



IMapBook collaborative discussions classification

Matej Rus, Aleks Vujić, and Nik Zupančič

Abstract

In today's world with increasing amount of information every day we require the help of artificial intelligence to help us process and understand the data. One of the areas of research is also the area of natural language processing which processes and analyzes natural language data. The goal of this project is to use the input data containing of chat messages during collaboration of students and try to classify each message in one of the predefined categories. We will also explore the options of including additional data like eBooks that were being discussed and how the classification results can be improved.

Keywords

natural language processing, text classification, logistic regression, naive bayes, rocchio classifier

Advisors: Slavko Žitnik

Introduction

During the past year many teachers and students around the world have been faced with a challenge of remote learning. Various tools can be used by teachers to engage their students and one of such tools is also **IMapBook** platform. IMapBook is a web-based application which is designed to improve students' reading comprehension through various games on the website. The activity from which provided dataset was composed is called **Collaborative discussion**.

In these collaborative discussions students were first asked to read several short stories. They were then assigned to groups and each group was given a question/topic to discuss using IMapBook platform. Members of the group were then able to discuss the topic within the group chat and together write the final response to teacher's question. The dataset contains around 800 chat messages which we will try and classify into one of the categories such as: content discussion, greeting, logistics, assignment instructions, feedback etc.

Existing solutions

Text classification is one of the challenges in the field of natural language processing. There are several different classification techniques that could be used for this task. In this section we will provide a short overview of some of these techniques.

Rocchio classification

Rocchio classification [1] algorithm was presented in 1971. It uses a simple approach to construct a prototype vector for each class - this is an average of all training vectors belonging to that class. Classification of the test document is then done by comparing the document's vector to prototype vectors of each class and choosing the class with the closest prototype vector. The centroid of a class c is computed as the vector average or center of mass of its members, where D_c is set of documents in D whose class is c : $D_c = \{d : (d, c) \in D\}$.

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \in D_c} \vec{v}(d) \quad (1)$$

Logistic regression

Another well known method of classification is logistic regression [2]. It is a statistical model that models probability of an input belonging to some class. By default, logistic regression can only be used for binary classification, but there are also **multinomial** logistic regression methods which generalize logistic regression to problems with multiple classes. Each document is represented as a vector of independent variables - features: $X_i = (x_{(1,i)}, x_{(2,i)} \dots x_{(M,i)})$, together with classification Y_i for each vector. A linear predictor function predicts the probability that some document i belongs to class k :

$$f(k, i) = \beta_{(0,k)} + \beta_{(1,k)}x_{(1,i)} + \dots + \beta_{(M,k)}x_{(M,i)} \quad (2)$$

where $\beta_{(m,k)}$ presents a weight associated with m -th feature and k -th class. In a vector form this can be written as:

$$f(k, i) = \beta_k \cdot X_i \quad (3)$$

We can construct a system of equations:

$$\ln P(Y_i = 1) = \beta_1 \cdot X_i - \ln Z$$

$$\ln P(Y_i = 2) = \beta_2 \cdot X_i - \ln Z$$

$$\vdots$$

$$\ln P(Y_i = K) = \beta_K \cdot X_i - \ln Z$$

In this case Z presents partition function for the distribution and can be computed with the following equation:

$$Z = \sum_{k=1}^K e^{\beta_k \cdot X_i} \quad (4)$$

This gives us a general equation for probability:

$$P(Y_i = c) = \frac{e^{\beta_c \cdot X_i}}{\sum_{k=1}^K e^{\beta_k \cdot X_i}} \quad (5)$$

Naive Bayes classifier

Naive Bayes classifier [3] is based on Bayes probability theorem. Given a vector $x = (x_1, x_2, \dots, x_n)$, representing some document with n features, we can use Bayes theorem to calculate probability that document x belongs to some class C_k . This is calculated for all possible classes:

$$P(C_k|x) = \frac{P(C_k)P(x|C_k)}{P(x)} \quad (6)$$

The most common use of naive Bayes classifier is to classify document to a most probable class:

$$y = \max P(C_k) \prod_{i=1}^n P(x_i|C_k) \quad (7)$$

Naive Bayes' advantage is that it only requires a small number of training examples to estimate classification parameters. It can handle continuous and discrete data, is not sensitive to irrelevant features and can make probabilistic predictions. It can also be easily updated with new training data.

Random Forest Classifier

Random Forest Classifier [4] is learning method for classification by constructing large number of decision trees. To be successful with multiple trees, we need a way to create trees that generalize independently. One of the methods to create multiple trees is to construct trees in randomly selected subspaces of the feature space. For given feature space of m dimensions there are 2^m subspaces in which decision tree can be constructed. Given t trees created in random subspaces, a discriminant function is needed to combine their classification. For a point x let $v_j(x)$ be terminal node that x is assigned to when it descends down tree $T_j (j = 1, 2, \dots, t)$. Given this, let

the posterior probability that x belongs to class c ($c = 1, 2, \dots, n$) be denoted by:

$$P(c|v_j(x)) = \frac{P(c, v_j(x))}{\sum_{l=1}^n P(c_l, v_j(x))} \quad (8)$$

Since the trees are fully split, most terminal nodes contains only single class and thus the value of the estimate $P(c|v_j(x))$ is almost always 1. The discriminant function is defined as:

$$g_c(x) = \frac{1}{t} \sum_{j=1}^t P(c|v_j(x)) \quad (9)$$

While some trees may be wrong, many other trees will be right, so as a group of trees are able to move in the correct direction. For this system to work the predictions made by individual trees needs to have low correlation with each other.

Related work

Online chat rooms have been around for a long time starting with IRC rooms and later continuing with web chat rooms and various applications. Due to the large amount of messages this is also an interesting area for natural language processing. Through the years many papers have researched the classification of online chat messages. However, data collected can differ from one platform to another. Some of the papers related to this area include:

- Paper *Using data mining methods for automated chat analysis* [5] describes the approach to classifying chat messages. It raises the problem of preprocessing the data which is usually written in informal language, leading to noisy data: misspelling of words, abbreviations, emoticons... The paper mentions two different representations of text that can be used: **bag of words**, where each message is presented as a dictionary of words and their frequencies and **shallow grammars**, in which they assume that grammatical structure of a message could imply the class of that message. For classification of messages into one of 4 predefined categories, the researchers used Naive Bayes classifier in combination with both bag of words and shallow grammar representation of text. Comparing the results of both we can notice that bag of words representation was the one with higher accuracy of classification.
- Another experiment relevant to our assignment is *Segmentation and Classification of Online Chats* [6]. Description of the preprocessing part mentions the frequency cutoff in which it discards words with very low or very high frequencies. We can assume that those words are not as important so they don't contribute to a better classification result. They used Naive Bayes and Viterbi classifiers with Naive Bayes achieving better classification results.

- Another interesting experiment on the topic of message classification is *Chat mining: Automatically determination of chat conversations' topic in Turkish text based chat mediums* [7]. The dataset used consists of messages in Turkish language, but many of the concepts apply for English as well. The paper compares 3 different classifier methods: Naive Bayes, k-nearest neighbors and SVM (Support Vector Machine) [8]. The highest accuracy on obtained dataset was achieved using SVM method.

Initial plans

1. In order to be able to use any of the approaches described in previous chapter we will initially have to import and read the data. As the dataset is given as a Microsoft office Excel Worksheet document we can use pandas library [9] in Python. The dataset is divided into 3 tables. The first table (CREW data) consist of all messages of readers after reading short story with all the additional informations about course, Bookclub etc. The second table is the same as the first one and contains only messages relevant to the story discussion. The third table contains crew final responses about the story that they read.
2. (Manual) analysis of the data: are the messages written in formal or informal language? Should we take common abbreviations of words into account as well (such as abbreviations *thx*, *tnx*, *thanx* for the word *thanks*)?
3. Next, we have to preprocess the text, removing stop-words, punctuation and capital letters in the messages. Once the preprocessing step is done, we have to vectorize input documents (in our case we represent each message as a vector of features). We can compare results using different vectorization methods: simple frequency count (bag of words), TF-IDF method...
4. Implement one of the classifiers described under the *Existing solutions* chapter to use as a baseline solution. The goal is to classify chat messages into specified categories. We have available about 800 chat messages to train and test out classifiers.
5. Explore possible areas of improvement of baseline solution:
 - Are there any typical words for each class whose presence would indicate that a message belong to

that class?

- Use the texts of stories being discussed to try and determine if a message relates to the contents of the story
- Analyze and compare different vector representations of messages and how they affect accuracy of the model.
- Treat the synonyms of a specific term as one word (for example in a conversation words like book, novel, story, text could all refer to the same term). How does this affect the classification?

References

- [1] Joseph Rocchio. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*, pages 313–323, 1971.
- [2] S Philip Morgan and Jay D Teachman. Logistic regression: Description, examples, and comparisons. *Journal of Marriage and Family*, 50(4):929–936, 1988.
- [3] Daniel Berrar. Bayes' theorem and naive bayes classifier. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics; Elsevier Science Publisher: Amsterdam, The Netherlands*, pages 403–412, 2018.
- [4] Tin Kam Ho. Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pages 278–282, 1995.
- [5] Anjo Anjewierden, Bas Kolloffel, and Casper Hulshof. Towards educational data mining: Using data mining methods for automated chat analysis to understand and support inquiry learning processes. In *International Workshop on Applying Data Mining in e-Learning (ADML 2007)*, pages 27–36. Citeseer, 2007.
- [6] Justin Weisz. Segmentation and classification of online chats, 2008.
- [7] Özcan Özyurt and Cemal Köse. Chat mining: Automatically determination of chat conversations' topic in turkish text based chat mediums. *Expert Systems with Applications*, 37(12):8705–8710, 2010.
- [8] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.
- [9] Python pandas library. <https://pandas.pydata.org/>. Accessed: 2021-03-24.