Using different approach to classify questions

Petr Lorenc
lorenpe2@fit.cvut.cz

May 9, 2017

Abstract

Main goal of this project was to build up a system which will give several answer to user's questions. The work will explore different ways to classify questions to several topics. Result of this work will be used in future work, which will be focused on automatic reply to user's messages. I have tried several approaches (begin with classical access like SVM and Naive Bayes to Neural network with word embeddings).

1 Introduction

There are a lot of emails, which are comming to your inbox. Most of them would be probably spam, but certain part will be classical messages which request for reply. Almost everybody wants to reply as quickly as possible, but it is very hard problem for computer to suggest right answer Unfortunately I don't have enough HW power and enough data like Google or Amazon, so I can not afford to use something like sequence-to-sequence neural network to generate answer to question direcly, that leads to classification question to some category and suggest some human-provided reply based on a priori experiences. Results can be used later until we have enough data - the glue to generate valid answer is if we know the topic of message. This work is focused on the classification part. I will try to develop system similar to [Figure 1] which can be used in bigger system like [Figure 2] for IR-based factoid question answering to get answer type. The answer type is very useful for filtering possible answers (if we know that question is about HUMAN we can remove answers about LOCATION).

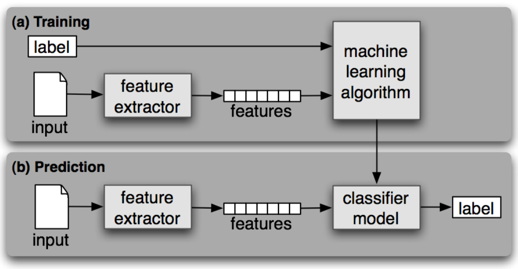


Figure 1: Supervised Classification[1]

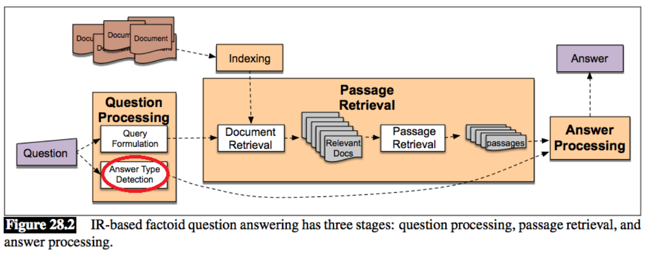


Figure 2: IR based factoid question answering[17]

Google, as one of the main email provider company, develop their system for smart reply[1]. They used several advance techniques (like system for semantic intent clustering, system for omitting redundant response, sequence-to-sequence neural network etc.). One part is also "Identifying a target response space" on which I focused.

I will try several methods like linear SVM, Naive Bayes or non-linear k-NN. Then I will focus on Neural network and their ability to remember some useful pattern in text (Recurent Neural Network). I will also explore some some existing way to represent word by their similarities like Word2Vec or Glove.

2 Related work

The classification topic was part of several research, on which I will focus in this section.

This work (Intent Classification of Short-Text on Social Media[4]) suggest to several useful steps for preprocessing input. Besides classical approach like removing stop words and stemming, it suggest to replace Named Entities with their representaion (for example instead of "How did serfdom develop in and then leave Russia ?" use "how do serfdom develop in and then leave GPE ?" where GPE is abbreviation for geo-political entity). I will go further and try to get information even from POS tagging (so example sentence would look like "how ADV do ADJ serfdom ADV develop VERB in ADP and CONJ then ADV leave VERB GPE NOUN ?")

In this work (Text Categorization with Support Vector Machines: Learning with Many Relevant Features[7]) is big emphasis on using Support Vector Machine for text classification. He get 86% accuracy on similar task (where he used Reuters dataset and had to classify to 90 categories). He claim that SVM (polynomial and rbf kernel) outperform k-NN classifier and Naive Bayes.

The work (Question Identification on Twitter[9]) was focused mainly on Twitter but some ideas can be used even for my work. It uses 5W1H to detect question in texts (this basic rule can be used in the future question detection)

Because our dataset contains only questions, we can define them as short texts. In the work on text similarities (Representation learning for very short texts using weighted word embedding aggregation[6]) is several baseline method how to represent such short texts. We will use classical Tf-Idf, Bag-of-words representation and vector representation, where we take a mean from all word's vectors and min-max (which lead to vector with doubled size). They also point out that if we use pre-trained model we can be sure that the vocabularies will overlap.

Word2Vec is model to represent word as vector - with property that semantically similar word will be close in vector space. This concept was introduced in Efficient Estimation of Word Representations in Vector Space[10]. Glove[11] is another approach how to represent word as vector. This models are only for words, so if we want some consistent representation of whole sentence, we have to combine these vector or use doc2vec[12]. I will explain their ideas in later part.

With Recurrent Neural network we can capture long term dependencies and with Convolution Neural network we can capture dependencies between part of words. The work (A C-LSTM Neural Network for Text Classification[13]) combine both neural network to create CNN with LSTM. They get very good results (accuracy about 94% but they also claim that SVM classifier which uses unigrams, bigrams, wh-word, head word, POS tags, parser, hypernyms, WordNet and 60 hand-coded rules have slightly better results).

3 Methods of preprocessing

3.1 Source of data

This work is focused on the questions. There are not enough data on the internet, which we can use (they are not labeled or it would be very time consuming to process them to some representative form. But because good data are very important, I focused on searching and preprocessing several data source to get as much information as possible.

1. Database of human-labeled question[2] - around 5500 sentences

2. Preprocessing of Enron[3] dataset
   Take raw data -> sort them to From-To objects -> extracting question with regular expression -> get unique question - around 120 000 sentences

3. Kaggle[4] database of questions - around 914 000 sentences

3.2 Data

Unless stated otherwise I will use first database (human-labeled question). This dataset was derivated from work [2] and [3], where is about 5500 human labeled question. They use two types of classes (coarse and fine). I focus only on coarse part (category are shown below).

3.3 Methology

1. Abbreviation (86)
   What does S.O.S. stand for ?

[2]http://cogcomp.cs.illinois.edu/Data/QA/QC/
[3]//www.cs.cmu.edu/enron/
[4]https://www.kaggle.com/c/quora-question-pairs

2. Description (1162)
   What is an annotated bibliography ?

3. Entity (1250)
   What films featured the character Popeye Doyle ?

4. Human (1223)
   What was the name of the first Russian astronaut to do a spacewalk ?

5. Location (835)
   What ocean is the largest in the world ?

6. Number (896)
   What 's the American dollar equivalent for 8 pounds in the U.K. ?

3.4 Representation of data

I was wonder if some type of preprocessing improve quality of results, so I try these:

1. RAW
   How did serfdom develop in and then leave Russia ?

2. lemmatizing + POS tagging
   how ADV do ADJ serfdom ADV develop VERB in ADP and CONJ then ADV leave VERB russia NOUN ?

3. lemmatizing + Replace entity
   how do serfdom develop in and then leave GPE ?

Removing of stopwords was in a separate part, where there we test removing all stopword, stopwords (without question words) and not removing anything.

I tried also several type of representation. I will go through all of them briefly.

1. Tf-Idf

2. Bi-grams + Tf-Idf

3. Vector representation (with/without scale normalization)

   (a) Word2Vec
   (b) Glove
   (c) Doc2Vec

3.5 Word2Vec

There are 2 main architecture (Continuous Bag-of-Word and Skip-gram). Main difference is how we train our model. In CBOW [Figure 3] we try predict central word with surrounding, in the skip-gram [Figure 4] is input central word and we try to predict surrounding. Training of Skip-gram slower but we get better result in general. I will used their pre-trained model[5] which was trained on almost 3 million word using skip-gram architecture and negative sampling (which is mainly optimization to avoid go through all vocabulary in a train part). I will use pre-trained model provided by Google.

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

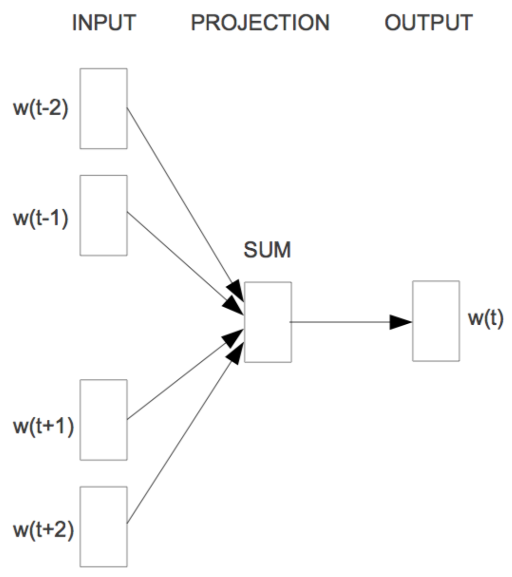


Figure 3: Continuous bag of word model[6]

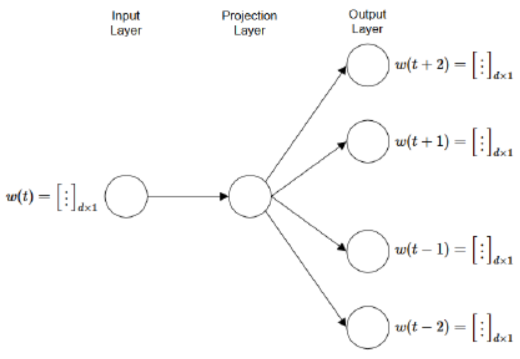

Input Layer
Projection Layer
Output Layer
$w(t) = [\vdots]_{d\times1}$
$w(t+2) = [\vdots]_{d\times1}$
$w(t+1) = [\vdots]_{d\times1}$
$w(t-1) = [\vdots]_{d\times1}$
$w(t-2) = [\vdots]_{d\times1}$

Figure 4: Skip-gram model [7]

3.6 Glove

In opposite to Word2Vec this is unsupervised algorithm based on reducing from very large matrix of co-occurence of words (how many times word1 occur in the context of word2) to low dimension. They have some pre-trained models too[8]. I will use pre-trained model.

3.7 Doc2Vec

The model is very similar to Word2Vec but we add a paragraph vector to help us predict words (as above here are also 2 architecture: Distributed Memory (DM) and Distributed Bag of Words (DBOW) - DM attempts to predict a word given its previous words and a paragraph vector. DBOW predicts a random group of words in a paragraph given only its paragraph vector [Figure 5]). It is also important to realize that context of paragraph can be much more informative than we think (for example if we have sentence like "He get all <TARGET>", we can say that there is high probability that target word is "votes" if we know that this sentence was from article "Presidential election"). I will use pretrained on wikipedia[9] and train on Enron emails and Kaggle question dataset.

[8]available https://nlp.stanford.edu/projects/glove/
[9]https://github.com/jhlau/doc2vec

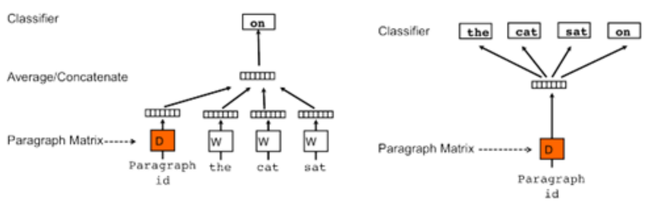


Figure 5: Distributed Memory (DM) and Distributed Bag of Words (DBOW) [12]

4 Classification

4.1 Naive Bayes

Naive Bayes classifiers are linear classifiers that are known for being simple yet very efficient. Especially for small sample sizes, naive Bayes classifiers can outperform the more powerful alternatives. But strong violations of the independence assumptions and non-linear classification problems can lead to very poor performances of naive Bayes classifiers. This can be our case if we represent our sentences as vector created from words.

4.2 Support Vector Machine

SVM searches for a separating hyperplane, which separates positive and negative examples from each other with maximal margin, in other words, the distance of the decision surface and the closest example is maximal Joachims [15] sums up, why SVMs are good for Text classification:

1. High-dimensional input space.
2. Few irrelevant features: almost all feature contain considerable information.
3. Document vectors are sparse
4. Most text categorization problems are linearly separable

But on the other hand, train SVM is very time consuming with large data and with multi-class data we have to do one-to-one classification, which will take even longer amount of time.

4.3 kNN

This is very basic, but also very powerful classification model. But we have to be very careful with parameters because our vector space will be huge.

4.4 Neural network

Neural network are very powerful tools nowadays. One of the reason is increasing computer power and moving computation from CPU to GPU. Classical neural network are basically matrix multiplication with adding a biases.

$$Output = K(\sum_{0}^{i} w_i * x_i) \tag{1}$$

where K is our activation function and x_i is a vector $(bias, x_1, x_2)$ of input to perceptron.

This architecture unfortunately does not keep information from one sample to another. This is very bad because we have a series of words which are very dependent. The LSTM[14] network solve this problem. They use both previous state and current input [Figure 6].

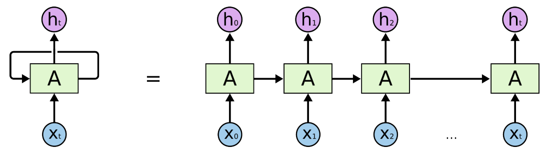


Figure 6: LSTM network [14]

We will focus on these recurrent network. For implementing Neural network we use Keras[10]. One drawback is that we need to have input in as fixed size vector (there are other variant but this is easiest). We choose as max size 39 which is suitable all sentences [Figure 7].

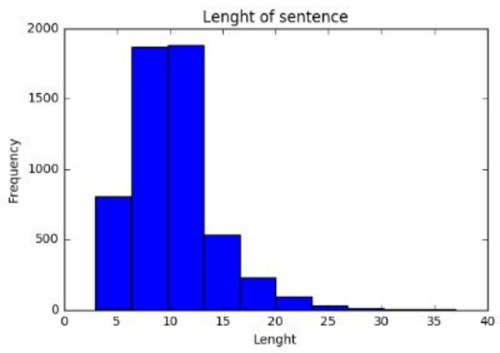


Figure 7: Length of sentences

4.4.1 FastText

Facebook release library for efficient learning of word representations and sentence classification. It is called FastText[11]. It is very fast implementation of Word2Vec model with some improvement which are described in original paper [8]. One of the creator of this library is Tomas Mikolov, which stands behind Word2Vec. This library can be train even on laptop in minutes and can be used to word to vector conversion or class classification as well. I will take a look at possibilities of this library.

5 Results

Strength of preprocessing are not suitable for vector representation of sentences [Figure 8]. It is mainly because if we have vector for New York (it already contain information that it is location) so if we replace it with tag LOC, we will not get much improvement (because we does not care about answer, but only the class). In the further work I will focus only on ENTITY preprocessing.

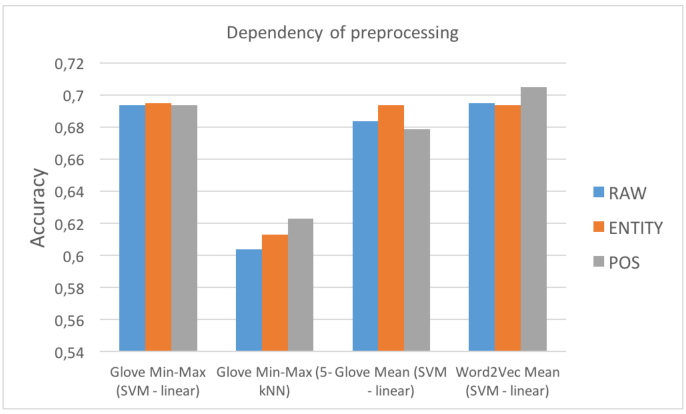


Figure 8: Strength of preprocessing

These result [Figure 9] show that for example Radius Nearest Neighbors are not suitable for large vector space. They have option that if we does not find a examples in some radius they does not classify it. It leads to big precision but small recall. From result we see that SVM with tf-idf outperform other method (it is mainly because of the reason I mention above). I tried train my own doc2vec model, on Enron dataset and on Kaggle question datasets. Results was very similar to each other - I used Distributed Memory model and train for 10 iteration with aim to represent sentence as 500-dim vector. I think that this method can be better if we train for a longer time (current 10 epoch was trained in 30 hours on 8 CPU and 16 GB RAM - using Metacentrum[12])

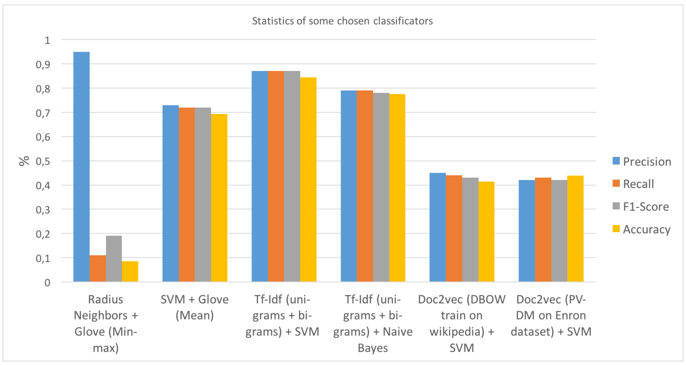


Figure 9: Precision/Recall/F1-score/Accuracy of several methods

5.1 Neural network

I tried also LSTM network to classify question. The result seems to be promising [Figure 10], accuracy goes to 90% and they are no big sight of overfitting, but these networks require strong HW, I have to use atleast 8 CPU and 16 GB RAM (and graphics card) - using Metacentrum. The architecture looks like shown on [Figure 11]. Input to network is 39*300 (where 39 is the longest sentence and 300 are the size of the embedding). I think that we can improve accuracy with adding more layers, but it will increase computation time. Dense layer in diagram mean fully connected neural network.

[10]https://keras.io/
[11]Available here https://github.com/facebookresearch/fastText

[12]metacentrum.cz

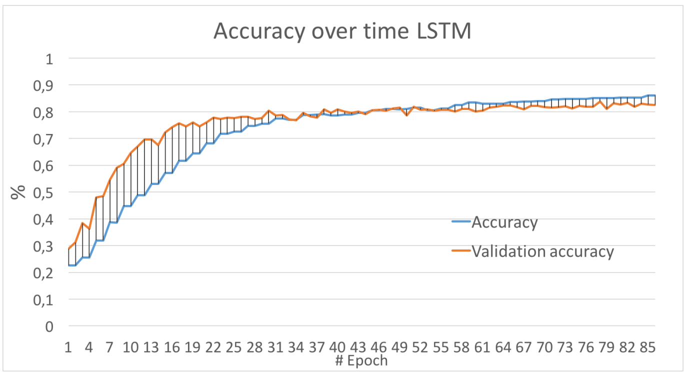


Figure 10: Training LSTM network

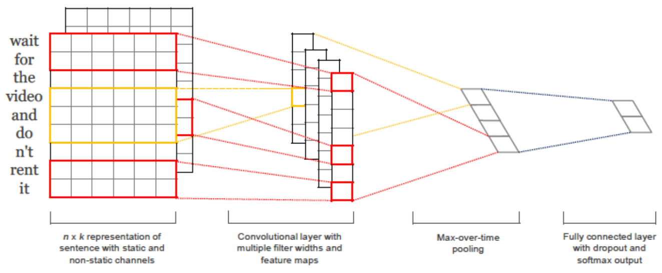

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 12: CNN network model with NLP[16]

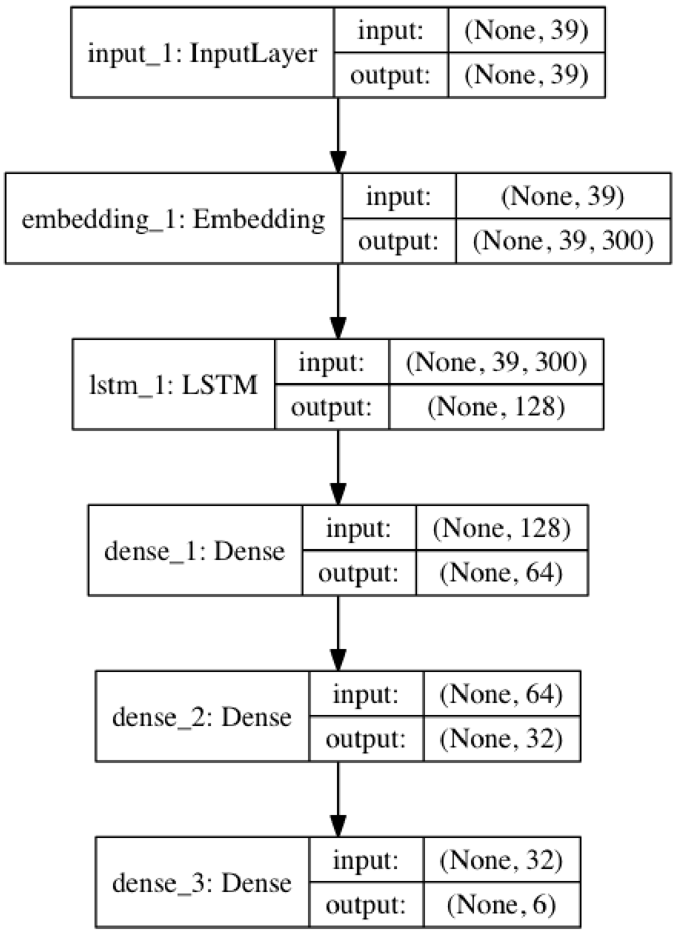


Figure 11: LSTM network model

The CNN can be also used for text classification (as was shown in [16]). Their results are a slightly worse than tuned SVM. They, same as me, use pretrained model to create vectors from words and then use 1D convolution network like shown on the [Figure 12] My architecture looks like on [Figure 13] and train process is shown at [Figure 14]. It is obvious, that we are getting overfitted after 15 epochs.

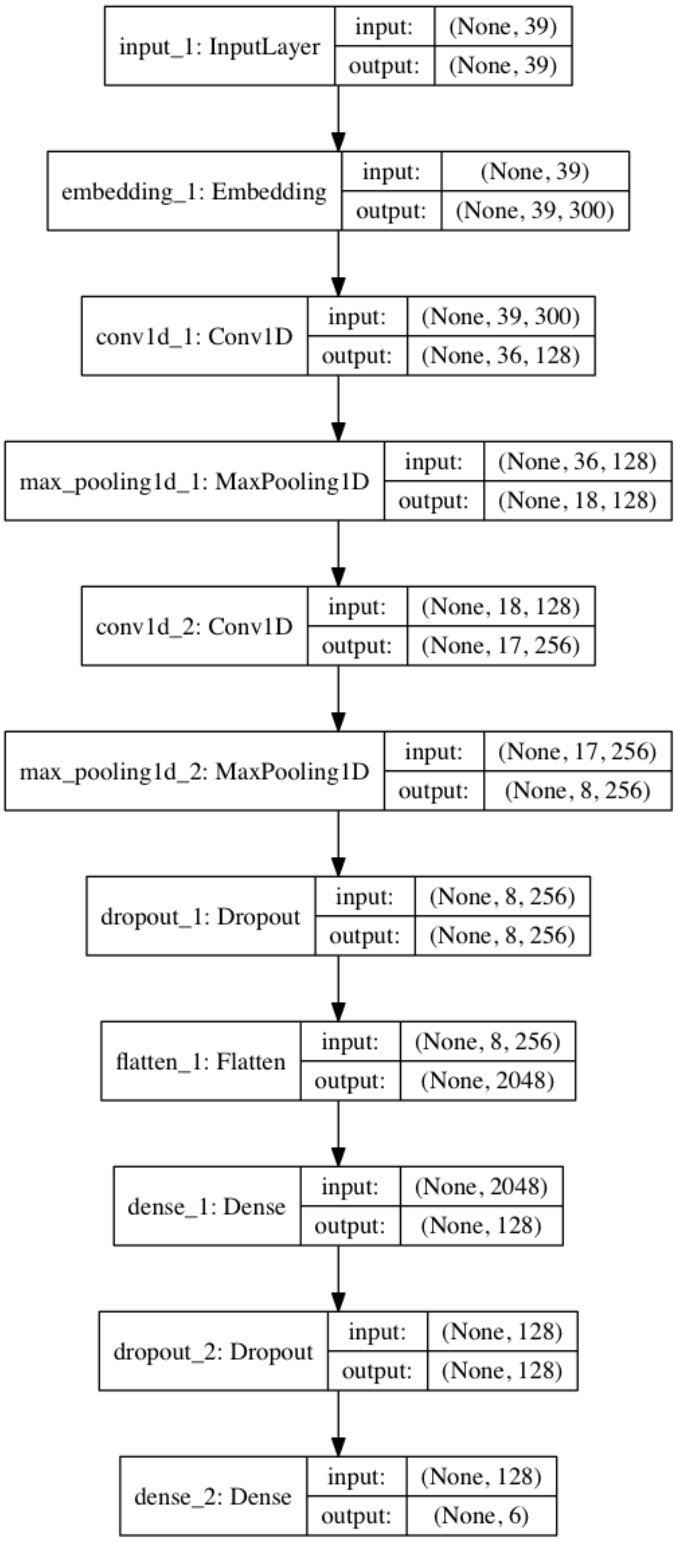

input_1: InputLayer | input: (None, 39) | output: (None, 39)
embedding_1: Embedding | input: (None, 39) | output: (None, 39, 300)
conv1d_1: Conv1D | input: (None, 39, 300) | output: (None, 36, 128)
max_pooling1d_1: MaxPooling1D | input: (None, 36, 128) | output: (None, 18, 128)
conv1d_2: Conv1D | input: (None, 18, 128) | output: (None, 17, 256)
max_pooling1d_2: MaxPooling1D | input: (None, 17, 256) | output: (None, 8, 256)
dropout_1: Dropout | input: (None, 8, 256) | output: (None, 8, 256)
flatten_1: Flatten | input: (None, 8, 256) | output: (None, 2048)
dense_1: Dense | input: (None, 2048) | output: (None, 128)
dropout_2: Dropout | input: (None, 128) | output: (None, 128)
dense_2: Dense | input: (None, 128) | output: (None, 6)

Figure 13: CNN network model

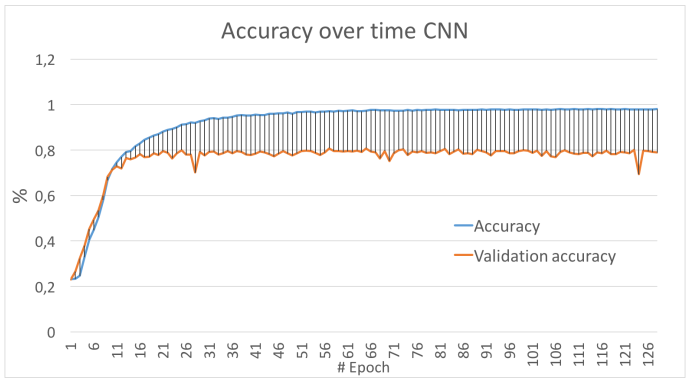


Figure 14: Training CNN network

As a natural conclusion from previous two statements is that combination of the LSTM+CNN (mentioned in [13]) in my work looks like on [Fiqure 15] and get results shown on [Figure 16]. Here are some overfitting too. Main reason why this models overfit is that they are to strong for a small dataset. It would be better to collect more data.

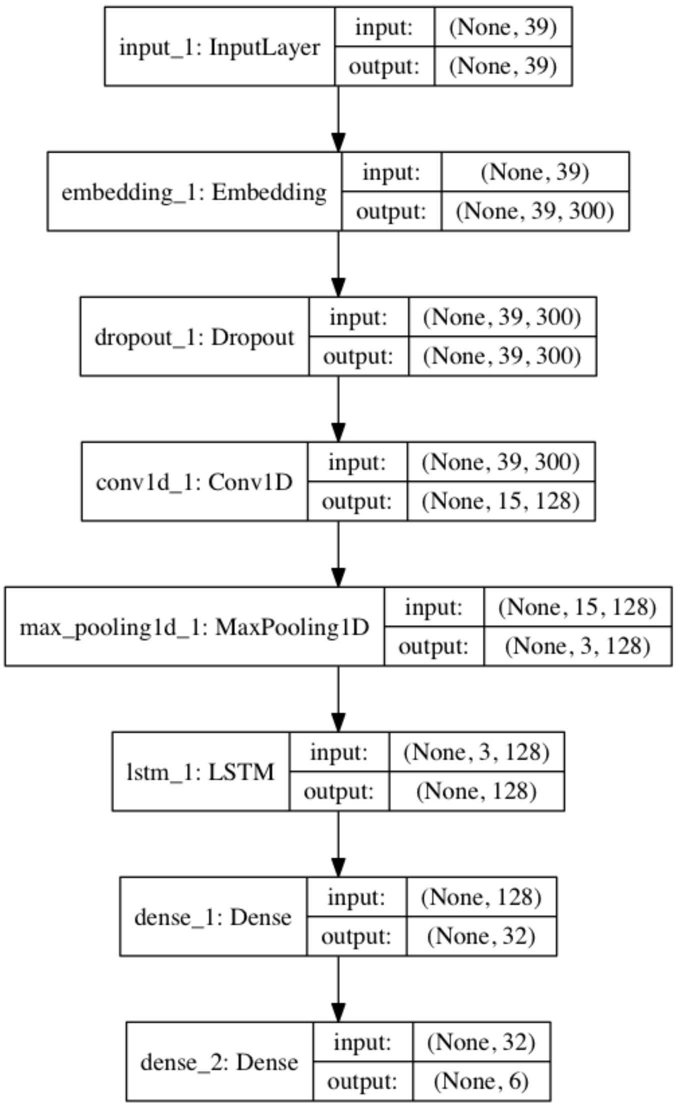

input_1: InputLayer | input: (None, 39) | output: (None, 39)
embedding_1: Embedding | input: (None, 39) | output: (None, 39, 300)
dropout_1: Dropout | input: (None, 39, 300) | output: (None, 39, 300)
conv1d_1: Conv1D | input: (None, 39, 300) | output: (None, 15, 128)
max_pooling1d_1: MaxPooling1D | input: (None, 15, 128) | output: (None, 3, 128)
lstm_1: LSTM | input: (None, 3, 128) | output: (None, 128)
dense_1: Dense | input: (None, 128) | output: (None, 32)
dense_2: Dense | input: (None, 32) | output: (None, 6)

Figure 15: C-LSTM network model

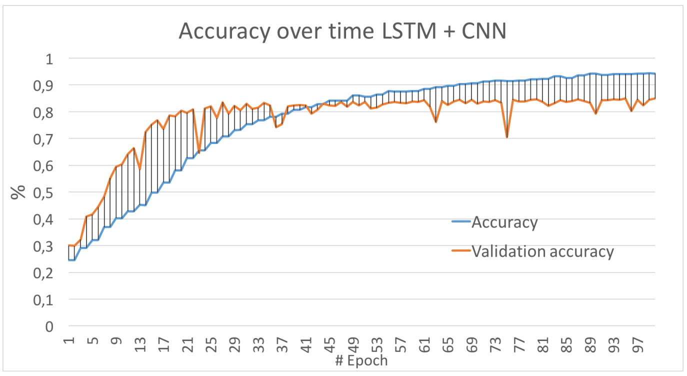
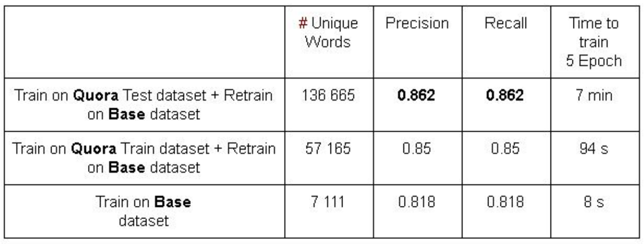


Figure 16: Training CNN + LSTM network

In a part about Neural network I mention library FastText. It's result are stunning. I ran their skipgram algorithm on Enron dataset and train model was used as pretrained model for supervised part (classification) result are shown on Figure 17



	# Unique Words	Precision	Recall	Time to train 5 Epoch
Train on **Quora** Test dataset + Retrain on **Base** dataset	136 665	**0.862**	**0.862**	7 min
Train on **Quora** Train dataset + Retrain on **Base** dataset	57 165	0.85	0.85	94 s
Train on **Base** dataset	7 111	0.818	0.818	8 s

Figure 17: Results of FastText

6 Discussion

I try several implementation of method to classify questions. The results suggest that tuned SVM is the winner, but SVM is not very suitable for very large data. The LSTM implementation seems also very promising and I want to explore their usability further. This work was introduction part of bigger project and I would really like to build on it.

I would like to focus on the work [13], where is combination of LSTM and CNN. I built very basic model which can be extend in many ways They claim that the results are very good compare to other methods (like LSTM or CNN alone).

I would really thank to Metacentrum, where a lot of my computation took place (I used almost 120 days of their CPU time). I would also like to thank to Jan Sedivy for some advises during this project.

All my codes and result are publicly available at https://github.com/petrLorenc/EmailReply.

Reference

[1] *Smart Reply: Automated Response Suggestion for Email* [online]. Authors: Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, Vivek Ramavajjala Available at:http://www.kdd.org/kdd2016/papers/files/Paper_1

[2] *Learning Question Classifiers* [online]. Authors: Xin Li, Dan Roth Available at:http://acl-arc.comp.nus.edu.sg/archives/acl-arc-090501d3/data/pdf/anthology-PDF/C/C02/C02-1150.pdf

[3] *Learning Question Classifiers: The Role of Semantic Information* [online]. Authors: Xin Li, Dan Roth Available at:http://l2r.cs.uiuc.edu/ danr/Papers/LiRo04a.pdf

[4] Intent Classification of Short-Text on Social Media Authors: Hemant Purohit, Guozhu Dong, Valerie Shalin, Krishnaprasad Thirunarayan, Amit Sheth

[5] Representation learning for very short texts using weighted word embedding aggregation Authors: Cedric De Boom, Steven Van Canneyt, Thomas Demeester, Bart Dhoedt Available at:https://arxiv.org/abs/1607.00570

[6] Semantic Text Similarity Using Corpus-Based Word Similarity and String Similarity Authors: AMINUL ISLAM and DIANA INKPEN URL: `<http://www.site.uottawa.ca/~diana/publications/tkdd.pdf>`

[7] Text Categorization with Support Vector Machines: Learning with Many Relevant Features Thorsten Joachims URL: `<https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf>`

[8] Bag of Tricks for Efficient Text Classification Authors: Armand Joulin ,Edouard Grave ,Piotr Bojanowski ,Tomas Mikolov URL: `<https://arxiv.org/pdf/1607.01759.pdf>`

[9] Question Identification on Twitter Authors: Baichuan Li, Xiance Si, Michael R. Lyu1, Irwin King13 and Edward Y. Chang2 Available at:https://static.googleusercontent.com/media/research

[10] *Efficient Estimation of Word Representations in Vector Space* [online]. Authors: Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean Available at:http://arxiv.org/pdf/1301.3781.pdf

[11] *GloVe: Global Vectors for Word Representation* [online]. Authors: Jeffrey Pennington, Richard Socher, Christopher D. Manning Available at:https://nlp.stanford.edu/pubs/glove.pdf

[12] *Distributed Representations of Sentences and Documents* [online]. Authors: Quoc Le, Tomas Mikolov Available at:http://proceedings.mlr.press/v32/le14.pdf

[13] *A C-LSTM Neural Network for Text Classification* [online]. Authors: Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau Available at:https://arxiv.org/abs/1511.08630

[14] *Long short-term memory. Neural computation* 1997 Authors: Sepp Hochreiter and Jurgen Schmidhuber Available at:http://www.bioinf.jku.at/publications/older/2604.pdf

[15] *Text categorization with support vector machines:learning with many relevant features* [1998] Authors: Thorsten Joachims From at:10th European-Conference on Machine Learning, Springer Verlag, Heidelberg, DE, 1998, pp. 137-142

[16] *Convolutional Neural Networks for Sentence Classification* [Online] Authors: Yoon Kim Available at:https://arxiv.org/abs/1408.5882

[17] *Question Answering - Lectures from Stanford* [Online] Authors: Daniel Jurafsky & James H. Martin Available at:https://web.stanford.edu/ jurafsky/slp3/28.pdf