google to rank the candidate questions.

We use the Mean Average Precision (MAP) as our primary evaluation metric,[5] but also report the Average Recall (AvgRec), Mean Reciprocal Rank (MRR), Precision (P), Recall (R), F1-measure (F1), and Accuracy (Acc) scores.

On the development set, we obtained the best MAP score with the majority voting model, but on the test set we did not see the same result. On the test set, the CNN model by itself obtained the best results. It is interesting to see that all three models separately performed better on the test set than on the development set, but the combined model did not.

Although models SS1 and SS2 both make use of string similarity measures, they produced different classification outputs for 60.2% of the development set queries. SS1's predictions differed from the CNN's in 54.8% of the development queries, and SS2's predictions differed from the CNN's in 72.6% of the development queries. The fact that the three models produced such different results, while each performed reasonably well, was the motivation for combining them.

One reason for the different results on the test and development sets might be the difference in the class balance. In the development set, 43% of the candidate questions were labeled as relevant, and 57% as irrelevant. In the test set this was 33% and 67% respectively. The training data resembled the development set more than the test set, with 45% relevant

[5] This is also what the official ranking of the participating systems is based on.

Test Set

Model	MAP(%)	AvgRec(%)	MRR	Acc(%)	P(%)	R(%)	F1(%)
Baseline 1(IR)	74.75	88.30	83.79	-	-	-	-
Baseline 2 (random)	46.98	67.92	50.96	45.20	40.43	32.58	73.82
SS1	71.11	85.71	81.20	70.00	55.02	54.08	54.55
SS2	72.95	88.51	82.26	70.29	55.92	50.64	53.15
CNN	**73.04**	87.72	82.21	73.43	60.09	60.09	60.09
Majority voting	70.20	86.21	78.58	74.57	63.96	54.08	58.60

Table 2: The official evaluation results for the SemEval-2016 Test Set

Development Set

Model	MAP(%)	AvgRec(%)	MRR	Acc(%)	P(%)	R(%)	F1(%)
Baseline 1(IR)	71.35	86.11	76.67	-	-	-	-
Baseline 2 (random)	55.95	73.23	62.23	48.80	44.42	76.64	56.16
SS1	70.06	85.95	77.33	68.80	68.35	50.47	58.06
SS2	68.96	85.58	74.00	64.60	64.12	39.25	48.70
CNN	72.71	87.86	79.33	75.20	76.47	60.75	67.71
Majority voting	**73.13**	88.93	80.00	72.20	64.48	78.04	70.61

Table 3: The official evaluation results for the SemEval-2016 Development Set

and 55% irrelevant candidate questions. We suspect that more training data is needed to obtain consistent results.

It would be interesting to see whether the scores improve when we add the string similarity features to the CNN directly (thereby losing the majority voting component), in the same way as we added the machine translation evaluation features. We leave this for future work.

4 Summary

In this paper, we proposed a method based on the combination of three different classifiers for the task of duplicate question ranking, in the context of SemEval 2016 Task 3-B. The classifiers we combined were a CNN using word embeddings and machine translation evaluation metrics, and two classifiers that used lexical similarity features: a naive Bayes classifier and a support vector machine (SVM). The results we obtained on the test set were quite different from the results on the development set, which may be explained by the small size of the training data set.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, volume 29, pages 65–72.

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, pages 69–72.

Yee Seng Chan and Hwee Tou Ng. 2008. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proceedings of the 46th Annual Meeting of the ACL: HLT (ACL 2008)*, pages 55–62.

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*, pages 160–167.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Spandana Gella, Bahar Salehi, Marco Lui, Karl Grieser, Paul Cook, and Timothy Baldwin. 2013. Unimelb nlp-core: Integrating predictions from multiple domains and feature sets for estimating semantic textual similarity. *Atlanta, USA*, page 207.

Yvette Graham, Bahar Salehi, and Timothy Baldwin.

2013. Umelb: Cross-lingual textual entailment with word alignment and string similarity features. *Atlanta, USA*, page 133.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLTNAACL)*, pages 182–190.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Preslav Nakov, Lluís Màrquez, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, San Diego, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Bahar Salehi and Paul Cook. 2013. Predicting the compositionality of multiword expressions using translations in multiple languages. In *Second Joint Conference on Lexical and Computational Semantics (*SEM 2013)*, pages 266–275.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, pages 223–231.

Matthew G Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. TER-Plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.