Effective Representation Aggregation for Supervised Detection of Equivalent Questions

Anonymous ACL submission

Abstract

Question and answer (Q&A) forums have recently become highly useful and popular in communities due to facilitating convenient acquisition and donation of expertise and insight between individuals all around the world. Automatically detecting whether two questions are equivalent has become an essential task in huge Q&A systems to avoid redundancy. In this paper, we address the task of learning to detect equivalent questions from data. In particular, we focus on how to best aggregate representations of a pair of questions to adequately detect their equivalence. While concatenation is a traditional and common method, we demonstrate that alternative approaches are more effective. In particular, we propose two methods for feature aggregation that are based on the outer product of question representations. We demonstrate the advantage of the proposed approaches on real world question equivalence datasets and show that they consistently improve the performance of the equivalence detection task compared to the state of the art methods.

1 Introduction

Question and answer (Q&A) community sites such as Reddit, Quora, JustDoc, StackOverflow, and StackExchange have become highly useful in today's digitally connected but busy world. They provide individuals with an opportunity to ask potential questions from peers or domain experts all around the world, provide answers to peers' questions, or read the already completed discussions. In such forums, on the one hand, knowledge seekers receive the answer to their questions from volunteer experts who hopefully enjoy such a donation and on the other hand, experts build reputation in their community by providing high quality answers that receive great attention (e.g. vote ups) by their readers. In general, such frameworks build a win-win situation for their users.

The problem of detecting redundant questions is central in QA systems both for providers to improve the quality of their service and for users (comprising of asking and answering parties) to be more continent participating in the communications. While several meanings for the redundancy of questions could be considered, in most QA systems, two questions are considered duplicate when all answers to one of them are suitable answers to the other[1]. This is more general than the two questions having the same meaning. In the contrary, it is possible that you ask two questions which are actually have different meanings, but the answer to which are effectively the same. (We will discuss this in Section **??**. todo) Ideally, there is a single representative question for a group of question wordings that all mean to ask the same thing, instead of having a group of scattered pages here and there. The lack of redundancy makes knowledge-sharing more efficient in many ways: for example, knowledge seekers can access all the answers to a question in a single page, and answer providers do not need to repeat the service multiple times. Also, they can reach a larger audience than if that questions were divided amongst several pages. How best to detect the redundancy and come up with such a canonical representative efficiently is a research question.

While humans can help by manually labeling redundant questions that they encounter, a high quality *automatic* detection system is expected to

[1]See for example the policy for Quora QA forum: https://www.quora.com/Whats-Quoras-policy-on-merging-questions.

be more cost effective, under control, consistent and reliable. Once detected that a question is duplicate of an already asked question, it can be prevented from being posted. Also duplicate pairs of questions can be combined together in a maintenance task after being asked and answered. How best to *automatically* diagnose the equivalence of two questions is an interesting research question in natural language processing (NLP).

This paper addresses the problem of automatic detection of question equivalence. The goal here is to learn a *supervised* model from data to classify if a pair of questions are redundant or not. A standard architecture is considered here where the input to the system is the representation of the words (tokens) of each question.

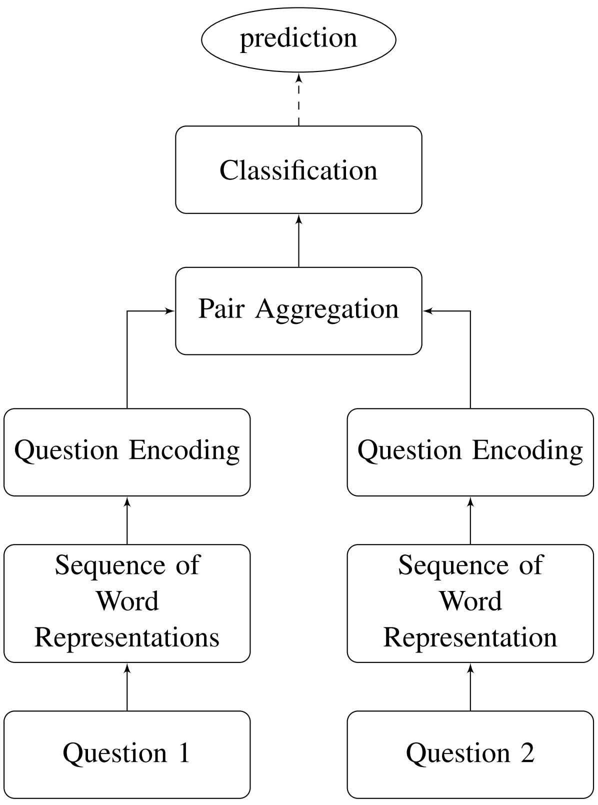


Figure 1: A Standard architecture for the problem

Each question is formed from a variable number of words. The encoding of each question is then performed using any appropriate aggregation method from the representation of its constituent words. Next, the representations of the pair of questions is somehow aggregated to form the representation of the main entity to be classified. Once the joint representation of the question pair is constructed, any desirable binary classification model is formed on top of it to learn a model for question duplicate detection from data. The diagram of Figure 1 illustrates the overall approach in this study.

An important phase in question deduplication is the pair aggregation phase, whose effectiveness is orthogonal to the question encoding stage. Coming up with a joint representation that sufficiently captures the interrelation between the pair of components is the key here. While different models for question deduplication has been proposed, few works focus on systematically studying the feature aggregation phase.

In this paper, we investigate the effect of the representation aggregation method in the deduplication process. We abstract away the question encoding method and the binary classification (or ranking) method used. We propose an aggregation operation based on the vectorized matrix multiplication of the question representations. Notably, this operation does not require the two question to have similar representation length. By applying it on real data, we show that this aggregation method consistently improves the performance of classification. Next we take a step further and propose a method based on concatenation followed by a vectorized outer product of the original representations. Further improvements are made by the second proposed method. The key contribution of this paper is to show that for the task of question deduplication, combining the question features by an outer product and then looking at each element of the matrix as a feature is much more effective that standard methods used in literature, by capturing interaction between all components. We also show that one could further improve the quality of aggregation, by not only including interactions between features of one question with the other, but also interactions between its own features.

The structure of the sequel is as follows. First, in Section 2, we formally set up the problem. In Section 3, we discuss existing works on this problem in the literature. In Section 5, we describe the proposed method. Finally, in Section 6, we experimentally evaluate the proposed method on real data. : To do: correct the structure based on exisiting sections

2 Problem setup

The overall question deduplication problem is formally defined as below. We are given a training set

S, $S = \big[\big((q_{11}, q_{12}), y_1\big), \quad \ldots, \quad \big((q_{t1}, q_{t2}), y_t\big)\big]$ consisting of t training examples, in which the i^{th} training example comprises two questions. Here q_{i1} is the first question and q_{i2} is the second. The output supervised signal for the i^{th} example is a binary class denoted with y_i, where $y_i \in \{0, 1\}$. Here label 1 denotes the class of a question pair whose contents are semantically equivalent and label 0 denoted the class when they are not semantically equivalent.

The goal is to learn a binary classifier from data to generalize, so that it can predict with high accuracy for an unseen pair of questions $Q_n = (q_{n1}, q_{n2})$ if it does contain an equivalent pair or not. The initial encoding for the first and second questions are denoted by $\phi(q_{i1})$ and $\psi(q_{i2})$ respectively.

Given the vector of representation for the first question $\phi(q_1) \in \mathbb{R}^m$ and of the second question $\psi(q_2) \in \mathbb{R}^n$ our focus is formally on how to get to the combined feature representation $\chi(q_1, q_2)$ to maximize classification accuracy, where $\chi(q_1, q_2)$ is a function of $\phi(q_1)$ and $\psi(q_2)$.

3 Related work

3.1 Related problems

A variety of NLP tasks aim at predicting relatedness between two pieces of input natural language segment. Looking at the data, such problems can be categorized according to the characteristic of the input, the characteristics of the output and how input and output are related to each other.

Input side – Type and size of language segments Equivalence of language units with different types and sizes is studied from words (Bollegala et al., 2007) and phrases (Wang et al., 2008; Saad and Kamarudin, 2013) to sentences (Socher et al., 2011; Hu et al., 2014; Kiros et al., 2015), questions (Wang et al., 2017), threads (Wu et al., 2011), and documents (Hofmann, 1999). The type of the input can impose different constraints on modeling. For example, if the input is a pair of words, the number of features is typically fixed. But for a pair of sentence or question, this is not the case. Also depending on the type of input entities meaningful representation could be different. For example, to decide if two complete questioning threads are effectively equivalent, not only the questions, but also signals coming from *subject tag* and the selected *best answers* are often very helpful (Wu et al., 2011).

Output side – The presence of human assigned label or any supervised signal to learn from The data could be supervised or unsupervised. In old fashion NLP tasks, supervised tasks were those that needed human labeling of data. Since labeling by human is an expensive task, unsupervised models was more cost effective and preferred to the supervised ones. The situation is now changed. Many of the more recent supervised works, do not need any manual labeling by human. Instead, efficient rule based methods are benefited from to automatically extract the output label from raw data[2]. For example, to label if two words are in the context of one another, not human labeling but only sliding a window is needed(Le and Mikolov, 2014). As another example, Tomar et al. (2017) propose to use rule based methods to extract labels as noisy proxies for real labels for the task of paraphrase identification and learn from them in a supervised manner.

Regardless of the source of the supervised output, as usual, the nature of it has the major role in defining the learning problem. Different candidates for output type here are binary (like in Quora question pair dataset), categorical levels Like in Stanford Natural Inference data, ordinal levels (like levels of 1-5 in SICK data (Marelli et al., 2014)), and continuous numbers. Depending on the type of output it makes sense to model it as either a classification (multi-class, multi-label, binary), ordinal regression, ranking or regression.

In unsupervised problems, there is no output supervised signal to learn from. Typically a simple rule or heuristic is to be defined on the two inputs and used in deciding if the two questions are equivalent or not. Example are (Pagliardini et al., 2017; Triantafillou et al., 2016). To do.

The type of association The semantics of relations that makes a pair be associated (Mirzazadeh, 2017) is varied in different applications. A variety of applications (Lintean and Rus, 2012) like paraphrase identification (Madnani et al., 2012), textual entailment (Bentivogli et al., 2016) and semantic textual similarity detection (Rychalska et al., 2016) has been investigated in different works. Different applications seek different levels of similarity as the desired level. (Bogdanova

[2]We consider models on unlabeled data with extracted supervised signal to be supervised. However, many authors call them unsupervised

et al., 2015).

In the so-called *duplicate and near-duplicate detection*, the goal is looking for exact or almost exact occurrence of a text in a corpus. The focus here is on finding redundant texts like (Ioannou et al., 2010; Zhang et al., 2010). It is noteworthy that duplicate pairs can be considered as semantically equivalent but not vice versa. New works in this area focus on improving the performance like (Wu et al., 2011).

The act of paraphrasing is generally defined as the restatement (or reuse) of sentences which gives the same meaning but in another form. In *paraphrase identification* we are looking for pairs that are paraphrase of each other. Notably, two paraphrased questions must be semantically equivalent but a given semantically equivalent pair of questions might not be the paraphrase of each other. For example, consider following question pairs from Quora dataset:

i. How can I be a good geologist? ii. What should I do to be a great geologist?
i. Why are dogs considered omnivores? ii. Are dogs carnivorous animals?

Both of the pairs above are considered equivalent as similar answers would be applicable to both. However, the questions in the second pair are neither duplicate nor paraphrases of each other. They are actually asking different things. It worths to note that despite this difference, *paraphrase identification* and *semantic similarity detection* terms are often used interchangeably in literature.

In the so-called *Semantic textual similarity evaluation*, one measures the degree of similarity in the underlying semantics of paired snippets of text, for example in the range of (0-5) where 0 stands for completely dissimilar pair and 5 stands for completely similar. A pair with higher degree have more overlap in meaning. The pairs with the highest degree of similarity (5 in 0-5 system) are also semantically equivalent. This can be done supervised like (Tai et al., 2015) or unsupervised like (Pagliardini et al., 2017; Triantafillou et al., 2016).

Textual entailment recognition expresses a graded semantic relationship but it is non-specific about the nature of the relationship with contradictory material still being a candidate for a high score (e.g., ”Apple” and ”Company” or ”Day” and ”Night” are highly related but not particularly similar).

Depending on the type of problem, different solutions would be appropriate.

While different task exist, in a machine learning context, it is preferable to approach all in a unified manner abstracting away the semantics of association to be learned.

Other details In detecting similarity of questions in a QA system, the detection stage can happen before or after posting questions. This slightly changes the task since the answers to the posted questions are strong signals, if available. But if the detection happens before publishing the questions, there are no answers available and the task is harder.

3.2 Related solutions

Examples of existing solutions in literature range from simple ones like word n-gram overlapping, string matching, to more complicated ones like semantic word similarity, word alignment, syntactic structure, etc (Vo et al., 2015).

For the **supervised** problem with binary outputs, classic classification models such as logistic regression (eg in (Triantafillou et al., 2016)), and support vector machines, as well as learning to rank models have been applied. At 2016, the Quora system used random forest in their system. With the advent of neural network models, a number of deep learning frameworks were used for this proposed: **”Siamese architecture”** and **”matching aggregation”** (Wang et al., 2017). In A Siamese architecture, two identical neural network encoders apply to sentences to obtain embedding of each sentence, (see (Tan et al., 2015)) and in matching aggregation small unit of each sentence such as words or contextual vectors match the other one like (Wang et al., 2016). As a number of other models using NNs, Bogdanova et al. (2015) use **Convolutional Neural Networks**, Wang et al. (2017) use **Character based Long Short Term Memory**, and use decomposable attention model.

Many of the works focus on feature engineering for questions of a specific and typically technical domain. For example, there has been plenty of works dedicated to question-answering about programming to assist software developments. For example Ahasanuzzaman et al. (2016),Bazelli et al. (2013) and Zhang et al. (2015) apply domain-specific feature engineering to the problem, like analyzing the type of questions of stack overflow in (Treude et al., 2011).

For **unsupervised** models, the frequently used older heuristics are, Jaccard [3], Euclidean distance and its variant, other distances such as Levenshtein distances, inner product, cosine similarity, and its variants. The most common unsupervised method for evaluating similarity is computing cosine similarity, which is popular in information retrieval and text mining. Here, the cosine of the angle between the two question vectors is used as the similarity metric. Next, if the cosine of this angle is less than a threshold, the pair are predicted to be similar and in the other case, they are different. One of the reasons for the popularity of cosine similarity is that it is very efficient to evaluate, especially for sparse vectors. Although, here, a cut-off threshold needs to be set.

Other variations of the solutions for **both supervised and unsupervised models** are based on different ways of representing words and sentences. In vocalbulary based models, bag of words or n-grams are used for representing a question. In embedding based models, words are embedded into a vector space to maintain context. In fact embedding can applied in different levels: character level, word level, and sentence level.

4 Feature aggregation

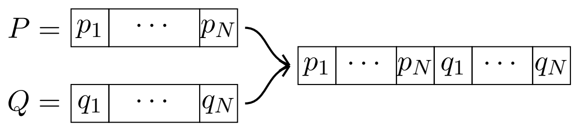
No matter how you represent the questions and do the classifcation, features of the two questions need to be combined. We review standard ways of aggregating, along with some simple generalizations.

4.1 Concatenation

In this method, the n_dimensional vectors of sentence pairs are concatenated to each other and form a 2n_dimension vector. With respect to the fact that each dimension of represented vectors of each sentence is the weight of a specific latent topic in that sentence, in this method, there is no direct connection between features of input questions. (Kiros et al., 2015)

$$e(q_1, q_2) = \langle \phi(q_1) \| \phi(q_2), w \rangle$$

size of rep= $2 \times n$



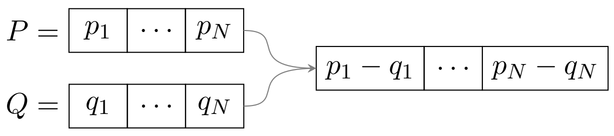
$P = $ p_1 $\cdots$ p_N
$Q = $ q_1 $\cdots$ q_N
p_1 $\cdots$ p_N q_1 $\cdots$ q_N

Figure 2: Concat

[3]Jaccard = $\frac{\text{size of intersection}}{\text{size of union}}$

4.2 Pairwise subtraction

In this method, the final representation of question pairs is an n-dimensional vector in which the i_th element of the vector is the subtract of i_th element of Q1 from i_th element of Q2. So the classifier should decide based on the difference of latent topic weights of question pairs.

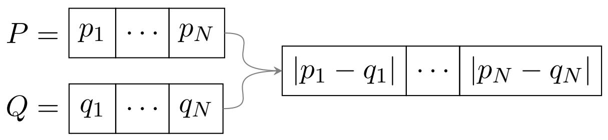


$P = $ p_1 $\cdots$ p_N
$Q = $ q_1 $\cdots$ q_N
$p_1 - q_1$ $\cdots$ $p_N - q_N$

Figure 3: subtract

4.3 Pairwise absolute difference

This method is similar to the previous one except that the i_th element of the final n-dimensional vector is the absolute difference of the i_th element of Q1 and the i_th element of Q2 instead of simple subtraction.

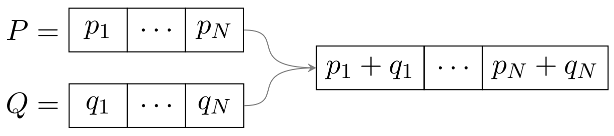


$P = $ p_1 $\cdots$ p_N
$Q = $ q_1 $\cdots$ q_N
$|p_1 - q_1|$ $\cdots$ $|p_N - q_N|$

Figure 4: absolute difference

4.4 Pairwise summation

Here the aggregated representation is an n-dimensional vector in which the i-th element of the vector is the sum of the i_th element of Q1 and the i_th element of Q2. In fact, the weight of a latent topic in the final representation is the sum of the weights of latent topic weights of question pairs.



$P = $ p_1 $\cdots$ p_N
$Q = $ q_1 $\cdots$ q_N
$p_1 + q_1$ $\cdots$ $p_N + q_N$

Figure 5: Pairwise sum

4.5 Pairwise multiplication

Here in the final representation of question pair, the weight of each latent topic is the multiplication of weight of this topic in each of the two questions.

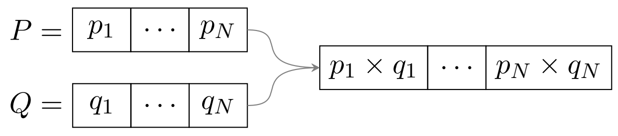


Figure 6: multiplication

4.6 Concatenation of product and absolute difference

In this method, the final representation of question pairs is a 2n-dimentional vector consists of the concatination of representation obtained in section 4.3 and 4.5. which is also used in (Kiros et al., 2015; Triantafillou et al., 2016)

attention: In the above methods, there was no connection between features of input questions or at least a single connection between just the related features. In the following methods, every single feature of the first question is in association with all features of the second question:

4.7 Sum of Question one times features of second question

In this method, the weight of each topic in the final representation is obtained from multiplication of weights of the first question, in the sum of all elements of the second question

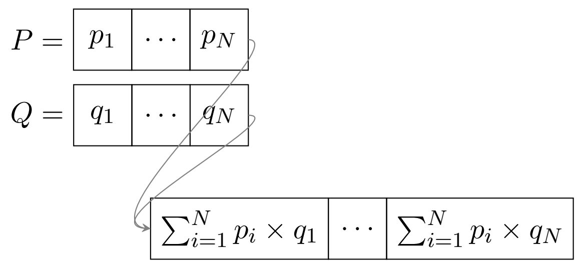

$P = p_1 \cdots p_N$
$Q = q_1 \cdots q_N$
$\sum_{i=1}^{N} p_i \times q_1 \cdots \sum_{i=1}^{N} p_i \times q_N$

Figure 7: Sum of Question one times features of second question

5 The proposed method

5.1 Vectorized Outer product

In this method the to find the final representation, at first, we calculate the outer product of first question's vector and the second questions vector which is a $n \times n$ matrix. Then we vectorize the archived matrix to obtain a vector with $n \times n$ dimension.

In this method, every topic from the first question directly get affected from all features of other question rather than the aggregation of other question (what happened in the previous method) Suppose the first question 's representation captures the properties below

{Yes/no question, WH question, social, sports }
Then the new features are:

Q1 yes/no Q2 yes/no; Q1 yes/no Q2 WH, Q1 yes/no Q2 social, Q1 yes/no Q2 sports; wh

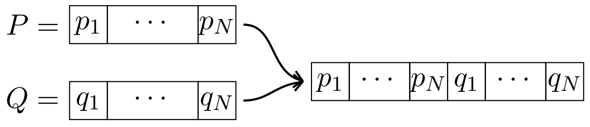
$$\begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} \begin{bmatrix} p_1 & p_2 & \cdots & p_N \end{bmatrix} = \begin{bmatrix} p_1q_1 & p_1q_2 & \cdots & p_1q_N \\ p_2q_1 & p_2q_2 & \cdots & p_2q_N \\ \vdots & & & \vdots \\ p_Nq_1 & p_Nq_2 & \cdots & p_Nq_N \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} p_1q_1 & p_1q_2 & \cdots & p_1q_N & p_2q_1 & \cdots & p_Nq_N \end{bmatrix}$$

Figure 8: outer product

5.2 Matrix multiplication of concatenations

As the last approach, here we combine the first and 6th method, in other words at first we concatenate question vectors of question pair, and then we computer the vectorized outer product on this new vector into its transpose. If the consider that we have n_dimensional question vectors, the final representation of this method is $4 \times n^2$ dimensional vector.



$$\begin{bmatrix} p_1 \\ \vdots \\ p_N \\ q_1 \\ \vdots \\ q_N \end{bmatrix} \begin{bmatrix} p_1 & \cdots & p_N & q_1 & \cdots & q_N \end{bmatrix} = \begin{bmatrix} p_1p_1 & \cdots & p_1q_N \\ p_2p_1 & \cdots & p_2q_N \\ \vdots & & \vdots \\ q_Np_1 & \cdots & q_Nq_N \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} p_1p_1 & p_1p_2 & \cdots & p_1q_N & p_2p_1 & \cdots & q_Nq_N \end{bmatrix}$$

Figure 9: Concat

Table 1: Dataset Statistics

	Quora		AskUbuntu	
	Train	Test	Train	Test
Dup.	120176(37.1%)	29087(35.9%)	24048(37.8%)	6041(37.9%)
nonDup.	203255(62.8%)	51770(64.2%)	39570(62.2%)	9863(62.1%)
total	323431(80%)	80857(20%)	63618(80%)	15904(20%)
	404288		**79522**	

6 Experimental evaluation

6.1 Dataset properties

We conduct experiments on two benchmark datasets for question de-duplication: the recent Quara dataset released in 2016? and AskUbuntu dataset. Table 1 shows the characteristics of the datasets we used. As shown, the number of training examples is around 300K and test examples around 80K. For the askUbuntu dataset, the number of training examples is around 63K and the number of test examples 16K.

The first dataset, Quora is recently published by the popular QA website [4]. This dataset contains 404288 question pairs. Questions are about all different topics. Figure 10 shows a sample of these questions. Here, each question pair is annotated based on human activity with a binary class label indicating whether the two questions are paraphrase of each other.

In particular, positive labels are created for questions that are merged by humans and negative label for the rest. It is expected the labeling has some amount of noise.

As the second dataset, we conducted our experiments on a freely available dataset which is published by stackExchange on its AskUbuntu subforum.

Here questions are about Linux operating system. We subsample about 80K question pairs from this dataset. For consistency, we performed the sampling in a way that the duplicate and non-duplicate fraction is similar to what we had for Quora dataset, i.e. 37% of pairs are duplicate in both data sets.

By looking carefully to these datasets, an important difference between the two is the nature of negative examples. The non-duplicate pairs of Quora dataset are still related questions sharing some level of similarity. By contrast, in AskUbuntu negative examples contain a pair of questions that are very different. So the second dataset could be considered an easier one in that respect.

[4]https://data.quora.com/

6.2 Experimental set-up

Words in each question are separated according to white spaces and non alphabetic tokens are removed. To distinguish the words of each question we used NLTK word tokenizer in Python Bird et al. (2009).The preprocessing applied to each sentence after tokenizing it using NLTK word tokenizer consists of changing to lower case, removing stop words and removing non-English alphabetic tokens (like numbers or words in other languages).

For representing the words, a pre-trained skip gram model (word2vec: (Mikolov et al., 2013)) is applied to come up with word features. The model is pre-trained on google-news[5] and returns a 300 dimensional feature vector for each word.

The next step is encoding the sentence, for which a weighted average sentence encoder is used. The *average* of word vectors is computed to form the representation of the sentence.This step for a sentence converts the matrix of word representations into a vector of question representation. Then we normalized this vector with its L_2 norm.

We compared the performance of the proposed word aggregation model with four other standard models. As the final classification, logistic regression and SVM classifiers with L2 squared regularization. Both datasets are divided into a 80/20 splits between training and test data.

Also due to the high amount of generated data, we partially fitted the classifier with 64 sample batch size.

[5]https://code.google.com/archive/p/word2vec

Table 2: Comparison of different aggregation methods on Quora dataset.The classification performance (%) after aggregation on test data is reported.

	Logistic Regression			SVM		
	Accuracy	F_1	AUC	Accuracy	F_1	AUC
Concatenation	69.51	48.73	63.12	69.92	45.96	62.41
Summation	69.59	48.29	63.01	69.91	45.45	62.25
Subtraction	63.62	0.23	49.78	63.92	0	50
Multiplication	69.64	51.79	64.32	71.69	56.66	67.25
abs diff	67.99	47.05	61.77	67.53	41.81	59.86
abs diff —— multi	71.60	$\boxed{\textit{56.31}}$	$\boxed{\textit{67.05}}$	72.27	$\boxed{\textit{57.80}}$	$\boxed{\textit{67.99}}$
OP1	$\boxed{\textit{73.53}}$	53.95	66.88	$\boxed{\textit{73.62}}$	51.86	66.18
OP2	**75.34**	**58.78**	**69.55**	**76.08**	**60.09**	**70.38**

Table 3: Comparison of different aggregation methods on AskUbuntu dataset.The classification performance (%) after aggregation on test data is reported.

	Logistic Regression			SVM		
	Accuracy	F_1	AUC	Accuracy	F_1	AUC
Concatenation	81.69	74.13	79.39	81.89	74.28	79.51
Summation	80.62	72.59	78.33	80.90	72.85	78.53
Subtraction	68.78	47.59	62.47	69.33	43.52	61.57
multiplication	77.31	60.69	71.11	79.02	64.59	73.38
abs diff	76.10	66.32	73.41	76.19	66.67	73.63
abs diff —— multi	80.48	72.16	77.85	82.07	74.32	79.47
OP1	$\boxed{\textit{83.93}}$	$\boxed{\textit{77.63}}$	$\boxed{\textit{82.08}}$	$\boxed{\textit{84.91}}$	$\boxed{\textit{80.09}}$	$\boxed{\textit{84.20}}$
OP2	**85.89**	**80.70**	**84.52**	**85.50**	**81.10**	**85.41**

6.3 Competitors

We compared the performance of the proposed methods with four standard methods that almost every paper uses them: **[Concatenation, summation, subtraction, multiplication]**. As mentioned in Section 4, except the first competitor, all others are operations that applied element-wise between the two representation.

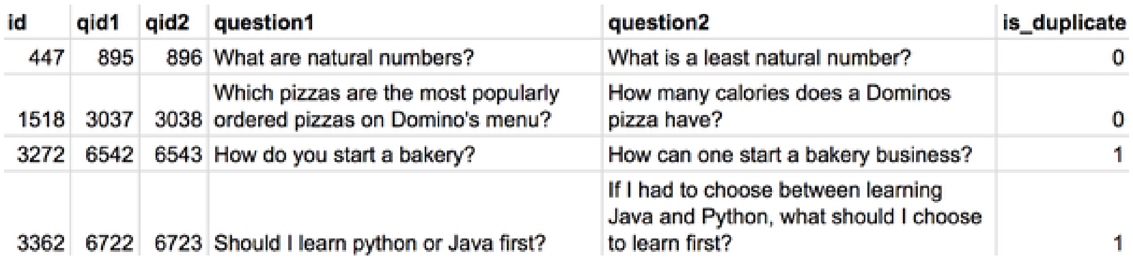
6.4 Results

The results are shown in Table 2 and Table 3. Table 2 reports test performance for Quora dataset and Table 3 shows the results for the AskUbuntu data. The first block is related to logistic regression while the second to support vector machines. Accuracy, F1 and AUC measures are reported. OP1 is our first proposal and Op2 the second proposal. The best performance is shown in bold. Interestingly in all cases OP2 gives the best performance. The second best results have boxes around them. Except for element-wise multiplication that wins in terms of F_1 and AUC, in one of the case, in all other cases the runner up is OP2. OP1 the second best, measured with all three criteria.

7 Conclusions

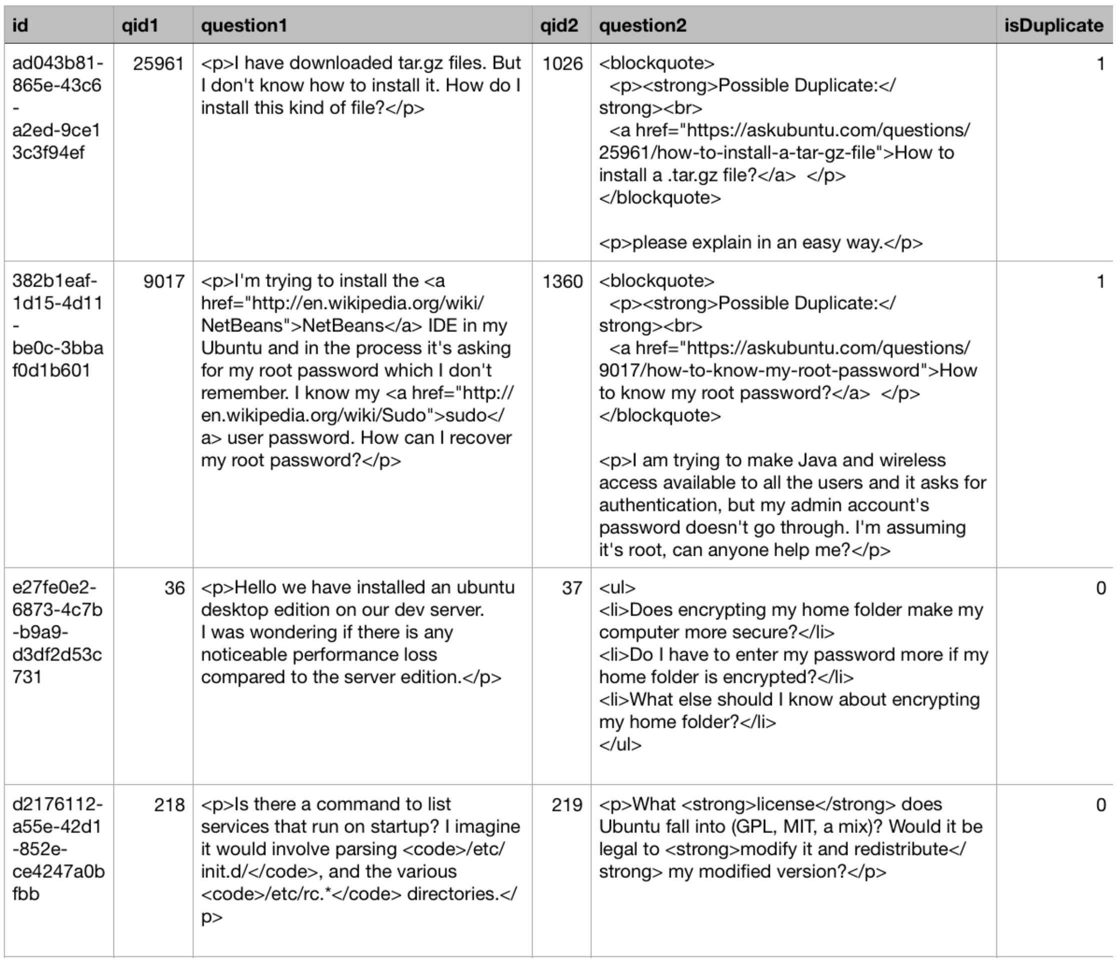
Summary of contributions

Future works



id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

Figure 10: Sample duplicate and non-duplicate questions from Quora dataset



id	qid1	question1	qid2	question2	isDuplicate
ad043b81-865e-43c6-a2ed-9ce13c3f94ef	25961	<p>I have downloaded tar.gz files. But I don't know how to install it. How do I install this kind of file?</p>	1026	<blockquote> <p><strong>Possible Duplicate:</strong> <a href="https://askubuntu.com/questions/25961/how-to-install-a-tar-gz-file">How to install a .tar.gz file?</a> </p> </blockquote> <p>please explain in an easy way.</p>	1
382b1eaf-1d15-4d11-be0c-3bbaf0d1b601	9017	<p>I'm trying to install the <a href="http://en.wikipedia.org/wiki/NetBeans">NetBeans</a> IDE in my Ubuntu and in the process it's asking for my root password which I don't remember. I know my <a href="http://en.wikipedia.org/wiki/Sudo">sudo</a> user password. How can I recover my root password?</p>	1360	<blockquote> <p><strong>Possible Duplicate:</strong> <a href="https://askubuntu.com/questions/9017/how-to-know-my-root-password">How to know my root password?</a> </p> </blockquote> <p>I am trying to make Java and wireless access available to all the users and it asks for authentication, but my admin account's password doesn't go through. I'm assuming it's root, can anyone help me?</p>	1
e27fe0e2-6873-4c7b-b9a9-d3df2d53c731	36	<p>Hello we have installed an ubuntu desktop edition on our dev server. I was wondering if there is any noticeable performance loss compared to the server edition.</p>	37	<ul> <li>Does encrypting my home folder make my computer more secure?</li> <li>Do I have to enter my password more if my home folder is encrypted?</li> <li>What else should I know about encrypting my home folder?</li> </ul>	0
d2176112-a55e-42d1-852e-ce4247a0bfbb	218	<p>Is there a command to list services that run on startup? I imagine it would involve parsing <code>/etc/init.d/</code>, and the various <code>/etc/rc.*</code> directories.</p>	219	<p>What <strong>license</strong> does Ubuntu fall into (GPL, MIT, a mix)? Would it be legal to <strong>modify it and redistribute</strong> my modified version?</p>	0

Figure 11: Sample duplicate and non-duplicate questions from AskUbuntu dataset

References

Muhammad Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K. Roy, and Kevin A. Schneider. 2016. Mining duplicate questions in stack overflow. In *MSR*. ACM.

Blerina Bazelli, Abram Hindle, and Eleni Stroulia. 2013. On the personality traits of stackoverflow users. In *ICSM*. IEEE Computer Society.

Luisa Bentivogli, Raffaella Bernardi, Marco Marelli, Stefano Menini, Marco Baroni, and Roberto Zamparelli. 2016. SICK through the semeval glasses. lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Language Resources and Evaluation* 50(1):95–124.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Dasha Bogdanova, Cícero Nogueira dos Santos, Luciano Barbosa, and Bianca Zadrozny. 2015. Detecting semantically equivalent questions in online user forums. In *Conference on Computational Natural Language Learning, CoNLL*.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *WWW*. ACM.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Neural Information Processing Systems (NIPS)*.

Ekaterini Ioannou, Odysseas Papapetrou, Dimitrios Skoutas, and Wolfgang Nejdl. 2010. Efficient semantic-aware detection of near duplicate resources. In *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Neural Information Processing Systems (NIPS)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*.

Mihai C. Lintean and Vasile Rus. 2012. Measuring semantic similarity in short texts through greedy pairing and word semantics. In *International Artificial Intelligence Research Society Conference*.

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

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *International Conference on Language Resources*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Farzaneh Mirzazadeh. 2017. *Solving Association Problems with Convex Co-embedding*. Ph.D. thesis, University of Alberta.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *CoRR* abs/1703.02507.

Barbara Rychalska, Katarzyna Pakulska, Krystyna Chodorowska, Wojciech Walczak, and Piotr Andruszkiewicz. 2016. Samsung poland NLP team at semeval-2016 task 1: Necessity for diversity; combining recursive autoencoders, wordnet and ensemble methods to measure semantic similarity. In *SemEval@NAACL-HLT*. The Association for Computer Linguistics.

Sazianti Mohd Saad and Siti Sakira Kamarudin. 2013. Comparative analysis of similarity measures for sentence level semantic measurement of text. In *ICCSCE*. IEEE, pages 90–94.

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 *Neural Information Processing Systems (NIPS)*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*. The Association for Computer Linguistics.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR* abs/1511.04108.

Gaurav Singh Tomar, Thyago Duque, Oscar Tackstrom, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. In *First Workshop on Subword and Character Level Models in NLP*.

Christoph Treude, Ohad Barzilay, and Margaret-Anne D. Storey. 2011. How do programmers ask and answer questions on the web? In *ICSE*. ACM.

Eleni Triantafillou, Ryan Kiros, Raquel Urtasun, and Richard Zemel. 2016. Towards generalizable sentence embeddings. In *Workshop on Representation Learning for NLP. ACL*.

Ngoc Phuoc An Vo, Simone Magnolini, and Octavian Popescu. 2015. FBK-HLT: an effective system for paraphrase identification and semantic similarity in twitter. In *International Workshop on Semantic Evaluation, SemEval@NAACL-HLT5*.

Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '08.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. In *COLING*. ACL.

Yan Wu, Qi Zhang, and Xuanjing Huang. 2011. Efficient near-duplicate detection for q&a forum. In *IJCNLP*. The Association for Computer Linguistics.

Qi Zhang, Yue Zhang, Haomin Yu, and Xuanjing Huang. 2010. Efficient partial-duplicate detection based on sequence matching. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 675–682.

Yun Zhang, David Lo, Xin Xia, and Jianling Sun. 2015. Multi-factor duplicate question detection in stack overflow. *J. Comput. Sci. Technol.* 30(5):981–997.