Sentiment Analysis in Czech

Capstone Project for Machine Learning Nanodegree at Udacity

Review #1

(Dated: November 1, 2017)

Radek Kyselý

I. Definition

A Project Overview

Sentiment analysis is one of the most researched topics in machine learning and natural language processing. Identifying a sentiment in one's speech helps us understand their underlying needs which aids us in maintaining more empathetic conversations; or enables us to conduct opinion mining for analyzing people's affection for a certain topic (e.g. product reviews, elections analysis, etc).

Despite the wealth of state-of-art solutions and APIs available for mainstream languages like English, there are very few publicly available sentiment analysis experiments in Czech. Throughout our project, we will be using findings and notes from the research by *Habernal et al. (2013)* at University of West Bohemia which explored the use of machine learning to recognize sentiments in Czech social media posts. Three data sets provided by the research team will also be used in this project.

B Problem Statement

While a classifier of 7 basic human emotions would be desirable for in-depth sentiment analysis, recognizing those is extremely challenging even for the best among humans.

Thus, the overall goal of this project is to create *a simple sentence-level sentiment classifier for the Czech language that will differentiate between neutral, negative and positive sentiments.*

We will leverage the power of word embeddings that allow us to represent text data in vector space and maintain the semantics of the words and ties between them. Then, a simple neural network with two convolutional layers, some pooling layers and one fully-connected layer will be used for sentiment classification. Since we have three data sets at hand, the experiment will be conducted multiple times on different data subsets and joints.

C Metrics

Main evaluation metric for our classifier will be *categorical accuracy* computed on the testing set. Accuracy is a simple metric that calculates how much of the data has been classified correctly.

However, accuracy might also be misleading indicator of the model's performance on imbalanced datasets. *For example, if 90% of our data is labeled "positive", even naïve classifier that would always predict "positive" will get 90% accuracy.*
To alleviate such readings; and to see how well our model generalizes on all labels, we will also evaluate *weighted F_1 score*.

F_1 score is defined as the mean of *precision* and *recall*. Simply put, precision measures what percentage of positive *predictions* were correct, while recall captures how many of our positive *samples* were correctly caught by the model.

$$precision = \frac{tp}{tp + fp} \qquad recall = \frac{tp}{tp + fn} \qquad F_1 = 2\frac{precision \times recall}{precision + recall}$$

F_1 scores defined above will be calculated for each label separately and their weighted average will be taken (with weights being the number of true instances for each label). In contrast to macro F meaure that takes harmonic mean of individual scores, weighted score takes label imbalances into consideration and thus is more appropriate for our data.

II. Analysis

A Data Exploration and Visualization

As stated above, we will be using the human-annotated resources created by *Habernal et al. (2013)* for their in-depth research of using machine learning methods for sentiment analysis in of Czech social media.

There are three corpora provided:

- **Facebook:** *10 000 samples*
  with 2 587 positive, 5 174 neutral, 1 991 negative and 248 bipolar *posts*
- **ČSFD (Czech Film Database):** *91 381 samples*
  with 30 897 positive, 30 768 neutral, and 29 716 negative *movie reviews*
- **MALL.CZ (largest all-round e-commerce in Czechia):** *145 307 samples*
  with 102 977 positive, 31 943 neutral, and 10 387 negative *product reviews*

All of the data sets only contain two columns—the text itself and its label. We will engineer more features for the classification later. Bipolar instances in *Facebook* corpus will ignored for this project.

The corpora however contain entries of arbitrary lengths. Since our goal is *sentence-level* classifier, we need to process our data to obtain samples that are only one sentence long. *Figure 1* below displays variation of text data lengths in the individual data sets, split by the labels.

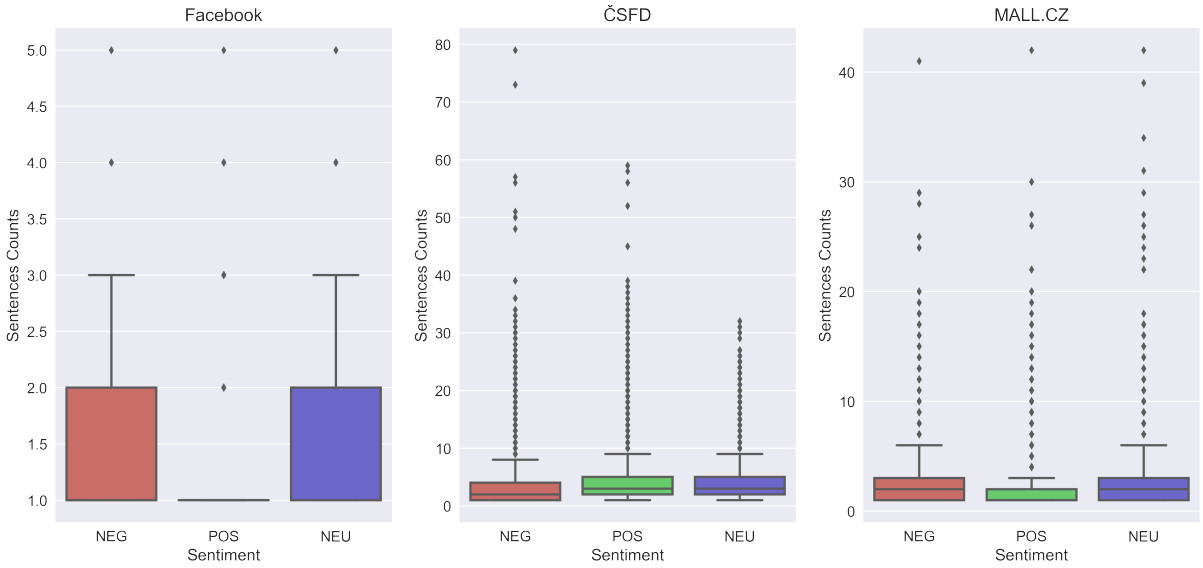

Figure 1: Analysis of posts lengths among the labels in our data sets

One way of achieving this would be to break down longer posts into individual sentences and label them with the sentiment of the origin post. However, we can't be sure the sentiments of individual sentences match the label of the whole post. Take the following review from *MALL.CZ* dataset (freely shortened and translated):

> *"The backpack is suitable for casual use, feels pleasant, fits nicely. However, the inner pocket is too small, my laptop doesn't fit in. Visually beautiful!"*

The first and the third sentences are clearly positive while the second one bears negative sentiment.

Creating three individual same-sentiment samples from entries like this one would contaminate our training data with too many badly labelled examples. Instead, we are going to simply filter out multi-sentence instances.

Figure 2 below displays the ratios of the number of multi- and single-sentence entries in our datasets.

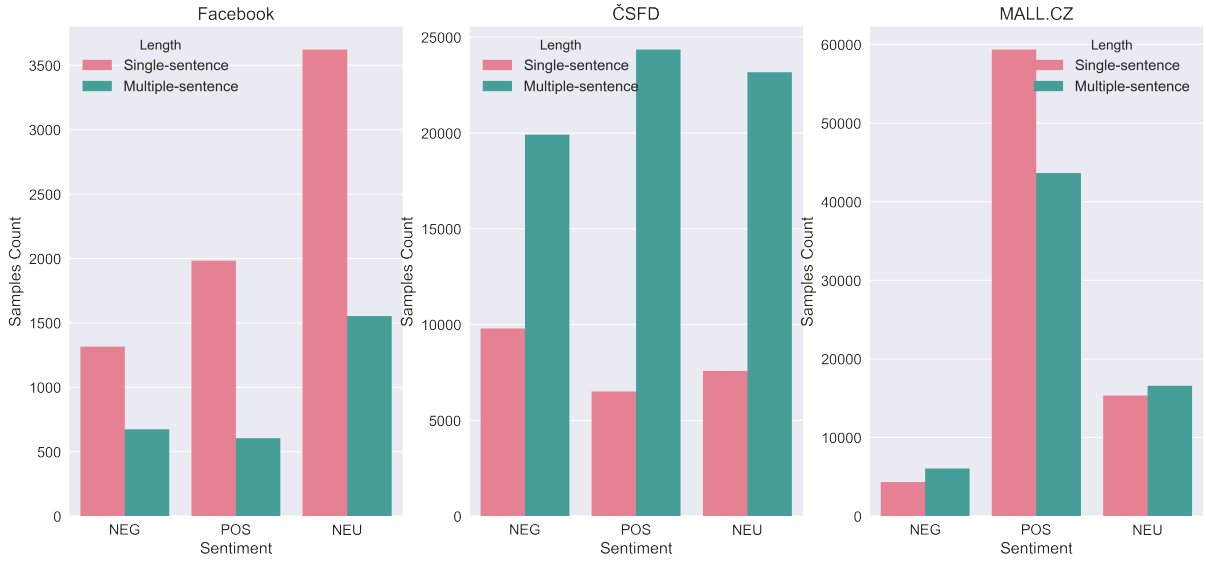

Figure 2: Number of multi- and single-sentence samples in our datasets

As you can see, filtering out multiple-sentence samples will reduce the size of our datasets rather drastically, but we are getting more appropriate samples in return.

Both figures 1 and 2 also illustrate conventions of communication within the data sets' origin channels.

Facebook is known for rather short messages and our plots prove it by showing that even outliers have maximum length of only 5 sentences. The second figure agrees on the hypothesis by showing there are at least two times more single- than multiple-sentence entries.

On the other hand, Czech Film Database (ČSFD) corpus showed there are mostly lengthy movie reviews, with extreme outliers up to nearly 80 sentences long. The convention of writing rather long film reviews is also supported by the second plot showing there are *many* more long samples than the short ones. Furthermore, we can see this holds true after visiting the website (in Czech) and checking the reviews length manually.

The last MALL.CZ data set presents more diverse results. There seems to be *about the same* amount of long and short entries for negative and neutral reviews, but a lot more short ones for the positive reviews. Also, as seen in Figure 1, the average sample length for positive label is lower than for the other two. This represents an interesting trend that can be confirmed by having a quick look at the data.
It seems that people who are satisfied with their purchase only leave generic positive comments like *"Pleasant scent"* , *"Great bargain!"* or simply *"Satisfied"* while neutral and negative reviewers tend to describe how they feel about the product in more detail.

Figure 2 also shows how imbalanced our data sets are. As discussed in *Metrics* section above, we need to be careful about huge imbalances (like in MALL.CZ corpus) and will address this issue later.

C Algorithms and Techniques

C. 1 Word Embeddings

Word embedding is a method of representing words in text data as vectors of real numbers while maintaining their semantic value. The words are represented in multidimensional vector space in a way that retains their former meaning and relationships.

Figure 3 displays an example how word embeddings can be used to extract semantic relationships between the words. The pink arrow depicts the vector mapping between the country and its capital. We can then add this vector to any other word vector to obtain some new word that is in the country→capital relationship with the former one.

We will use word embeddings for encoding our input sentences as vectors that can be used in machine learning models. This is highly appropriate for recognizing latent sentiment in a sentence because we are not losing any of its words or inter-words meanings.

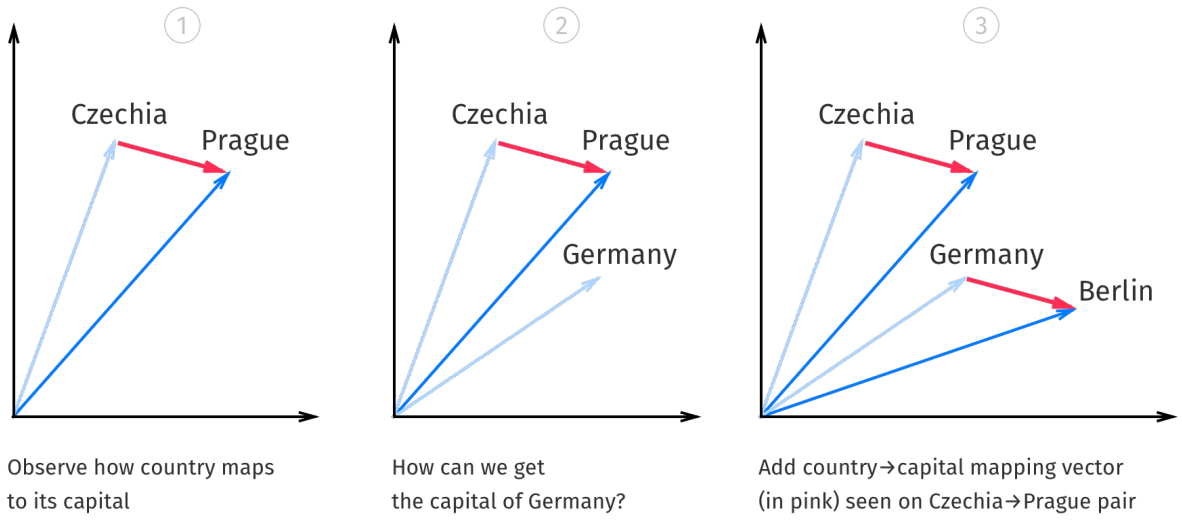

Figure 3: Two-dimensional example of how word embeddings can be used

C. 2 Convolutional Neural Networks

Convolutional neural network (CNN) is a special kind of neural network that is commonly used for feature extraction. Convolutional layer slides its filters across the input and generates map of detected features. Filters are automatically learnt during the network training and sliding them across the input enables us to pick up the hints about our sentiment label in anywhere in the input.

Although CNNs are mostly known for their use in computer vision, we will use two-layer CNN to extract features from our multi-dimensional input sentences encoded using word embeddings. Then, a single fully connected layer will be used for recognizing positive, neutral or negative sentiment from the extracted features.

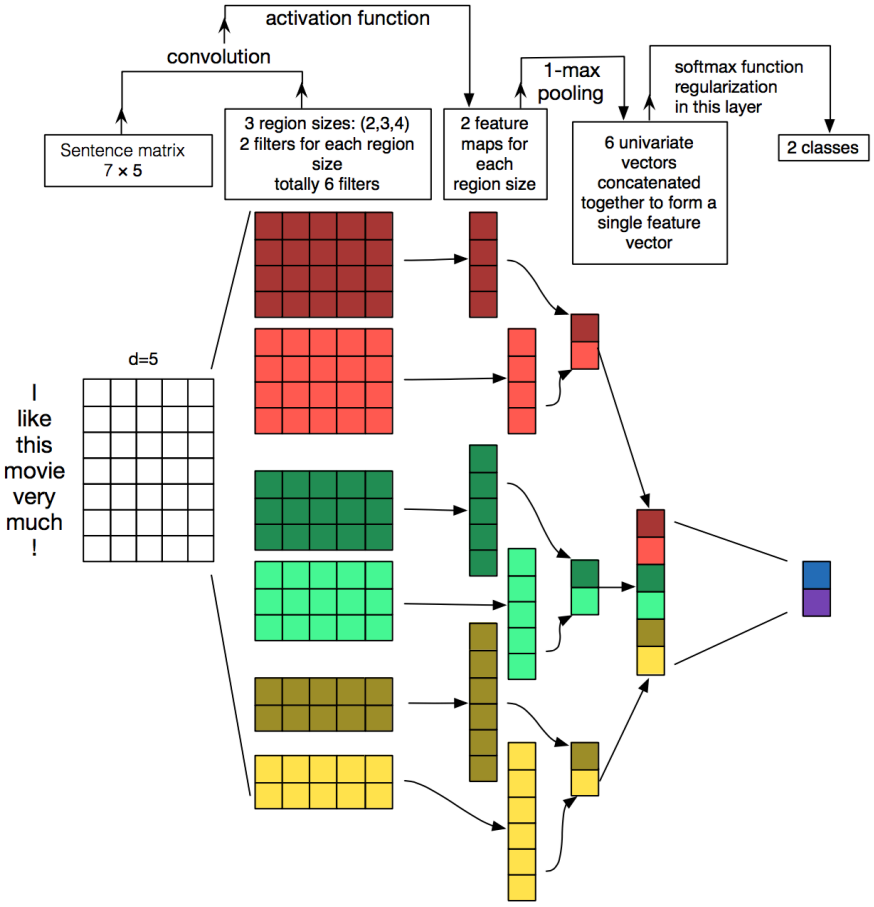

Figure 4: An example illustration of a multi-kernel CNN architecture for sentence classification. *Source: Zhang, Y., & Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.*

D Benchmark

We will compare our classifier against *weighted random guess*. In an uniformly distributed three-label dataset, random guess should get us about 33% accuracy and a similar F_1 score. Since we are going to use imbalanced data sets, it is only appropriate to use *weighted* random choice as that will give us much more robust comparison to our main imbalanced models.

However, it's important to state that sentiment analysis also has its own accuracy limits. As discussed in the research paper by *Habernal et al. (2013)*,

> *"Although the gold labels were chosen after a consensus of at least two people, there were many conflicting cases that must have been solved by a third or even a fourth person. Thus even the original annotators do not achieve 1.00 F-measure on the gold data."*

They continue by comparing the best classification model and both annotators in three confusion matrices. The machine learning model got a macro F measure of *0.69* while two annotators got *0.86* and *0.92* respectively.

Important takeaway for us is that even human beings are not capable of 100% accurate sentiment recognition in written text and we should evaluate out results accordingly.

III. Methodology

A Data Preprocessing

All of the data sets provided by *Habernal et al. (2013)* contain two columns: the text itself and its label. This section discusses the engineering of the features from the text data so that we can use it for training our neural network.

The following process matches the order of operations in the attached project's Jupyter Notebook environment:

- **Load the “raw” data sets.** Given the corpora are provided separately, they have different category labelling. We unify the labels upon loading the sets. Facebook corpus contains few posts that are labelled “bipolar”. Bipolarity is out of this project scope, so we drop these upon loading the set, too.

- **Extract single-sentence samples.** As discussed above, we only need entries that are one sentence long. Our CNN also requires certain input shape. Quick statistical query showed the average number of words in sentences is 15, so we also reject any sentences that are longer than *twice the average*. This limit was chosen manually as it seems to be the one that accommodates most scenarios while maintaining fixed-size input shape.

- **Create one joint dataset.** We want to train one of the models on all available data. For that, we create one joint dataset with balanced representation of our labels.

- **Handle imbalance in the rest corpora.** By leaving them alone. We already reduced our data sets drastically and don’t want to lose any more samples. The data imbalance will however be taken into consideration when evaluating the models

The current state is 4 Pandas DataFrames containing one-sentence text data.

- **Process the text data.** We run all of our data sets through the text processor that: replaces the emojis with happy/sad keywords; strips off punctuation, numbers and excessive whitespace; converts to lower case and applies light word stemming that reduces our vocabulary size. Stemming was chosen over lemmatization as it provides better handling of semantics. Given that Czech is morphologically rich language, stemming the words is the key data transformation for the task.

Now, we also need data for word vectors training. Because we want as much text as possible, *the three raw corpora (including all posts)* are concatenated into one temporary set; ran through the same text processor and exported in `.txt` format. Words vectors of *100 dimensions* are trained using *fastText* library using the skip-gram model described in *Bojanowski et al. (2016)*. The advantage of training our own word representation model is that it exactly shares the vocabulary in our training datasets. These 100-dimensional representation of words are the new engineered features for sentiment analysis.

- **Vectorize the text data.** Once the word vectors are ready, we can finally convert our sentences from text format to array representation. We do that by obtaining vectors for all words in a sentence and wrapping them in a new array. This creates a 2D array of shape `(no. of words)×100`—and we pad each sentence array with zeroes to match our input shape `30×100`. This is then repeated for each sentence in a corpus. The 3D array representing complete dataset has a shape of `(no. of samples)×30×100`.

- **Save the new tensors.** Since the data processing is time-consuming task, we export our fresh arrays in `.npy` format to `data` directory for the future use. In the previous step, we already split the samples and labels into separate numpy arrays, thus we are saving `X` and `y` values separately for each of the 4 data sets, resulting in eight `.npy` files in data folder.

Note that all of the saved data is still intact. They will be split into random training and testing sets upon loading them to the memory using `load_xy` function. Since we already reduced our corpora rather drastically, only 10% of the data is being withheld for testing.

B Implementation

As stated above, 4 individual models will be trained on different data sets (three original corpora and one combined). All models will share the same convolutional

neural network (ConvNet) architecture, but will be trained on different batch sizes and number of epochs.

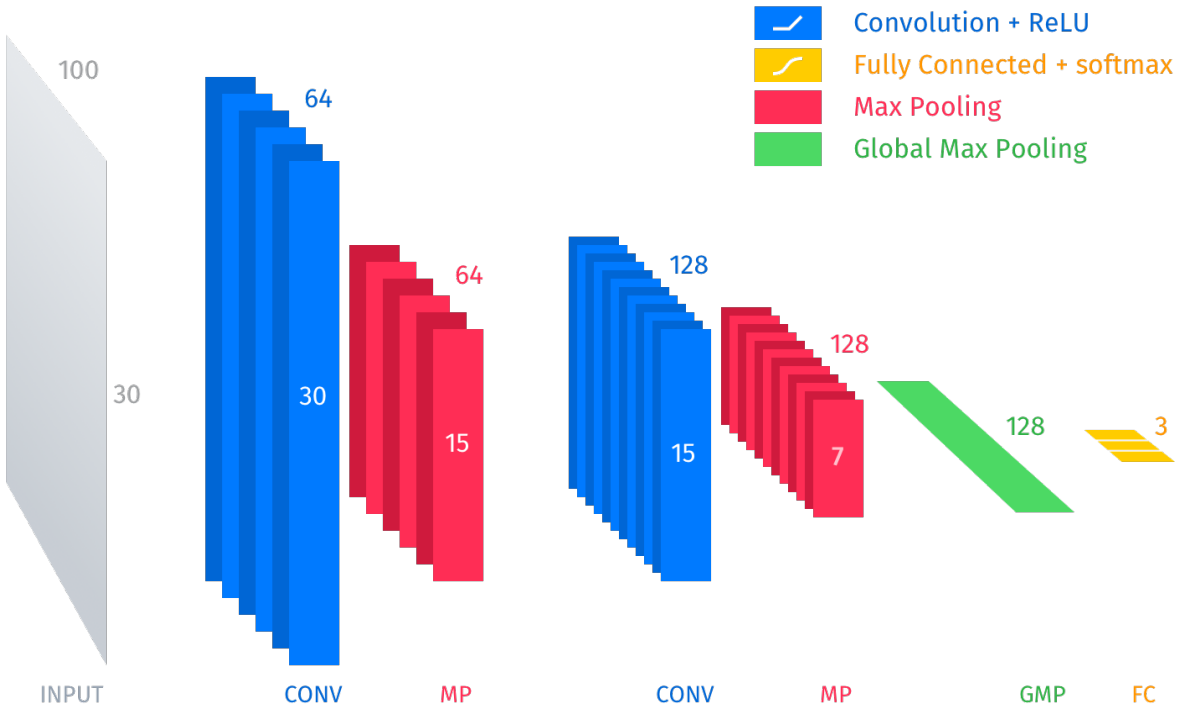

Figure 5: ConvNet architecture used for sentiment analysis in Czech

Figure 5 above displays the schema of the final architecture chosen for our task. Our normalized input has a shape of 30×100 and goes through the following:

- **1st Convolutional Layer.** This one looks for the features using kernel of size 4 and generates 64 feature maps. Activates the ouput using rectified linear unit (ReLU).
- **1st Max Pooling Layer.** Activated feature maps are down-sized to half (15).
- **2nd Convolutional Layer.** The second convolutional layer searches for features in the previous down-sized activations using kernel size 8 and generates 128 feature mappings. Activates the ouput using ReLU.

- **2nd Max Pooling Layer.** Activated feature maps are down-sized to half (7).
- **Global Max Pooling Layer.** Grabs the most *(maximally)* prominent feature from each of the 128 feature maps.
- **Fully Connected Layer.** Three-node densely connected layer identifies the sentence sentiment in extracted features and activates the output using *softmax* function to return the predicted label.

During its training, network tries to minimalize the *cross entropy loss* evaluated on the cross validation set which is obtained by withholding *15% of the training data*.

Since the architecture is shared by multiple models, we don't want to rewrite the code for each model. Thus, a simple `CNN` class is written for handling everything related to a model—from the definition to bechmark testing. The neural network inside the `CNN` class is implemented in *Keras* with *TensorFow* backend. The result models' and benchmark scores are computed by the same class instance using methods *scikit-learn's* `metrics` module.

For data pre-processing, mostly *Pandas*, *numpy* and Python built-in modules were used. Famous *NLTK* library for natural language processing provided us with the text representations of emojis used for extracting emojis sentiment; as well as with pre-trained sentence and word tokenizers used for identifying single-sentence posts. A third-party Python script for *Czech stemming* by Luís Gomes was also used in the project.

The coding process was in general straightforward. However, it soon became clear that handling differently processed versions of three data sets in memory is quite expensive. To free up the memory, explicit data deletion commands are invoked throughout the project to significantly improve the Notebook's performance.

C Refinement

The initial architecture and hyperparameters were *inspired* by Yoon Kim's research paper *CNNs for Sentence Classification (2014)*. After extensive grid search, Kim proposes single-layer architecture with *multiple* kernel sizes, *100* feature maps, global max pooling, one fully connected layer with dropout of *0.5*, batch size of *50*, L2 regularization of *3* and *Adadelta* update rule.

Although only *single* kernel size was used, the rest of the architecture followed Kim's design. Accuracy and F_1 scores obtained from this model were however close to those of our baseline naïve classifier; with the cross entropy on the validation set floating higher than *1.25*.

	Accuracy Score	F_1 Score
Inspired by Kim (suggested L2)	~45 %	~0.43
Inspired by Kim (lighter L2)	~49 %	~0.48
Inspired by Kim (removed L2)	~67 %	~0.66
Refined (added conv. layer)	~71 %	~0.71

Table 1: Comparing scores for the four of tested architectures

It turned out the low scores were caused by heavy L2 weights regularization. Setting L2 to lighter regularization (like *0.5* or *0.1*) improved the results, but not significantly. The greatest improvement happened after removing L2 altogether. The models without L2 performed much better and gained scores that were only about *5% lower than the final ones*.

Many experiments using different kernel sizes (from *2* to *10*), number of filters (from *16* do *256*) or batch sizes (*20, 50, 100, 200, 1000*) were conducted, but none of the settings improved the models' performance.

The next leap forward was made by introducing one more convolutional layer. Although many less or more extreme settings were tested, it turned out setting the first layer similarly to Kim's proposal (kernel size of 4 and little less filters, 64) and conventionally doubling the parameters in the second layer yielded the most optimal results, and climbed the models to ~70% accuracy.

Then, introducing max pooling after each convolution contributed to the overall accuracy boost with about 1%.

Despite not improving the accuracy score on average, *Adagrad* update rule was chosen over formerly proposed Adadelta. After training the models multiple times, Adagrad seems to converge to the best solution more frequently than Adadelta.

Dropout on the fully connected layer was tested with three different values—*0.25*, *0.5* and *0.75*. The last one made the training too slow (in terms of convergence) and in the result about 1% less accurate. The first two yielded about the same results as the model without dropout. Thus, the dropout was removed from the CNN design altogether.

Table 1 summarizes scores of individual models discussed in this section.

The initial data processor included stop words removal. However, it turned out Czech stop words hold semantic value that is important for sentiment analysis. Thus, this step was removed from the text processor and stop words are kept in our data.

IV. Results

A Model Evaluation and Validation

All of our models were validated on a cross validation set during their training. Since they share the same architecture, we can directly compare one to each other.

Table 2 below displays batch sizes and numbers of epochs that were used for each of the models' training, together with their gained scores.

	Facebook	ČSFD	MALL.CZ	Combined
Accuracy Score	71.62 %	71.34 %	82.52 %	67.82 %
F_1 Score	0.71	0.71	0.81	0.67
Naïve Accuracy	38.43 %	33.88 %	62.51 %	32.23 %
Naïve F_1	0.39	0.33	0.62	0.32
Cross Entropy	0.6102	0.6283	0.4495	0.7333
No. of Epochs	5	5	20	20
Batch Size	20	20	1000	1000

Table 2: Summary of our models' results.
F_1 scores for individual labels are to be found in the project's Jupyter Notebook.

A 1 Facebook Model

The result scores of this model were generally surprising, given the imbalance and size of the training data set.

It has some difficulties recognizing sentences with negative sentiment, but that is expected since only about 1 000 negative samples were present in the training set. In contrast, neutral and positive labels were represented with about 3 000 and 1 500 samples respectively.

It's also worth pointing out that the original data set contained a dictionary that is very specific to its origin channel (Facebook) and the model doesn't respond well to more general language.

A 2 ČSFD (Czech Film Database) Model

This model was trained on the most reduced data set. From promising 90 thousand samples in the raw corpus, only 19 000 sentences were fit for our use.

Despite that, we were still able to get a competitive score, a reasonable cross entropy loss and *the best spread* between the benchmark and trained model.

A 3 MALL.CZ Model

The model with the highest scores. This is however only an illusion described in *Metrics* section in the beginning of this report. The MALL.CZ data set is extremely imbalanced, with *56 712 positive, 13 959 neutral and only 3 682 negative samples*. Although we're getting amazing overall score, looking at the F_1 scores for individual labels reveals that the negative class is getting only *0.55*, neutral *0.48*, but positive having insanely high score of *0.90*.

However, thanks to the weighted nature of our benchmark, we can see that even the naïve classifier is getting high accuracy for such imbalanced corpus. When we compare the result to the baseline, the CNN model is only *1.3 times* better than benchmark while the rest of our models are on average *2 times* better.

Thus, this model is generally the worst of all. On the other hand, it shows that CNNs are capable of learning accurate representations of sentiment when given a lot of samples.

A 4 Combined Model

This was a test of creating as general model as possible. It also has the *lowest scores* and *highest cross entropy*. However, this makes sense as we are *forcing* the network to generalize by feeding much more diverse samples from all datasets. The lowest scores tell us that if we want truly generalizing model (the one that acts well on all kinds of vocabularies), we would need to obtain *much bigger* corpus of

such general text examples in order for the network to learn all of different features.

B Justification

As discussed in the *Benchmark* section of this report, the sentiment analysis has it upper accuracy limits. The in-depth research paper on Czech sentiment analysis by *Habernal et al. (2013)* compares the performance of two human data annotators to the final "gold" labels chosen after agreement of two, three or even four people. It turns out sentiment analysis is highly subjective and even the human annotators on average received a F score of *0.9*.

If we exclude the highly imbalanced *MALL.CZ* model, the average F_1 score of our naïve classifiers is *~0.35* while the average F measure of our models is *~0.70*.

Our models do perform twice as better as their baseline naïve classifiers, but will still *misclassify one in the three* examples. That makes the models not much useful for fully automated tasks such as opinion mining in huge corpora or recognizing sentiment in chatbot conversations for communication adjustments. However, they might find some use in an assistive applications where the predicted sentiment will later be subject to human evaluation as well.

The infamous *MALL.CZ model* also shows that accurate sentiment analysis in Czech is possible when given enough of training examples. This might be a nice takeaway for further investigation of supervised methods for Czech sentiment analysis.

Until then, the problem stays only partially solved.

V. Conclusion

A Free-Form Visualization

Figure 6 displays four new fabricated questions and visualizes their predictions using our 4 models. Freely translated English versions are attached for reference.

	Facebook	ČSFD	MALL.CZ	Combined
Rozbila se po prvním použití, je na hovno! It broke after the first use, it's shitty!				
Rozbila se až za rok It broke after a year of use				
S manželem jsme si víkend moc užili Me and my husband enjoyed the weekend				
Ok, ale nic zajímavého Ok, but nothing interesting				

Figure 6: Four example sentences and their predictions from our models.
Red circle = negative sentiment; green circle = positive; blue circle = neutral.

Note how the second sentence was chosen in contrast to the first one. It also illustrates the mentioned upper accuracy limit of sentiment analysis. While some might see it as clearly neutral, it does have indications of negativity and might cause different labelling among humans.

We can see the *Facebook model* predicted all sentences correctly. Although it was said that this model doesn't perform well on general language, one must keep in mind it shines at predicting neutral labels which are overrepresented in our table.

ČSFD model got only 1 sentence out of 4 wrong, and it was the conflicting one. Looking at individual label F_1 scores does show the poorest performance on neutral sentences, so this classification makes sense.

MALL.CZ clearly showed why it is the poorest model despite its high scores. Since it was trained on data with too many positive samples, the network learned the assumption that most of sentences in the real world are positive and thus have classification rules skewed towards positive label.

Combined model is the same scenario as *ČSFD*. The conflicting neutral/negative sentence was classified as negative since it performs significantly better on this label.

B Reflection

We can summarize our sentiment analysis attempt in the following steps:

1. Obtaining the data sets for rather rare project in the Czech language
2. Making decisions about the provided data and project's goal
3. Data preprocessing and interpretting text data in vector space
4. Finding the most optimal CNN architecture for multiple models *(manually since a systematic grid search would be computationally too expensive)*
5. Training and evaluating multiple models

The most difficult and time-consuming part of this project was making the decisions about our data and its transformations. Maybe the goal of making a *sentence-level* classifier was not well aligned with provided corpora full of mixed-length entries. We had to manually select the most suitable ones and balance between their relevancy and amount.

Having said that, the whole project was an extraordinary introduction to solving machine learning problems in the real world using real messy and limited data.

C Improvement

There is one huge improvement that could be made—*get more of relevant data*. During the experimentation phase, it became clear how the amount and type of data used for training influences the final results of our models. Going for post- or review-level classifier would probably be more appropriate given the provided corpora. Furthermore, it seems that all models struggle with learning the representation of neutral sentiment, so obtaining extra neutral samples might help. It would most probably be worth trying out a classifier with only two, negative and positive, target labels as well.

Although we were able to reach the scores from the research on sentiment analysis in Czech by *Habernal et al. (2013)*, there clearly have to be better solutions. Maybe they just weren't researched as deeply as solutions for mainstream languages.

Another possible improvement would be to use completely different approach, given the data entries for such task usually will be multiple sentences long. For example, a recent research by *Radford et al. (2017)* explores unsupervised methods that can learn sentiment representations on character level with an excellent accuracy. Although this is computationally very expensive approach, it yields very robust solution—both precision and flexibility wise.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomáš Mikolov. 2016.
Enriching Word Vectors with Subword Information.

Tomáš Brychcín and Miloslav Konopík. 2015.
HPS: High precision stemmer.

Yoav Goldberg. 2015.
A Primer on Neural Network Models for Natural Language Processing.

Ivan Habernal, Tomáš Ptáček and Josef Steinberger. 2013.
Sentiment Analysis in Czech Social Media Using Supervised Machine Learning.
In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis,* pages 65—74. Atlanta, Georgia. 14 June 2013. Association for Computational Linguistics.

Yoon Kim. 2014.
Convolutional Neural Networks for Sentence Classification.

Alec Radford, Rafal Jozefowicz and Ilya Sutskever. 2017.
Learning to Generate Reviews and Discovering Sentiment.