Ukrainian Catholic University

Faculty of Applied Sciences

Business Analytics & Computer Science Programmes

Using classifiers in NLP

Linear Algebra final project report

Authors:
Denys Udud
Ilya Konstantynenko
Nazar Liubas

Github project:
https://github.com/Ikonsty/NPL-emotion-LA

May 2021

Abstract

Small research for testing accuracy of a few classifiers as SVM, Biyes classifier and KNN. The five emotions was classified and got plots of accuracy by each emotion for each classifier. Also, a telegram bot was created to use SVM in practice on real data.

1 Introduction

Our project is on emotional recognition in text. We will use LA and it's algorithms to recognise which emotions are placed in messages, emails, reviews. We will work recognition of five main emotions: joy, sad, anger, fear, and neutral. Our solution has a potential to optimize work of customer support via prioritizing messages that convey strong negative emotions. We build a telegram bot where customers can copy paste the message and get it's emotional inclination. In this report we are going to explain three algorithms that will help to recognise which emotions are fielded in the text. These algorithms are: SVM, NBC and KNN.

2 Literature review

There are some works related to our topic of text classification. Bruno Trstenjak and others[3] used KNN with TF-IDF to classify documents in 4 groups (Sport, politics, Finance, Daily News) and they achieved good accuracy results that differ from group to group, but in general is higher than 50%. Angel Urbano Romeu[4] used such classifiers as SVM, KNN, GMM and NBC to classify audio recordings by its emotions (emotions he focused on are Happiness, Anger, Fear, Sadness and Neutral). Their accuracies were between 67% and 81%, and the best result was shown by KNN. Dey Lopamudra[5] and others used NBC and KNN to classify hotel and movie reviews with two labels (positive, negative). Their results show that the classifiers yielded better results for the movie reviews with the Naïve Bayes‘ approach giving above 80% accuracy and outperforming k-NN approach. However for the hotel reviews, the accuracies were much lower and both the classifiers yielded similar results.

3 Mathematical description of the chosen approaches

3.1 Support Vector Machine (SVM)

3.1.1 Short explanation of algorithm

SVM or support vector machine is a supervised learning algorithm, means that firstly we need to give completed data with answers and then an algorithm can find answers to new data. In this research the classifier based on SVM was built, so there is a short explanation of how it works[6].

Define $Y = y1, y2$ some labels that we can give to a message. Some object that we take, in this case a sentence, can have N attributes $x = (x_1, x_2, ..., x_n)$ for example as words. The algorithm takes the value x and and builds function $F(x) = y$ and returns label y. The main aim of SVM is to find this function called linear classifier $w_1x_1 + w_2x_2 + ... + w_nx_n + w_0 = 0 \; in \; R^n$. Global formula F of x looks like:

$$F(x) = sign(w^T x - b), where \; w = (w_1, w_2, ..., w_n), b = -w_0$$

After building the hyperplane points that will be one one side will mark as first class and others as a second one. The best hyperplane that can be placed is that one when the object of two labels lies as far from the hyperplane as possible. The nearest points to the classifier are called support vectors because the biggest margin between them and line are calculated. On figure they are red with and without filling for black and white points respectively.

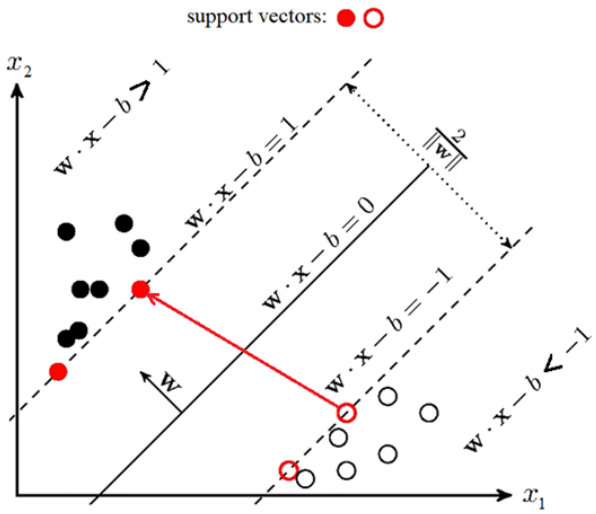

Figure 1: Example of SVM.

Rules to choose best weights for SVM

Vector w is a normal to linear classificator. If we find a projection of vectors created by support vectors of different classes onto w we will get the width of the margin.

$$< (x_+ - x_-), w/||w|| >= (< x_+, w > - < x_+, w >)/||w|| = ((b+1)-(b-1))/||w|| = 2/||w||$$

$$2/||w|| -> max$$

$$||w|| -> min$$

$$(w^T w)/2 -> min$$

Margin is called a value $M = y(w^T x - b)$. SVM:

- make a mistake if $M < 0$(y and $(w^T x - b)$ have different values, as label do not match with position upper or button the line)
- object goes to margin line if $Min(0; 1)$ that is also not good, because we can‘t be sure what label is for an object
- make true classification if $M > 1$, the object is out of margin zone

So the correct classification will be when:

$$y(w^T x - b) \geq 1$$

Taking the previous statement we make a system for default hard-margin SVM, where objects can't be placed in the margin zone.

$$\Big\{(w^T w)/2 -> min \quad , \Big\{y(w^T x - b) \geq 1 \tag{1}$$

If data is linearly inseparable we can not use this system, so we add an error for learning data. $\xi_i > 0$ are values of how big the error is for each x_i:

$$\Big\{(\mathrm{w}^T w)/2 + i -> min \quad , \Big\{\mathrm{y}(\mathrm{w}^T x_i - b) \geq 1 - i \quad , \Big\{\xi_i > 0 \tag{2}$$

Now we will count a penalty - amount of all errors that occurred:

$$Penalty = \sum [M_i < 0], where\ [M_i < 0] = 1,\ if\ Mi < 0;\ 0,\ if\ M_i \geq 0$$

Add a sensitivity to how big the error is(How small is M) and how far it is to a margin:

$$Penalty = \sum [M_i < 0] = \sum (1 - M_i)_+ = \sum max(0, 1 - M_i)$$

Adding it to a standard formula that we have got earlier we get a classical loss function for soft-margin SVM[7]:

$$Q = max(0, 1 - M_i) + (w^T w)/2$$

$$Q = max(0, 1 - yw^T x) + (w^T w)/2$$

The task is to minimize a loss function using gradient descent algorithm[8]. The rule of changing weights, where is a descent step:

$$w = w - \bigtriangledown Q$$

$$\bigtriangledown Q = w - yx,\ if\ yw^T x < 1; or\ w,\ if\ yw^T x_1$$

Of course this classification is only for two labels at the same time but sentences must be classified in six different emotions. There will be a one-vs-many algorithm where we classify between one current emotion and the others and then check which emotion has the highest chance to be in this sentence[9].

3.1.2 Pros and cons.

Pros:

- Works good with small data and large feature space
- The width of the margin will be the maximum so number of errors would be minimal
- Have only one solution

Cons:

- Needs big data for learning
- One exception in learning data can build really bad linear classifier
- By default classify data into two groups

3.2 Naive Bayes classifier (NBC)

3.2.1 Short explanation of algorithm

Naïve Bayes classifier is a simple probabilistic model that works well on text classification. This probabilistic classifier uses Bayes theorem to determine the probability that an observation belongs to some class with a strong (naive) independence assumption[10].

Let's describe Bayes theorem and how it is used there[11].

This theorem states the next relationships, given given class variable y and dependent feature vector x_1 through x_n:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n|y)}{P(x_1, \ldots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$$

This relationship is simplified to:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\Pi_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y|x_1, \ldots, x_n) \propto P(y)\Pi_{i=1}^{n} P(x_i|y)$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i|y)$; the former is then the relative frequency of class y in the training set.

$$\hat{y} = argmax_y P(y)\Pi_{i=1}^{n} P(x_i|y)$$

Now, let's describe a simple example which shows how Naive Bayes classifier works[12]. Let's imagine we have two datasets. In the first one there are keywords characteristic to author A (boy, girl, love, love, cafe, date, date, data, cafe, dinner, vegetables) and probability to come across certain word (for author A the probability to came across word "date" is equal to: number of occurrence "date" / number words in the set = 3/11) . And in the second one we have the same for author B (set of words for B (vegetables, food, food, dinner, cafe, dinner)). Also we are given phrases (vegetables, dinner) and need to know, which author (A or B) wrote this phrase. For that problem we can use Naive Bayes classifier. Firstly we should calculate a common guess from given (training data). For A it will be p(A) = num of words characteristic to author A divided by sum of num of words characteristic to author A and B = 11/17. Than we can calculate the probability for phrase to be written by A given, that phrase is "vegetables dinner" :

$$P(A|phrase) = P(A) * P(dinner|A) * P(vegetables|A) = 0,65 * \frac{1}{11} * \frac{1}{11} = 0,0052$$

The same we calculate for author B. And If $P(B|phrase) > P(A|phrase)$ - it is more likely that phrase was written by author B. In such a way we can determine which emotion is described in the message, who wrote it and in what mood he was.

3.2.2 Pros and cons.

Pros:[15]

- Easy implementation
- Fast
- Noise resilience (which means that outliers do not seriously affect results)
- One of the easiest machine learning algorithms to train

Cons:

- Needs big data for learning
- Do not work well, when there are dependent features
- Naive Bayes processes consider all features as independent, so some features might be processed with very high bias

3.3 k-nearest neighbors (KNN)

3.3.1 Short explanation of algorithm

The KNN is a classificator that among other applications can be used for classification of written text. It works as follows: we have a set of training points in multidimensional space. These points are classified. Then we add a new point with an unknown class and want to label it. To do this, we calculate K nearest points to this one and label new one by class that has the most members among these K. There are few ways of calculating distance between points, most often used is Euclidian one (but not always the best):

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Where x and y are points in Euclidean n-space, x_i and y_i are i-th coordinates of x and y respectively.

Let's consider an example:

There are only 2 classes (blue and red) and we classify black point. As $K = 5$, the algorithm is looking for 5 nearest neighbours, and as the majority of these neighbours are blue points, black one will be classified as blue.

As our issue is to classify text (we will call different texts as documents), we should somehow bring it into numerical values. And this can be done by a weight matrix. This is the $N \times M$ matrix where N is the number of unique words in the dataset and M is the number of documents in our dataset . Each (i, j)-enry of this matrix is a weight of i-th word in the j-th document. Also there should be a table with id-s of documents and their classes. Tables looks as follows:

Weights of words should be calculated by us. Bruno Trstenjak highlights such main way to calculate weights of words in weight matrix:[14]

$$(i,j)\ entry\ of\ matrix\ = TF \times log_2(\frac{Quantity\ of\ documents\ in\ dataset}{Quantity\ of\ i-th\ word\ in\ dataset})$$

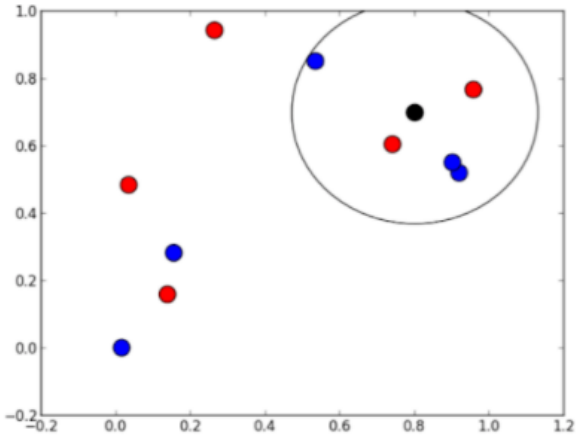

Figure 2: k-nearest neighbours example with two classes (blue and red) and k=5 [13]

Words\Document id	1	2	3
Word1	2	0	1
Word2	0	0	3
Word3	1	2	1
Word4	1	0	0

Document id	Label
1	Label 1
2	Label 2
3	Label 1

Figure 3: Example of weight matrix and label table

And then, to calculate a distance between two documents, we will use cosine metric (because during experiments it showed the best accuracy results):

$$cos(\theta) = \frac{A \times B}{||A|| \times ||B||}$$

Where $cos(\theta)$ is the measure of vectors similarity (1 is perfect similarity), A and B are like columns of weight matrix.

3.3.2 Pros and cons.

Pros:[15]

- It is simple, intuitive and easy to implement
- There is no need for training this model
- This model immediately adapts as we collect new training data
- Have only one parameter
- Can be used for multi-class problem

Cons:

- It is algorithm of high complexity that also need a lot of memory
- Data should be converted properly to number format
- It is not easy to choose parameter
- Data should be balanced in order to this model show a good results
- It is sensitive to outliers

4 Implementation and experiments

4.1 Data description

We have found a dataset[16] with 2 columns - Text and its Emotion. There are five labels in column Emotion: joy, sad, anger, fear, and neutral. Also, data is divided into training and testing set that compose, respectively, 80% and 20% of all data. Distribution of data is on a figure:

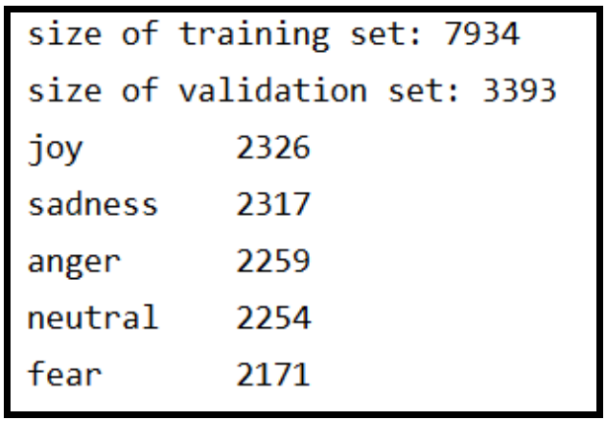

Figure 4: Distribution of labels in dataset

In order to start training classifiers, text from dataset should be cleaned. We done it by the next schema:

1. remove html markup (signs like <, >, ?, etc)
2. remove urls (that starts with 'http')
3. remove hashtags (that starts with # and @)
4. remove punctuation and non-ascii signs
5. remove whitespaces from beginning and end of sentence
6. lower all letters

4.2 Implementation

Due to luck of time we didn't implemented classifiers by ourselves, but used existing solutions from python library sklearn.

Also, as the basis the project of Lukas Garbas who get very similar accuracy because we have trained our classifiers on the same dataset. When we understand that SVM is the best classifier for this, we use it for telegram chat bot. It is written on python using python-telegram-bot and have pretty easy implementation: it takes a message from user, use SVM that was already trained, predict an emotion of this text and gives result to chat. Also, the trained model saves in 'tfidf-svm.sav' file to prevent model from teaching every time when user sent a message.

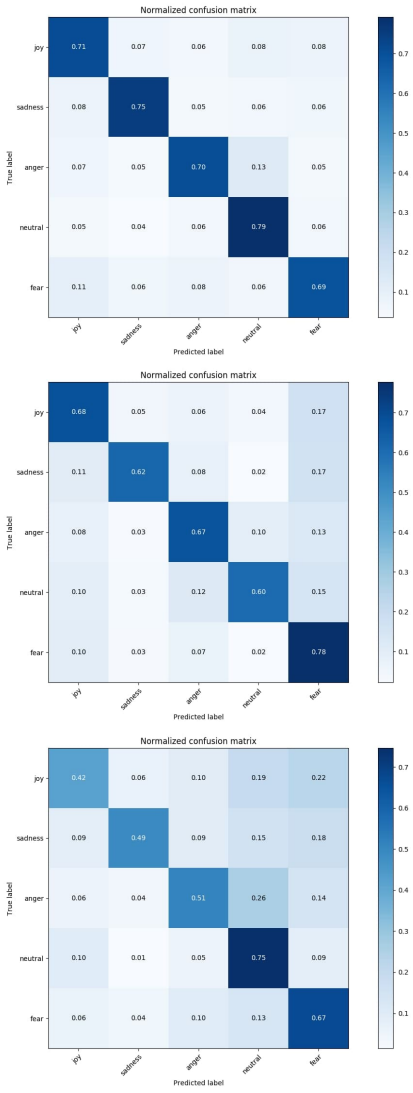

Figure 5: Confusion matrix for SVM (upper), NBC (middle) and KNN (bottom)

4.3 Results of experiments

We measure goodness of classifiers by accuracy of their predictions. It is calculated by the next formula:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ all\ predictions}$$

It is well applicable measure in our setting, as our data is balanced. So, accuracies of classifiers are:

- SVM: 72.71%
- NBC: 67.02%
- KNN: 56.29%

Also we get such graphical representations for each method. The right scale represents the true value of statement and the bottom one represents labels that classifier gave. All three classifiers have dark-blue diagonal on this graphics - that means that mostly

messages were classified as it should be. Also there is some incorrect determinations, for example SVM we can observe that there is a 5 percent chance to classify anger as sadness and so on. The darker the square is the more accurate the prediction was. Also from such diagrams we can get information what classifier is the best and what emotion it the easiest to predict. For example, for support vector machine the easier text turns out a text with neutral meaning or fear for Naive Bayes classifier.

5 Conclusions

In our project on emotional recognition in text we realised a telegram bot solution, which, by using such mathematics tools as KNN, SVM and BC, classifies which emotion is placed in message, emails, reviews. Our solution works well and can recognise 5 main emotions : joy, sadness, anger, neutral, fear. Moreover, while we were working on the project, we investigated that support vector machine works the best of these three classifiers for NLP: emotion classification. Although, SVM model is pretty accurate and works well in our app, we think that it's accuracy could be even higher if we had bigger dataset. So the next steps in developing our solution is searching and forming the bigger dataset for training the model and expansion and improvement of UX/UI design.

References

[1] Gilbert Strang, *Introduction to Linear Algebra*, 5th Edition, Wellesley–Cambridge Press, 2016.

[2] http://github.com/linear-algebra

[3] Bruno Trstenjak, Sasa Mikac, Dzenana Donko, \KNN with TF-IDF based Framework for Text Categorization", Procedia Engineering, Volume 69, 2014, Pages 1356-1364, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2014.03.129. (https://www.sciencedirect.com/science/article/pii/S1877705814003750)

[4] Romeu, A.. \Emotion recognition based on the speech, using a Naive Bayes classifier." 2016, (https://www.semanticscholar.org/paper/Emotion-recognition-

[5] Dey, Lopamudra Chakraborty, Sanjay Biswas, Anuraag Bose, Beepa Tiwari, Sweta. (2016). \Sentiment Analysis of Review Datasets Using Naïve Bayes‘ and K-NN Classifier". International Journal of Information Engineering and Electronic Business. 8. 54-62. 10.5815/ijieeb.2016.04.07. (https://www.researchgate.net/publication/305001704_Sentiment_Analysis_of_Review_Data

[6] SVM. First look and realisation. Detailed analysis, Lagg, 2020 - (https://habr.com/ru/company/ods/blog/484148/)

[7] Sentiment Analysis using SVM, Vasista Reddy, 2018 (https://medium.com/@vasista/sentiment-analysis-using-svm-338d418e3ff1)

[8] Linear Algebra, Gradient Descent Algorithm in machine learning,
Ghassan Karwchan, 2019 (https://blog.gisspan.com/2019/12/linear-algebra-grandient-d

[9] Personality classification based on Twitter text using Naive Bayes,
KNN and SVM, 2015, IEEE (https://ieeexplore.ieee.org/abstract/document/7436992)

[10] Romeu, A.. \Emotion recognition based on the speech, using a Naive
Bayes classifier." 2016, (https://www.semanticscholar.org/paper/Emotion-recognition

[11] Jason Brownlee,' Naive Bayes Classifier From Scratch in Python',
October 18, 2019 https://machinelearningmastery.com/naive-bayes-classifier-scratch-

[12] Sckit Learn Org \Naive Bayes", site: https://scikit-learn.org/stable/modules/naiv

[13] Romeu, A.. \Emotion recognition based on the speech, using a Naive
Bayes classifier." 2016, (https://www.semanticscholar.org/paper/Emotion-recognition

[14] Bruno Trstenjak, Sasa Mikac, Dzenana Donko, \KNN with
TF-IDF based Framework for Text Categorization", Procedia
Engineering, Volume 69, 2014, Pages 1356-1364, ISSN
1877-7058, https://doi.org/10.1016/j.proeng.2014.03.129.
(https://www.sciencedirect.com/science/article/pii/S1877705814003750)

[15] \Pros and Cons of K-Nearest Neighbors", https://www.fromthegenesis.com/pros-and-co

[16] https://github.com/lukasgarbas/nlp-text-emotion/blob/master/data/data$_test.csv$