Machine Learning Engineer Nanodegree

Capstone Project

Yelp Open Dataset Sentiment Analysis: supervised and unsupervised approach
Maksim Lebedev

December, 2018

I. Definition

Project Overview

Yelp's website, Yelp.com, is a crowd-sourced local business review and social networking site. Its user community is primarily active in major metropolitan areas The site has pages devoted to individual locations, such as restaurants or schools, where Yelp users can submit a review of their products or services using a one to five star rating system. Yelp Dataset is subset of businesses, reviews and user data.

Reviews and comments are one of the most important data sources that business has, especially in a customer-centric, digital market. There are lots of insights that we can get from these information - it could be sentiments, topics extraction, as well as clustering and categorization.

Sentiment analysis of the comments that are represented in Yelp dataset is the goal of this project.

This is a natural language processing project.

Problem Statement

Comments or reviews, which people are leaving in the Internet are really important, especially now, when we live in a digital era. Lots of companies are not using this source of information, some - because they do not think about it, some because they do not have resources. Some companies are still doing that “old” way - when people are reading comments and categorizing them, identifying sentiment as well, which makes this process slow, not giving a business an opportunity to react fast or anticipate some changes in client’s behaviour.

Those companies that are doing this analysis - sometimes spending time and money to prepare datasets and label this datasets. Most successful machine-learning projects are using supervised-learning algorithms, however there are opportunities to use unsupervised-learning algorithms to help addressing some of the issues.

Having labeled dataset is very good, but in most of the companies, it is not the case so, in this project I want to compare usage of supervised learning algorithms with unsupervised learning (or semi-unsupervised).

For supervised learning - I will be using word2vec or doc2vec and then compare several algorithms: naive bayes, logistic regression and neural networks.
For unsupervised sentiment analysis: use word2vec or doc2vec together with SentiWordNet, also look at OpenAI method described here (https://github.com/openai/generating-reviews-discovering-sentiment)

Metrics

As Yelp dataset contains sentiment they will be used to calculate accuracy. For the simplicity reviews with rating higher or equal to 4 - will be positive, 3 - neutral, less than 3 - negative. For supervised learning I will split dataset into three - training, test and validation. Precision,recall and F1 score would be used for model evaluation as well

For unsupervised part: it will be accuracy for the sentiment.

As it is a comparison exercise: another metric is time spent to both, as one of the goal of this project is to understand - what could be done with unsupervised learning, and how small businesses could benefit from it. I want to evaluate how much time is needed for unsupervised learning versus supervised, where, in supervised - dataset preparation should be considered as well.

II. Analysis

Data Exploration

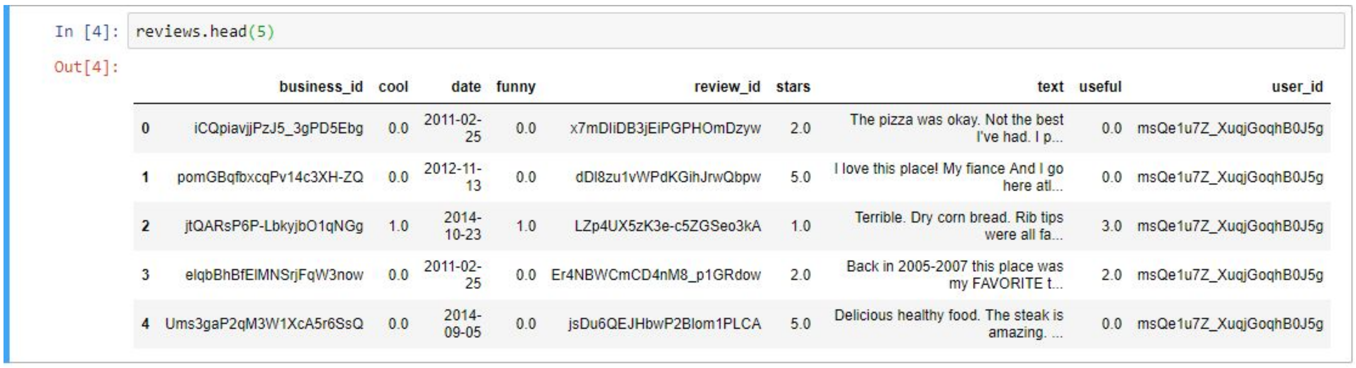
Dataset that is used for this project could be found on https://www.yelp.com/dataset. As mentioned above it's provided by Yelp and contains businesses reviews.

The main file is review.json. Contains full review text data including the user_id that wrote the review and the business_id the review is written for. There are more than 5 million reviews stored in the file.

Reviews dataset contains following columns:

- buisiness_id: unique business id;
- cool: 1 or 0 - flag whether the comment is cool;
- date: date when review was left;
- funny: flag whether the comment is funny 1 or 0;
- review_id: unique review id;
- stars: number of stars that user left for the business;
- useful: notification of whether the comment is useful or not

- user_id: unique user id code

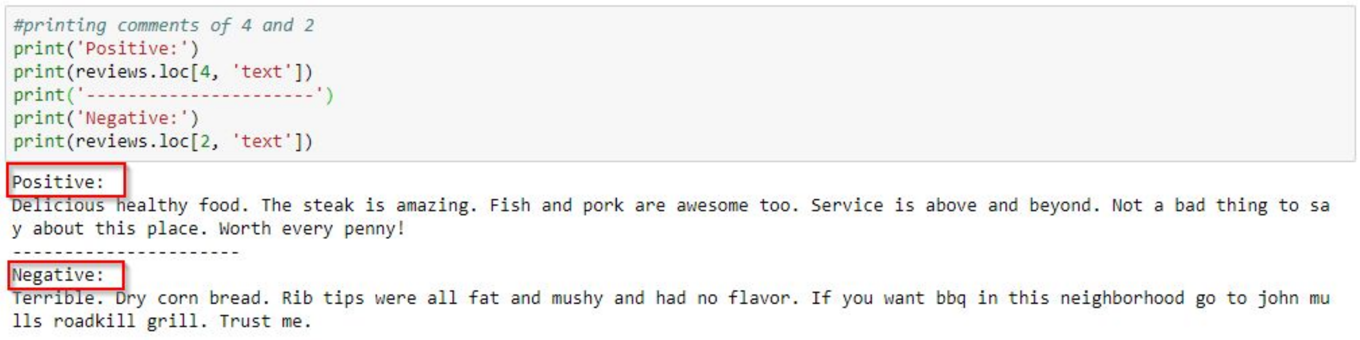


```
In [4]: reviews.head(5)
Out[4]:
```

	business_id	cool	date	funny	review_id	stars	text	useful	user_id
0	iCQpiavjjPzJ5_3gPD5Ebg	0.0	2011-02-25	0.0	x7mDIiDB3jEiPGPHOmDzyw	2.0	The pizza was okay. Not the best I've had. I p...	0.0	msQe1u7Z_XuqjGoqhB0J5g
1	pomGBqfbxcqPv14c3XH-ZQ	0.0	2012-11-13	0.0	dDl8zu1vWPdKGihJrwQbpw	5.0	I love this place! My fiance And I go here atl...	0.0	msQe1u7Z_XuqjGoqhB0J5g
2	jtQARsP6P-LbkyjbO1qNGg	1.0	2014-10-23	1.0	LZp4UX5zK3e-c5ZGSeo3kA	1.0	Terrible. Dry corn bread. Rib tips were all fa...	3.0	msQe1u7Z_XuqjGoqhB0J5g
3	elqbBhBfElMNSrjFqW3now	0.0	2011-02-25	0.0	Er4NBWCmCD4nM8_p1GRdow	2.0	Back in 2005-2007 this place was my FAVORITE t...	2.0	msQe1u7Z_XuqjGoqhB0J5g
4	Ums3gaP2qM3W1XcA5r6SsQ	0.0	2014-09-05	0.0	jsDu6QEJHbwP2Blom1PLCA	5.0	Delicious healthy food. The steak is amazing. ...	0.0	msQe1u7Z_XuqjGoqhB0J5g

Figure 1. Example of dataset

For project purposes only two columns: text and stars will be useful. "text" column contains comments and "stars" would be using the identify the sentiment. To see some concrete examples, I will take rows with index 4 and 2. The first one is a positive sentiment and second is negative.

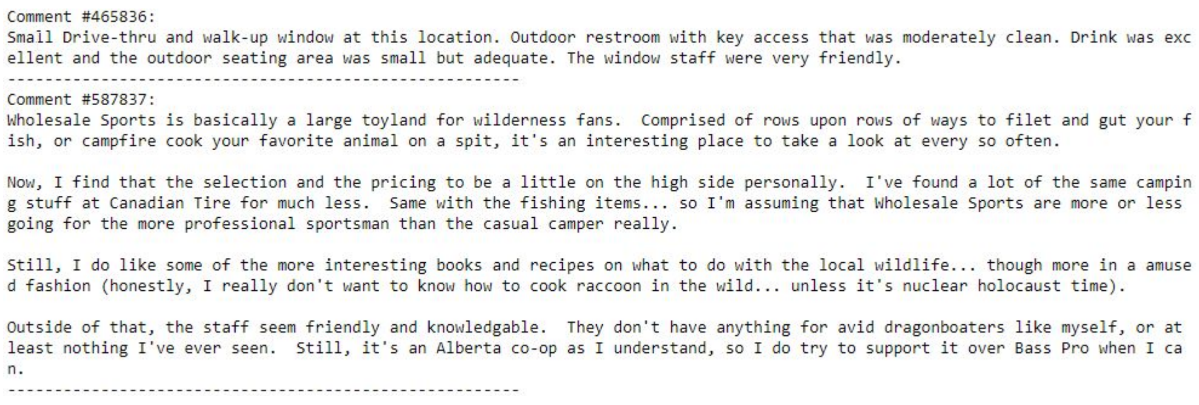


```
#printing comments of 4 and 2
print('Positive:')
print(reviews.loc[4, 'text'])
print('----------------------')
print('Negative:')
print(reviews.loc[2, 'text'])
```

```
Positive:
Delicious healthy food. The steak is amazing. Fish and pork are awesome too. Service is above and beyond. Not a bad thing to sa
y about this place. Worth every penny!
----------------------
Negative:
Terrible. Dry corn bread. Rib tips were all fat and mushy and had no flavor. If you want bbq in this neighborhood go to john mu
lls roadkill grill. Trust me.
```

Figure 2. Positive and Negative comments

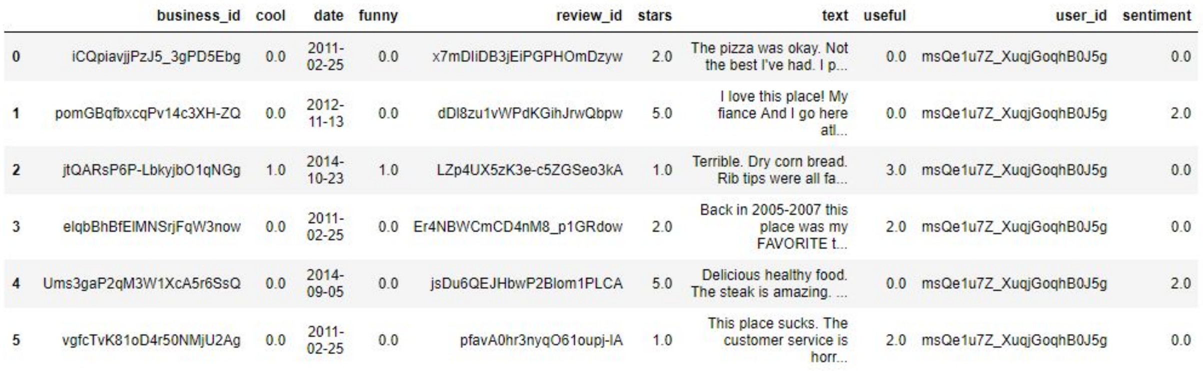
For simplification purposes, comments with stars number higher than 4 will be considered as positive, comments with stars value equals or less than 2 - are negative. After evaluation of comments with 3 stars - it was decided that they should be in category neutral. Several comments with 3 stars are presented below:



```
Comment #465836:
Small Drive-thru and walk-up window at this location. Outdoor restroom with key access that was moderately clean. Drink was exc
ellent and the outdoor seating area was small but adequate. The window staff were very friendly.
-----------------------------------------------------
Comment #587837:
Wholesale Sports is basically a large toyland for wilderness fans.  Comprised of rows upon rows of ways to filet and gut your f
ish, or campfire cook your favorite animal on a spit, it's an interesting place to take a look at every so often.

Now, I find that the selection and the pricing to be a little on the high side personally.  I've found a lot of the same campin
g stuff at Canadian Tire for much less.  Same with the fishing items... so I'm assuming that Wholesale Sports are more or less
going for the more professional sportsman than the casual camper really.

Still, I do like some of the more interesting books and recipes on what to do with the local wildlife... though more in a amuse
d fashion (honestly, I really don't want to know how to cook raccoon in the wild... unless it's nuclear holocaust time).

Outside of that, the staff seem friendly and knowledgable.  They don't have anything for avid dragonboaters like myself, or at
least nothing I've ever seen.  Still, it's an Alberta co-op as I understand, so I do try to support it over Bass Pro when I ca
n.
-----------------------------------------------------
```

Figure 3. Example of neutral comment

To differentiate sentiment - a new column has been created, which is using ‘stars’ column input to map it to sentiment. 0 - is a negative sentiment, 1 - neutral sentiment, 2 - positive sentiment.



	business_id	cool	date	funny	review_id	stars	text	useful	user_id	sentiment
0	iCQpiavjjPzJ5_3gPD5Ebg	0.0	2011-02-25	0.0	x7mDIiDB3jEiPGPHOmDzyw	2.0	The pizza was okay. Not the best I've had. I p...	0.0	msQe1u7Z_XuqjGoqhB0J5g	0.0
1	pomGBqfbxcqPv14c3XH-ZQ	0.0	2012-11-13	0.0	dDl8zu1vWPdKGihJrwQbpw	5.0	I love this place! My fiance And I go here atl...	0.0	msQe1u7Z_XuqjGoqhB0J5g	2.0
2	jtQARsP6P-LbkyjbO1qNGg	1.0	2014-10-23	1.0	LZp4UX5zK3e-c5ZGSeo3kA	1.0	Terrible. Dry corn bread. Rib tips were all fa...	3.0	msQe1u7Z_XuqjGoqhB0J5g	0.0
3	elqbBhBfElMNSrjFqW3now	0.0	2011-02-25	0.0	Er4NBWCmCD4nM8_p1GRdow	2.0	Back in 2005-2007 this place was my FAVORITE t...	2.0	msQe1u7Z_XuqjGoqhB0J5g	0.0
4	Ums3gaP2qM3W1XcA5r6SsQ	0.0	2014-09-05	0.0	jsDu6QEJHbwP2Blom1PLCA	5.0	Delicious healthy food. The steak is amazing. ...	0.0	msQe1u7Z_XuqjGoqhB0J5g	2.0
5	vgfcTvK81oD4r50NMjU2Ag	0.0	2011-02-25	0.0	pfavA0hr3nyqO61oupj-IA	1.0	This place sucks. The customer service is horr...	2.0	msQe1u7Z_XuqjGoqhB0J5g	0.0

Figure 4. Dataset with sentiment column

For the final dataset - we leave only review_id, in case we need to understand what is the row that we work with, text and stars.

From the dataset was remove one row, which contained no comment and moreover was starred as a 0.

For the rest of the comments - there were no anomalies.

Exploratory Visualization

Before starting dataset should be splitted to three different datasets: training, validation and test set.

However, during the exploratory analysis - it’s been identified that around 65% of the comments are positive. This means that algorithm will be biased if leave it as is. This is shown on the histogram below.

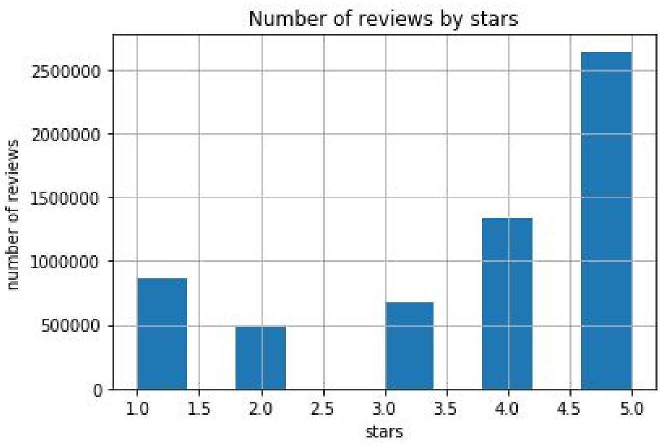


Figure 5. Histogram - number of reviews by stars

In total we have around 4 mlns of positive comments, ~700k of neutral comments and almost 1 million and a half of negative. To prepare a good training set we should leave it proportionally. It means that some part of positive comments should be left outside of the dataset, it could be then used as well to test the algorithm.

Before doing this, we also need to identify, if there is any correlation between the length of the review and number of stars given.

Following histogram is showing number of reviews by length of the comment for the whole dataset and for only positive comments.

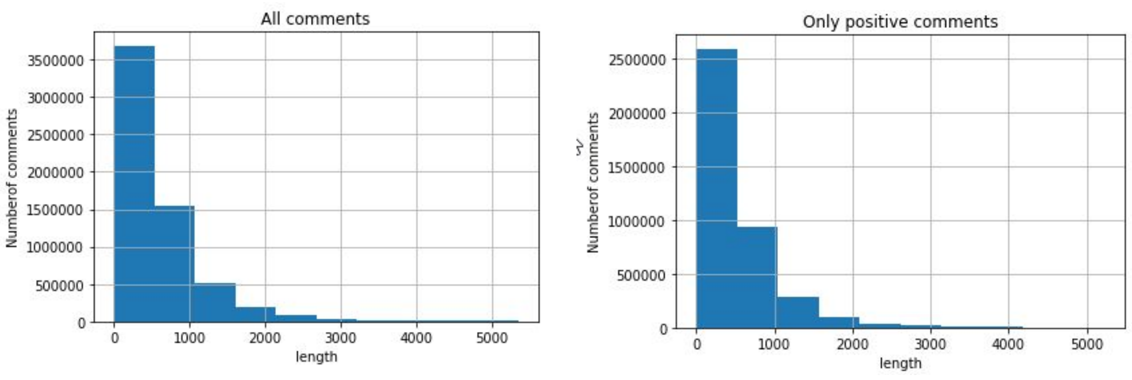
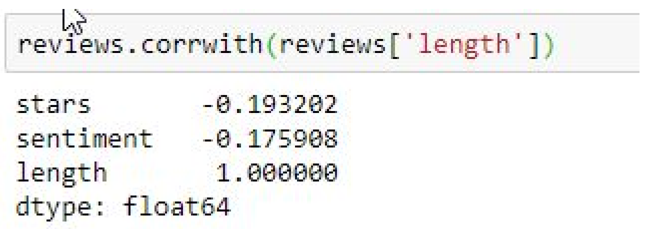


Figure 6. Number of reviews by length of the comment

As shown on the histogram, positive comments tend to have the same distribution of length, as the whole dataset. To make sure there is no dependency:

1. Let's compute pairwise correlation of columns:



```
reviews.corrwith(reviews['length'])
```

```
stars        -0.193202
sentiment    -0.175908
length        1.000000
dtype: float64
```

2. Let's build a stacked bar chart with all five categories, which will be divided by bins beforehand. Each bin with the step of 500 symbols

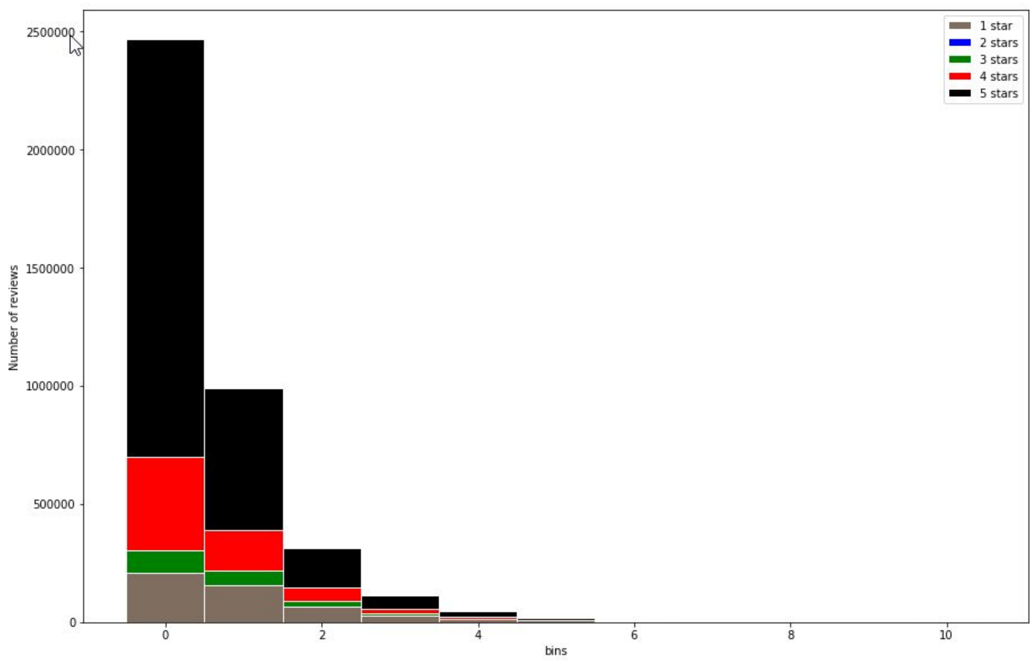


Figure 7. Stacked bar chart by stars and length

This bar chart was built to identify if there is any tendency or correlation between length and number of stars. There is no clear correlation, but **interestingly** that still when comment is long - it's more likely that it's either 1 or 5 stars. Also there seem to be very few comments with 2 stars.

Algorithms and Techniques

There are two main task for this capstone project: use supervised and unsupervised learning algorithms to do the sentiment analysis.

For supervised learning Naive Bayes, Logistic Regression, Random Trees and ensembling with a voting classifier are used to achieve a base line. Text of each sentence will be tokenized, and then algorithms will be fit by them. For tokenization - spacy package will be used.

After word2vec model will be built to train a neural network. There will be two vector sizes 100 and 120 tried. Convolutional neural network will be used for this. Keras will be used for building a neural network. Conv1d and dense layers will be used, in order to avoid overfitting - dropout and early stopping will be added.
For activation functions: relu, elu and tanh or their combinations will be tried.
Categorical cross entropy - will be used as loss function, optimizer - Adam. And last layer of CNN will be a softmax, as we have three different labels.

For unsupervised part - tokens that were gotten using spacy will be used. Before starting with sentiwordnet - part-of-speech tags should be assigned for all the tokens in each of the sentences. Nltk.pos_tag will be used for this. Tags then should be converted to wordnet tags format, for instance wn.ADJ == JJ. So this should be converted before sentiment calculation.

Benchmark

Best result of “simple algorithms” will be used as a benchmark for both CNN and unsupervised part. Expecting that these results will be giving around 70% of accuracy on test subset.

However, aim for the CNN is to achieve 80% of accuracy on test set.

III. Methodology

Data Preprocessing

Data preprocessing part is very important for any of the machine learning projects. NLP projects are not exception.

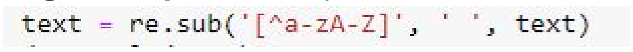
As described in data analysis, dataset contains 65% of positive comments. It is very important that negative and neutral reviews will be represented in the dataset.

To make sure that performance of the system will not be an issue, from 6 millions of rows, it was decided to leave only 150000, this is due to performance issues.. Each type of sentiment is equally represented in dataset (50001 x 3).

Then scikit-learn library will be used for splitting datasets to three subsets. Train, validation and test subsets. Proportion will be 60-20-20.

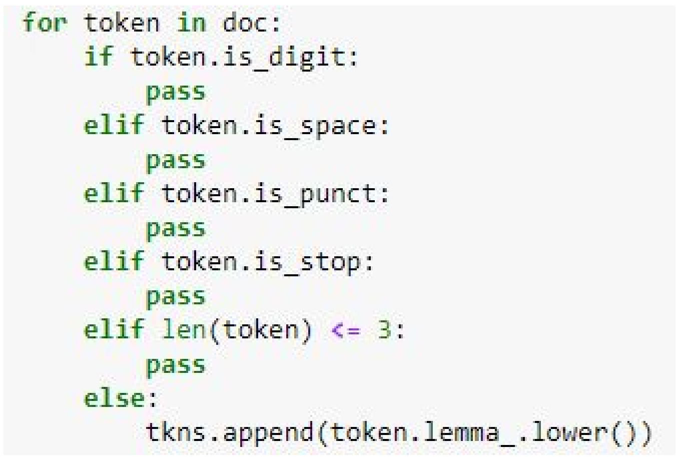
For text preprocessing will be used:

- **Regular expression** operators to clean the text from anything but text



```
text = re.sub('[^a-zA-Z]', ' ', text)
```

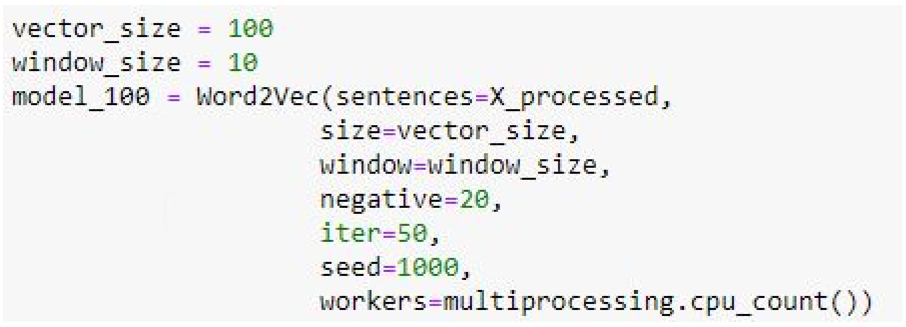
- **Spacy**: to remove numbers, to remove spaces, remove punctuation, check whether word is in the stopwords and also identifying the lemma of the word. Also only tokens longer than three symbols will be considered.



```
for token in doc:
    if token.is_digit:
        pass
    elif token.is_space:
        pass
    elif token.is_punct:
        pass
    elif token.is_stop:
        pass
    elif len(token) <= 3:
        pass
    else:
        tkns.append(token.lemma_.lower())
```

Lemmatization, and not stemming, was chosen as it supposes to “do things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the *lemma*”. (https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html)

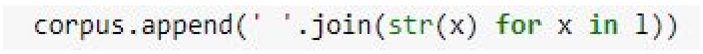
After above mentioned actions text will be represented as list of token’s lists. It will be ready for building word2vec model with **gensim** module of python. For this gensim.Word2Vec will be used.



```
vector_size = 100
window_size = 10
model_100 = Word2Vec(sentences=X_processed,
                     size=vector_size,
                     window=window_size,
                     negative=20,
                     iter=50,
                     seed=1000,
                     workers=multiprocessing.cpu_count())
```

Figure 8. gensim.word2vec

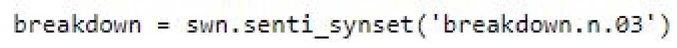
However, due to the fact that Naive Bayes model does not work with word2vec - bag of words model is needed as well. For this **nltk** package is used. Tokens will be merged together to the string, and corpus will be formed. List of strings of tokens separated by spaces.



```
corpus.append(' '.join(str(x) for x in l))
```

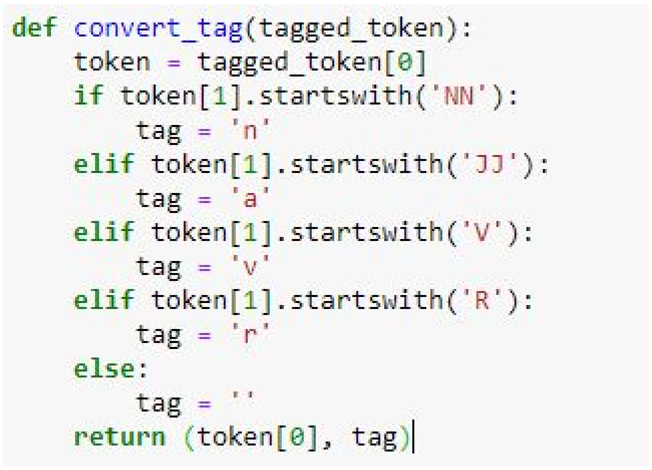
After this both word2vec and BoW models could be built.

For the unsupervised part - tokens from the first part will be used. However, as it’s planned to work SentiWordNet - part-of-speech tagging for all the tokens should be implemented first.



```
breakdown = swn.senti_synset('breakdown.n.03')
```

From an example above, could be noted that word should be represented as tuple: (‘word’, ‘pos’, usability). Nltk.pos_tag will be used for this operation all the tags should be converted to wordnet type. Convert tag function has been created for this purpose:



```python
def convert_tag(tagged_token):
    token = tagged_token[0]
    if token[1].startswith('NN'):
        tag = 'n'
    elif token[1].startswith('JJ'):
        tag = 'a'
    elif token[1].startswith('V'):
        tag = 'v'
    elif token[1].startswith('R'):
        tag = 'r'
    else:
        tag = ''
    return (token[0], tag)
```

Figure 9.pos tagging

After above-listed operations, we could start with implementation of supervised and unsupervised parts.

Implementation

Initially it's been planned to use the whole dataset. However due to several reasons the final solution contains only 150k of comments:

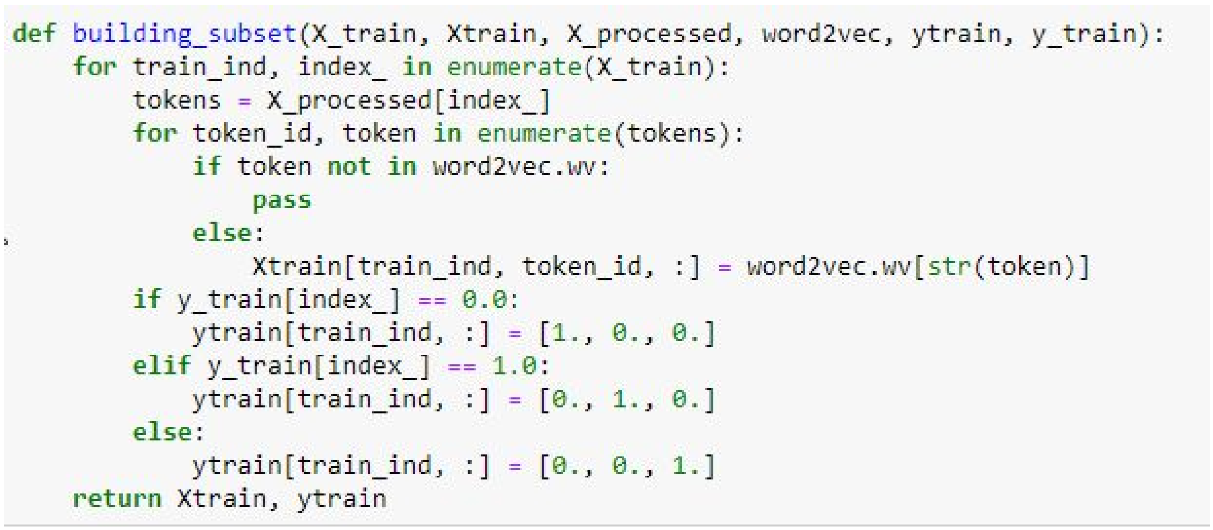
1. Positive comments are dominating the dataset, so to make all the labels represented equally - only part of it, around 1,5 mlns could be used;
2. Due to memory issues, it's not possible to prepare an array 900000, 547, 150 for training the neural network. 900000 - is the size of training set, 547 is the maximum number of tokens in one comments, 150 - is a vector size for word2vec.
3. After few experiments - it was decided to leave 150k of comments, and then after training the neural network - check how it works for those comments which are left behind.
4. Two types of word2vec vectors has been created 100 and 120 words - to check what is the impact on the neural network.

The most difficult part was a dataset preparation. For having a labels being represented equally in all of the subsets, initially I was creating a subsets split by positive, negative and neutral labels.

As an array size that could be used without having a MemoryError - is about 90001, 80, 120 - I was filtering comments that coming to those subsets. Their length should have been equal or less than 100 tokens.

After all the three subset were merged together, and for all of them I run the tokenizer that has been described in Data Preprocessing section. The process of cleansing was the longest one - and took around 1 hour (initially for 1,5 mlns of comments it took 10 hours).

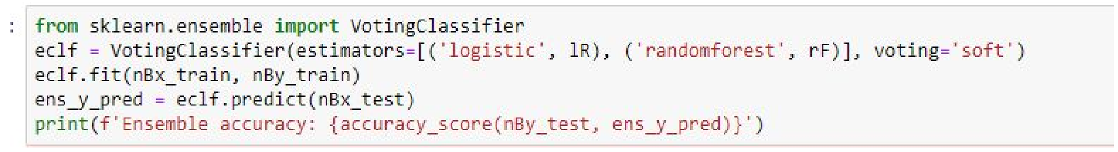
To split dataset following function has been created:



```python
def building_subset(X_train, Xtrain, X_processed, word2vec, ytrain, y_train):
    for train_ind, index_ in enumerate(X_train):
        tokens = X_processed[index_]
        for token_id, token in enumerate(tokens):
            if token not in word2vec.wv:
                pass
            else:
                Xtrain[train_ind, token_id, :] = word2vec.wv[str(token)]
        if y_train[index_] == 0.0:
            ytrain[train_ind, :] = [1., 0., 0.]
        elif y_train[index_] == 1.0:
            ytrain[train_ind, :] = [0., 1., 0.]
        else:
            ytrain[train_ind, :] = [0., 0., 1.]
    return Xtrain, ytrain
```

After splitting the dataset and also preparing tokenized and word2vec representation I started with algorithms.

Naive Bayes, Random Forest, Logistic Regression - were not giving accuracy higher than 67%. At the end - ensemble method of Logistic Regression and Random forest was used. Accuracy achieved 67.8%



```python
from sklearn.ensemble import VotingClassifier
eclf = VotingClassifier(estimators=[('logistic', lR), ('randomforest', rF)], voting='soft')
eclf.fit(nBx_train, nBy_train)
ens_y_pred = eclf.predict(nBx_test)
print(f'Ensemble accuracy: {accuracy_score(nBy_test, ens_y_pred)}')
```

Convolutional Neural Network achieved 75.62% of accuracy. Architecture has been chosen based on the research paper: Convolutional Neural Networks for Sentiment Classification on Business Reviews [https://arxiv.org/pdf/1710.05978.pdf]

The final architecture: two conv1d layers, both with elu activation function with "same" padding, 32 filters, first one with kernel size 3, second one with kernel 2. To avoid overfitting - dropout is used after first and second conv1d layers.

After flatten is used and then fully connected layer with 256 neurons. With activation tanh, then dropout. Last layer is fully connected with 3 neurons and activation softmax.

Loss function - categorical crossentrophy, optimizer Adam with learning rate of 0.0001. To train CNN - batch size that was used is 32 and number of epochs - 100.

As after lots of epoch neural networks tends to overfit - Early Stopping was added. It worked with validation dataset, minimum delta - 0.00025, and patience of 2, meaning that it was waiting for two occurrences of loss function increase.

In this architecture - CNN stopped after epoch number 22, and gave - 75.62% on test dataset, 76.97% on training and 75.48% on validation set.

Epoch 22/100
 - 28s - loss: 0.5397 - acc: 0.7697 - val_loss: 0.5630 - val_acc: 0.7548

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_1 (Conv1D)            (None, 81, 32)            11552     
_________________________________________________________________
dropout_1 (Dropout)          (None, 81, 32)            0         
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 81, 32)            2080      
_________________________________________________________________
dropout_2 (Dropout)          (None, 81, 32)            0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 2592)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               663808    
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 771       
=================================================================
Total params: 678,211
Trainable params: 678,211
Non-trainable params: 0
```

Below is a visualization of training process.

```
visualize_history(history)
```

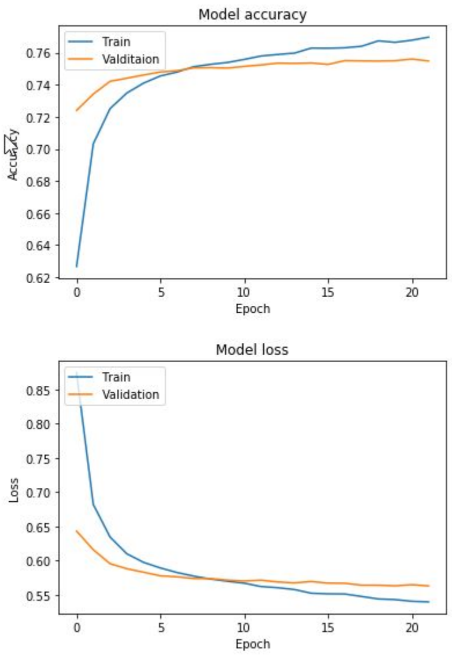
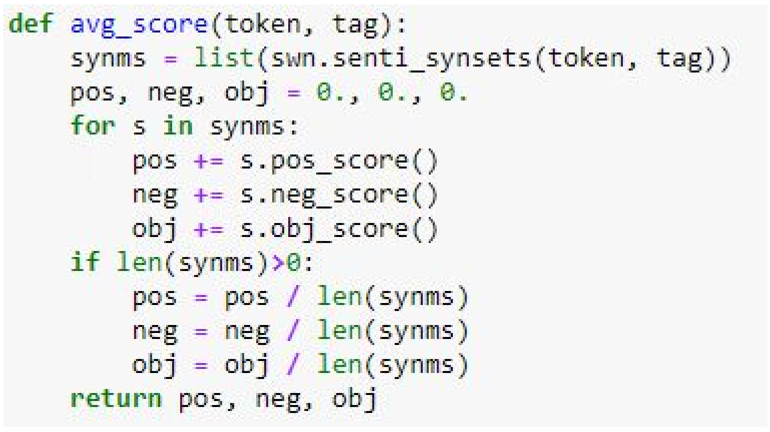


Figure 10. Model accuracy/loss

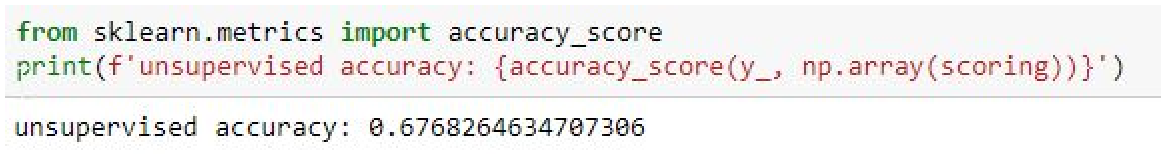
For unsupervised part - nltk sentiwordnet is used. Sentiwordnet is a lexical resource for opinion mining. After pos tagging the result could be passed to sentiwordnet and we can get the synonyms list, where each synonym has a positive, negative and objective score. Going forward this creates a problem with a neutral comments - class 1.0. To avoid this problem, only positive and negative comments has been left for a training.

For each of the token in the sentence - and average positive and negative score for all its synonyms has been calculated. In order to identify the class - a sentence score is calculated as following: positive - negative. Where if score is higher than 0 - then label is 2.0, else 0.0. Function avg_score is presented below:



```
def avg_score(token, tag):
    synms = list(swn.senti_synsets(token, tag))
    pos, neg, obj = 0., 0., 0.
    for s in synms:
        pos += s.pos_score()
        neg += s.neg_score()
        obj += s.obj_score()
    if len(synms)>0:
        pos = pos / len(synms)
        neg = neg / len(synms)
        obj = obj / len(synms)
    return pos, neg, obj
```

After those calculations - process run for all 100002 of examples, original labels were used to calculate accuracy - which was 67.68%



```
from sklearn.metrics import accuracy_score
print(f'unsupervised accuracy: {accuracy_score(y_, np.array(scoring))}')
```

```
unsupervised accuracy: 0.6768264634707306
```

Refinement

Standard algorithms:

In order to achieve better results with a standard algorithms, ensemble method has been used. There were two weak classifiers used for this - logistic regression and random forest.

Grid Search was used to identify the best parameters for logistic regression algorithm in order to identify penalty and “C” (“l2”and 1.0 were used)

After implementing ensemble method: Voting Classifier - accuracy score improved from 67.35% and 67.45% to 67.8%.

CNN:

During neural network training I was doing several changes. I started with following architecture:*[Conv1d -32/5] - [Conv1d - 32/3] - [Dense-256] - [Dense-3].* This architecture gave around 73% of accuracy with elu activation functions, and Adam optimizer. After several iterations I stopped on following architecture: *[Conv1d -32/3] - [Dropout(0.25)] - [Conv1d-32/2] - [Dropout(0.25)] - [Dense - 256] - [Dropout(0.5)] - [Dense -3]*

Things that were tried on the way:

- Optimizers: Adam (best), Nadam (75.22%), Adamax (74.7%), RMSprop (75.15%)
- Batch sizes: 32 (worked the best), 64 (around 73% of accuracy), 16 (slowing down the process)
- Batch normalization: batch normalization was added after first and second nodes of CNN - it did not bring any improvement.
- Learning rates: 0.0001 (works the best), 0.00001 (slowed down the process), 0.00005 (was not giving any improvement)
- Word2vec vs fasttext vs pre-trained glove. Several approaches were tried - word2vec was giving better results rather than fasttext. Pre-trained glove vectors (100, 200) were giving worse results that vector created from dataset
- Word2vec: 100 dimensions, 120 dimension, 200 dimension, 250 dimension. Despite performance problems after running some parts on gpu - I was checking results with different vector sizes. 120 - gave the best results.

Unsupervised:

For unsupervised approach - there were two different libraries of tagging the data used (spacy seems to be using nltk library for pos): spacy and nltk.

Using pos tag from nltk - gave 67% accuracy. After changing pos tagger from nltk to spacy, also filtering out some of the part where tag couldn't be identified - model achieved 71.6% of accuracy.

IV. Results

Model Evaluation and Validation

The dataset that was created for this model contains only comments with number of tokens less than 100. This means that the behavior could be different with longer comments.

Standard algorithms used in this work gave good results and they were fast to implement. They were proven lots of times, and have place in the world of nlp. Some fast prototyping and benchmarking - could be a good use case for them.

CNN. Convolutional Neural Networks for Sentiment Classification on Business Reviews - paper was taken as a basis for building CNN. The final model does not have lots of changes comparing with initial one. Adding additional layers, batch normalization and changing optimizers did not help that match. Instead - increasing a word2vec vector size - showed an improvement. Initially it was 100. Vector size of 120 showed better performance.

The final model has been tested on unseen data and gave good results as well. Result of the model are correlating with research paper where an own word2vec vector has been used.

Unsupervised. As it was mentioned before, unsupervised model has its downside. It does not work with a neutral comments, so the data has been tweaked accordingly. In general it gives a good result on the dataset that has been used for the project. It works good on simple cases (https://en.wikipedia.org/wiki/Sentiment_analysis) and not on complex cases

Justification

In general the aim of the project was to check which of the methods of supervised learning is performing better, also taking into account time spent. For CNN the goal was to achieve 80% of accuracy.

In details:

- Standard algorithms - 67% of accuracy, around 70% was expected;
- CNN - 75% of accuracy, 80% was expected;
- Unsupervised accuracy score is higher than Naive Bayes. 71% versus 67%.

To sum this up, results that were achieved are around the numbers that were expected for standard algorithms and unsupervised learning part. CNN - accuracy is lower than expected. This is due to the fact that not all the samples were used (only 150k versus 1.5 mls), and also limitation on the size of word2vec.

The unsupervised learning part took around 18 hours of work, where supervised part took 32 hours. Significant part of this time has been spent for tokenization process (included in both numbers - 15 hours)

With this result - all the solutions presented could have its place. CNN for more precise sentiment analysis, unsupervised for tagging before human check, and standard algorithms for fast prototyping and sampling.

V. Conclusion

Reflection

In this NLP project three different methods of machine learning has been used and analyzed. As result - each of them has its place.
As expected the most time consuming part was data preparation and analysis. After standard machine learning algorithms: Logistic Regression, Naive Bayes, Random Forest and Voting Classifier were used to create a baseline or benchmark.
To feed CNN - word2vec embedding has been used, it also has been compared with fasttext and pre-built vectors such as glove200 and glove100.
Finally unsupervised learning and sentiwordnet were used create a sentiment analysis model (only for positive and negative).

Final results are correlating with expectations, except the fact that due to performance problems - there were limitations on ways of CNN training. Avoiding these problems was the most difficult part of this project.
This project also helped me to understand why infrastructure is important and when and where gpu could be used.

Improvement

This is a reflection of everything that could be tried in order to improve the models:

1. Word2vec with 300/400/500 vector size;
2. More training data
3. Create two different model - one for short comments, one for long;
4. Try keras embedding layer in the NN architecture;
5. Try LSTM (RNN) to improve performance;
6. Combination of CNN and RNN;
7. Unsupervised: different lexicons could be tried to achieve better results;

Most of the point from the list above are very easy to correct, if no computational limitations persist.