Profile-Based Retrieval

IREI Assignment

David Cabornero Pascual
david.cabornero@alumnos.upm.es

April 4, 2022

1 Introduction

With the huge escalation of information in our times, text categorization systems have become more necessary than ever. For example, search engines need to find the best results that fit with a certain query in a few seconds, and some systems have to find information in many documents at the same time. Not only is accuracy a key point in these systems, but also the response time.

In this case, the issue consists of the need of recommending certain user text snippets according to their interests. To achieve this goal, some tagged text snippets are provided to the system to train some machine learning algorithm that allows us to classify other not-tagged text snippets. With this algorithm, we can recommend not-tagged texts to certain users once they have indicated explicitly which are their preferences.

Two machine learning techniques are going to be used for this purpose: k-Nearest-Neighbors and Deep Neural Networks. The dataset OHSUMED is provided with different tags related to health, and we will extract some conclusions with different metrics. In the same way, these metrics will be utilized to decide in which aspects a certain algorithm is better.

2 Methodology

The implementation of this assignment has been carried out in *Python 3*, mainly with the libraries *sklearn* for general purposes and *nltk* for preprocessing purposes.

The workflow of the assignment is explained in Figure 1. One seed will be fixed in order to keep the experiments repeatable. A 30% of the dataset will be reserved for testing purposes. Both datasets (*OHSUMED* and *Reuters*) have been extracted from

[1]. In particular, *Reuters* with 90 categories and *OHSUMED* with 20.000 abstracts have been used.

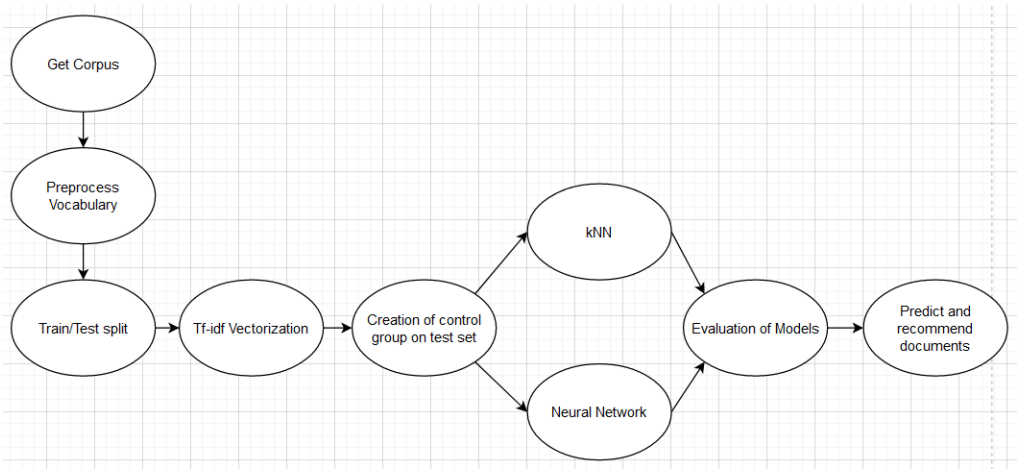


Figure 1: Workflow of the assignment.

2.1 Data preprocessing

Firstly, this dataset is composed of text snippets and every document will have some words that are irrelevant to our task. For instance, prepositions or linkers are not generally interested to determine the class of a document. For this purpose, some common preprocessing techniques in natural language problems are going to be applied:

- Removal of every punctuation mark, since they are not a real part of the words.
- Replacement of every capital letter by its proper lower case.
- Every word whose length is shorter than 4 will be removed since they are normally linkers, propositions or very common words.
- The library *nltk* includes a corpus with stopwords, which are those words that are empty without another keyword. They are, for instance, articles, prepositions or pronouns. In this case, we will remove also these stopwords.
- Finally, a lemmatization will be carried out. This method covers a problem with some words that, although they are different, their meaning is the same or very similar. For instance, words in plural, conjugation of verbs or gender of nouns are variations of a word and must be lemmatized in order to be identified as a single word.

Moreover, we have trimmed documents, but they are not readable yet by the most common machine learning algorithms. In order to perform this, we need a vector

of a fixed dimension to represent each document. In this case, the method tf-idf will provide this functionality. This method will provide a vector for each document with a dimension equal to the number of words in the corpus. Each value will be between 0 and 1 and will be dependent on the frequency of the word in the document (positive dependence) and on the frequency of the word in the whole corpus (negative dependence), because an uncommon word is supposed to be a more relevant word if it appears several times in a document.

2.2 Dataset description

The current dataset that is going to be used for all the trials is the OHSUMED one. This dataset contains a collection of about 20.000 abstracts of papers related to cardiovascular diseases. It contains 23 different categories and each one is related to a different disease. They are divided into two groups: 10.000 abstracts are prepared for training and the other 10.000 for testing.

Nonetheless, this dataset is too big for this assignment, and the classes are very unbalanced. In order to solve this problem, only 6 categories have been selected from the training set (ignoring the test set during the whole assignment). In particular, they all have between 450 and 650 text snippets. With this measure, the chosen classes are 6, 8, 10, 12, 20 and 21. Now, we only have 3.244 text snippets instead of 20.000. After preprocessing the data with the criteria of the previous section, there are 16.078 useful words, and consequently 16.078 attributes in the following algorithms. It is a huge number of features and we have to take it into account.

Apart from this dataset, we need to check that other text snippets (or some random text) are not classified as a certain category. Obviously, not every text snippet has to be predicted as an abstract of a paper about cardiovascular diseases. To perform that, some texts for the *Reuters* dataset have been chosen. In particular, 433 text snippets make up this set, tagged with a *None* tag.

2.3 k Nearest Neighbors

For the nearest neighbors problem, we have the main question: which distance is more useful to detect similar text snippets. In this case, the similarity function will be used only to find the top k most similar snippets:

$$d(\vec{x}, \vec{y}) = 1 - \text{sim}(\vec{x}, \vec{y}) = 1 - \cos(\vec{y}), \tag{1}$$

where $\vec{x}$ and $\vec{y}$ are two vectors encoded with the *tf-idf* vectorization and the similarity is obtained by the dot product.

After that, we have to decide the weight of each neighbor in the decision. One of the most common implementations of *kNN* is the uniform weighting or weights based on the distance. However, here is a more correct approach: take advantage of the

similarity. We can sum the similarities of each category with our target and conclude that the greater sum is associated with the predicted class. In other words, the score of each category c is calculated as follows:

$$\text{score}(c, \vec{t}) = \sum_{\vec{d} \in \mathcal{S}_k} \mathbf{I}(c, \vec{d}) \cos(\vec{t}, \vec{d}), \tag{2}$$

where $\vec{t}$ is the vector associated with the target, $\mathcal{S}_k$ is the set of vectors associated with the k nearest documents, and $\mathbf{I}(c, \vec{d})$ is the indicator function that is 1 if the document d is associated with the c category, and 0 in any other case.

Apart from that, the only hyperparameter that we need to specify is k, the number of neighbors. In this case, 15 neighbors will be selected to determine which category is the correct one.

The good part of this scoring is that, somehow, we have created a probability of being associated with every category for each text snippet. It could be useful to create a threshold, as we will see in the following sections.

The computational cost of this algorithm is quite different from the rest. Apart from preprocessing time, the training phase does not consume time at all. Nonetheless, the test phase is linearly dependent on only these three components:

- Size of the training set.
- The number of features, i.e. the vocabulary of the documents.
- The number of neighbors k.

Each sample will be classified in a certain time determined only by these three factors, so there are some advantages and disadvantages. On the one hand, huge datasets are strongly penalized because of the size, but also because of the implicit extension of the vocabulary that is supposed to be in a large dataset. On the other hand, unlike other algorithms, it is not dependent on the number of categories, so it is very suitable for multiclass problems.

2.4 Deep Learning

In this case, a deep neural network with several layers will be implemented in order to solve this classification problem. In the input layer, we have a neuron for each word in the vocabulary whose value is determined by the corresponding tf-idf coefficient of the word in the document. The output layer is composed of a number of neurons equal to the number of categories: six in this case. Each neuron will output the probability of belonging to a certain class.

Regarding the hyperparameters, the neural network has four fully-connected hidden layers with 256, 128 and 64 neurons respectively. Since the proposed neural network

is deep, the activation function is $ReLU$, and the backpropagation is made via $ADAM$ algorithm. In order to prevent overfitting, early stopping after 15 epochs without improvement in the validation set has been implemented.

As well as kNN, these algorithms return the probabilities of belonging to a certain category, which will be useful again in order to calculate a threshold.

In this case, training efficiency is very different: it depends linearly on the number of weights, the number of training examples and the number of epochs (since we have early stopping, we can forget about the last one). If we depend on the number of weights, training efficiency is dependent on the number of input and output layers. In conclusion, we have several drawbacks in this case, as the vocabulary and the number of samples tends to be huge and we are struggling with a multiclass problem.

Nonetheless, once our network has been trained, the testing efficiency only depends on the size of the neural network. It means that the size of the training set is not relevant anymore, which is especially good in an information retrieval problem. Consequently, we can conclude that deep learning techniques are much more efficient than knn once they are trained if we are dealing with big datasets.

Finally, the size could be an issue with such a big input layer. Let's calculate the explicit size of the weight matrix. We have a float per value, which size is 4 Bytes. We have more or less 16.000 neurons in the input layer and 256 in the first hidden layer. After multiplying these three values, we obtain that the first layer has a size of 16,5 Mbytes. The following layers are much smaller, so we can forget about them. This size could be quite big if we compare it but other algorithms, but every domestic computer can run these neural networks without problems.

2.5 Selection of a threshold

In addition to the text snippets that are provided by the official dataset, we need to check those text snippets that have nothing to do with our categories are not classified as one of our categories. To perform this task, a control set should be added to the test set. Thus, we will be able to know whether our system can detect if a text belongs to any of our categories. The chosen dataset is from *Reuters*, which is about some economy text snippets related to some financial assets. The number of texts must be balanced with the rest of the categories, and the representation of each category is more or less $500 \cdot 0,3 = 150$.

In order to perform this new feature, we have to decide when we do not have enough certainty to determine that a text snippet belongs to a known category. In this case, two algorithms are implemented and both of them provide the probability of belonging to the predicted class. The idea is to provide a threshold that allows us to reject the hypothetical category if there is not enough certainty. One of the most common ways

to calculate the better threshold is by maximizing this function:

$$f(\theta) = TPR(\theta)(1 - FPR(\theta)), \tag{3}$$

where θ is the threshold, TPR is the True Positive Rate (or recall) and FPR is the False Positive Rate (or probability of false alarm). Here, we understand as *positive* every category of OHSUMED and the text snippets of *Reuters* as *negative*.

Finally, we have to decide which value is better for the threshold. There are mainly two charts that plot the performance of the algorithms depending on the threshold: the ROC curve and the precision-recall curves. In both cases, every category is considered as a *positive* result and the absence of a category is considered as *negative* result. Both will be used to determine the threshold, but ROC curves are usually better when the classes are balanced.

2.6 Comparison between algorithms

There are several ways to compare both algorithms. Firstly, we can take into account that the selection of the threshold could have influenced the rest of the metrics. For this reason, we could start analyzing the ROC curves and the area under ROC curve (AUROC), which are not dependent on the threshold selection method.

Once we have chosen a threshold, we must be concerned about the class imbalance. One of the classes could have bad predictions, of maybe the threshold is causing many *false not-classified* documents. In order to perform this task, we can use the confusion matrix.

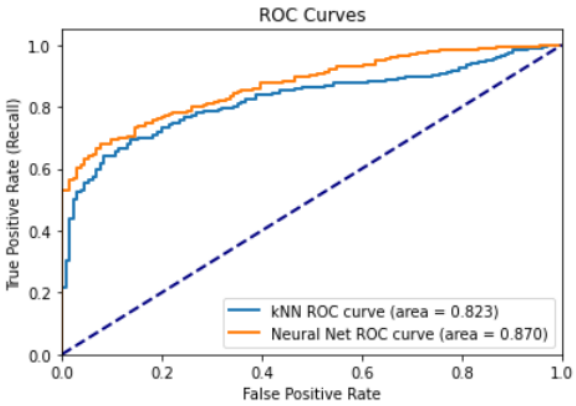
Finally, we can expect that a 7x7 matrix is not the best way to analyze imbalance. Some metrics could help us to perform this task. In this work four common metrics have been chosen: accuracy, precision, recall, F1-score and AUROC of the previous ROC curve. The accuracy and AUROC have a clear definition in multiclass problems, but we have to specify how the rest of the metrics work here. In this case, the *micro average* has been selected, that is, we will sum the true positives, false positives and false negatives over all the labels and will calculate precision, recall and F1-score with these values.

2.7 Recommendations to a certain user

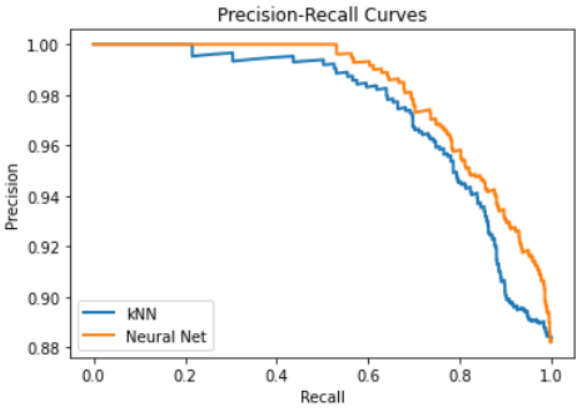
At last, a small simulation of recommendations will be carried out. To perform this task, a small amount of users is created, and each one will have one or more preferences about the kind of documents they are interested in. Finally, one of our algorithms will predict the class of the document and will send it to the users that indicated their preference for this type of document.

3 Results

Firstly, we can see in Figure 2 both ROC and precision-recall curves. In both cases, the neural network is slightly above, so this neural network is going to behave better independently of the threshold that we have chosen. In the precision-recall curve, we have to be careful with the scale of the axis: the precision axis has values between 0.88 and 1, so both lines are nearer than expected.



(a) ROC curves.



(b) Precision-recall curves.

Figure 2: ROC and precision-recall curves of both algorithms.

After applying the Equation 3, the best threshold in kNN is 0.456 and in the Neural Network is 0.835. It means that when we have the Neural Network, we are going to be much more demanding, requiring that the algorithm returns a higher value if we want to be sure of the result. In Figure 3 we can observe how, as a result of a higher threshold in the Neural Network, kNN has a greater False Positive Rate and True Positive Rate. However, the TPR does not seem to be much larger, since the threshold is almost double in the Neural Network. One explanation could be that Neural Networks have

more certainty about their predictions, while kNN always has noise because of its nature (it always has near neighbors of different classes).

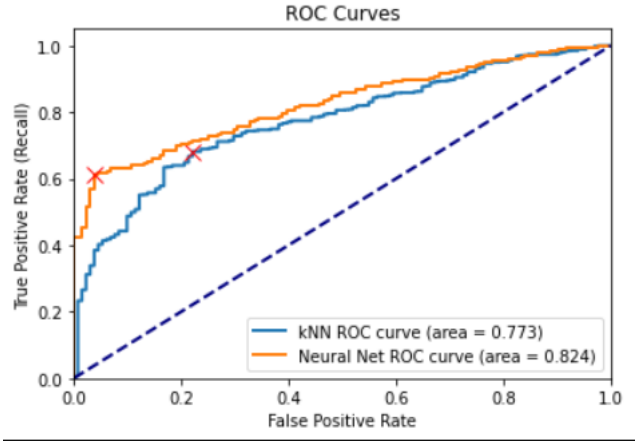
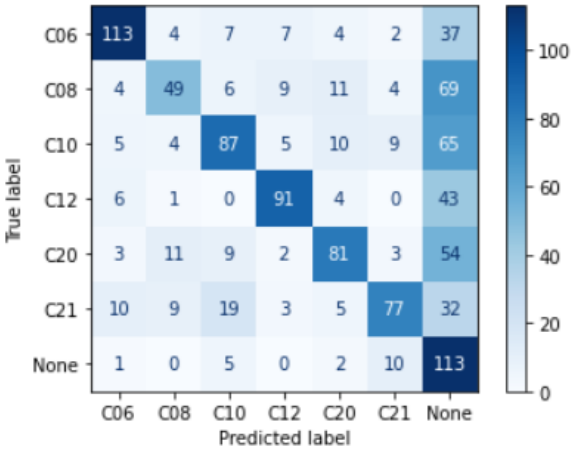


Figure 3: ROC curve after selecting the threshold. The points indicates the position of the selection in the curve.

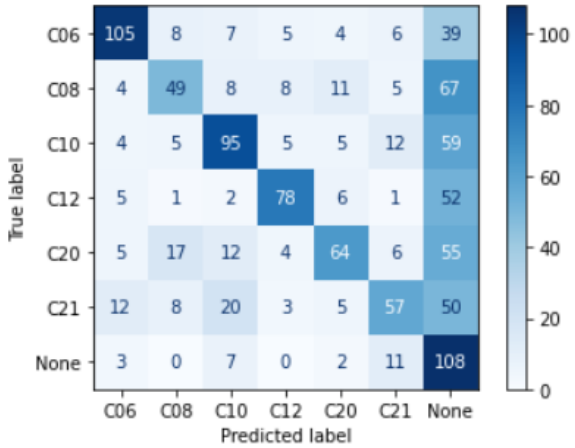
Besides, apart from analyzing the accuracy of the threshold we can analyze which is the performance of each classifier with every class. In order to do this, we will use both confusion matrices in Figure 4. As the comparison between both matrices could be exhausting, the difference between kNN confusion matrix and the Neural Network one has been performed in Figure 5 with a heatmap. Some of the conclusions that we can extract from these matrices are:

- Due to a greater threshold, the Neural Network has fewer problems finding control group elements. However, it has more False Negatives for the same reason.
- In general, kNN is better to find more True Positives of the classes, but more control elements are misclassified.
- C06 is the class of documents that are classified more easily in both cases.
- C08 is the hardest class in both cases, and it is confused very often with C20 and the control group.
- The Neural Network is almost perfect if you send it an element of the control group. However, in some cases, the kNN algorithm classifies elements of the control group as elements of class C21.

Nonetheless, some 7x7 matrices have too much information. We can compress them into some metrics. In particular, the metrics that have been performed are in Table 1. As we can see, almost every metric is better in the Neural Network case. We already knew that the AUROC metric is better because of the shape of the ROC curves, but



(a) k-Nearest-Neighbors



(b) Neural network.

Figure 4: Confusion matrices of each algorithm.

the rest of the information is new. For instance, the accuracy, precision and recall are better in the Neural Network, which means that it is better in terms of hits, false positives and false negatives (once we have averaged the result of the classes).

However, F1-score is better in kNN. As a matter of fact, F1-score is known because it measures better imbalance between classes, so here we can assert that kNN is more beneficial if we want to get similar results in every class.

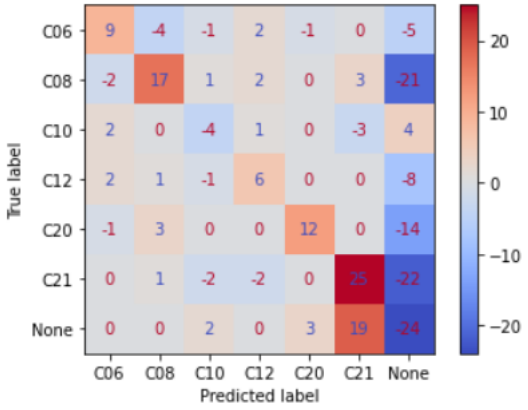


Figure 5: Difference between kNN confusion matrix and the one of the Neural Network. Greater values (in red) are greater in kNN, while lower values (in blue) are greater in the Neural Network.

	Accuracy	Precision	Recall	F1-score	AUROC
kNN	51.4	52.4	61.4	53.1	0.823
Neural Net	55.9	54.3	63.5	51.1	0.870

Table 1: Table of metrics. Except for AUROC, they are all expressed in percentage.

4 Conclusion

Several classes of documents in OHSUMED dataset have been used to evaluate a predictive model that can predict the class of a given random document. The final purpose of this model is a recommendation of not classified papers to users of our system. To perform this task, a control group has been created, because some of the papers that are provided could be not relevant at all.

After preprocessing the vocabulary by removing irrelevant words and transforming each document into a vector via tf-idf algorithm, two algorithms have been performed to carry out the main task: kNN and Neural Networks. In the first case, the class of the most similar documents to the document we want to classify is taken into account. In the second case, a Neural Network is performed, where the input is the vector that represents the document that we want to classify and the output is a *softmax* layer that predicts the probability of belonging to a certain class.

We have a control group, but we cannot train our model with this data to classify elements that do not belong to any class, because this control group is not representative of every element of this kind. To predict elements in the control group, a threshold has been performed, and when the probability of belonging to the majoritarian class is not enough (less than the threshold), no class is predicted.

Finally, some metrics such as accuracy, precision, recall, F1-score, ROC curves,

precision-recall curves, confusion matrices and AUROC have been performed or calculated to guess which of these algorithms are better. Generally, all the indicators state that the neural network has a better performance. One single metric, F1-score, states that kNN is better, so we can understand that kNN is better balanced between classes, but not enough (as we can see regarding the rest of the metrics).

References

[1] Text categorization corpora. http://disi.unitn.it/moschitti/corpora.htm. Accessed: 2022-04-04.

[2] N. R. B. de Araujo and R. Baeza-Yates. *Modern Information Retrieval*. Pearson Higher Education, 2010.

[3] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2018.