

Uwe Kloos • Natividad Martínez • Gabriela Tullius

# Tagungsband **Informatics Inside connect(IT);**

Informatik-Konferenz der Hochschule Reutlingen  
20. Mai 2020

*Herausgeber*

Prof. Dr. Uwe Kloos	Prof. Dr. Natividad Martínez	Prof. Dr. Gabriela Tullius
Fakultät Informatik	Fakultät Informatik	Fakultät Informatik
Hochschule Reutlingen	Hochschule Reutlingen	Hochschule Reutlingen
Reutlingen, Deutschland	Reutlingen, Deutschland	Reutlingen, Deutschland



ISBN: 978-3-00-065431-2

Hochschule Reutlingen  
Fakultät Informatik  
Human-Centered Computing  
Alteburgstraße 150  
72762 Reutlingen

© 2020 Hochschule Reutlingen

Autoren: Jessica Konecki, Katharina Lohr, Adrian Schmidt, Antonia Schöning,  
Sven Vennebusch und Dominik Walzcak

Gestaltung, Illustration: Jessica Konecki

Lektorat: Jessica Konecki, Katharina Lohr, Adrian Schmidt, Antonia Schöning,  
Sven Vennebusch und Dominik Walzcak

Weitere Mitwirkende: Claudiu Bräuer, Yoan Dimitrov, Laura Discher, Alexander  
Kahabka, Nicolai Kraus, Markus Rein, Wafa Sadri und Marc Viertel

infoinside@reutlingen-university.de  
www.infoinside.reutlingen-university.de

Das Werk, einschließlich seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung ist ohne Zustimmung des Verlages und des Autors unzulässig. Dies gilt insbesondere für die elektronische oder sonstige Vervielfältigung, Übersetzung, Verbreitung und öffentliche Zugänglichmachung.

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;  
detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

# Inhaltsverzeichnis

## Teil I Medizin

<b>Computergestützte Lösungen für Herausforderungen in der Kryochirurgie</b> .....	1
Nicolai Kraus	
<b>Haptisches Feedback in der medizinischen Robotik</b> .....	10
Adrian Schmidt	
<b>Disease Detection and Health Monitoring Through Breath Analysis</b> .....	21
Marc Viertel	
<b>Benutzerstudie an einer Software von Kontrastmittelinjektoren für CT &amp; MRT</b> .....	29
Jessica Konecki	
<b>Regulatorische Anforderungen an eine Qualitätsmanagement-Software für Medizinprodukte</b> .....	41
Laura Discher	
<b>Analyse des Requirements Engineering eines KIS-Herstellers</b> .....	53
Katharina Lohr	
<b>Traceability für Usability Engineering bei medizintechnischen Produkten</b> .....	65
Antonia Schöning	

## Teil II Neuronale Netze

<b>Combining word embeddings and convolutional neural networks to detect duplicated questions</b> .....	74
Yoan Dimitrov	
<b>Visualisierung von neuronalen Netzen zur Unterstützung beim Training</b> .....	83
Claudiu Bräuer	
<b>Testumgebungen für Augmented Reality zur Anzeige autonomer Fahrfunktionen</b> .....	95
Markus Rein	

# Combining word embeddings and convolutional neural networks to detect duplicated questions

Yoan Dimitrov

Reutlingen University

Yoan.Dimitrov@student.Reutlingen-University.de

## Abstract

Detecting semantic similarities between sentences is still a challenge today due to the ambiguity of natural languages. In this work, we propose a simple approach to identifying semantically similar questions by combining the strengths of word embeddings and Convolutional Neural Networks (CNNs). In addition, we demonstrate how the cosine similarity metric can be used to effectively compare feature vectors. Our network is trained on the Quora dataset, which contains over 400k question pairs. We experiment with different embedding approaches such as Word2Vec, Fasttext, and Doc2Vec and investigate the effects these approaches have on model performance. Our model achieves competitive results on the Quora dataset and complements the well-established evidence that CNNs can be utilized for paraphrase recognition tasks.

## Keywords

Natural Language Processing, Word Embeddings, Sentence Classification, Convolutional Neural Networks, Deep Learning

## CCS Concepts

• Computing methodologies ~ Machine learning ~ Artificial intelligence ~ Natural language processing

## 1 Introduction

Over the past decade, numerous online community forums such as Stack Overflow and Quora have emerged, enabling users to ask domain-specific questions and obtain appropriate answers. Since many users share the same interests and often ask the same questions, these online forums face the challenge of identifying semantically similar or equivalent questions in order to avoid redundancies and improve their service quality and user experience [3]. Successfully measuring the semantic similarity between pairs of questions is, however, a very difficult hurdle to overcome, since natural languages are ambiguous at all major levels of linguistics. Two questions could vary in length, punctuation, sentence structure, and choice of words and still have the same intention. For example, consider the following two questions: *How old are you?* *What is your age?*

---

Supervisor: Prof. Dr. Benjamin Himpel  
Reutlingen University  
Benjamin.Himpel@Reutlingen-  
University.de

Informatics Inside  
Wissenschaftliche Vertiefungskonferenz  
20. Mai 2020, Reutlingen University  
Copyright 2020 Yoan Dimitrov

Both questions have the same intention and can be answered with the same answer, but they both have a different vocabulary and sentence structure. These aspects, although seemingly trivial, are not always easy to detect. Recent years have seen great progress in detecting semantically similar questions with deep learning. In particular, Recurrent Neural Networks (RNNs), which are inherently ideal for sequential data such as text, have achieved state-of-the-art results [6, 7, 9, 19, 20, 23].

In this work, we tackle the task of identifying semantically similar questions from the officially released Quora dataset. We adopt the question similarity definition suggested by Bogdonova et al. [3]:

*“Two questions are considered to be semantically equivalent if they can be answered by the same answer.”*

Contrary to popular approaches, we propose a simple and yet effective method based on word embeddings and a Convolutional Neural Network (CNN) architecture introduced by Yoon Kim [21]. Our experimental results are comparable to state-of-the-art and also shed light on the importance of cleaning noisy text data before creating word embeddings, as it drastically affects the performance of the model.

The outline of the paper is as follows. Section 2 gives a brief overview of related work on the detection of semantically equivalent questions. All details of our proposed model as well as the necessary data preprocessing steps are presented in section 3. Our model is evaluated in section 4 and compared with other approaches. Lastly, concluding remarks and future research directions are discussed in section 5.

## 2 Related work

There have been numerous academic contributions dealing with paraphrase detection and the detection of semantically equivalent questions. The following section

gives an overview of recent research conducted with the Quora dataset.

Wang et al. [23] were the first to publish impressive results on the detection of duplicated questions with the Quora dataset. They proposed a bilateral multi-perspective matching (BiMPM) model, which compares sentence embeddings  $P$  and  $Q$  from multiple perspectives and in both directions ( $P \rightarrow Q$  and  $P \leftarrow Q$ ) [23]. The matching vectors are then aggregated with a Long Short-Term Memory Network (LSTM). The authors achieved an impressive 88.17% classification accuracy [23]. Tomar et al. [7] also took on a similar LSTM approach, but used character-level  $n$ -gram embeddings and pretrained their decomposable attention (DECATT) model with noisy data from other domains in order to improve overall performance [7]. They managed to obtain an accuracy of 88.40%.

Homma, et al. employed a Siamese Gated Recurrent Unit (GRU) and applied data augmentation to their model for better results [20]. They achieved 85% accuracy on the Quora dataset. Furthermore, they also experimented with the following distances metrics to calculate the similarity between sentence vectors: Euclidian distance, Cosine distance, and weighted Manhattan distance. According to their studies, the weighted Manhattan distance performed best [20].

A relatively unique approach for detecting duplicated questions from the Quora dataset was demonstrated in [9] by Deudon. The author uses a Variational Autoencoder to derive the intentions of the questions as normal distributions. He argues that decomposing the representation of a sentence in a mean vector and a diagonal covariance matrix accounts for uncertainty and ambiguity in natural languages and is therefore more appropriate than single vectors which capture relationships between multiple words and phrases. The Gaussian intents are fed into a Variational Siamese network and compared using a Wasserstein

distance metric. He recorded an accuracy of 88.86% on the Quora dataset [9].

Gong et al. [19] introduced a Densely Interactive Inference Network (DIIN) to achieve high-level text comprehension by hierarchically extracting semantic features from interaction space. They make use of a CNN to acquire important features, which are then aggregated and passed to an output layer. The output layer predicts a confidence level for each class. Their model achieved state-of-the-art performance on multiple, well known datasets. On the Quora dataset, they were able to achieve an 89% accuracy [19].

Shen et al. conducted a comparative study between Simple Word-Embedding based Models (SWEMs) and RNN/CNN based models [6]. Their aim was to understand when and why simple pooling strategies such as max-pooling and hierarchical pooling can be applied to word embeddings without any further processing. They found that such strategies are sufficient in some cases of natural language understanding. They experimented with 17 different datasets. Their best model on the Quora dataset achieved an accuracy of 83.03% [6].

### 3 Approach

In the following sections, we present the implementation details and background of our proposed model. We particularly focus on the data preprocessing pipeline and the relevant operations and layers of our network architecture.

#### 3.1 Problem formulation

Let  $P = (p_1, \dots, p_M)$  and  $Q = (q_1, \dots, q_N)$  be two questions consisting of  $M$  and  $N$  words, respectively [23]. Furthermore, let  $y \in \{0, 1\}$  be the binary label indicating whether questions  $P$  and  $Q$  are duplicated or not. Our model's goal is to predict the  $y$  label that best reflects the semantic relationship between  $P$  and  $Q$ . As Wang et al. point out [23], the problem can also be seen as the estimation of the conditional probability  $P(y | P, Q)$ .

#### 3.2 Text Preprocessing

The Quora dataset consists of 404,351 question pairs with positive and negative labels indicating whether or not two questions are duplicated. Roughly 37% of the question pairs are duplicates. The question pairs go through several preprocessing steps before being converted into numerical representations. First, every question is stripped of all punctuation and other non-alphanumeric characters. Then each question is tokenized with the Natural Language Toolkit (NLTK) tokenizer [13]. We decide not to remove stop words as they are considered to be semantically important in many cases. Furthermore, we do not discard rare words that occur only once or twice since they make up 43.21% of the text corpus. To train our neural network, all input questions need to have a fixed length. Therefore, we clip all sentences longer than 40 words. We decided on the length 40 after finding out that less than 1% of the questions contain more than 40 tokens.

In an attempt to fix numerous spelling errors which could negatively affect the performance of the model, we employ a pre-trained Word2Vec model [14, 15] and SymSpellpy. The Word2Vec model was trained on a Wikipedia dataset and therefore has a huge vocabulary. It is assumed that words which cannot be found by the Word2Vec model either contain spelling errors or are unknown terms. Such words are passed to SymSpellpy, a tool that corrects spelling errors and segments words accordingly if necessary. Words, which SymSpellpy cannot correct, are not discarded. We assume that such words are unknown names of people or simply outside the vocabulary (OOV). A similar approach can be found in [20].

#### 3.3 Word Embeddings

The preprocessed questions are transformed into numerical representations before being used as input to our neural network. We experiment with three different embedding approaches: Word2Vec [14, 15], FastText [11]

and Doc2Vec [16]. In addition, we introduce a fourth approach by combining Word2Vec and Term Frequency-Inverse Document Frequency (TF-IDF) [4, 8].

Word2Vec was introduced by Mikolov et al. in 2013 [14, 15]. The authors proposed two closely related neural network architectures for learning word vector representations: Continuous Bag-Of-Words (CBOW) and Skip-Gram [14, 15]. CBOW attempts to learn the vector representation of a target word by taking some surrounding context words into account. Contrary to CBOW, Skip-gram predicts some corresponding context words for a given target word.

Bojanowski et al. [11] introduced Fasttext, an extension to Word2Vec, after recognizing that rare words are often poorly estimated or even regarded as insignificant to some algorithms. Instead of only learning vector representations on whole words in a text corpus like Word2Vec does, Fasttext is capable of learning the  $n$ -grams within a word [2, 11]. This, in turn, allows Fasttext to be more sensitive to OOV and rare words. The model first splits words into  $n$ -grams and then feeds the  $n$ -grams to a CBOW or Skip-Gram model in order to learn the embeddings.

Doc2Vec, also considered to be an extension to Word2Vec, was introduced by Le and Mikolov in 2014 [16]. This model attempts to learn how to map both words and entire sentences to an embedding space while capturing important semantics [16].

In this work, Word2Vec, Fasttext, and Doc2Vec are trained using the Skip-Gram architecture. This is due to the fact that Skip-Gram works well on smaller data sets and is more susceptible to rare words [14, 15]. Each model is trained for 30 epochs with the same parameters and generates 300-dimensional word vectors.

For the fourth embedding approach, we first calculate the TF-IDF score of every word in the text corpus. Frequent words like “the, a,

and, or”, which are found across all sentences, are penalized; simultaneously rare words across the whole text corpus receive high scores [4, 8]. We iterate through the vocabulary  $V$ , which represents all unique words in the dataset, and multiply the vector representation  $w$  of each word with its corresponding  $tf\_idf$ -score as shown in Equation 1. The result is an embedding matrix  $E \in \mathbb{R}^{n \times d}$ , the rows of which are the TF-IDF-weighted word vectors across 300 dimensions.

$$E = \sum_{i=0}^V w_i \times tf\_idf_i \quad (1)$$

### 3.4 Network Architecture

The network used to classify question pairs as duplicates or non-duplicates is largely based on the CNN architecture for sentence classification introduced by Yoon Kim [21] in 2014.

We begin with an embedding layer, which maps all words in a pair of questions  $\{P, Q\}$  to their corresponding  $d$ -dimensional vector representations. This results in the matrices  $P \in \mathbb{R}^{n \times d}$  and  $Q \in \mathbb{R}^{n \times d}$ , where  $n$  is the maximum (null-padded) sentence length, and  $d$  is the dimensionality of the vector representations [1, 21, 22]. Note that the word vector representations could come from any of the four embedding approaches mentioned previously. The matrices  $P$  and  $Q$  are then fed to separate 1d convolutional layers to extract different levels of semantic features for each question. To avoid repetition we will only explain how features are extracted from the matrix  $P$ , since the exact same procedure is used for matrix  $Q$ .

Given the matrix  $P$ , whose rows consist of a sequence of word representations, we denote  $P[i : j]$  as a sub-matrix of  $P$  from row  $i$  to row  $j$  [16]. In order to extract features, we apply a filter  $w \in \mathbb{R}^{h \times k}$  to a window of  $h$  words in the matrix  $P$ , while the width  $k$  of the filter remains fixed [17, 21, 22].

A feature map  $c_i \in \mathbb{R}^{n-h+1}$  is generated by repeatedly applying the filter  $w$  on the submatrices of  $P$ :

$$c_i = \alpha(w \cdot P[i : i + h - 1] + b) \quad (2)$$

Here,  $b \in \mathbb{R}$  is a bias term and  $\alpha()$  is a non-linear activation function [21, 22]. In this work, we use the element-wise Rectified Linear Unit (ReLU) activation function. Generally, we apply five filter windows  $\{2, 3, 4, 6, 8\}$  200 times to matrices  $P$  and  $Q$ . This means each convolution layer outputs a matrix  $c \in \mathbb{R}^{(n-h+1) \times 200}$ . After generating the feature maps for all corresponding filter sizes, we then apply a global max-pooling operation (Eq. 3) as well as a global min-pooling operation (Eq. 4) over the matrix  $c$  [21]. Furthermore, we split the matrix  $c$  into  $i$  equal chunks, apply the global max-pooling operation to each chunk, and then concatenate their max-values (Eq. 5).

$$\begin{aligned} c_{max} &= \max(c) \\ \text{with } c_{max} &\in \mathbb{R}^{1 \times 200} \end{aligned} \quad (3)$$

$$\begin{aligned} c_{min} &= \min(c) \\ \text{with } c_{min} &\in \mathbb{R}^{1 \times 200} \end{aligned} \quad (4)$$

$$c_{max_i} = \max(z_1) \oplus \dots \oplus \max(z_i) \quad (5)$$

where  $z_1, \dots, z_i$  represent four equal submatrices  $\in \mathbb{R}^{(n-h+1) \times 50}$  of the matrix  $c$ . The symbol  $\oplus$  is used to denote the concatenation operator.

As Kim mentions in [21], max-pooling captures the most important feature for every feature map. In the same manner, the min-pooling operation captures the least important feature for every feature map. We include this feature as well as the  $c_{max_i}$  features, as they capture additional important information about each question.

In order to find out if matrices  $Q$  and  $P$  are semantically equivalent or not, we compare

their respective  $c_{max}$ ,  $c_{min}$  and  $c_{max_i}$  features, which are extracted from each feature map. More formally, we calculate their cosine similarities:

$$\cos(\theta) = \frac{f_1 \cdot f_2}{\|f_1\| \times \|f_2\|} + \lambda \quad (6)$$

where  $f_1$  and  $f_2$  represent the corresponding  $c_{max}$ ,  $c_{min}$  or  $c_{max_i}$  values for matrices  $Q$  and  $P$  and  $\lambda$  is an additional, trainable parameter with a range from -1 to 1. Recall that our model uses five filters with varying window sizes to obtain multiple features for each question. Therefore, we compare the features  $f_1$  and  $f_2$  obtained from each filter in both directions ( $f_1 \rightarrow f_2$  and  $f_2 \leftarrow f_1$ ) [23]. After matching all feature vectors for  $Q$  and  $P$ , we concatenate their cosine similarity values as follows:

$$x = x_1 \oplus x_2 \oplus \dots \oplus x_n, \quad (7)$$

The concatenated vector  $x$  is then passed through a softmax function, which produces the distribution over the classification classes  $\{0, 1\}$ , from which the strongest class is outputted.

### 3.5 Network inputs

Originally, we designed the CNN architecture to compare two questions in the form of  $d \times n$  matrices. After some experimentation, however, we decided to also explicitly include the unique words  $\{P_u, Q_u\}$  and keywords  $\{P_k, Q_k\}$  of each question pair as additional, embedded inputs to the network, resulting in three inputs for each question. Thus, we apply convolutional filters to the pairs  $\{P, Q\}$ ,  $\{P_u, Q_u\}$ , and  $\{P_k, Q_k\}$  and extract their  $max$ ,  $max_i$ , and  $min$  features. Thereafter, we compare the features of  $P$ ,  $P_k$ , and  $P_u$  with the features of  $Q$ ,  $Q_k$ , and  $Q_u$  using the cosine similarity function (Eq. 6). The resulting vectors are then concatenated and fed to the softmax function to generate the final prediction. This approach is depicted in Figure 1. It should be noted that activation and regularization layers are omitted to avoid further complexity.



In order to find the unique words for every question, we simply retrieve the words that do not intersect in a question pair. The keyword extraction from each question is based on the words' *tf-idf*-scores. The three words with the highest *tf-idf*-scores in a question are selected as keywords. Since frequent words are penalized by the TF-IDF-algorithm and therefore have a lower score, they are usually not included as key words.

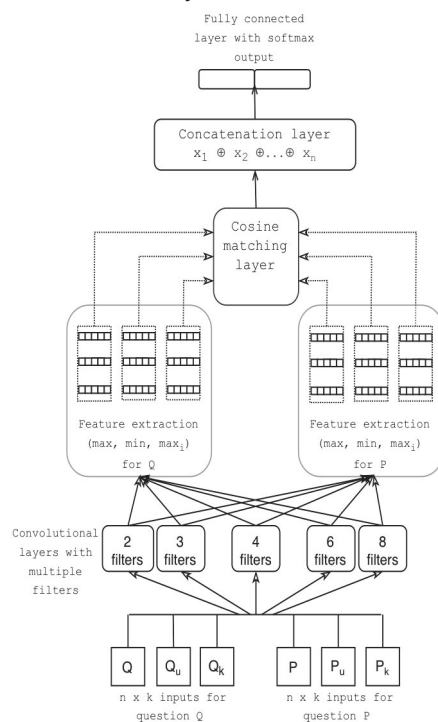


Figure 1: Proposed CNN architecture

### 3.6 Parameters and Training

For all experiments, we train with a mini-batch size of 64 and use the non-linear ReLU activation function on all convolutional layers. All the weights of these layers are randomly initialized with the *Xavier* initializer [18]. To avoid overfitting and stabilize training, we apply a dropout [10] rate of 0.1 as well as batch normalization with default parameters [12].

Our model is trained and evaluated on the exact same data partitions that Wang et al. published [23]. This allows us to compare our results fairly with theirs and other published results [6, 7, 9, 19, 20, 23], which also use the same data partitions. The dataset is split into three partitions: 10,000 question pairs for validation, 10,000 question pairs for testing, and the rest for training.

Our model's goal is to minimize the binary cross entropy of the predicted and true distributions. We employ the ADAM optimizer [5] with ( $lr = 0.001$ ,  $\beta_1 = 0.5$ ,  $\beta_2 = 0.999$ ) to update model parameters. The learning rate  $lr$  is decayed after every two epochs by a factor of 0.1. During training, we also update the pre-trained word embeddings for the pairs  $\{P, Q\}$  and  $\{P_u, Q_u\}$ . The weights of the keywords pair  $\{P_k, Q_k\}$ , are left static during training [21], since they do not improve model performance.

As we train our model with each of the four embedding approaches, we select the models that work best on the validation set and then evaluate them on the test set [23]. Accuracy is used as the evaluation metric for the dataset. The training is aborted if there is no update to the best accuracy on the validation set for the last 3 epochs.

## 4 Experimental Results

In Table 1 we compare our baselines with other models trained on the Quora dataset. Each model's validation accuracy (if published) and test accuracy are listed in the columns. The models are sorted by decreasing test accuracy. Note that all models with  $\diamond$  were reported by Wang et al. in their work [23]. Our baselines are denoted in bold in the last four rows of the table. Each baseline is labeled by the embedding approach used during training. Our experimental results surpass some models in terms of test accuracy and are also competitive with the state-of-the-art models. Among our models, we found Word2Vec to be the most consistent embedding approach.

**Table 1: Performance comparison of different models on the Quora dataset**

Model	Val Acc.	Test Acc.
DIIN [19]	89.44	89.06
VAR-Siamese [9]	89.05	88.86
DECATT [7]	88.89	88.40
BiMPM [23]	88.69	88.17
L.D.C. $\diamond$	-	85.55
Siamese-GRU [20]	-	85.0
Multi-Perspective-LSTM $\diamond$	-	83.21
SWEM-concat [6]	-	83.03
SWEM-aver [6]	-	82.68
Siamese-LSTM $\diamond$	-	82.58
SWEM-max [6]	-	82.20
Multi-Perspective-CNN $\diamond$	-	81.38
Siamese-CNN $\diamond$	-	79.60
<b>W2V</b>	<b>87.45</b>	<b>86.73</b>
<b>D2V</b>	<b>87.29</b>	<b>86.52</b>
<b>FAST</b>	<b>87.35</b>	<b>86.59</b>
<b>W2V+TFIDF</b>	<b>87.03</b>	<b>86.02</b>

In Table 2 we show several correct and incorrect label predictions on the validation set from our best model. The P column represents the predicted binary label, while the T column represents the true binary label from the validation set. The first two rows contain pairs of questions that our model classified correctly. The last three rows show questions that were incorrectly classified by our model.

During the training of our models we observed two interesting, behavioral patterns: A model sometimes tends to correctly predict most non-

duplicated pairs of questions, but at the same time has difficulties in predicting duplicates. The opposite case can also occur. In addition, we found that our models often predict question pairs correctly from a human point of view, but incorrectly according to the data set. Further investigation revealed that the Quora dataset has many question pairs that are labeled with a high degree of subjectivity. This evidently affects the performance of our model and its ability to generalize.

**Table 2: Model prediction examples**

Q1	Q2	P	T
How do I get funding for my web-based startup idea?	How do I get seed funding pre-product?	1	1
When did the first world war happen?	What were the causes of World War I?	0	0
How deep is the Titanic wreck?	How deep is Atlantic Ocean where Titanic sank?	0	1
What is the physical meaning of divergence, curl and gradient of a vector field?	Why curl represents rotation of a vector field?	0	1
What is definition of instrumentation?	What is biomedical instrumentation?	1	0

## 5 Conclusion

We presented a simple CNN approach built on top of word embeddings to detect duplicated questions from the Quora dataset. Correcting spelling errors in the data set with the pre-trained Word2Vec model improved our model performance by 3% on average. We employed four different embedding approaches, which

all led to similar results. Using global max-pooling, min-pooling and max-pooling on chunks to extract important features from each convolutional layer proved to be very effective. We also experimented with the average-pooling operation, but it performed poorly for our task. Overall, our competitive results show the potential of CNNs for the detection of semantically equivalent questions, a task that is typically solved by using RNNs.

With regard to future improvements of our approach, we believe using character-level embeddings [17] instead of word embeddings or even combining both methods could boost performance. Future work may also involve the study of other approaches for extracting features from convolutional layers, model ensemble techniques, and other similarity measures for comparing features.

## 6 References

- [1] A. Jacovi, O. S. Shalom, and Y. Goldberg. Understanding convolutional neural networks for text classification. *In Proc. EMNLP Blackbox NLP. ACL*. 2018.
- [2] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. 2017.
- [3] D. Bogdanova, C. dos Santos, L. Barbosa, and B. Zadrozny. Detecting semantically equivalent questions in online user forums. *Proceedings of the 19th Conference on Computational Natural Language Learning*, 1:123–131, 2015.
- [4] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson. Prentice Hall, *Third Edition draft*. 2018.
- [5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *In CoRR, abs/1412.6980*. 2014.
- [6] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 2018.
- [7] G. S. Tomar, T. Duque, O. Täckström, J. Uszkoreit, and D. Das. Neural paraphrase identification of questions with noisy pretraining. *In Proceedings of the 1st Workshop on Subword and Character Level Models in NLP*, page 142–147. 2017.
- [8] J. Ramos. Using tf-idf to determine word relevance in document queries. *In ICML*. 2003.
- [9] M. Deudon. Learning semantic similarity in a continuous space. *In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Advances in Neural Information Processing Systems 31, pages 993–1004. Curran Associates, Inc.* 2018.
- [10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958. January 2014.
- [11] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *In TACL 5:135–146*. 2017.
- [12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In ICML*. 2015.

- [13] S. Bird, E. Klein, and E. Loper. Natural language processing with Python: analyzing text with the natural language toolkit. *O'Reilly Media, Inc.*, 2009.
- [14] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J Dean. Distributed representations of words and phrases and their compositionality. *In Advances in Neural Information Processing Systems*. 2013a.
- [15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *In Proceedings of ICLR Workshop*. 2013.
- [16] V. Q. Le and T. Mikolov. Distributed Representations of Sentences and Documents. *In Proceedings of ICML*. 2014.
- [17] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *In NIPS*. 2015.
- [18] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256*. 2010.
- [19] Y. Gong, H. Luo, and J. Zhang. Natural language inference over interaction space. *In ICLR*. 2018.
- [20] Y. Homma, S. Sy, and C. Yeh. Detecting duplicate questions with deep learning. *In Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, pp. 1–8. 2017.
- [21] Y. Kim. Convolutional neural networks for sentence classification. *In Proceedings of the 2014 Conference on Empirical Methods for Natural Language Processing*. 2014.
- [22] Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*. 2015.
- [23] Z. Wang, W. Hamza, and R. Florian. Bilateral multi-perspective matching for natural language sentences. *In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI), pages 4144–4150*. 2017.