Available online at http://icsejournal.com/index.php/JCSE
Journal of Computer Science an Engineering (JCSE)
e-ISSN 2721-0251
Vol. 5, No. 2, August 2024, pp. 116-136

Class-Oriented Text Vectorization for Text Classification: Case Study of Job Offer Classification

Ghislain Wabo Tatchum[1*], Armel Jacques Nzekon Nzeko'o[2], Fritz Sosso Makembe[3], Xaviera Youh Djam[4]

[1,2,3,4]University of Yaounde I, Faculty of Sciences, Computer Science Department, Yaounde, 812 Cameroon.
[2]Sorbonne University, IRD, UMI 209 UMMISCO Bondy, F-93143 France.
[1]ghislain.wabo@facsciences-uy1.cm*; [2] armel.nzekon@facsciences-uy1.cm; [3] fritz-oswald.makembe@facsciences-uy1.cm; [4]xaviera.kimbi@facsciences-uy1.cm
* corresponding author

ARTICLE INFO

Article History:
Received July 24, 2024
Revised August 28, 2024
Accepted August 30, 2024

Keywords:
Supervised machine learning
Natural Language Processing (NLP)
Text classification
Class-oriented text vectorization
Pre-processing

Correspondence:
E-mail: ghislain.wabo@facsciences-uy1.cm

ABSTRACT

Advances in data science have made it possible to solve many real-life problems using automatic text classification applications. This is the case in e-recruitment, where job offers are classified and recommended to jobseekers. In natural language processing, text classification involves a vectorization step, whereby each document is represented as a vector of coordinates linked to a keyword. Those keywords are obtained by vectorizing the entire corpus, and are used to distinguish one document from another in the corpus. However, it is preferable for each keyword to distinguish one class from another. To obtain these types of keywords, the authors consider the class of documents in the vectorization process. They first create a class-oriented document for each class by merging all documents from the same class, and then apply a vectorization algorithm. Experiments are carried out using datasets from Minajobs, Nigham, and Monster with the classification models Decision Tree, Naive Bayes, Support Vector Machine, and a deep neural network self-attention transformer (TFM). The vectorization methods used on class-oriented documents are Doc2Vec and TF-IDF combined with our class-oriented vectorization strategies, including OC, ZIPF, and OWDC. To evaluate these experiments, we used the precision, MAP, and F1-Score metrics. According to the results, the TFM methods can improve accuracy by 29, 40, and 33% compared to previous work and the traditional way of classifying text documents. The NB methods can improve accuracy by 19, 22, and 20%, while the DT methods can improve accuracy by 34, 37, and 34%. The SVM methods can improve accuracy by 33, 34, and 34% in the Monster, Nigham, and Minajobs datasets. In addition, we validate our contribution by comparing ourselves with three other works in the literature using four datasets (RE'16, Wap, WebKB, and Kla) and obtain improvements in accuracy and F1-score up to 55%.

1. Introduction

The task of text classification consists of cataloguing text into a set of categories corresponding to predefined labels. It is an indispensable task in Natural Language Processing, which has a wide spectrum of applications in fields like sentiment analysis which is a field of application of automatic language processing that tries to detect translated emotions in text, generally whether they are positive or negative, has been explored by Liang et al. [1], by Cheng et al. [2] and also in the work of Storey and O'Leary [3]. Another application is question answering, a task consisting in automatically answering questions posed by humans - a topic addressed in the work of Calijorne et al. [4]. Or topic labelling, an NLP task as applied by Hao et al. [5] to Chinese text.

We can also mention job classification, a NLP task, as job offers are text-based and therefore require language processing to classify them. This is addressed in works such as those by Parmar et al. [6] or Akter et al. [7]. In the age of the explosion of digital technologies, and with an abundance of information, the classification task is proving to be a time-consuming challenge. When it is done by humans, tiredness and lack of knowledge in the field influence the results negatively. For thoses reasons machine learning models are better to automate the task of text classification to optimise execution time and efficiency on large volumes of data.

Classification of textual data requires some sort of mathematics pre-treatment to transform it into supervised machine learning understandable form. The performance of classification models depends on their understanding of textual data. It is imperative to have good vectorization if we expect good data representation and therefore good classification. There are several works such as Jin et al. [8] and Tiun et al. [9], who initiated the idea of class-oriented vectorization but with features and meta-data or works such as [10,11] that have dealt with text classification but on employment data or works of Afandi et al. [12] who have tackled the field of text classification based on social data (texts obtained from social networks such as tweeter and instagram). They all address text classification according to a standard scheme illustrated in Figure 1. In all these works, the vectorization process involved providing a vector representation of the job offers. These vectors are, in a base vector space, the keywords extracted from the entire corpus as features, and thus discriminate the offers from one another. However, in the context of text classification, it would be desirable for vector representations of job offers to be able to discriminate not one from another, but rather each class (in this case, user profiles) from the others. In other words, the keywords should be more representative of the classes than of the offers, to better distinguish one class from another and also take into account all classes, even the under-represented ones; referring here to class-oriented text vectorization.

Moreover, Information and Communication Technologies (ICTs) have transformed the traditional job supply and demand market into a new online form through e-recruitment platforms. Hjort et al. [14], demonstrate the benefits in some African countries of e-recruitment platforms. In the same vein, Suvankulov et al. [15], demonstrate the benefits both for users of these platforms, who see their probability of obtaining other jobs increase, and for recruiters, who save an enormous amount of time in the recruitment process. E-recruitment platforms solve a number of problems for recruiters, by reducing the time it takes to analyse applications and the costs associated with advertising, while at the same time increasing the amount of information processed. This has been made possible by the job recommendation systems for the right profile in the platform.

Therefore, in this work, the authors decide to apply class-oriented text vectorization to text classification methods in the context of job offer classification. Indeed, the classification of job offers is a domain to which the authors have applied class-oriented text vectoring, but not being the only domain, the authors propose to vary the types of classification algorithms according to the models on which they are based, to show the generalizability of the contribution.

In this work, recommendation systems is define as a subsets of information screening, consisting of predicting the suitability of an element for a user, with the aim of proposing a set of elements that are likely to be of interest to him. In this case, the aim is to provide a list of job offers for a given user profile. In order for the machine learning models to well understanding jobs offers, the jobs must be well vectorized. Note that as a prelude to a recommendation system, the job offer recommendation problem in this article is approach as a classification problem. Here, the authors consider our job offer classes as profiles, and the top-n recommendation of job offers here consists in classifying them, but this time in descending order of class membership probabilities.

To this end, this work proposes to extend the conference work previously presented [16], which initiated the beginnings of class-oriented document set vectorization and thus a more optimal vector representation of documents. In this work, classification is performed using two machine learning algorithms (decision tree and naive bayes) applied to two datasets named Minajobs and Nigham. In this extension of the work, we first introduce new parameters whose variation highlights the impact of class-oriented document size, the number of keywords in class-oriented document collections and the different combinations of approaches on document classification performance. In order to show that the idea is generalizable and reproducible, two other classification algorithms are added to the previous two, namely the Support Vector Machine (SVM) algorithm and the neural networks with attention mechanisms or transformers (hereinafter referred to as TFM). Following on from the previous work [16], a complete and detailed presentation of the idea is proposed and a dataset named Monster is added to the experimental data. Finally, in order to test the robustness of the proposal in this paper and to validate it, we decices to compare ourselves with three previous works using four datasets (RE'16, Wap, WebKB and Kla).

For the rest of this paper, the authors continue by presenting firstly a state of the art on text classification, vectorization techniques and text classification methods, secondly the standard text classification process, thirdly they are presenting the vectorization of class-oriented documents, fourthly the experiments and their results and finally fifthly a conclusion is made.

2. Methods

In this section, the authors present the state of the art of the various existing methods, followed by a presentation of the classic text classification process, and finally a presentation of vectorization on class-oriented documents.

2.1. Background of Text Classification, Vectorization Techniques and Classification Algorithms

We will start this sub-section by presenting some existing work on text classification, followed by some existing techniques for text vectorization. And finally, some learning algorithms applied on text classification problems in the previous works.

2.1.1. Background on Text Classification

Many works have focused on the text classification task with the aim of proposing solutions to improve the efficiency of this task. For example, previous work by Tiun et al. [10], focuses on text classification by comparing several combinations of text vectorization techniques and classification algorithms on a requirements dataset. Their experiments show that techniques such as TF-IDF, FastText and Word2Vec perform well in combination with Logistic Regression (LR) Convolutional Neural Networks (CNN) and SVM algorithms.

Other works, notably those by Cunha et al. [10], are also interested in text classification. More explicitly, they demonstrate that the efficiency of text classification systems depends as much on the classification algorithms as, if not more so, on the preprocessing tasks prior to training. To this end, they introduce a new step in the traditional text classification pipeline, which reduces the size of the resulting vector representation while increasing density and reducing the number of training instances. The result is an increase in performance and a reduction in text classification time.

In these earlier works, the vectorization step remains traditional, although Cunha et al. [10], add a new step just after vectorization to reduce dimensionality, it nevertheless remains a classical vectorization over the whole corpus. To this end, Jin et al. [8], initiate the idea of class-oriented vectorization by proposing to add to the traditional vectorization process the consideration of feature class frequencies, which optimises representation and increases classification performance. However, the fact that this modification is made in the vectorization technique and applied to the entire corpus still poses limits in terms of the representativeness of all classes and well-discriminating representations.

2.1.2. Vectorization Techniques for Classification

In case of classification problems where datasets are in textual form, one of the first challenges is vectorization. Indeed, machine learning algorithms for text classification do not take text as input, but rather a vector representation of that text. Vectorization, in the case of textual data, is the placement of this text in a representation space (vector space) which will enable the comprehension and assimilation of this data by machine learning models. Textual data is usually a collection of documents containing hundreds, thousands or even millions of words, so the aim is to provide a representation in a smaller space indexed by keywords, so as to preserve as much of the original information as possible. Good vectorization is therefore essential for good understanding of the text by machine algorithms, and therefore for good classification. Several vectorization methods have been proposed in the literature, the authors present just a few of them below.

a) Word Count

In this method, the documents are represented by the occurrences in the document of the words in the corpus. Indeed, considering a corpus consisting of 04 job offers (documents) {J_1, J_2, J_3, J_4}, with a dictionary of 05 words {*A*, *B*, *I*, *K*, *L*} such that:

J_1 : *B K A B A K I I;*
J_2 : *A L I B A L I B A B;*
J_3 : *B I B I B A L A;*
J_4 : *K L L K L I L K K K I L I I;*

The representation of this corpus using this method shown in table 1 :

Table 1. An example of word count

Documents	A	B	I	K	L
J_1	2	2	3	1	2
J_2	3	3	2	0	2
J_3	2	3	2	0	1
J_4	0	0	4	5	5

This method is very simple to design, but quickly becomes inappropriate as the corpus grows and the dictionary increases, because we end up with very large vectors of a size equal to the cardinal of the dictionary, and also a lot of hollow vectors.

b) TF-IDF

In an attempt to overcome the limitations of the previous technique (Word count), the TF-IDF technique attempts to select important words according to a coefficient that is both proportional to the occurrence of the word in a document and inversely proportional to the occurrence of the word in the entire corpus. The term TF-IDF therefore stands for term frequency and inverse document frequency. The formula proposed by Sabri et al. [17], is as following :

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \tag{1}$$

t : terms, d : document, D : documents collection.
Here *tf(t,d)* is the function that provides the frequency of *t* in *d*, as defined following :

$$tf(t,d) = \frac{count(t)}{|d|} \tag{2}$$

count(t) : occurrences of t , $|d|$ *: number of all words in d.*

The idf is defined as following:

$$idf(t,D) = log\left(\frac{|d|}{1+|\{d\in D: t\in d\}|}\right) \tag{3}$$

$|d|$: size of the document space, $|\{d \in D: t \in d\}|$: occurrences of in d.

c) Word2Vec

This is a vector-based word representation method based on neural networks. It is made up of three layers (hidden, input and output): Common Bag Of Words (CBOW) and Skip-gram. The idea is to rely on the context of adjacent words in a corpus and identify similar words based on their representation. Kowsari et al. [18], present this method in more detail, specifying inputs and outputs for each layer, and also the task and roles of each layers.

This method represents words by vectors in such a way as to highlight the relationship between words, i.e. vectors of similar words will have close values. Here, the authors want to represent documents as collections of words. Such a method will lead us to have a document representation in the form of a matrix set of words instead of a single vector.

d) Doc2Vec

Unlike the Word2Vec method, which represents a document as a matrix of vectors representing its words, Doc2Vec will provide a single vector representation for a document. This is a neural approach to unsupervised learning, providing a vector representation for sequences of words of variable length, or phrases. They are two main Doc2Vec versions: Distributed Bag of Words (DBOW) and Distributed Memory (DM) [10].

The first variant, known as DM for Distributed Memory, is based on the principle of learning to construct a vector of invariable size as a representation from a portion of text and its context. Consisting of a projection layer (whose role is to create word/document vectors during learning) and an output layer (whose role is to predict the target word from the distributed context representation), it is then passed a document and its context as input.

The second variant, DBOW or Distributed Bag of Words, relies more on the structure and distribution of words in the document than on their meaning. The DBOW version is interesting for extracting the distribution properties of words in the corpus, and is easier and faster than the DM version, which is more efficacious for extracting the meaning and significance of words.

Doc2Vec is therefore a technique that allows a single representation vector to be obtained for a document. It is therefore less space-hungry than the Word2Vec technique. Doc2Vec reduces memory requirements up to 80% compared with Word2Vec, while remaining just as efficient [16].

These vectorization techniques are basic, but other works [20, 21, 22] have tried to ameliorate classification algorithms performances by using deep learning models both as classifiers and as vectorizers (feature extractors), In some cases, they have been able to upgrade the results of the classification algorithms, but generally, the characteristics obtained by deep learning are not explicable, and above all, they do not use the basic methods and also the upstream knowledge of the document classes, since this is supervised learning.

2.1.3. Models of Machine Learning

They are several text classification algorithms based on machine learning proposed in literature, each operating on different principles but based on knowledge learned from the data. Some of the models used in this work are shown below.

a) Naive Bayes

The problem of classification can be likened to a selection of hypotheses (h) given a set of data (d). The Naive Bayes model uses Bayes' theorem to calculate the probability that a hypothesis is the best given a set of data. Bayes' theorem is stated as follows:

$$P(H \vee d) = \frac{P(d \vee H) * P(H)}{P(d)} \quad (4)$$

$P(H \vee d)$: Probability of H with knowledge of data d; $P(d \vee H)$: probability of the data knowing H; $P(H)$: Hypotheses H probability; $P(d)$: data d probability.

Training a naive Bayes model is simply a matter of calculating the probabilities of each class for different documents, so there's no need for parameter adjustments or optimizations.

Advantages: naive bayes is relatively simple to understand and construct, explicable, fast and easy to train, even with a small dataset.

Disadvantages: because of the assumption of word independence in this model, it's often referred to as naive or simple. In general, this type of algorithm can do the same classification work as the other algorithms that already exist, but its performance is limited when it comes to large amounts of data to process, because as data grows, so do the dependencies between the set of words, and therefore, the verification of the Naïve Bayes hypothesis decreases.
It remains one of the easiest classification algorithms to implement, since it is based solely on a probability law, and its use is also justified by the fact that this paper is an extension of previous work [16] which used this algorithm. The extension must therefore be at least as effective as the algorithms previously used.

b) Decisions Tree

It's a tree structure similar to an organisational chart, in which an internal node represents a feature, the branch represents a decision rule, and each leaf node represents the result. The highest node is the root node. The aim of this model is to create a decision tree that performs a criterion-by-criterion analysis. Significant criteria are determined according to statistical weights of the values as present in some previous work [31].

A decision tree classifies a document, starting at the root of the tree and moving successively downwards through the branches whose conditions are satisfied by the document, until a leaf node is reached. The class that labels the leaf node is assigned to the document.

There are several decision tree algorithms [23], of which the most popular are: (i) ID3 (Iterative Dichotomiser 3) : which, as its name suggests, performs a dichotomous traversal of the tree, each time exploiting entropy and gain information to select the correct branch; (ii) C4.5 : which uses gain information or gain ratios to evaluate splitting points within decision trees. This is a later version; (iii) CART: The term, CART, hense for "Classification And Regression Trees" this algorithm use a measure names Gini impurity to identify the ideal attribute for splitting.

The decision tree model is one of the most explicable machine learning models, and is often used in work on the explicability of black-box models. Other models, such as random drills and set models, are also based on its working principle, which justifies its use in this work.

c) Support Vector Machine (SVM)

In machine learning, the svm algorithm is one of the most widely used models; it was initially built for binary classification problems [9] but was soon adapted for multi-class classification problems [18]. In principle, this algorithm attempts to construct a hyperplane that will separate the data to be classified as best as possible, so as to maximize the distance between the data closest to the hyperplane on either side. The dimension of the hyperplane is a function of the number of classes. The theoretical formalization of this model is given below [8, 10]:

$$f(x) = \underset{t}{argmax}\left(\sum_f f_t, f(x)\right) \tag{5}$$

Intuitively, the solution to a k-class classification problem is to construct a decision function for all k classes at once, which amounts to an optimization problem defined as follows:

$$\underset{w_1, w_2, \ldots, w_k, \zeta}{min} \frac{1}{2} \sum_k w_k^T w_k + c \sum_{(x_t, y_t) \in D} \zeta_t \tag{6}$$

st.

$$w_{y_t}^T x - w_k^T \leq i - \zeta_t,$$
$$\forall (x_t, y_t) \in D, k \in \{1,2 \ldots K\}, k \neq y_t.$$

where (x_i, y_i) represent the training data points such that $(x_i, y_i) \in D$, C is the penalty parameter, ζ is a slack parameter, and k stands for the class.

d) Neural Network : Transformers

In NLP, deep neural network models such as CNNs and RNNs have often been used for text classification, but recently they have begun to be replaced by neural networks with attention mechanisms (self attention)[13]. Attention mechanisms enable transformers to easily pass information through input sequences. In their general architecture, transformers consist of decoders and encoders with fully connected layers [24]. The encoder is the input and constructs an optimized representation of the input it receives, while the decoder uses this representation and other attributes to provide an output.

Our standard model will consist of input, dropout and output layers. To these layers we've added dense layers with softmax and Relu activation functions respectively, as well as a 1D global average pooling layer of course, as this is the text. The training parameters used were the adam activation function and the sparse_categorical_crossentropy loss function for a batch size of 32 and a number of epochs of 40.

e) Summary of state of art.

After presenting a review of all the existing methods and techniques in the field of text classification, a summary of the work that has focused on increasing the performance of classification algorithms by trying to optimize the pre-processing and vectorization stage is presented in Table 2.

Table 2. summary of the literature on text vectorization for classification

Authors	Problems	Approaches	limits
Tiun et al. [10]	Classification of functional and non-functional requirement in software requirement	Word2vec and fast Text techniques for vectorization	Vectorization is performed on the entire corpus, which can discriminate between certain classes and give a non-optimal representation.
Cunha et al. [9]	Improve text vectorization to boost the performance of text classification algorithms.	Extended pre-processing pipeline: On the role of meta-feature representations, sparsification and selective sampling	The metadata extracted here do not allow us to make a representation of documents that discriminates them by class, but rather from one another.
Jin et al. [8]	Optimize text vectorization to improve the performance of text classification algorithms.	Feature selection based on absolute deviation factor for text classification.	The features extracted enable us to discriminate documents not by class, but rather from one another.
Afandi et al. [12]	Detect emotions in social network comments (sentiments analysis)	Text classification with classical text vectorization using the TF-IDF technique	Vectorization is performed on the entire corpus, which may discriminate certain classes and result in a non-optimal representation.

2.2. Classic Flowchart of Text Classification

Figure 1 summarizes the main stages of a text classification system with machine learning algorithms. In this process, the authors start generally by set of cleaning and preprocessing tasks on the data, in this case job offers. These tasks may involve transforming the offers into a list of words known as tokens (this is tokenization), then into a list of radical words known as sterms (in this case, sterming), and finally removing unwanted elements from these lists. These are words, sterms or characters that are superfluous or don't provide any significant information.

After this step, they transform the lists of words (sterms) into numerical vectors (vectorization), traditionally by selecting keywords (features) from the entire corpus and representing the offers in the space formed by these keywords. Techniques such as TF-IDF[32] and Doc2Vec are used. This is followed by the actual classification, using machine learning algorithms to produce lists of offers by class (user profiles) and thus a top-n recommendation.

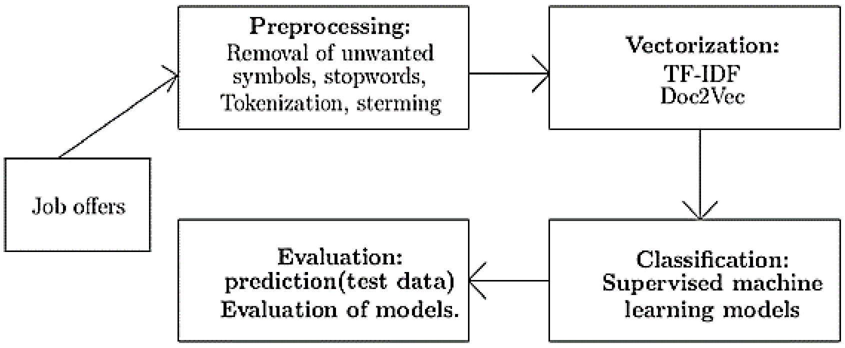


Figure 1. Standard flowchart of the text classification process with machine learning algorithms.

2.2.1. Preprocessing of Job Offers

In data science, before using the data, an essential step is to pre-process it. Generally, data sets obtained directly after collection contain a lot of anomalies and need to be pre-processed. This is an important step because the data quality has a direct influence on the results of the processing carried out on it. This has also been illustrated in the case of text classification that the text pre-processing has an important influence on the classification algorithms performance [25, 12].

In this stage, the unwanted terms are removed, words and characters (i.e. those not providing information relevant to our task). Then, as in the previous work [26, 27], the offers are transformed into a list of words without separators by tokenization, and then extracted the roots of compound words to leave only a list of sterms.

2.2.2. Vectorization

In this step, the text are plunged from their initial representation into a vector representation in a new, reduced space. In other words, learning from the entire corpus, extract keywords (features) and use them as the basis for the new representation space, with coordinate values calculated variably according to the vectorization techniques used. Inspired by previous work [28], the following techniques are use : TF-IDF and Doc2Vec. This choice was motivated by the fact that, in the literature [12, 13, 28], these techniques have been shown to be among the best vectorization techniques, and also because the TF-IDF technique, being frequency-based, is an explicable one (i.e. a happy medium between efficiency and explicability), but also because the new vectorization paradigm also works well on deep learning-based techniques: Doc2Vec.

2.2.3. Supervised Machine Learning Models for Classification and Top-N Recommendation

To match job offers and user profiles, the job offer recommendation task are assimilate to a classification of offers in classes representing user profiles. So, machine learning algorithms that were trained on a proportion of the data and then tested on the other proportionhave been used. After testing, we had offers arranged in classes with probabilities of belonging to these classes, so from these probabilities a ranking is made to provide the top-N recommendation.

2.3. Vectorization on Class-Oriented Document Collection

In this sub- section, we present in greater detail the operating mechanism of class-oriented document vectorization, starting with architecture, followed by parameterizations, then by the definition of each of class-oriented vectorization strategies, and finally by classification and recommendation.

2.3.1. General Architecture

In the classic text vectorization the entire corpus are passing under its textual representation, directly to the vectorization techniques, which will select from the entire corpus the keywords that will be the features of the numerical representation. Except that in some cases the corpus is unbalanced (certain classes are under-represented), resulting in a non-representative selection of keywords and therefore poor vectorization. What's more, even when the corpus is balanced, the keywords used here are generally the most recurrent ones, or those that only serve to distinguish one offer from another, and therefore does not characterize the classes themselves.

That's why a class-oriented approach to text vectorization is proposing, which adds a new step to the classic vectorization process by creating class-oriented documents (a set of keywords or features) that will be passed on to the vectorization techniques. More explicitly, instead of passing documents directly to the vectorization technique, they start by creating a class-oriented document of the same size for each class (profile), containing the set of words that characterize it and therefore distinguish it from other classes. These documents are then used for vectorization. This ensures that each class can be represented in the same way as the others, and above all that the digital representations of the initial documents are in a vector space that is representative and discriminating by class, since this is the task of classification.

Figure 2 shows the new text classification process (job offers in our case) with class-oriented text vectorization. It clearly shows new step added to the classic scheme presented earlier in figure 1.

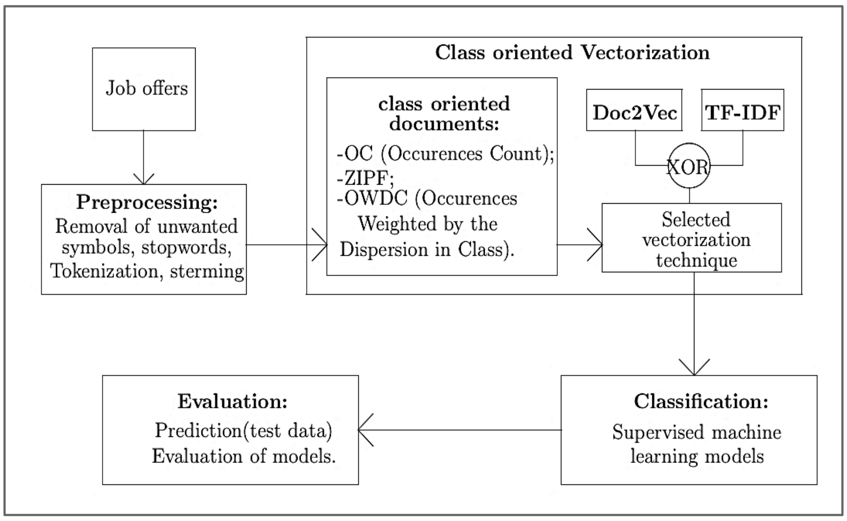

Job offers
Class oriented Vectorization
class oriented documents:
-OC (Occurences Count);
-ZIPF;
-OWDC (Occurences Weighted by the Dispersion in Class).
Doc2Vec
TF-IDF
XOR
Preprocessing:
Removal of unwanted symbols, stopwords, Tokenization, sterming
Selected vectorization technique
Evaluation:
Prediction(test data)
Evaluation of models.
Classification:
Supervised machine learning models

Figure 2. Flowchart of the text classification process using machine learning algorithms with the vectorization step applied to class-oriented documents *collection.*

2.3.2. Experimental settings

In this work, three class-oriented vectorization strategies were used. Each of these strategies uses a set of parameters, which we'll start by defining here : (i) **Parameter K**: this is an integer taking its value from the set {100, 300, 500, 1000, 1300, 1500, 2000, 2300, 2500, 3000}. It's the size of the class-oriented documents or in other words, it's the number of keywords chosen per class to form the class-sorted document. (ii) **Parameter P**: This too is an integer, whose values are in the set {10, 30, 50, 100, 300, 500, 1000, 1500, 2000, 3000}. Except that this time it's the size of a vectorized offer. In other words, it's the length of the vector representation of the offers. (iii) **Parameter e**: referring to the keyword rebalancing strategy in the event of a word deficit in a class for a high K value. e is included in the set {1, 2, 3}, with the meanings randomly duplicating words in the event of deficits to fill in, randomly reducing other class-oriented documents to be at the same level, leaving the respective imbalance for values 1, 2 and 3.

2.3.3. Class Oriented Document

Occurrence Count (OC) : to form the class-oriented document with the *K* most frequent words in each class. We assume that the more frequent a word is in a class, the more it discriminates from the others. We no longer vectorize all the documents in each class, but rather, for each class, the document made up of *K* words, the most frequent in all the documents in each class. However, this strategy can quickly prove limited when several words are frequent in several classes at the same time. These words are no longer discriminating in these cases, and therefore not interesting as keywords.

Zipf law (Zipf): In order to resolve the limitation posed by the OC strategy, the latter proposes to form class-oriented documents by the *K* words with the highest ZIPF score (defined in [29]) in each class. This score inversely weights the frequency of words by their rank when they are ranked in descending order of frequency. Although this technique rebalances the distribution of keywords, it doesn't completely solve the previous problem. Indeed, many words still manage to have a good ZIPF score despite being in several other classes, particularly when the class is not very large.

Occurrences Weighted by the Dispersion in the Class (OWDC): In this strategy, word frequencies is penalize by the number of classes in which they are found. A new score *C(m)* is define given by the equation 7, *m* being a word. For each class, the class-oriented documents will be made up of the words with the highest score *C(m)*, i.e. the most frequent words in the class and the least frequent in other classes.

$$C(m) = \frac{WordOccurenceInClass * NumbersOfJobsContainingWords}{NumbersOfJobsInClass} \tag{7}$$

Let's assume, as shown in figure 3, a corpus made up of four classes C1, C2, C3 and C4, in a vocabulary of words made up of {A, B, C, D, E, F, I, K, L, X, Y}. The classes each possess documents, but have specific words that discriminate them from one another. For C1 we have the words A and B, for C2 we have C and D, F and E for C3 and X and Y for C4. The standard process can provide us with feature sets of words (I, K, L, E, X, F, Y, C) that represent the corpus, but not the individual classes. We can also see that the fact that classes C1 and C2 don't have enough documents discriminates them. Moreover, words like I, K and L are present in all classes and therefore have no discriminating information.

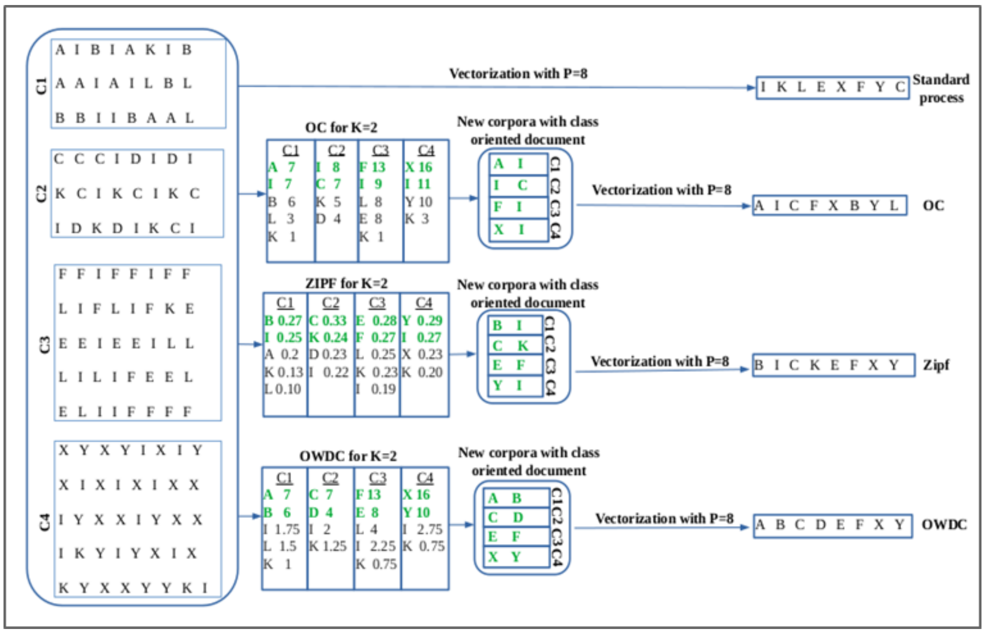


Figure 3. Scheme showing how text vectorization works on a class-oriented document collection, applied to a corpus of four classes whose documents are sets of words (letters). The class-oriented document collection is shown in green.

The selection of keywords for vectorization are significantly upgraded by the use of strategies. We can see for example, the OC strategy prioritizes the most occurrently words in classes, allowing each class to be represented in the keywords, even though uninteresting words such as I and L remain. The ZIPF strategy inversely prioritizes the words of each class by their rank, improving keyword choice despite the fact that I and K remain in the selection. Finally, OWDC selects only those words that exactly characterize each class.

2.3.4. Classification and Top-n recommendation in our case.

Once the new vectorization paradigm has been applied and obtained the vector representations, we pass them on to machine learning algorithms including: Naive bayes, decision tree, svm, neural network: transformers. These algorithms will learn from the vector representations of the offers, and classify the offers into classes based on the trained models, with a view to a future top-n recommendation. In this way, we'll be able to see the impact of this new paradigm on the performance of our algorithms.

In order to show that the proposed new vectorization paradigm is generalizable, the authors proposed to apply it to machine learning algorithms based on different models. The Naive Bayes algorithm, which is a probabilistic model, the decision tree algorithm, which is the basic model of randomized drills and boosting algorithms, the SVM algorithm, which is a regression model, and neural networks, which are deep learning models. It also showed that the contribution can be applied to both explainable and black-box models.

The top-n recommendation problem here is assimilated to a classification task, where the profiles are the classes. We therefore classify job offers in the standard way, arranging them by class and thus by profile. In addition, in each class (for each profile), the jobs that have been classified are ranked in descending order of the probability provided by the classification algorithm, and we thus obtain our top-N recommendation.

3. Results and Discussion

This section first presents the datasets used, then the evaluation protocol and results.

3.1. Datasets Used

For our experiments, three datasets are proposed, namely “Monster”, “Nigham” and “Minajobs”, these names were given according to their sources. The Monster dataset is a set of 679 job offers in the IT field scoped on the Monster[1] job publication site, and divided into three classes C1 for software architect, C2 for graphic designer and C3 for software engineering. The Nigham dataset is a large dataset used in previous work [11] and available on kaggle[2], from which we extracted 43,083 IT job offers divided into 07 classes including C1 for Developer, C2 for Digital Marketing, C3 for Designer, C4 for IT security, C5 for Computer maintenance, C6 for Community manager and C7 for Network Administrator.

The Minajobs dataset is a set of 1,005 job offers scrapped on minajobs[3] a job publication site in cameroon, they are divided into 11 classes namely : C1 for Developer, C2 for Web master, C3 for Database manager, C4 for Analyst, C5 for Digital Marketing, C6 for Designer, C7 for IT security, C8 for Computer maintenance, C9 for Community manager, C10 for Archivist and finally C11 for System Administrator. For these three datasets, the choice to use only IT job offers was justified by the fact that the labelling was done by ourselves, and manually.

[1]https://www.monster.fr/ visited in April 2022

[2]https://www.kaggle.com/jsrshivam/job-recommendation-case-study visited in April 2022

[3]http://minajobs.net visited in June 2022

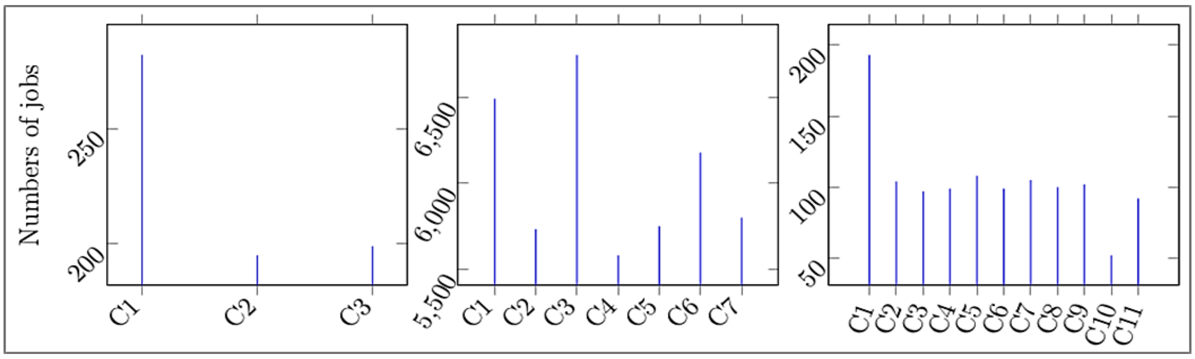


Figure 4. Diagram Showing the Distribution of Job Offers by Class in the Monster, Nigham and Minajob Corpora Respectively.

Figure 4 presents three bar charts showing the distribution of job offers by class for the three corpora Monster, Nigham and Minajobs. Each diagram is on a different scale, given the difference in proportion between the number of offers in each corpus. We can see from these diagrams that the corpora are unbalanced, with some classes over-represented (C1 for Monster, C1, C3 and C6 for Nigham and C1 for Minajobs) and others under-represented (C2 and C3 for Monster, C4 for Nigham and C10 for Minajobs). It should be noted that the authors only focused on IT offers because, as the labelling was done manually, they did not have any experience in other fields to help us do this.

3.2. Evaluation Protocol

To evaluate the work, each dataset is splited into two: 70% for training and 30% for testing. The classification test was evaluated using Precision and F1-score. But they didn't stop there. Based on the probabilities obtained by the classification models, they produced a recommendation using top-n values (5, 10, 50), which was evaluated using the MAP (Mean Average Precision) metric [30].

Thus, the precision metric is defined as a ratio between the number of well-classified or well-recommended offers and the total number of classifications or recommendations [31]; its formal definition is given by formula (8) with TP for true positives, FP for false positives and FN for false negatives. The F1-score metric is an average between the precision and recall metrics; formula (9) gives its formal definition [23].

$$Precision = \frac{\sum_{i=1}^{L} TP_i}{\sum_{i=1}^{L} TP_i + FP_i} \quad (8)$$

and

$$F1 - score = \frac{2*Precision*Recall}{Precision+Recall} = \frac{\sum_{i=1}^{L} 2TP_i}{\sum_{i=1}^{L} TP_i + FP_i + FN_i} \quad (9)$$

And the MAP metric is a sort of average of the precision averages, which enables us to evaluate the proposed recommendation approximation by taking into account not only the correct classifications (recommendation), but also their position in the top-N. It is defined as follows [23]:

$$MAP@N = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (10)$$

where :

$$AP@N = \frac{1}{GTP}\sum_{k=1}^{N} P@k * rel@k. \quad (11)$$

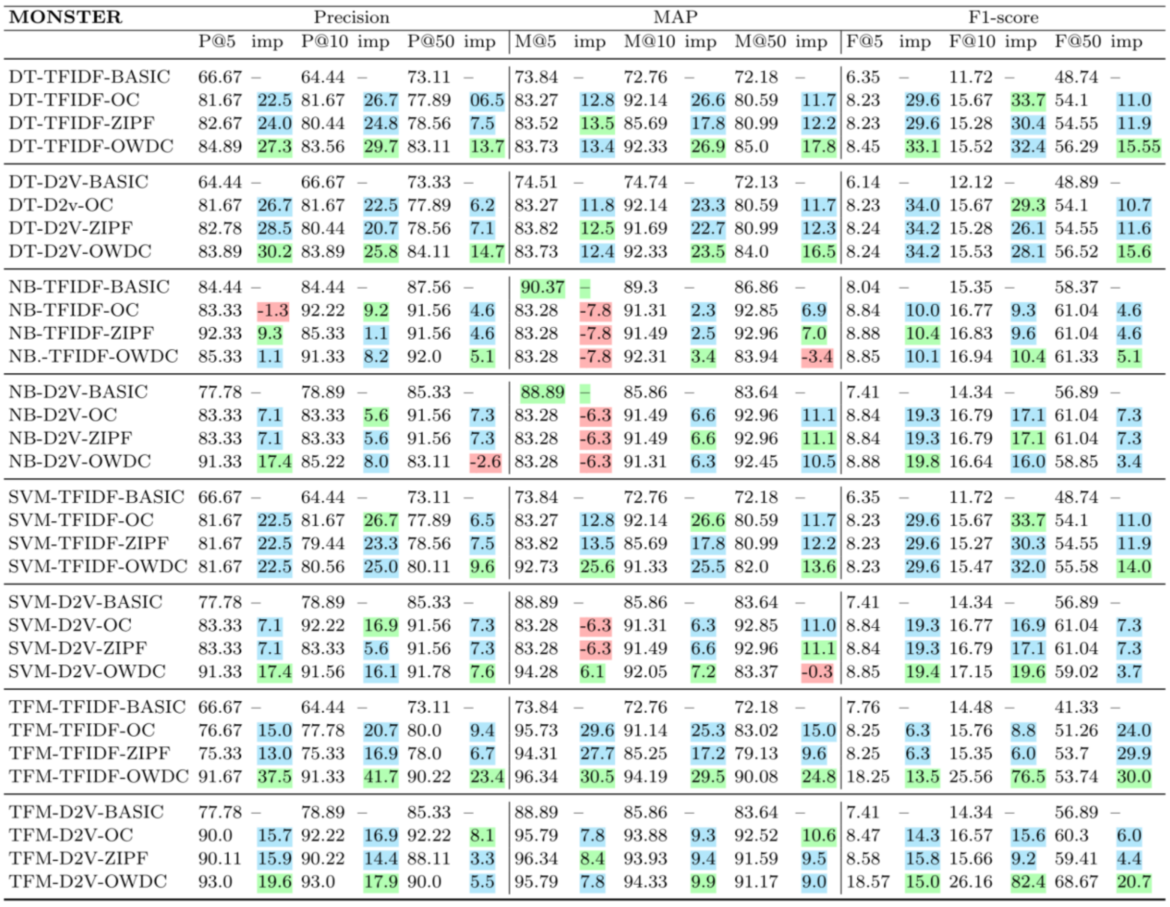
Where GTP is the total number of true positives, n is the number of documents of interest to the user, P@k is the precision at rank k, and rel@k is the relevance function which will be 1 if the document at rank k is relevant and 0 otherwise.

e-ISSN 2721-0251

3.3. Results and Comments

Figures 5, 6, 7 each represent tables, respectively for the Monster, Minajob and Nigham datasets, the best results of classification and ranking of job offers as a recommendation system; with the best combinations of parameters and metrics previously presented. Each table contains 08 sections with 09 blocks. First line in each section is the performance of machine learning algorithms (DT, NB, SVM and TFM) combined with classical vectorization (BASIC with Doc2Vec or TF-IDF). Each block of the different sections compares the first-line classification system (BASIC) with different proposed strategies (OC, ZIPF, OWDC) that make up our grant.

The use of the color green indicates that the classification system for this line has obtained the best result according to its metric and has no ties. The use of the color blue means that the line's classification system performs better than the base system, but has either ties or another system that is better than it. The color red is used when the performance of the classification system is below that of the base system. And white marks the basic performance.



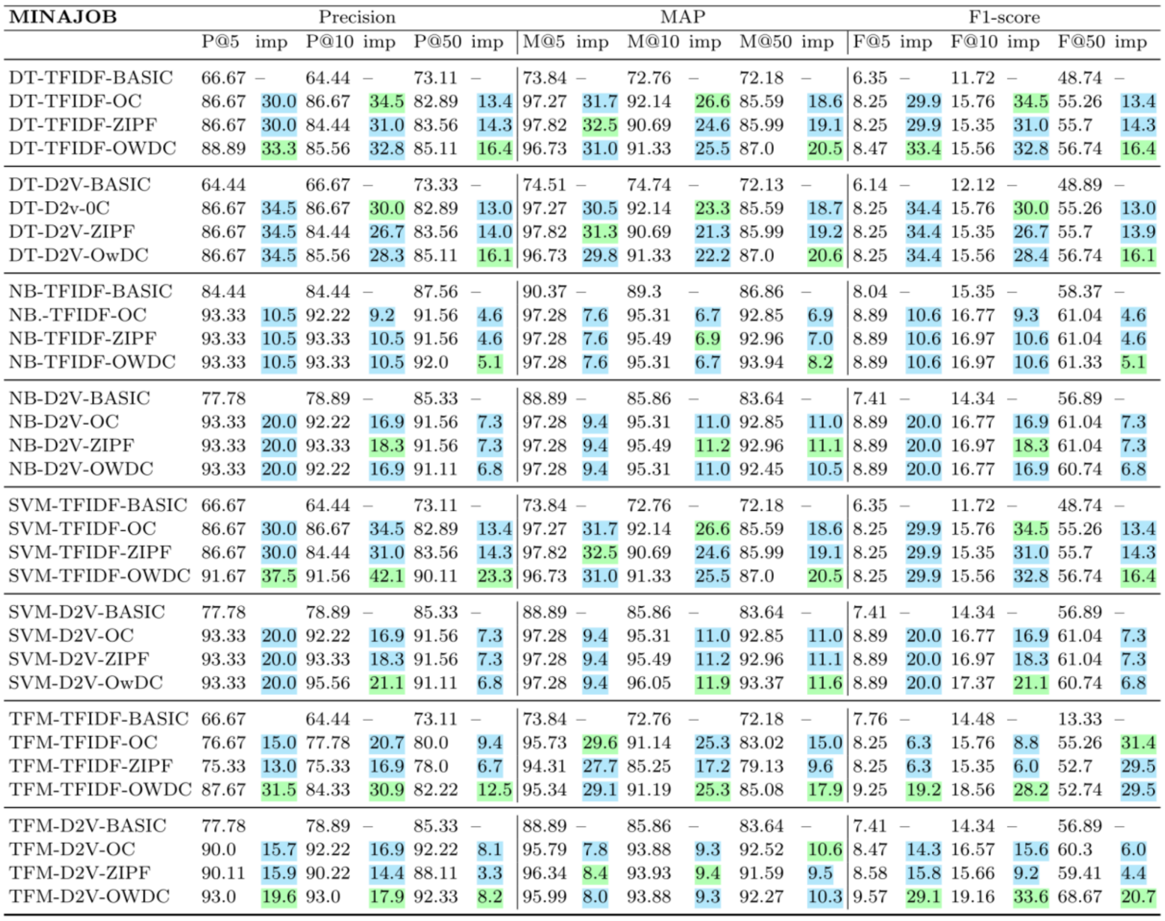
MONSTER	Precision						MAP						F1-score					
	P@5	imp	P@10	imp	P@50	imp	M@5	imp	M@10	imp	M@50	imp	F@5	imp	F@10	imp	F@50	imp
DT-TFIDF-BASIC	66.67	–	64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	6.35	–	11.72	–	48.74	–
DT-TFIDF-OC	81.67	22.5	81.67	26.7	77.89	06.5	83.27	12.8	92.14	26.6	80.59	11.7	8.23	29.6	15.67	33.7	54.1	11.0
DT-TFIDF-ZIPF	82.67	24.0	80.44	24.8	78.56	7.5	83.52	13.5	85.69	17.8	80.99	12.2	8.23	29.6	15.28	30.4	54.55	11.9
DT-TFIDF-OWDC	84.89	27.3	83.56	29.7	83.11	13.7	83.73	13.4	92.33	26.9	85.0	17.8	8.45	33.1	15.52	32.4	56.29	15.55
DT-D2V-BASIC	64.44	–	66.67	–	73.33	–	74.51	–	74.74	–	72.13	–	6.14	–	12.12	–	48.89	–
DT-D2v-OC	81.67	26.7	81.67	22.5	77.89	6.2	83.27	11.8	92.14	23.3	80.59	11.7	8.23	34.0	15.67	29.3	54.1	10.7
DT-D2V-ZIPF	82.78	28.5	80.44	20.7	78.56	7.1	83.82	12.5	91.69	22.7	80.99	12.3	8.24	34.2	15.28	26.1	54.55	11.6
DT-D2V-OWDC	83.89	30.2	83.89	25.8	84.11	14.7	83.73	12.4	92.33	23.5	84.0	16.5	8.24	34.2	15.53	28.1	56.52	15.6
NB-TFIDF-BASIC	84.44	–	84.44	–	87.56	–	90.37	–	89.3	–	86.86	–	8.04	–	15.35	–	58.37	–
NB-TFIDF-OC	83.33	-1.3	92.22	9.2	91.56	4.6	83.28	-7.8	91.31	2.3	92.85	6.9	8.84	10.0	16.77	9.3	61.04	4.6
NB-TFIDF-ZIPF	92.33	9.3	85.33	1.1	91.56	4.6	83.28	-7.8	91.49	2.5	92.96	7.0	8.88	10.4	16.83	9.6	61.04	4.6
NB.-TFIDF-OWDC	85.33	1.1	91.33	8.2	92.0	5.1	83.28	-7.8	92.31	3.4	83.94	-3.4	8.85	10.1	16.94	10.4	61.33	5.1
NB-D2V-BASIC	77.78	–	78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
NB-D2V-OC	83.33	7.1	83.33	5.6	91.56	7.3	83.28	-6.3	91.49	6.6	92.96	11.1	8.84	19.3	16.79	17.1	61.04	7.3
NB-D2V-ZIPF	83.33	7.1	83.33	5.6	91.56	7.3	83.28	-6.3	91.49	6.6	92.96	11.1	8.84	19.3	16.79	17.1	61.04	7.3
NB-D2V-OWDC	91.33	17.4	85.22	8.0	83.11	-2.6	83.28	-6.3	91.31	6.3	92.45	10.5	8.88	19.8	16.64	16.0	58.85	3.4
SVM-TFIDF-BASIC	66.67	–	64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	6.35	–	11.72	–	48.74	–
SVM-TFIDF-OC	81.67	22.5	81.67	26.7	77.89	6.5	83.27	12.8	92.14	26.6	80.59	11.7	8.23	29.6	15.67	33.7	54.1	11.0
SVM-TFIDF-ZIPF	81.67	22.5	79.44	23.3	78.56	7.5	83.82	13.5	85.69	17.8	80.99	12.2	8.23	29.6	15.27	30.3	54.55	11.9
SVM-TFIDF-OWDC	81.67	22.5	80.56	25.0	80.11	9.6	92.73	25.6	91.33	25.5	82.0	13.6	8.23	29.6	15.47	32.0	55.58	14.0
SVM-D2V-BASIC	77.78	–	78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
SVM-D2V-OC	83.33	7.1	92.22	16.9	91.56	7.3	83.28	-6.3	91.31	6.3	92.85	11.0	8.84	19.3	16.77	16.9	61.04	7.3
SVM-D2V-ZIPF	83.33	7.1	83.33	5.6	91.56	7.3	83.28	-6.3	91.49	6.6	92.96	11.1	8.84	19.3	16.79	17.1	61.04	7.3
SVM-D2V-OWDC	91.33	17.4	91.56	16.1	91.78	7.6	94.28	6.1	92.05	7.2	83.37	-0.3	8.85	19.4	17.15	19.6	59.02	3.7
TFM-TFIDF-BASIC	66.67	–	64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	7.76	–	14.48	–	41.33	–
TFM-TFIDF-OC	76.67	15.0	77.78	20.7	80.0	9.4	95.73	29.6	91.14	25.3	83.02	15.0	8.25	6.3	15.76	8.8	51.26	24.0
TFM-TFIDF-ZIPF	75.33	13.0	75.33	16.9	78.0	6.7	94.31	27.7	85.25	17.2	79.13	9.6	8.25	6.3	15.35	6.0	53.7	29.9
TFM-TFIDF-OWDC	91.67	37.5	91.33	41.7	90.22	23.4	96.34	30.5	94.19	29.5	90.08	24.8	18.25	13.5	25.56	76.5	53.74	30.0
TFM-D2V-BASIC	77.78	–	78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
TFM-D2V-OC	90.0	15.7	92.22	16.9	92.22	8.1	95.79	7.8	93.88	9.3	92.52	10.6	8.47	14.3	16.57	15.6	60.3	6.0
TFM-D2V-ZIPF	90.11	15.9	90.22	14.4	88.11	3.3	96.34	8.4	93.93	9.4	91.59	9.5	8.58	15.8	15.66	9.2	59.41	4.4
TFM-D2V-OWDC	93.0	19.6	93.0	17.9	90.0	5.5	95.79	7.8	94.33	9.9	91.17	9.0	18.57	15.0	26.16	82.4	68.67	20.7

Figure 5. Results of the Best Combinations of Parameters (Best Combination of P, T and e) of job offer Classification as a top-N Ranking on the Monster Corpora.

3.3.1. Description of Best Results Table

The figure 8 presents a table that contains 72 blocks of results where we compared BASIC to our contribution (OWDC, OC, ZIPF). This figure has a total of 216 cells, each one related to a block of the tables in figures 5, 6, 7 and contains the job classifier system that has the best performance in the associated block. We are not *imp.* the improvement compared to the classic classification system in the first line of the block (BASIC).

In Figure 8, the color blue indicates that in this section it's the OC class-oriented vectorization strategy that has obtained the best result, the color green it's used for the ZIPF strategy and yellow for the OWDC strategy. We leave in white with a “>” sign the sections where it's the basic system that has obtained the best results.



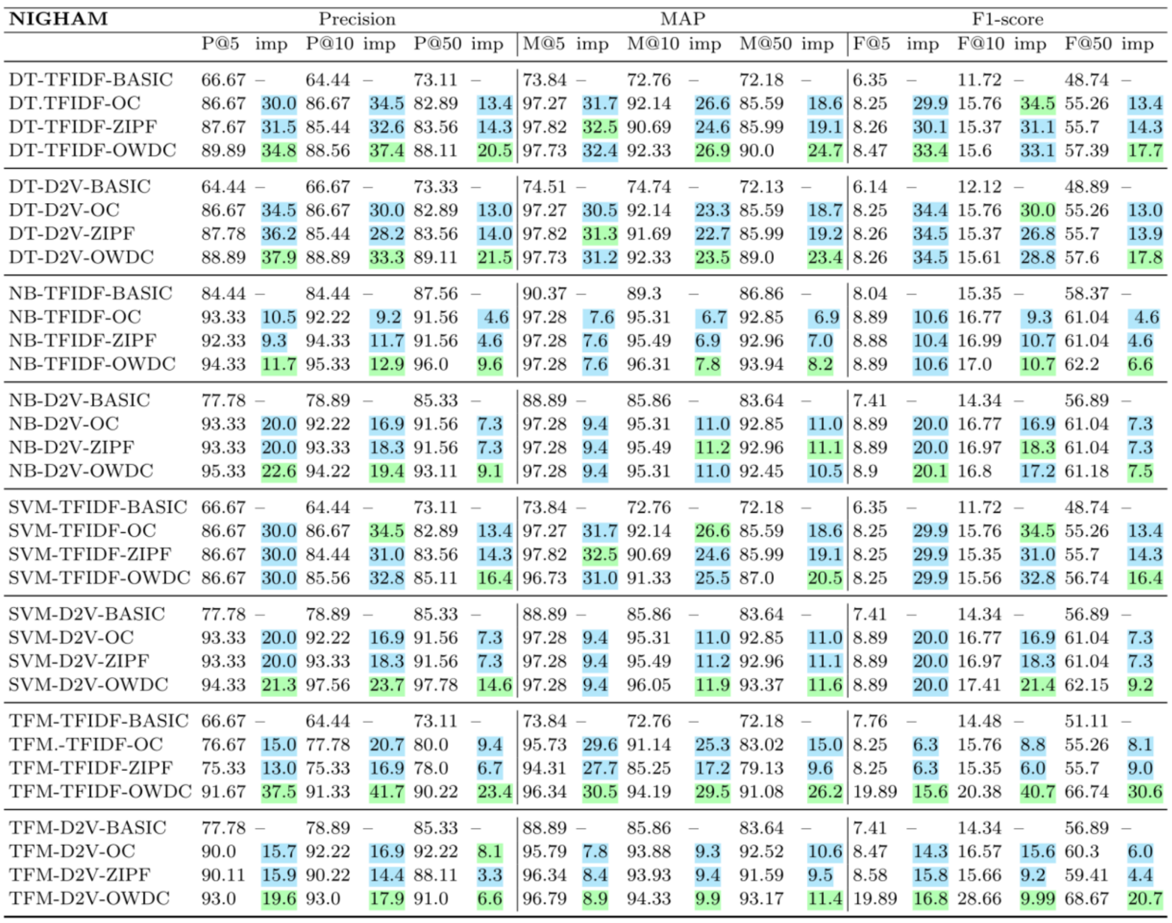
MINAJOB	Precision						MAP						F1-score					
	P@5	imp	P@10	imp	P@50	imp	M@5	imp	M@10	imp	M@50	imp	F@5	imp	F@10	imp	F@50	imp
DT-TFIDF-BASIC	66.67	–	64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	6.35	–	11.72	–	48.74	–
DT-TFIDF-OC	86.67	30.0	86.67	34.5	82.89	13.4	97.27	31.7	92.14	26.6	85.59	18.6	8.25	29.9	15.76	34.5	55.26	13.4
DT-TFIDF-ZIPF	86.67	30.0	84.44	31.0	83.56	14.3	97.82	32.5	90.69	24.6	85.99	19.1	8.25	29.9	15.35	31.0	55.7	14.3
DT-TFIDF-OWDC	88.89	33.3	85.56	32.8	85.11	16.4	96.73	31.0	91.33	25.5	87.0	20.5	8.47	33.4	15.56	32.8	56.74	16.4
DT-D2V-BASIC	64.44		66.67	–	73.33	–	74.51	–	74.74	–	72.13	–	6.14	–	12.12	–	48.89	–
DT-D2v-0C	86.67	34.5	86.67	30.0	82.89	13.0	97.27	30.5	92.14	23.3	85.59	18.7	8.25	34.4	15.76	30.0	55.26	13.0
DT-D2V-ZIPF	86.67	34.5	84.44	26.7	83.56	14.0	97.82	31.3	90.69	21.3	85.99	19.2	8.25	34.4	15.35	26.7	55.7	13.9
DT-D2V-OwDC	86.67	34.5	85.56	28.3	85.11	16.1	96.73	29.8	91.33	22.2	87.0	20.6	8.25	34.4	15.56	28.4	56.74	16.1
NB-TFIDF-BASIC	84.44		84.44	–	87.56	–	90.37	–	89.3	–	86.86	–	8.04	–	15.35	–	58.37	–
NB.-TFIDF-OC	93.33	10.5	92.22	9.2	91.56	4.6	97.28	7.6	95.31	6.7	92.85	6.9	8.89	10.6	16.77	9.3	61.04	4.6
NB-TFIDF-ZIPF	93.33	10.5	93.33	10.5	91.56	4.6	97.28	7.6	95.49	6.9	92.96	7.0	8.89	10.6	16.97	10.6	61.04	4.6
NB-TFIDF-OWDC	93.33	10.5	93.33	10.5	92.0	5.1	97.28	7.6	95.31	6.7	93.94	8.2	8.89	10.6	16.97	10.6	61.33	5.1
NB-D2V-BASIC	77.78		78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
NB-D2V-OC	93.33	20.0	92.22	16.9	91.56	7.3	97.28	9.4	95.31	11.0	92.85	11.0	8.89	20.0	16.77	16.9	61.04	7.3
NB-D2V-ZIPF	93.33	20.0	93.33	18.3	91.56	7.3	97.28	9.4	95.49	11.2	92.96	11.1	8.89	20.0	16.97	18.3	61.04	7.3
NB-D2V-OWDC	93.33	20.0	92.22	16.9	91.11	6.8	97.28	9.4	95.31	11.0	92.45	10.5	8.89	20.0	16.77	16.9	60.74	6.8
SVM-TFIDF-BASIC	66.67		64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	6.35	–	11.72	–	48.74	–
SVM-TFIDF-OC	86.67	30.0	86.67	34.5	82.89	13.4	97.27	31.7	92.14	26.6	85.59	18.6	8.25	29.9	15.76	34.5	55.26	13.4
SVM-TFIDF-ZIPF	86.67	30.0	84.44	31.0	83.56	14.3	97.82	32.5	90.69	24.6	85.99	19.1	8.25	29.9	15.35	31.0	55.7	14.3
SVM-TFIDF-OWDC	91.67	37.5	91.56	42.1	90.11	23.3	96.73	31.0	91.33	25.5	87.0	20.5	8.25	29.9	15.56	32.8	56.74	16.4
SVM-D2V-BASIC	77.78		78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
SVM-D2V-OC	93.33	20.0	92.22	16.9	91.56	7.3	97.28	9.4	95.31	11.0	92.85	11.0	8.89	20.0	16.77	16.9	61.04	7.3
SVM-D2V-ZIPF	93.33	20.0	93.33	18.3	91.56	7.3	97.28	9.4	95.49	11.2	92.96	11.1	8.89	20.0	16.97	18.3	61.04	7.3
SVM-D2V-OwDC	93.33	20.0	95.56	21.1	91.11	6.8	97.28	9.4	96.05	11.9	93.37	11.6	8.89	20.0	17.37	21.1	60.74	6.8
TFM-TFIDF-BASIC	66.67		64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	7.76	–	14.48	–	13.33	–
TFM-TFIDF-OC	76.67	15.0	77.78	20.7	80.0	9.4	95.73	29.6	91.14	25.3	83.02	15.0	8.25	6.3	15.76	8.8	55.26	31.4
TFM-TFIDF-ZIPF	75.33	13.0	75.33	16.9	78.0	6.7	94.31	27.7	85.25	17.2	79.13	9.6	8.25	6.3	15.35	6.0	52.7	29.5
TFM-TFIDF-OWDC	87.67	31.5	84.33	30.9	82.22	12.5	95.34	29.1	91.19	25.3	85.08	17.9	9.25	19.2	18.56	28.2	52.74	29.5
TFM-D2V-BASIC	77.78		78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
TFM-D2V-OC	90.0	15.7	92.22	16.9	92.22	8.1	95.79	7.8	93.88	9.3	92.52	10.6	8.47	14.3	16.57	15.6	60.3	6.0
TFM-D2V-ZIPF	90.11	15.9	90.22	14.4	88.11	3.3	96.34	8.4	93.93	9.4	91.59	9.5	8.58	15.8	15.66	9.2	59.41	4.4
TFM-D2V-OWDC	93.0	19.6	93.0	17.9	92.33	8.2	95.99	8.0	93.88	9.3	92.27	10.3	9.57	29.1	19.16	33.6	68.67	20.7

Figure 6. Results of the Best Combinations of Parameters (Best Combination of P, T and e) of Job Offer Classification as a top-N Ranking on The Minajob Corpora.

3.3.2. Best Results Comments

General comparisons with classic job classification systems : A global observation of table presented in Figure 8 immediately reveals that the BASIC classification system is better only 2 times (on the Monster corpus, the NB-D2V and NB-TFIDF lines for the MAP metric alone are only in the top 5). For a total of 216 blocks, this represents around 0.93% (2/216), leaving 99.07% (214/216) for our contribution. In other words, our contribution is 99.07% better. Within this 99.07%, the ZIPF strategy (green color) is represented at 13.88% (30/216), the OC strategy (blue color) at 25.46% (55/216) and the OWDC strategy (yellow color) at 59.72% (129/216).

Improvement rate in Monster dataset : If we look at the Monster dataset alone, we can make the following observations: the rate of improvement of the decision tree model (DT) lies within the ranges [6%, 30%], [11%, 26%] and [10%, 34%]. The Naive Bayes (NB) model lies within the ranges [0%, 17%], [0%, 11%] and [3%, 19%]. Similarly, the rate of improvement of the SVM model is in the ranges [5%, 26%], [0%, 26%] and [3%, 33%]. Finally, that of the TFM model is in the ranges [3%, 41%], [7%, 30%] and [4%, 82%], respectively for the Accuracy, MAP and F1-score metrics for all models.



NIGHAM	Precision						MAP						F1-score					
	P@5	imp	P@10	imp	P@50	imp	M@5	imp	M@10	imp	M@50	imp	F@5	imp	F@10	imp	F@50	imp
DT-TFIDF-BASIC	66.67	–	64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	6.35	–	11.72	–	48.74	–
DT.TFIDF-OC	86.67	30.0	86.67	34.5	82.89	13.4	97.27	31.7	92.14	26.6	85.59	18.6	8.25	29.9	15.76	34.5	55.26	13.4
DT-TFIDF-ZIPF	87.67	31.5	85.44	32.6	83.56	14.3	97.82	32.5	90.69	24.6	85.99	19.1	8.26	30.1	15.37	31.1	55.7	14.3
DT-TFIDF-OWDC	89.89	34.8	88.56	37.4	88.11	20.5	97.73	32.4	92.33	26.9	90.0	24.7	8.47	33.4	15.6	33.1	57.39	17.7
DT-D2V-BASIC	64.44	–	66.67	–	73.33	–	74.51	–	74.74	–	72.13	–	6.14	–	12.12	–	48.89	–
DT-D2V-OC	86.67	34.5	86.67	30.0	82.89	13.0	97.27	30.5	92.14	23.3	85.59	18.7	8.25	34.4	15.76	30.0	55.26	13.0
DT-D2V-ZIPF	87.78	36.2	85.44	28.2	83.56	14.0	97.82	31.3	91.69	22.7	85.99	19.2	8.26	34.5	15.37	26.8	55.7	13.9
DT-D2V-OWDC	88.89	37.9	88.89	33.3	89.11	21.5	97.73	31.2	92.33	23.5	89.0	23.4	8.26	34.5	15.61	28.8	57.6	17.8
NB-TFIDF-BASIC	84.44	–	84.44	–	87.56	–	90.37	–	89.3	–	86.86	–	8.04	–	15.35	–	58.37	–
NB-TFIDF-OC	93.33	10.5	92.22	9.2	91.56	4.6	97.28	7.6	95.31	6.7	92.85	6.9	8.89	10.6	16.77	9.3	61.04	4.6
NB-TFIDF-ZIPF	92.33	9.3	94.33	11.7	91.56	4.6	97.28	7.6	95.49	6.9	92.96	7.0	8.88	10.4	16.99	10.7	61.04	4.6
NB-TFIDF-OWDC	94.33	11.7	95.33	12.9	96.0	9.6	97.28	7.6	96.31	7.8	93.94	8.2	8.89	10.6	17.0	10.7	62.2	6.6
NB-D2V-BASIC	77.78	–	78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
NB-D2V-OC	93.33	20.0	92.22	16.9	91.56	7.3	97.28	9.4	95.31	11.0	92.85	11.0	8.89	20.0	16.77	16.9	61.04	7.3
NB-D2V-ZIPF	93.33	20.0	93.33	18.3	91.56	7.3	97.28	9.4	95.49	11.2	92.96	11.1	8.89	20.0	16.97	18.3	61.04	7.3
NB-D2V-OWDC	95.33	22.6	94.22	19.4	93.11	9.1	97.28	9.4	95.31	11.0	92.45	10.5	8.9	20.1	16.8	17.2	61.18	7.5
SVM-TFIDF-BASIC	66.67	–	64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	6.35	–	11.72	–	48.74	–
SVM-TFIDF-OC	86.67	30.0	86.67	34.5	82.89	13.4	97.27	31.7	92.14	26.6	85.59	18.6	8.25	29.9	15.76	34.5	55.26	13.4
SVM-TFIDF-ZIPF	86.67	30.0	84.44	31.0	83.56	14.3	97.82	32.5	90.69	24.6	85.99	19.1	8.25	29.9	15.35	31.0	55.7	14.3
SVM-TFIDF-OWDC	86.67	30.0	85.56	32.8	85.11	16.4	96.73	31.0	91.33	25.5	87.0	20.5	8.25	29.9	15.56	32.8	56.74	16.4
SVM-D2V-BASIC	77.78	–	78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
SVM-D2V-OC	93.33	20.0	92.22	16.9	91.56	7.3	97.28	9.4	95.31	11.0	92.85	11.0	8.89	20.0	16.77	16.9	61.04	7.3
SVM-D2V-ZIPF	93.33	20.0	93.33	18.3	91.56	7.3	97.28	9.4	95.49	11.2	92.96	11.1	8.89	20.0	16.97	18.3	61.04	7.3
SVM-D2V-OWDC	94.33	21.3	97.56	23.7	97.78	14.6	97.28	9.4	96.05	11.9	93.37	11.6	8.89	20.0	17.41	21.4	62.15	9.2
TFM-TFIDF-BASIC	66.67	–	64.44	–	73.11	–	73.84	–	72.76	–	72.18	–	7.76	–	14.48	–	51.11	–
TFM.-TFIDF-OC	76.67	15.0	77.78	20.7	80.0	9.4	95.73	29.6	91.14	25.3	83.02	15.0	8.25	6.3	15.76	8.8	55.26	8.1
TFM-TFIDF-ZIPF	75.33	13.0	75.33	16.9	78.0	6.7	94.31	27.7	85.25	17.2	79.13	9.6	8.25	6.3	15.35	6.0	55.7	9.0
TFM-TFIDF-OWDC	91.67	37.5	91.33	41.7	90.22	23.4	96.34	30.5	94.19	29.5	91.08	26.2	19.89	15.6	20.38	40.7	66.74	30.6
TFM-D2V-BASIC	77.78	–	78.89	–	85.33	–	88.89	–	85.86	–	83.64	–	7.41	–	14.34	–	56.89	–
TFM-D2V-OC	90.0	15.7	92.22	16.9	92.22	8.1	95.79	7.8	93.88	9.3	92.52	10.6	8.47	14.3	16.57	15.6	60.3	6.0
TFM-D2V-ZIPF	90.11	15.9	90.22	14.4	88.11	3.3	96.34	8.4	93.93	9.4	91.59	9.5	8.58	15.8	15.66	9.2	59.41	4.4
TFM-D2V-OWDC	93.0	19.6	93.0	17.9	91.0	6.6	96.79	8.9	94.33	9.9	93.17	11.4	19.89	16.8	28.66	9.99	68.67	20.7

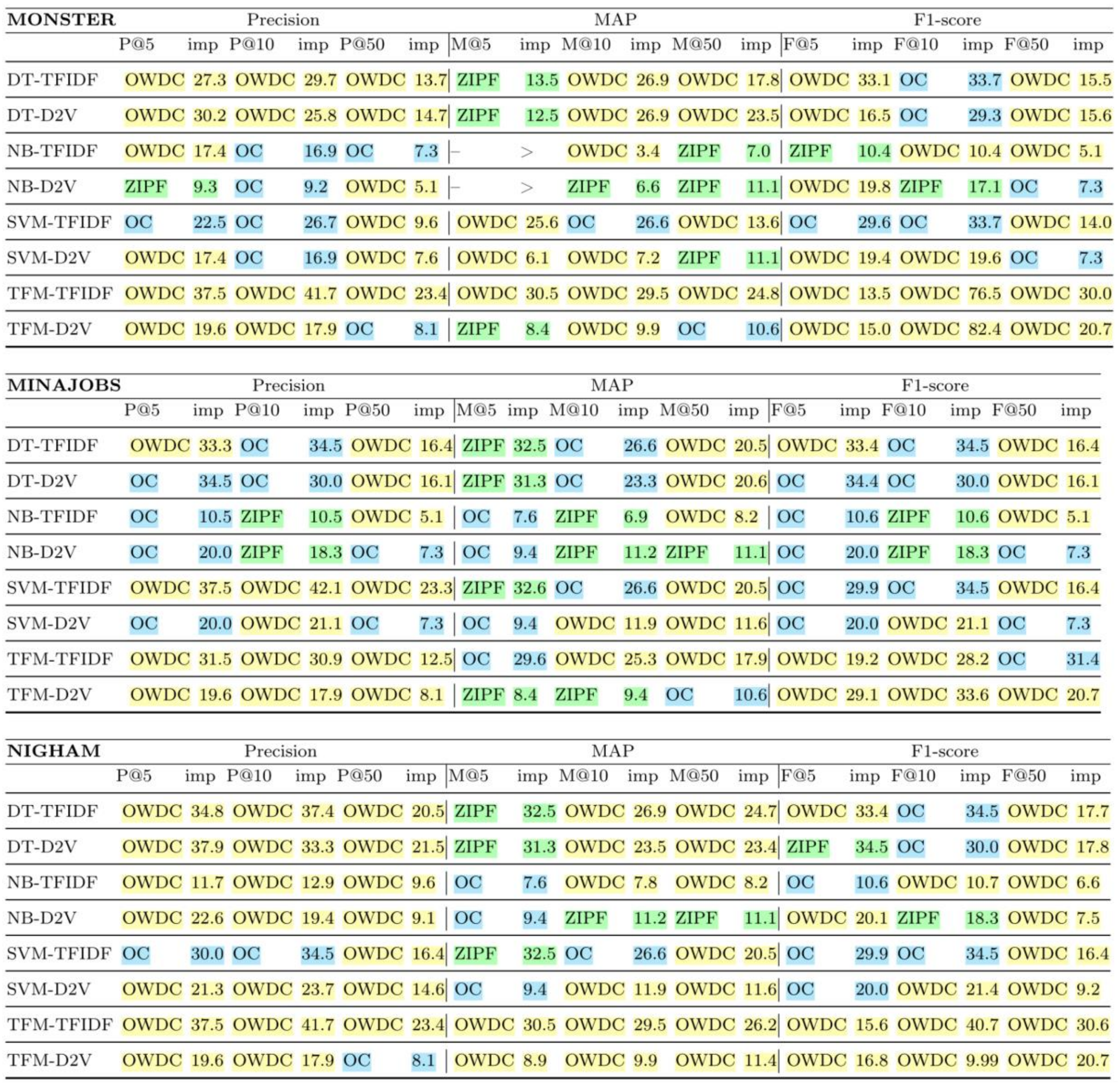
Figure 7. Results of the Best Combinations of Parameters (Best Combination of P, T and e) of Job Offer Classification as a top-N ranking on the Nigham Corpora.

Improvement rate in Minajobs dataset : Looking at the Minajob dataset in the same way, we note that: the rate of improvement of the decision tree model (DT) lies within the ranges [13%, 34%], [18%, 32%] and [13%, 34%]. The Naive Bayes (NB) model lies within the ranges [4%, 20%], [6%, 11%] and [4%, 20%]. Similarly, the rate of improvement of the SVM model is in the ranges [6%, 42%], [9%, 32%] and [6%, 34%]. Finally, that of the TFM model is in the ranges [3%, 31%], [9%, 32%] and [6%, 34%], respectively for the Accuracy, MAP and F1-score metrics.

Improvement rate in Nigham dataset : Focusing on the Nigham corpus, the rate of improvement from the point of view of the decision tree (DT) model lies within the ranges [13%, 37%] for the Precision metric, [18%, 32%] for the MAP metric and [13%, 34%] for the F1-score metric. From the point of view of the Naive Bayes (NB) model, it lies within the ranges [4%, 22%] for the Precision metric, [6%, 11%] for the MAP metric and [4%, 20%] for the F1-score metric. From the point of view of the SVM model, it lies within the ranges [4%, 34%] for the Precision metric, [9%, 32%] for the MAP metric and [7%, 34%] for the F1-score metric. And from the point of view of the TFM model, it lies within the ranges [3%, 37%] for the Precision metric, [7%, 30%] for the MAP metric and [4%, 40%] for the F1-score metric.

Use of OWDC, OC and ZIPF with the machine learning algorithms : If we now look at the table of best results (Figure 8) for TFM, DT, NB and SVM, we can do the following remarks. OWDC(45/129) and OC(6/55) for TFM, ZIPF (16/30) and OC (15/55) are best for NB, OWDC(29/129) and OC (9/55) are best for SVM, ZIPF (16/30) and OC (15/55) are best for NB and OWDC (34/129) and ZIPF (8/30) are best for DT. A general remark is that OWDC is significantly better for all models, and the combination with TFM gives better performance than any other combination. We recommend the use of the combination TFM-OWDC.

Use of OWDC, OC and ZIPF with the vectorization technique : when we analyze table shown in figure 8 from the point of view of the vectorization techniques (Doc2Vec and TF-IDF), we conclude that the strategies are fairly stable when combined with either technique. In fact, for the OWDC 70/129 for TF-IDF and 59/129 for Doc2Vec, for OC strategy we have 28/55 for Doc2Vec and 27/55 for TF-IDF and for ZIPF 19/30 for Doc2Vec and 11/30 for TF-IDF



MONSTER	Precision						MAP						F1-score					
	P@5	imp	P@10	imp	P@50	imp	M@5	imp	M@10	imp	M@50	imp	F@5	imp	F@10	imp	F@50	imp
DT-TFIDF	OWDC	27.3	OWDC	29.7	OWDC	13.7	ZIPF	13.5	OWDC	26.9	OWDC	17.8	OWDC	33.1	OC	33.7	OWDC	15.5
DT-D2V	OWDC	30.2	OWDC	25.8	OWDC	14.7	ZIPF	12.5	OWDC	26.9	OWDC	23.5	OWDC	16.5	OC	29.3	OWDC	15.6
NB-TFIDF	OWDC	17.4	OC	16.9	OC	7.3	-	>	OWDC	3.4	ZIPF	7.0	ZIPF	10.4	OWDC	10.4	OWDC	5.1
NB-D2V	ZIPF	9.3	OC	9.2	OWDC	5.1	-	>	ZIPF	6.6	ZIPF	11.1	OWDC	19.8	ZIPF	17.1	OC	7.3
SVM-TFIDF	OC	22.5	OC	26.7	OWDC	9.6	OWDC	25.6	OC	26.6	OWDC	13.6	OC	29.6	OC	33.7	OWDC	14.0
SVM-D2V	OWDC	17.4	OC	16.9	OWDC	7.6	OWDC	6.1	OWDC	7.2	ZIPF	11.1	OWDC	19.4	OWDC	19.6	OC	7.3
TFM-TFIDF	OWDC	37.5	OWDC	41.7	OWDC	23.4	OWDC	30.5	OWDC	29.5	OWDC	24.8	OWDC	13.5	OWDC	76.5	OWDC	30.0
TFM-D2V	OWDC	19.6	OWDC	17.9	OC	8.1	ZIPF	8.4	OWDC	9.9	OC	10.6	OWDC	15.0	OWDC	82.4	OWDC	20.7

MINAJOBS	Precision						MAP						F1-score					
	P@5	imp	P@10	imp	P@50	imp	M@5	imp	M@10	imp	M@50	imp	F@5	imp	F@10	imp	F@50	imp
DT-TFIDF	OWDC	33.3	OC	34.5	OWDC	16.4	ZIPF	32.5	OC	26.6	OWDC	20.5	OWDC	33.4	OC	34.5	OWDC	16.4
DT-D2V	OC	34.5	OC	30.0	OWDC	16.1	ZIPF	31.3	OC	23.3	OWDC	20.6	OC	34.4	OC	30.0	OWDC	16.1
NB-TFIDF	OC	10.5	ZIPF	10.5	OWDC	5.1	OC	7.6	ZIPF	6.9	OWDC	8.2	OC	10.6	ZIPF	10.6	OWDC	5.1
NB-D2V	OC	20.0	ZIPF	18.3	OC	7.3	OC	9.4	ZIPF	11.2	ZIPF	11.1	OC	20.0	ZIPF	18.3	OC	7.3
SVM-TFIDF	OWDC	37.5	OWDC	42.1	OWDC	23.3	ZIPF	32.6	OC	26.6	OWDC	20.5	OC	29.9	OC	34.5	OWDC	16.4
SVM-D2V	OC	20.0	OWDC	21.1	OC	7.3	OC	9.4	OWDC	11.9	OWDC	11.6	OC	20.0	OWDC	21.1	OC	7.3
TFM-TFIDF	OWDC	31.5	OWDC	30.9	OWDC	12.5	OC	29.6	OWDC	25.3	OWDC	17.9	OWDC	19.2	OWDC	28.2	OC	31.4
TFM-D2V	OWDC	19.6	OWDC	17.9	OWDC	8.1	ZIPF	8.4	ZIPF	9.4	OC	10.6	OWDC	29.1	OWDC	33.6	OWDC	20.7

NIGHAM	Precision						MAP						F1-score					
	P@5	imp	P@10	imp	P@50	imp	M@5	imp	M@10	imp	M@50	imp	F@5	imp	F@10	imp	F@50	imp
DT-TFIDF	OWDC	34.8	OWDC	37.4	OWDC	20.5	ZIPF	32.5	OWDC	26.9	OWDC	24.7	OWDC	33.4	OC	34.5	OWDC	17.7
DT-D2V	OWDC	37.9	OWDC	33.3	OWDC	21.5	ZIPF	31.3	OWDC	23.5	OWDC	23.4	ZIPF	34.5	OC	30.0	OWDC	17.8
NB-TFIDF	OWDC	11.7	OWDC	12.9	OWDC	9.6	OC	7.6	OWDC	7.8	OWDC	8.2	OC	10.6	OWDC	10.7	OWDC	6.6
NB-D2V	OWDC	22.6	OWDC	19.4	OWDC	9.1	OC	9.4	ZIPF	11.2	ZIPF	11.1	OWDC	20.1	ZIPF	18.3	OWDC	7.5
SVM-TFIDF	OC	30.0	OC	34.5	OWDC	16.4	ZIPF	32.5	OC	26.6	OWDC	20.5	OC	29.9	OC	34.5	OWDC	16.4
SVM-D2V	OWDC	21.3	OWDC	23.7	OWDC	14.6	OC	9.4	OWDC	11.9	OWDC	11.6	OC	20.0	OWDC	21.4	OWDC	9.2
TFM-TFIDF	OWDC	37.5	OWDC	41.7	OWDC	23.4	OWDC	30.5	OWDC	29.5	OWDC	26.2	OWDC	15.6	OWDC	40.7	OWDC	30.6
TFM-D2V	OWDC	19.6	OWDC	17.9	OC	8.1	OWDC	8.9	OWDC	9.9	OWDC	11.4	OWDC	16.8	OWDC	9.99	OWDC	20.7

Figure 8. Summary of Tables in Figures 5, 6 and 7 Showing the Best Results for Each Block; with Yellow Showing the Times when OWDC was Best, Blue Showing the Times when OC was best and green showing the times when Zipf was Best.

3.3.3. Best Parameters Comments

The graphs in figures 9 and 10 show the evolution of the performances according to the parameters K and P in top 10. In these figures, the left column corresponds to the TF-IDF technique and the right column to Doc2Vec, while the lines represent the Precision, MAP and F1-score metrics for the first, second and third respectively.

Performance by K top 10: Looking at figure 9, we can see several graphs showing the evolution of different performances (according to metrics and classification systems) as a function of *K* parameter values. As a result, we can see that Precision and MAP evolve in much the same way, with low performance at low *K* values and stabilizing around 1500. The F1-score metric, on the other hand, quickly becomes stable at higher *K* values. It can also be noted that, generally speaking, the DT and TFM models are stable for different values of *K* according to different metrics.

Performance by P top 10 : Similarly, looking at figure 10, we note that the Accuracy and MAP metrics become generally constant from *P* values to 500 and reach their peak values for *P* around 1000. The F1-score arrives at its higher values for a P of around 300, then falls back around 600 to be more or less constant thereafter, precisely with NB, and is fairly stable with SVMs and the DT.

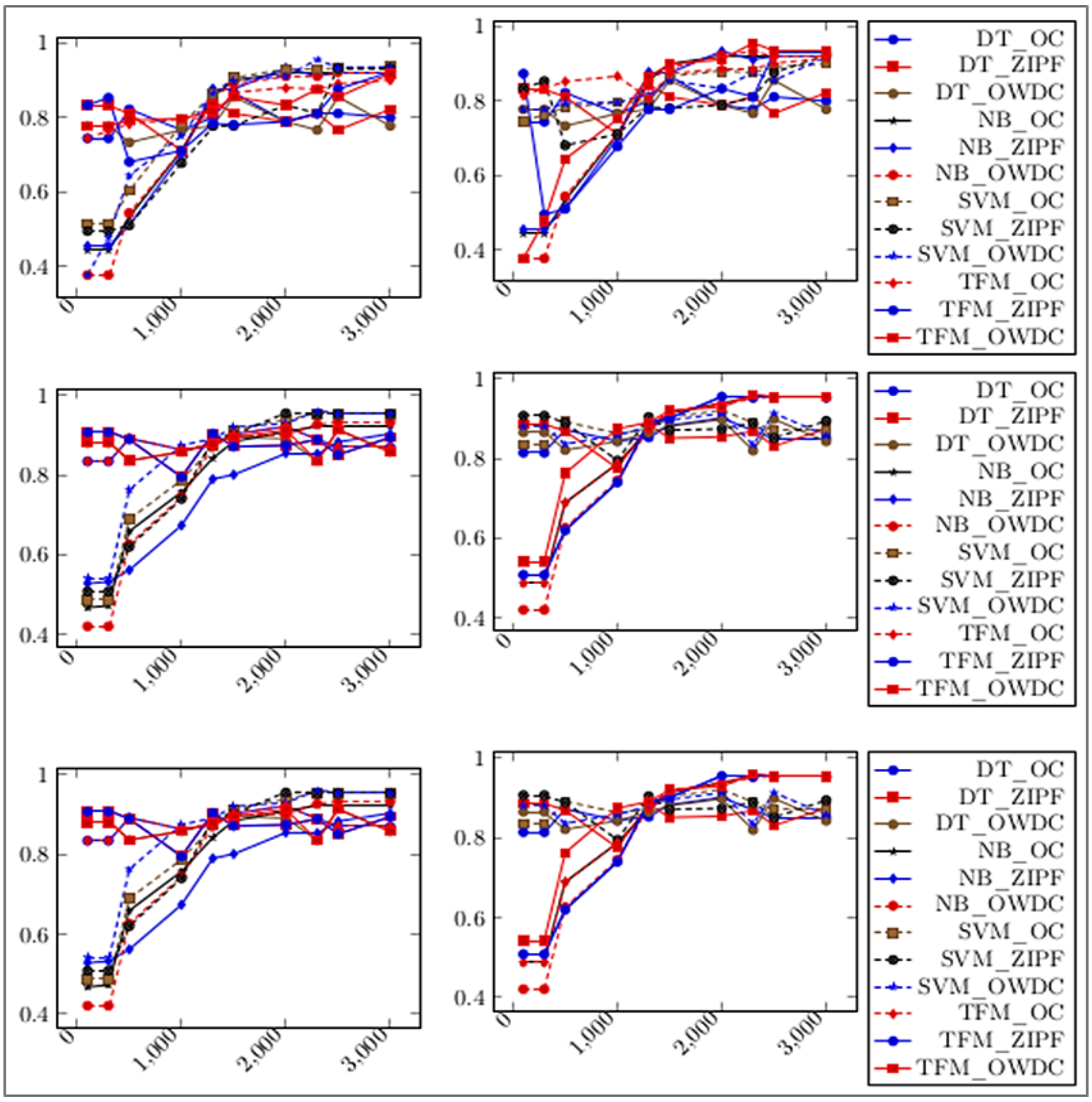


Figure 9. Evolution of Performance as a Function of K top 10.

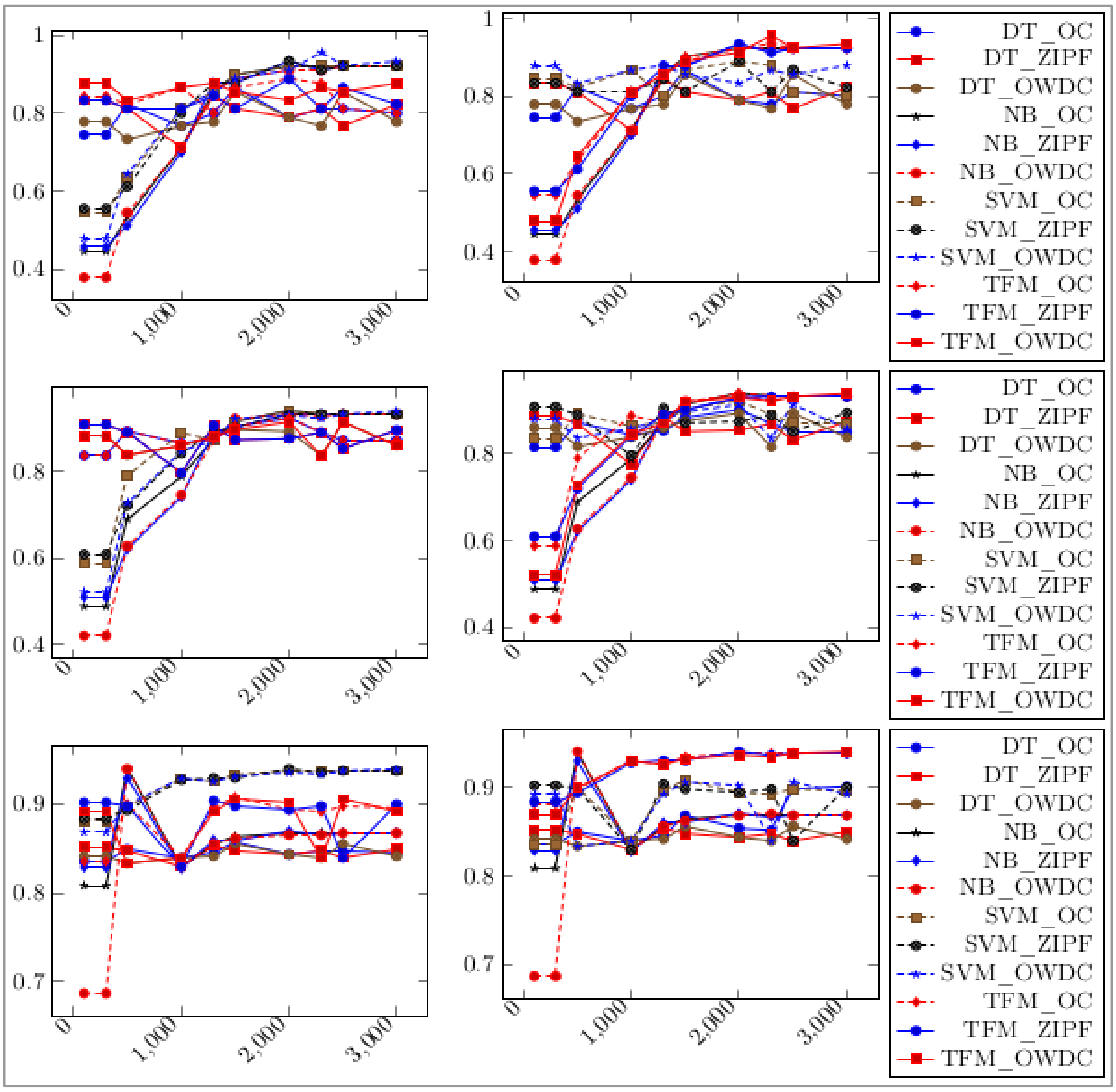


Figure 10. Evolution of performance as a function of P top 10.

3.4. Results Comparisons with State of Art

After presenting the performance of the contributions on the different datasets, the authors propose to test our vectorization paradigm on the RE '17 Data Challenge Area 2 [9] dataset, which focuses on the identification of functional (FR) and non-functional (NFR) requirements. In the dataset, the requirement text in the FR category is labelled with 'F' and NFR as 'A, L, LF, MN, O, PE, SC, SE, US, FT, PO'. These are 625 requirements texts used as a data set. 56% of which belong to the NFR class and the rest to the FR class.

They also applied our vectorization paradigm to the three other datasets (Wap, WebKB and Kla) detailed in [8]. These datasets consist respectively of 1560, 4199, and 2340 documents divided into 20, 4 and 20 classes respectively. And also on WebKB datasets presented by [10] which is a completed version of WebKB datasets [8] 'versions.

To conform to the work of [9], among the previously used machine learning algorithms, they have chosen only Naive Bayes and SVM and the metrics Precision and F1-score. The table shown in figure 11 presents the best results obtained with the new vectorization paradigm (obtained with the OWDC technique) on the data from [9], [8] and [10], compared with the results that they themselves obtained. We can see that our results stand out for each of the combination cases, with a performance improvement ranging from 2% for the NB-TFIDF combinations to 55% for the SVM-TFIDF combinations.

Table 3. Comparison of the results of [10] on the RE'16 dataset and [8] and [9] on Wap, WebKB and Kla datasets with our results on the same dataset.

		Precision				**F1-score**			
Datasets Works		**NB-D2V**	**NB-TFIDF**	**SVM-D2V**	**SVM-TFIDF**	**NB-D2V**	**NB-TFIDF**	**SVM -D2V**	**SVM-TFIDF**
RE'16	Tiun et al., [10]	82.11	**93.48**	83.46	31.2	82.55	91.13	73.47	38.42
RE'16	Class-Oriented Vect.	**95.11**	93	**94.33**	**86.67**	**92.45**	**93.94**	**93.37**	**87**
Wap	Jin et al., [8]	***	***	***	***	***	***	83.65	84.17
Wap	Class-Oriented Vect.	91.12	89.52	90.04	90	91.07	89.69	**92.37**	**90.21**
WebKB	Jin et al., [8]	***	***	***	***	***	***	86.04	87.56
WebKB	Cunha et al., [9]	***	***	***	***	***	***	***	82.26
WebKB	Class-Oriented Vect.	93.02	91.50	93.04	91.05	91.87	90.61	**92.42**	**91.27**
Kla	Jin et al., [8]	***	***	***	***	***	***	86.28	87.02
Kla	Class-Oriented Vect.	92.02	91.50	91.14	89.25	92.77	91.69	**93.70**	**93.13**

4. Conclusion

In this paper the authors propose a new text vectorization paradigm for text classification problems, called class-oriented text classification. The aim of this paradigm is to improve the numerical representation of text in order to increase the performance of text classification algorithms. They therefore propose three strategies in this new paradigm: One strategy, based on word occurrence, selects the most frequent words in a class to form the class-oriented document (called OC); another strategy selects the words in the class-oriented document based on the ZIPF score, hence the name ZIPF; and a final strategy that selects words for the class-oriented document based on word recurrence in classes, but this time penalizing it with the number of classes in which the word appears, to ensure that the selected keywords discriminate only one class, they call it OWDC.

To demonstrate the effectiveness of our contribution, experiments are conducted on a set of three datasets named Monster, Nigham, Minajobs, with the classification models DT for Decision Tree, NB for Naive Bayes, SVM for Support Vector Machine and TFM for Transformers (deep neural networks with self-attention). The vectorization methods used on class-oriented documents were TF-IDF and Doc2Vec combined with our class-oriented vectorization strategies including OC, ZIPF and OWDC. To evaluate these experiments, we used the precision, MAP and F1-Score metrics and, as a prelude to a recommendation, classification of job offers as top-N has been made with N having the values 5, 10 and 50 in the classes used here as profiles.

At the end of the evaluation of these experiments, results which, for different vectorization strategies of our contribution, improve the classical results have been obtained. For example, for the Naive Bayes classifier, the improvement ranged from 19% to 22%, depending on the corpus and strategies used (for our lowest performances), while for the transformers classifier, the improvement ranged from 29% to 40%, depending on the corpus and strategies used.

In addition, to validate their work, we proposed to show that it can be applied to data other than job offers. To this end, they compared themselves with the work of [9], [8] and [10] by experimenting with their contribution on the datasets of [9], [8] and [10]. It emerges that in a general way they have an increase in the performances going from 2% to 55%; which demonstrates the positive impact of our contribution.

This work has enabled us to show that we can increase the performance of text classification algorithms by taking classes into account during vectorization. In the future, we intend to tackle not only text but also image and sound classification. In other words, we will be proposing a class-oriented vectorization paradigm for images and sounds, in order to improve their classification.

Declaration of Interests, Data Availability, Ethical and Informed Consent for Data Used

For this article the authors guarantee that there are no personal, financial or relational conflicts of interest that could influence the publication of this article.
The authors also guarantee that the data used can be made available to those who need it, on demand for those datasets not yet on open platforms. And they declare that the data has been collected in a way that does not violate any ethical rules.

References

[1] Y. Liang, H. Liu, and S. Zhang, "Micro-blog sentiment classification using Doc2vec + SVM model with data purification," The Journal of Engineering, vol. 2020, no. 13, 2020. doi: 10.1049/joe.2019.1159

[2] J. Chen, Z. Gong, and W. Liu, "A Dirichlet process biterm-based mixture model for short text stream clustering," Applied Intelligence, vol. 50, no. 5, 2020. doi: 10.1007/s10489-019-01606-1

[3] C. Storey and D. E. O'Leary, "Text Analysis of Evolving Emotions and Sentiments in COVID-19 Twitter Communication," Cognitive Computation, vol. 16, no. 4. Springer Science and Business Media LLC, pp. 1834–1857, Jul. 28, 2022. doi: 10.1007/s12559-022-10025-3.

[4] M. A. Calijorne Soares and F. S. Parreiras, "A literature review on question answering techniques, paradigms and systems," Journal of King Saud University - Computer and Information Sciences, vol. 32, no. 6, 2020. doi: 10.1016/j.jksuci.2018.08.005

[5] M. Hao, B. Xu, J. Y. Liang, B. W. Zhang, and X. C. Yin, "Chinese short text classification with mutual-attention convolutional neural networks," ACM Transactions on Asian and Low-Resource Language Information Processing, vol. 19, no. 5, 2020. doi: 10.1145/3388970

[6] J. Parmar, S. S. Chouhan, and V. Raychoudhury, "A machine learning based framework to identify unseen classes in open-world text classification," Information Processing and Management, vol. 60, no. 2, 2023. doi: 10.1016/j.ipm.2022.103214

[7] S. Akter, N. Nawal, A. Dey, and A. Das, "Analyzing the IT job market and classifying IT jobs using machine learning algorithms," in Applied Intelligence for Industry 4.0, 2023. doi: 10.1201/9781003256083-19

[8] L. Jin, L. Zhang, and L. Zhao, "Feature selection based on absolute deviation factor for text classification," Information Processing and Management, vol. 60, no. 3, 2023. doi: 10.1016/j.ipm.2022.103251

[9] W. Cunha, S. Canuto, F. Viegas, T. Salles, C. Gomes, V. Mangaravite, E. Resende, T. Rosa, M. A. Gonçalves, and L. Rocha, "Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling," Information Processing and Management, vol. 57, no. 4, 2020. doi: 10.1016/j.ipm.2020.102263

[10] S. Tiun, U. A. Mokhtar, S. H. Bakar, and S. Saad, "Classification of functional and non-functional requirement in software requirement using Word2vec and fast Text," Journal of Physics: Conference Series, vol. 1529, no. 4, 2020. doi: 10.1088/1742-6596/1529/4/042077

[11] A. Nigam, A. Roy, H. Singh, and H. Waila, "Job recommendation through progression of job selection," in Proc. of 2019 6th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS 2019), 2019. doi: 10.1109/CCIS48116.2019.9073723

[12] M. Afandi and K. N. Isnaini, "Analyzing Public Trust in Presidential Election Surveys: A Study Using SVM and Logistic Regression on Social Media Comments," Journal of Computer Science and Engineering (JCSE), vol. 5, no. 1, pp. 1-11, 2024.

[13] P. Atanasova, J. G. Simonsen, C. Lioma, and I. Augenstein, "A diagnostic study of explainability techniques for text classification," in EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, 2020. doi: 10.1007/978-3-031-51518-7_7

[14] J. Hjort and J. Poulsen, "The arrival of fast internet and employment in Africa," American Economic Review, vol. 109, no. 3, 2019. doi: 10.1257/aer.20161385

[15] F. Suvankulov, M. C. K. Lau, and F. H. C. Chau, "Job search on the internet and its outcome," Internet Research, vol. 22, no. 3, 2012. doi: 10.1108/10662241211235662

[16] G. Wabo, A. Nzekon, F. Sosso, and X. Djam, "Vectorization on class-oriented documents for job recommendation based on supervised machine learning models," in CARI 2022, Yaoundé, Cameroon, 2022.

[17] T. Sabri, O. el Beggar, and M. Kissi, "Comparative study of Arabic text classification using feature vectorization methods," Procedia Computer Science, vol. 198, 2021. doi: 10.1016/j.procs.2021.12.239

[18] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," Information, vol. 10, no. 4, 2019. doi: 10.3390/info10040150

[19] K. N. Prafajar, H. Vallyan, N. L. P. A. Candradewi, I. S. Edbert, and D. Suhartono, "Multiclass Job Recommendation System in the IT Field between Classification and Prediction Method," in 2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST 2022), 2022. doi: 10.1109/GECOST55694.2022.10010659

[20] O. Ben-Porat, S. Hirsch, L. Kuchy, G. Elad, R. Reichart, and M. Tennenholtz, "Predicting strategic behavior from free text," Journal of Artificial Intelligence Research, vol. 68, 2020. doi: 10.1613/JAIR.1.11849

[21] A. Shen, B. Salehi, J. Qi, and T. Baldwin, "A multimodal approach to assessing document quality," Journal of Artificial Intelligence Research, vol. 68, 2020. doi: 10.1613/JAIR.1.11647

[22] B. Xing and I. W. Tsang, "Exploiting Contextual Target Attributes for Target Sentiment Classification," Journal of Artificial Intelligence Research, vol. 80, pp. 419–439, Jun. 2024. doi: 10.1613/jair.1.14947

[23] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," Journal of Applied Science and Technology Trends, vol. 2, no. 01, 2021. doi: 10.38094/jastt20165

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.

[25] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," Information Processing and Management, vol. 50, no. 1, 2014. doi: 10.1016/j.ipm.2013.08.006

[26] V. Wisdom and R. Gupta, "An introduction to twitter data analysis in python," Artigence Inc, 2016.

[27] J. Perkins, Python Text Processing with NLTK 2.0 Cookbook, 2010.

[28] L. Singh, "Clustering Text: A Comparison Between Available Text Vectorization Techniques," 2022. doi: 10.1007/978-981-16-1249-7_3

[29] B. Habert and M. Jardino, "R. Harald Baayen — Word Frequency Distributions. Text, Speech and Language Technology n°18, Dordrecht : Kluwer Academic Publishers, 2001, 334 p. + 1 CD-Rom," Corpus, vol. 2, 2003. doi: 10.4000/corpus.42

[30] D. Valcarce, A. Bellogín, J. Parapar, and P. Castells, "Assessing ranking metrics in top-N recommendation," Information Retrieval Journal, vol. 23, no. 4, 2020. doi: 10.1007/s10791-020-09377-x

[31] M. A. Hambali, M. D. Gbolagade, and Y. A. Olasupo, "Heart Disease Prediction Using Principal Component Analysis and Decision Tree Algorithm," Journal of Computer Science and Engineering (JCSE), vol. 4, no. 1, 2023.

[32] R. Egger, “Text Representations and Word Embeddings,” Tourism on the Verge. Springer International Publishing, pp. 335–361, 2022. doi: 10.1007/978-3-030-88389-8_16.