Detecting Semantically Equivalent Questions in Online User Forums

Dasha Bogdanova*, **Cícero dos Santos†, Luciano Barbosa† and Bianca Zadrozny†**
*ADAPT centre, School of Computing, Dublin City University, Dublin, Ireland
dbogdanova@computing.dcu.ie
†IBM Research, 138/146 Av. Pasteur, Rio de Janeiro, Brazil
{cicerons,lucianoa,biancaz}@br.ibm.com

Abstract

Two questions asking the same thing could be too different in terms of vocabulary and syntactic structure, which makes identifying their semantic equivalence challenging. This study aims to detect semantically equivalent questions in online user forums. We perform an extensive number of experiments using data from two different Stack Exchange forums. We compare standard machine learning methods such as Support Vector Machines (SVM) with a convolutional neural network (CNN). The proposed CNN generates distributed vector representations for pairs of questions and scores them using a similarity metric. We evaluate in-domain word embeddings versus the ones trained with Wikipedia, estimate the impact of the training set size, and evaluate some aspects of domain adaptation. Our experimental results show that the convolutional neural network with in-domain word embeddings achieves high performance even with limited training data.

1 Introduction

Question-answering (Q&A) community sites, such as Yahoo! Answers,[1] Quora[2] and Stack Exchange,[3] have gained a lot of attention in the recent years. Most Q&A community sites advise users to search the forum for an answer before posting a new question. However, this is not always an easy task because different users could formulate the same question in completely different ways. Some user forums, such as those of the Stack Exchange online community, have a duplication policy. Exact duplicates, such as copy-and-paste questions, and nearly exact duplicates are usually quickly detected, closed and removed from the forum. Nevertheless, some duplicate questions are kept. The main reason for that is that *there are many ways to ask the same question, and a user might not be able to find the answer if they are asking it a different way*.[4]

In this study we define two questions as semantically equivalent if they can be adequately answered by the exact same answer. Table 1 presents an example of a pair of such questions from Ask Ubuntu forum. Detecting semantically equivalent questions is a very difficult task due to two main factors: (1) the same question can be rephrased in many different ways; and (2) two questions could be asking different things but look for the same solution. Therefore, traditional similarity measures based on word overlap such as shingling and Jaccard coefficient (Broder, 1997) and its variations (Wu et al., 2011) are not able to capture many cases of semantic equivalence.

In this paper, we propose a convolutional neural network architecture to detect semantically equivalent questions. The proposed CNN first transforms words into word embeddings (Mikolov et al., 2013), using a large collection of unlabeled data, and then applies a convolutional network to build distributed vector representations for pairs of questions. Finally, it scores the questions using a similarity metric. Pairs of questions with similarity above a threshold, defined based on a held-out set, are considered duplicates. CNN is trained using positive and negative pairs of semantically equivalent questions. During training, CNN is *induced* to produce similar vector representations for questions that are semantically equivalent.

We perform an extensive number of experiments using data from two different Stack Exchange forums. We compare CNN performance with a traditional classification algorithm (Support

[1] https://answers.yahoo.com/
[2] http://www.quora.com
[3] http://stackexchange.com/
[4] http://stackoverflow.com/help/duplicates

Proceedings of the 19th Conference on Computational Language Learning, pages 123–131,
Beijing, China, July 30-31, 2015.

Title: I can't download anything and I can't watch videos	**Title:** How can I install Windows software or games?
Body: Two days ago I tried to download skype and it says an error occurred it says *end of central directory signature not found Either this file is not a zipfile, or it constitutes one disk of a multi-part archive. In the latter case the central directory and zipfile comment will be found on the last disk(s) of this archive. zipinfo: cannot find zipfile directory in one of /home/maria/Downloads/SkypeSetup-aoc-jd.exe* or */home/maria/Downloads/SkypeSetup-aoc-jd.exe.zip, and cannot find /home/maria/Downloads/SkypeSetup-aoc-jd.exe.ZIP* pe... this happens whenever I try to download anything like games and also i can't watch videoss it's looking for plug ins but it doesn't find them i hate this [sic!]	**Body:** Can *.exe* and *.msi* files (Windows software) be installed in Ubuntu? [sic!]
Link: `http://askubuntu.com/questions/364350`	`http://askubuntu.com/questions/988`
Possible Answer (Shortened version): .exe files are not binary-compatible with Ubuntu. There are, however, compatibility layers for Linux, such as Wine, that are capable of running .exe.	

Table 1: An example of semantically equivalent questions from Ask Ubuntu community.

Vector Machines (Cortes and Vapnik, 1995)) and a duplicate detection approach (shingling (Broder, 1997)). The results show CNN outperforms the baselines by a large margin.

We also investigate the impact of different word embeddings by analyzing the performance of the network with: (1) word embeddings pre-trained on in-domain data and all of the English Wikipedia; (2) word vectors of different dimensionalities; (3) training sets of different sizes; and (4) out-of-domain training data and in-domain word embeddings. The numbers show that: (1) word embeddings pre-trained on domain-specific data achieve very high performance; (2) bigger word embeddings obtain higher accuracy; (3) in-domain word embeddings provide better performance independent of the training set size; and (4) in-domain word embeddings achieve relatively high accuracy even using out-of-domain training data.

2 Task

This work focuses on the task of predicting semantically equivalent questions in online user forums. Following the duplication policy of the Stack Exchange online community,[5] we define semantically equivalent questions as follows:

Definition 1. *Two questions are semantically equivalent if they can be adequately answered by the exact same answer.*

Since our definition of semantically equivalent questions corresponds to the rules of the Stack Exchange duplication policy, we assume that all questions of this community that were marked as duplicates are semantically equivalent. An example of such questions is given in Table 1. These questions vary significantly in vocabulary, style, length and content quality. However, both questions require the exact same answer.

The exact task that we approach in this study consists in, given two problem definitions, predicting if they are semantically equivalent. By *problem definition* we mean the concatenation of the title and the body of a question. Throughout this paper we use the term *question* as a synonym of problem definition.

3 Related Work

The development of CNN architectures for tasks that involve sentence-level and document-level processing is currently an area of intensive research in natural language processing and information retrieval, with many recent encouraging results.

Kim (2014) proposes a simple CNN for sentence classification built on top of `word2vec` (2013). A multichannel variant which combines static word vectors from `word2vec` and word vectors which are fine-tuned via backpropagation is also proposed. Experiments with different variants are performed on a number of benchmarks for sentence classification, showing that the simple CNN performs remarkably well, with state-of-the-art results in many of the benchmarks, highlighting the importance of using unsupervised pre-training of word vectors for this task.

Hu et al.(2014) propose a CNN architecture for hierarchical sentence modeling and, based on that, two architectures for sentence matching.

[5] http://blog.stackoverflow.com/2010/11/dr-strangedupe-or-how-i-learned-to-stop-worrying-and-love-duplication/; http://meta.stackexchange.com/questions/32311/do-not-delete-good-duplicates

They train the latter networks using a ranking-based loss function on three sentence matching tasks of different nature: sentence completion, response matching and paraphrase identification. The proposed architectures outperform previous work for sentence completion and response matching, while the results are slightly worse than the state-of-the-art in paraphrase identification.

Yih et al.(2014) focus on the task of answering single-relation factual questions, using a novel semantic similarity model based on a CNN architecture. Using this architecture, they train two models: one for linking a question to an entity in the DB and the other mapping a relation pattern to a relation in the DB. Both models are then combined for inferring the entity that is the answer. This approach leads to a higher precision on Q&A data from the WikiAnswers corpus than the existing rules-based approach for this task.

Dos Santos & Gatti (2014) developed a CNN architecture for sentiment analysis of short texts that jointly uses character-level, word-level and sentence-level information, achieving state-of-the-art results on well known sentiment analysis benchmarks.

For the specific task of semantically equivalent questions detection that we address in this paper, we are not aware of any previous work using CNNs. Muthmann and Petrova (2014) approach the task of identifying topical near-duplicate relations between questions from social media as a classification task. They use a simple lexico-syntactical feature set and different classifiers are evaluated, with logistic regression reported as the best performing one. However, it is not possible to directly compare our results to theirs because their experimental methodology is not clearly described in the paper.

There are several tasks related to identifying semantically equivalent questions. These tasks include near-duplicate detection, paraphrase identification and textual semantic similarity estimation. In what follows, we outline the differences between these tasks and the one addressed in this work.

Duplicate and Near-Duplicate Detection aims to detect exact copies or almost exact copies of the same document in corpora. Duplicate detection is an important component of systems for Web crawling and Web search, where it is important to identify redundant data in large corpora. Common techniques to detect duplicate documents include shingling (Broder, 1997; Alonso et al., 2013) and fingerprinting (Manku et al., 2007). State-of-the-art work also focuses on the efficiency issues of the task (Wu et al., 2011). It is worth noting that even though all duplicate and near-duplicate questions are also semantically equivalent, the reverse is not true. Semantically equivalent questions could have small or no word overlap (see Table 1 for an example), and thus, are not duplicates.

Paraphrase Identification is the task of examining two sentences and determining whether they have the same meaning (Socher et al., 2011). If two questions are paraphrases, they are also semantically equivalent. However, many semantically equivalent questions are not paraphrases. The questions shown in Table 1 significantly differ in the details they provide, and thus, could not be considered as having the same meaning. State-of-the-art approaches to paraphrase identification include using Machine Translation evaluation metrics (Madnani et al., 2012) and Deep Learning techniques (Socher et al., 2011).

Textual Semantic Similarity is the task of measuring the degree of semantic similarity between two texts, usually on a graded scale from 0 to 5, with 5 being the most similar (Agirre et al., 2013) and meaning that the texts are paraphrases. All semantically equivalent questions are somewhat semantically similar, but semantic equivalence of questions defined here does not correspond to the highest value of the textual semantic similarity for the same reasons these questions are not always paraphrases.

4 Neural Network Architecture

In this section, we present our neural-network strategy for detecting semantically equivalent questions.

4.1 Feed Forward Processing

As detailed in Figure 1, the input for the network is tokenized text strings of the two questions. In the first step, the CNN transforms words into real-valued feature vectors, also known as word embeddings or word representations. Next, a convolutional layer is used to construct two distributed vector representations r_{q_1} and r_{q_2}, one for each input question. Finally, the CNN computes a similarity score between r_{q_1} and r_{q_2}. Pairs of questions with similarity above a threshold, defined based on

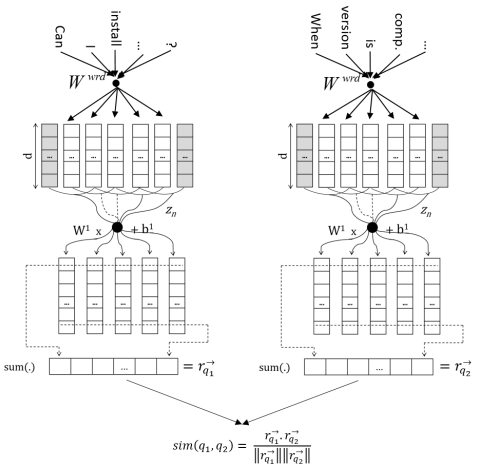

Figure 1: Convolutional neural network for semantically equivalent questions detection.

a heldout set, are considered duplicates.

4.2 Word Representations

The first layer of the network transforms words into representations that capture syntactic and semantic information about the words. Given a question consisting of N words $q = \{w_1, w_2, ..., w_N\}$, every word w_n is converted into a real-valued vector r^{w_n}. Therefore, for each question, the input to the next NN layer is a sequence of real-valued vectors $q^{emb} = \{r^{w_1}, r^{w_2}, ..., r^{w_N}\}$

Word representations are encoded by column vectors in an embedding matrix $W^0 \in \mathbb{R}^{d \times |V|}$, where V is a fixed-sized vocabulary. Each column $W^0_i \in \mathbb{R}^d$ corresponds to the word embedding of the i-th word in the vocabulary. We transform a word w into its word embedding r^w by using the matrix-vector product:

$$r^w = W^0 v^w \quad (1)$$

where v^w is a vector of size $|V|$ which has value 1 at index w and zero in all other positions. The matrix W^0 is a parameter to be learned, and the size of the word embedding d is a hyper-parameter to be chosen by the user.

4.3 Question Representation and Scoring

The next step in the CNN consists in creating distributed vectors from word embeddings representations of the input questions. To perfom this task, the CNN must deal with two main challenges: different questions can have different sizes; and important information can appear at any position in the question. The convolutional approach (Waibel et al., 1989) is a natural choice to tackle these challenges. In recent work, convolutional approaches have been used to solve similar problems when creating representations for text segments of different sizes (dos Santos and Gatti, 2014) and character-level representations of words of different sizes (dos Santos and Zadrozny, 2014). Here, we use a convolutional layer to compute the question-wide distributed vector representations r_{q_1} and r_{q_2}. For each question, the convolutional layer first produces local features around each word in the question. Then, it combines these local features using a sum operation to create a fixed-sized feature vector (representation) for the question.

Given a question q_1, the convolutional layer applies a matrix-vector operation to each window of size k of successive windows in $q_1^{emb} = \{r^{w_1}, r^{w_2}, ..., r^{w_N}\}$. Let us define the vector $z_n \in \mathbb{R}^{dk}$ as the concatenation of a sequence of k word embeddings, centralized in the n-th word:[6]

$$z_n = \left(r^{w_{n-(k-1)/2}}, ..., r^{w_{n+(k-1)/2}}\right)^T$$

The convolutional layer computes the j-th element of the vector $r_{q_1} \in \mathbb{R}^{cl_u}$ as follows:

$$[r_{q_1}]_j = f\left(\sum_{1<n<N} \left[f\left(W^1 z_n + b^1\right)\right]_j\right) \quad (2)$$

where $W^1 \in \mathbb{R}^{cl_u \times dk}$ is the weight matrix of the convolutional layer and f is the hyperbolic tangent function. The same matrix is used to extract local features around each word window of the given question. The *global* fixed-sized feature vector for the question is obtained by using the sum over all word windows.[7] Matrix W^1 and vector b^1 are parameters to be learned. The number of convolutional units cl_u (which corresponds to the size of the question representation), and the size of the word context window k are hyper-parameters to be chosen by the user.

Given r_{q_1} and r_{q_2}, the representations for the input pair of questions (q_1, q_2), the last layer of the CNN computes a similarity score between q_1 and q_2. In our experiments we use the cosine similarity $s(q_1, q_2) = \frac{\vec{r_{q_1}} . \vec{r_{q_2}}}{\|\vec{r_{q_1}}\| \|\vec{r_{q_2}}\|}$.

[6] Words with indices outside of the sentence boundaries use a common *padding embedding*.

[7] Using max operation instead of sum produces very similar results.

4.4 Training Procedure

Our network is trained by minimizing the mean-squared error over the training set D. Given a question pair (q_1, q_2), the network with parameter set θ computes a similarity score $s_\theta(q_1, q_2)$. Let $y_{(q_1,q_2)}$ be the correct *label* of the pair, where its possible values are 1 (equivalent questions) or 0 (not equivalent questions). We use stochastic gradient descent (SGD) to minimize the mean-squared error with respect to θ:

$$\theta \mapsto \sum_{(x,y)\in D} \frac{1}{2}\left(y - s_\theta(x)\right)^2 \tag{3}$$

where $x = (q_1, q_2)$ corresponds to a question pair in the training set D and y represents its respective label $y_{(q_1,q_2)}$.

We use the backpropagation algorithm to compute gradients of the network. In our experiments, we implement the CNN architecture and the backpropagation algorithm using Theano (Bergstra et al., 2010).

5 Experimental Setup

5.1 Data

In our experiments we use data from the Ask Ubuntu Community Questions and Answers (Q&A) site.[8] Ask Ubuntu is a community for Ubuntu users and developers, and it is part of the Stack Exchange[9] Q&A communities. The users of these communities can ask and answer questions, and vote up and down both questions and answers. Users with high reputation become moderators and can label a new question as a duplicate to an existing question.[10] Usually it takes five votes from different moderators to close a question or to mark it as a duplicate.

We use the Ask Ubuntu data dump provided in May 2014. We extract all question pairs linked as duplicates. The data dump we use contains 15277 such pairs. For our experiments, we randomly select a training set of 24K pairs, a test set of 6K and a validation set of 1K, making sure there are no overlaps between the sets. Half of each set contains pairs of semantically equivalent questions (positive pairs) and half are pairs of questions that are not semantically equivalent. The latter pairs are randomly generated from the corpus. The data was tokenized with NLTK (Bird et al., 2009), and all links were replaced by a unique string.

For the experiments on a different domain (see Section 6.4) we use the Meta Stack Exchange[11] data dump provided in September 2014. Meta Stack Exchange (Meta) is used to discuss the Stack Exchange community itself. People ask questions about the rules, features and possible bugs. The data dump we use contains 67746 questions, where 19456 are marked as duplicates. For the experiments on this data set, we select random balanced disjoint sets of 20K pairs for training, 1K for validation and 4K for testing. We prepare the data in exactly the same manner as the Ask Ubuntu data.

5.2 Baselines

We explore three main baselines: a method based on the Jaccard coefficient which was reported to provide high accuracy for the task of duplicate detection (Wu et al., 2011), a Support Vector Machines (SVM) classifier (Cortes and Vapnik, 1995) and the combination of the two.

For the first baseline, documents are first represented as sets of shingles of lengths from one to four, and then the Jaccard coefficient for a pair of documents is calculated as follows:

$$J(S(d_1), S(d_2)) = \frac{S(d_1) \cap S(d_2)}{S(d_1) \cup S(d_2)},$$

where $S(d_i)$ is the set of shingles generated from the ith document. High values of the Jaccard coefficient denote high similarity between the documents. If the value exceeds a threshold T, the documents are considered semantically equivalent. In this case, the training data is used to select the optimal threshold T.

For the SVM baseline, we represent documents with n-grams of length up to four. For each pair of questions and each n-gram we generate three features: (1) if the n-gram is present in the first question; (2) if the n-gram is present in the second question; (3) the overall normalized count of the n-gram in the two questions. We use the RBF kernel and perform grid search to optimize the values of C and γ parameters. We use a frequency threshold[12] to reduce the number of features. The

[8] http://askubuntu.com/

[9] http://stackexchange.com

[10] More information about Stack Exchange communities could be found here: http://stackexchange.com/tour

[11] meta.stackexchange.com

[12] Several values (2, 5, 35 and 100) were tried with cross-validation, the threshold with value 5 was selected

implementation provided by LibSVM (Chang and Lin, 2011) is used.

In order to combine the two baselines, for a pair of questions we calculate the values of the Jaccard coefficient with shingles size up to four, and then add these values as additional features used by the SVM classifier.

5.3 Word Embeddings

The word embeddings used in our experiments are initialized by means of unsupervised pre-training. We perform pre-training using the skip-gram NN architecture (Mikolov et al., 2013) available in the word2vec[13] tool. Two different corpora are used to train word embeddings for most of the experiments: the English Wikipedia and the Ask Ubuntu community data. The experiments presented in Section 6.4 also use word embeddings trained on the Meta Stack Exchange community data.

In the experiments with the English Wikipedia word embeddings, we use the embeddings previously produced by dos Santos & Gatti (2014). They have used the December 2013 snapshot of the English Wikipedia corpus to obtain word embeddings with word2vec.

In the experiments with Ask Ubuntu and Meta Stack Exchange word embeddings, we use the Stack Exchange data dump provided in May 2014 to train word2vec. Three main steps are used to process all questions and answers from these Stack Exchange dumps: (1) tokenization of the text using the NLTK tokenizer; (2) image removal, URL replacement and prefixing/removal of the code if necessary (see Section 6.1 for more information); (3) lowercasing of all tokens. The resulting corpora contains about 121 million and 19 million tokens for Ask Ubuntu and Meta Stack Exchange, respectively.

6 Experimental Results

6.1 Comparison with Baselines

Ask Ubuntu community gives users an opportunity to format parts of their posts as code by using *code* tags (an example is in italic in Table 1). It includes not only programming code, but commands, paths to directories, names of packages, error messages and links. Around 30% of all posts in the data dump contain *code* tags. Since the rules for code formatting are not well defined, it was not clear if a learning algorithm would benefit from including it or not. Therefore, for each algorithm we tested three different approaches to handling code: keeping it as text; removing it; and prefixing it with a special tag. The latter is done in order to distinguish between the same term used within text or within code or a command (e.g., a *for* as a preposition and a *for* in a for loop). When creating the word embeddings, the same approach to the code as for the training data was followed.

[13] http://code.google.com/p/word2vec/

System	Valid. Acc.	Test Acc.
SVM + shingles	85.5	82.4
CNN + Askubuntu	93.4	92.9

Table 3: CNN and SVM accuracy on the validation and the test set using the full training set.

The 1K example validation set is used to tune the hyper-parameters of the algorithms. In order to speed up computations, we perform our initial experiments using a 4K examples balanced subset of the training set. The best validation accuracies are reported in Table 2.

We test the shingling-based approach with different shingle sizes. As Table 2 indicates, the accuracy decreases with the increase of the shingle size. The fact that much better accuracy is achieved when comparing questions based on simple word overlap (shingle size 1), suggests that semantically equivalent questions are not duplicates but rather have topical similarity. The SVM baseline performs well only when combined with the shingling approach by using the values of the Jaccard coefficient for shingle size up to four as additional features. A possible reason for this is that n-gram representations do not capture enough information about semantic equivalence. The CNN with word embeddings outperforms the baselines by a significant margin.

The results presented in Table 2 indicate that the algorithms do not benefit from including the code. This is probably because the code tags are not always used appropriately and some code examples include long error messages, which make the user generated data even more noisy. Therefore, in the following experiments the code is removed.

The validation accuracy and the test accuracy using the full 24K training set is presented in Table 3. The SVM with four additional shingling features is found best among the baselines (see Table 2) and is used as a baseline in this experiment. Again, the CNN with word embeddings outperforms the best baseline by a significant margin.

Algorithm	Features	Code	Best validation acc.	Optimal hyper-parameters
SVM-RBF	binary + freq.	kept	66.2	C=8.0, γ ≈3.05e-05
SVM-RBF	binary + freq.	removed	66.53	C=2.0, γ ≈1.2e-04
SVM-RBF	binary + freq.	prefixed	66.53	C=8.0, γ ≈3.05e-05
Shingling (size 1)	-	kept	72.35	-
Shingling (size 1)	-	removed	72.65	-
Shingling (size 1)	-	prefixed	70.94	-
Shingling (size 2)	-	kept	69.24	-
Shingling (size 2)	-	removed	66.83	-
Shingling (size 2)	-	prefixed	67.74	-
Shingling (size 3)	-	kept	65.23	-
Shingling (size 3)	-	removed	62.93	-
Shingling (size 3)	-	prefixed	64.43	-
SVM-RBF	binary + freq. + shingles	kept	74.0	C=32.0, γ ≈3.05e-05
SVM-RBF	binary + freq. + shingles	removed	**77.4**	C=32.0, γ ≈3.05e-05
SVM-RBF	binary + freq. + shingles	prefixed	73.6	C=32.0, γ ≈3.05e-05
CNN	Askubuntu word vectors	kept	91.3	
CNN	Askubuntu word vectors	removed	**92.4**	d=200, k=3, cl_u=300, λ=0.005
CNN	Askubuntu word vectors	prefixed	91.4	

Table 2: Validation Accuracy and best parameters for the baselines and the Convolutional Neural Network.

6.2 Impact of Domain-Specific Word Embeddings

We perform two experiments to evaluate the impact of the word embeddings on the CNN accuracy. In the first experiment, we gradually increase the dimensionality of word embeddings from 50 to 400. The results are presented in Figure 2. The vertical axis corresponds to validation accuracy and the horizontal axis represents the training time in epochs. As has been shown in (Mikolov et al., 2013), word embeddings of higher dimensionality trained on a large enough data set capture semantic information better than those of smaller dimensionality. The experimental results presented in Figure 2 correspond to these findings: we can see improvements in the neural network performance when increasing the word embeddings dimensionality from 50 to 100 and from 100 to 200. However, the Ask Ubuntu data set containing approximately 121M tokens is not big enough for an improvement when increasing the dimensionality from 200 to 400.

In the second experiment, we evaluate the impact of in-domain word embeddings on the network's performance. We obtain word embeddings trained on two different corpora: Ask Ubuntu community data and English Wikipedia (see Section 5.3). Both word embeddings have 200 dimensions. The results presented in Table 4 show that training on in-domain data is more beneficial for the network, even though the corpus used to create word embeddings is much smaller.

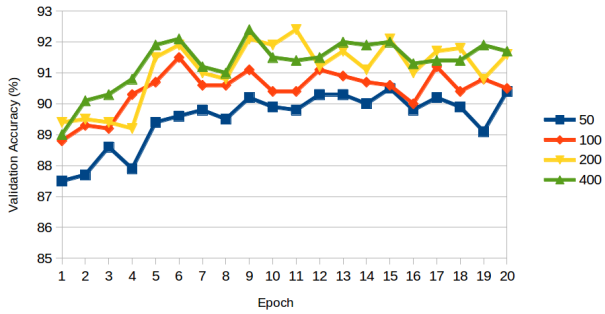

Figure 2: CNN accuracy depending on the size of word embeddings

Word Embeddings	Num.tokens	Valid.Acc.
Wikipedia	≈1.6B	85.5
AskUbuntu	≈121M	92.4

Table 4: Validation Accuracy of the CNN with word embeddings pre-trained on different corpora.

6.3 Impact of Training Set Size

In order to measure the impact of the training set size, we perform experiments using subsets of the training data, starting from 100 question pairs and gradually increasing the size to the full 24K training set.[14] Figure 3 compares the learning curves for the SVM baseline (with parameters and features described in Section 6.1) and for the CNN with word embeddings trained on Ask Ubuntu and English Wikipedia. The vertical axis corresponds to the validation accuracy, and the horizontal axis represents the training set size. As Figure 3 indicates, increasing the size of the training set pro-

[14] We use sets of 100, 1000, 4000, 12000 and 24000 question pairs.

vides improvements. Nonetheless, the difference in accuracy when training with the full 24K training set and 4K subset is about 9% for SVM and only about 1% for the CNN. This difference is small for both word embeddings pre-trained on Ask Ubuntu and Wikipedia but, the in-domain word embeddings provide better performance independently of the training set size.

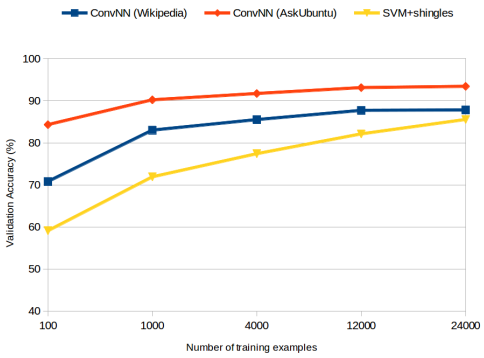

Figure 3: Validation accuracy for the baseline and the CNN depending on the size of training set.

6.4 Domain Adaptation

Muthmann and Petrova (2014) report that the Meta Stack Exchange Community[15] is one of the hardest for finding semantically equivalent questions.

We perform the same experiments described in previous sections using the Meta data set. In Table 5, we can see that the CNN accuracy on Meta test data (92.68%) is similar to the one for Ask Ubuntu community on test data (92.4%) (see Table 3).

Also, in Table 5, we show results of a domain adaptation experiment in which we do not use training data from the Meta forum. In this case, the CNN is trained using Ask Ubuntu data only. The numbers show that even in this case using in-domain word embeddings helps to achieve relatively high accuracy: 83.35% on the test set.

7 Error Analysis

As we have expected, the CNN with in-domain word vectors outperforms the vocabulary-based baselines in identifying semantically equivalent questions that are too different in terms of vocabulary. The CNN is also better at distinguishing questions with similar vocabulary but different meanings. For example, the question pair, (q_1) *How can I install Ubuntu without removing Windows?* and (q_2) *How do I upgrade from x86 to x64 without losing settings?*[16] is erroneously classified as a positive pair by the SVM, while the CNN classifies it correctly as a negative pair.

[15] http://meta.stackexchange.com/

Train.Data	Size	Word Vect.	Val.Acc.	Test.Acc.
META	4K	META	91.1	89.97
META	4K	Wikipedia	86.9	86.27
META	20K	META	**92.8**	**92.68**
META	20K	Wikipedia	90.6	90.52
AskUbuntu	24K	META	83.9	83.35
AskUbuntu	24K	AskUbuntu	76.8	80.0

Table 5: Convolutional Neural Network Accuracy tested on Meta Stack Exchange community data.

There are some cases where both CNN and SVM fail to identify semantic equivalence. Some of these cases include questions where essential information is presented as an image, e.g., a screenshot, which was removed during preprocessing.[17]

8 Conclusions and Future Work

In this paper, we propose a method for identifying semantically equivalent questions based on a convolutional neural network. We experimentally show that the proposed CNN achieves very high accuracy especially when the word embeddings are pre-trained on in-domain data. The performance of an SVM-based approach to this task was shown to depend highly on the size of the training data. In contrast, the CNN with in-domain word embeddings provides very high performance even with limited training data. Furthermore, experiments on a different domain have demonstrated that the neural network achieves high accuracy independently of the domain.

The next step in our research is building a system for retrieval of semantically equivalent questions. In particular, given a corpus and a question, the task is to find all questions that are semantically equivalent to the given one in the corpus. We believe that a CNN architecture similar to the one proposed in this paper might be a good fit to tackle this problem.

Acknowledgments

The work of Dasha Bogdanova is supported by Science Foundation Ireland through the CNGL

[16] Due to space constraints we only report the titles of the questions

[17] For instance, http://askubuntu.com/questions/450843

Programme (Grant 12/CE/I2267) in the ADAPT Centre (www.adaptcentre.ie) at Dublin City University. Her contributions were made during an internship at IBM Research.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1*, pages 32–43, Atlanta, Georgia, USA, June.

Omar Alonso, Dennis Fetterly, and Mark Manasse. 2013. Duplicate news story detection revisited. In Rafael E. Banchs, Fabrizio Silvestri, Tie-Yan Liu, Min Zhang, Sheng Gao, and Jun Lang, editors, *Information Retrieval Technology*, volume 8281 of *Lecture Notes in Computer Science*, pages 203–214. Springer Berlin Heidelberg.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.

A. Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES'97, Washington, DC, USA. IEEE Computer Society.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning, Volume 20(3)*, pages 273–297.

Cícero Nogueira dos Santos and Maíra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, Dublin, Ireland.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML), JMLR: W&CP volume 32*, Beijing, China.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 2042–2050, Montreal, Quebec, Canada.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods for Natural Language Processing*, pages 1746–1751, Doha, Qatar.

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

Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 141–150, New York, NY, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations*, ICLR 2013, Scottsdale, AZ, USA.

Klemens Muthmann and Alina Petrova. 2014. An Automatic Approach for Identifying Topical Near-Duplicate Relations between Questions from Social Media Q/A . In *Proceedings of Web-scale Classification: Classifying Big Data from the Web WSCBD 2014*.

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 John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 801–809.

Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. 1989. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Volume 37(3), pages 328–339.

Yan Wu, Qi Zhang, and Xuanjing Huang. 2011. Efficient near-duplicate detection for q&a forum. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1001–1009, Chiang Mai, Thailand.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648.