Customer mail categorization

Maroš Dubíny, 445408

1. Introduction

This project focuses on categorization of incoming emails to Kiwi.com to selected categories, which can be redefined at will. For this I will use different approaches of Machine Learning, mainly LSTM and pre-learned transformers.

2. State of the art

Currently this task is done by so called “Front office” agents to send mail to correct departments, in case it is not so simple so they can respond themselves. This categorization aims to ease load on these agents and at least partially do categorization by itself. Later it can be connected to our SmartFAQ system to respond to some of the incoming mail in fully autonomous way.

3. Implementation

3.1 File structure

3.1.1 Base directory

This directory contains run.py file that tests most successful network on test inputs and prints results. You can find there requirements.txt which can be processed by PIP.

3.1.2. Preprocessing

This Directory contains python scripts which wer used in pre-processing of data from raw database selects dumped to JSON dump files. They result in files later used in categorization. Files that are connected to unsuccessful embedding/LSTM approach are prefixed with `embedding_` prefix and are left there for more/less documentation purposes

3.1.3. Categorization

This Directory contains files used for categorization by ML techniques. Again, files that are connected to embedding/LSTM approach are prefixed with `embedding_` prefix. Here lies saves directory as well with all results mentioned later.

3.2. LSTM

Last semester I have explored a possibility to use LSTM to categorize e-mail. This did not lead to a lot of success with results not too different from random result generation. This may have been caused by factors like:

- Too shallow networks
- Impure data
- My limited knowledge of LSTM

This approach was abandoned for current semester in favor of `simpletransformers` library. Decision like this was made for their good general architecture, pre-trained models and they are generally advised to be used in NLP community as well as by my tutor.

3.3. Transformers

This semester I have decided to use `simpletransformers` library from https://simpletransformers.ai/ which is based on https://github.com/huggingface/transformers. For this I have used pre-trained models namely:

- BERT – github , used bert-base-cased
- ALBERT – github , used albert-base-v2
- RoBERTa – github, used roberta-base

4. DATA

From this point on,I will be talking only about transformers approach. Data used in learning and testing were in 3 categories. Google news test set, small set of emails from Kiwi.com and bigger set.

4.1. Google news

Not much to be said about this data set. It is common data set for testing NLP approaches. Files are located in data folder as test.csv and train.csv . It was used just to test architecture.

4.2. Emails – small set

This set was used to check data quality and they are emails that are very high quality from a specific email queue. After filtration resulting set consists of around 5 000 emails.

4.3. Emails – big set

This is proper learning set consisting of last 200 000 incoming emails to kiwi.com after filtration – removing of pure html or just links and spam, non-English emails and formatting., resulting set is something above 80 000 emails which can be considered as solid data set.

5. Results

5.1. Google news

Again, not a point of our interest. A well defined data set resulting in mcc approximately 0.9

5.2. Emails – small set

This set was very carefully picked, however I was not able to reach considerable accuracy with it.

BERT: eval_loss = 3.0813085388492896, mcc = 0.2838441260197735

ALBERT: eval_loss = 2.481212126242148, mcc = 0.14703977772385585

RoBERTa: eval_loss = 1.822548424875414, mcc = 0.2535355643253822

0.28 mcc could be considered not random but it is way too low anyway.

5.3. Emails – big set

Only difference here was increase in the data set side by approximately 16 times. This seems to be sufficient and was able to reach in my opinion good results considering purity of data.

BERT: eval_loss = eval_loss = 0.6751383292331066, mcc = 0.5577036861010718

ALBERT: eval_loss = eval_loss = 0.6344813069429349, mcc = 0.5866294300167191

RoBERTa: eval_loss = 0.6446254906053573, mcc = 0.5732704119143343

BERT 10 epochs = eval_loss = 1.1268075010900287, mcc = 0.5791220151516601

As we can see adding additional 10 epochs did not help much.

6. Manual testing

For results ~0.6 mcc is in my opinion mainly responsible that the test data used contains a lot of categorization errors. My estimation is that little less than half of the labels in data is wrong so this is a great result. I have tried to do the testing manually by run.py file in base directory. Output of this script with BERT model fully matches expectations , however I am sure I could find a case where it works incorrectly. Overall I would say that this classification is at least as thorough as average agent.

Script output:

Predicting for: Dear partner, Greetings from Airline. Regarding your order 1234 for ticket submitted is not refundable. That means there is no money can be refunded.

result = 1, expected = 1.

Predicting for: Dear Colleagues, Please, check the status of the booking number 1234

result = 2, expected = 2.

Predicting for: Dear sir or madam, could you please check price for baggage and priority boarding on flight 1234.

result = 3, expected = 3.

Predicting for: Please use 2 eggs and milk for pancakes.

result = 4, expected = 4.

7. Takaways

For me this was a great learning experience and this are my takeaways:

1) Working with real life data is much harder than with prepared data set. This data is wrongly labeled and without huge manual effort it is hard to do it properly.

2) Working with restricted data is hard as they are very hard to access, acquiring them takes long for all the approvals, are not possible to present outside of their restriction and it is not possible to process them on external hardware.

3 It is hard to combine School and Work as school process takes longer and high priority tasks will not get delegated to be properly documented as school requires. On the other hand Low prio tasks are hard to push in company as nobody cares and there is no product push.

4) As consequence 90, if not 95% of the work was spent on acquiring data not ML.