Vectorization on class-oriented documents for job recommendation based on supervised machine learning models

Ghislain Wabo, Armel Nzekon, Fritz Sosso, Xaveria Djam

Vectorization on class-oriented documents for job recommendation based on supervised machine learning models

Ghislain WABO[*1], **Armel NZEKON**[1,2], **Fritz SOSSO**[1], **Xaveria DJAM**[1]

[1]Université de Yaoundé I, Département d'Informatique, BP 812, Yaoundé, Cameroun
[2]Sorbonne Université, IRD, UMI 209 UMMISCO, F-93143, Bondy, France

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

Abstract

Nowadays, job recommender systems are more useful in the fight against unemployment due to their strong presence in e-recruitment platforms that are becoming very popular. Most of the recommender systems based on machine learning that recommend jobs to profiles use a vector representation of job offers based on keywords. However, these keywords are results of vectorization which is applied on a collection of documents where each one is a job offer. In this case, each keyword discriminates one job offer from another, whereas it can be preferable that each keyword discriminates one class from another. The aim of this paper is to improve the recommendation of job offers to users, by proposing to apply vectorization on a class-oriented collection of documents in order to obtain more useful keywords for the representation of job offers. In the case of job recommendation, each class-oriented document corresponds to a user profile. Experiments are done on two datasets (Nigam and Minajobs), using TF-IDF and Doc2Vec as vectorization techniques, Naive Bayes and Decision Trees as supervised machine learning models for top-N recommendation, and Precision, MRR and MAP as evaluation metrics. Our results show that, whatever the case, the best performance is always reach by a job recommender system resulting from our contribution. Compared to classic job recommender systems, the improvement rates can go up to 13% and 24% for systems based on Naive Bayes and go up to 55% and 46% for those based on Decision tree, respectively in the Nigam and Minajobs datasets.

Keywords

Job recommender systems, Machine learning, TF-IDF, Doc2Vec, class-oriented documents.

I INTRODUCTION

According to the International Labor Organization[1], we talk about underemployment when the duration or the productivity of the job of a person is insufficient compared to other employment opportunities that the person is willing and able to pursue, and in 2021 the Organization for Economic Cooperation and Development[2] estimated the global underemployment rate at 86.1%. This finding justifies the interest to be given to work that aims to give job seekers the possibility of having offers that best match their profile.

[1]https://www.ilo.org/wcmsp5/groups/public/
[2]https: //www.oecd.org/en/

Moreover, with the advent of Information and Communication Technologies (ICT), we are witnessing a new form of labor market: that of online job supply and demand. In fact, Hjort et al. [3] provide evidence of the positive impact of the internet on the labor market in twelve African countries and Suvankulov et al. [9] show that job seekers who used the Internet saw their probability of being rehired within 12 months increase from 7.1% to 12.7%.

E-recruitment platforms are a solution for solving the problem of recruiting professionals, as they reduce file processing time and advertising costs, and have the advantage of increasing the volume of data that HR professionals can deal with. All this is possible thanks to the use of job recommender systems to appropriate profiles that are integrated into these platforms.

A recommendation system is thus a subclass of information filtering which consists in predicting the score or the preference of a user on an item; with the aim of proposing the items likely to be the most appreciated by this one. In our case it is about trying to predict the job offers that will be preferred or that will best suit the user profiles. Vectorizing the offers well is therefore imperative to have better recommendations.

Several papers on job recommendation are based on the use of machine learning techniques: neural networks [5], forest of decision trees [6] and naive bayes [4]. In all these previous work, the vectorization stage which consists in giving a vectorial representation of the job offer text by using some keywords as the vector space basis, is essential.

However, these keywords are results of vectorization which is applied on a collection of documents where each one is a job offer. In this case, each keyword discriminates one job offer from another, whereas it can be preferable that each keyword discriminates one user-profile from another, which corresponds to a vectorization applied to a class-oriented document collection. That is why in this paper, we propose to apply vectorization on a class-oriented document collection to get more useful keywords for job offer representation.

The rest of this paper is structured as follows: the section II presents some work that uses vectorization in job recommender system, and the section III describes how we proceed to apply vectorization on class-oriented document collection. The section V is dedicated to experiments and results, and we conclude in Section VI.

II VECTORIZATION STAGE FOR TEXT CLASSIFICATION BY SUPERVISED MACHINE LEARNING MODELS

In the context of job offer recommendations, which are textual data, with supervised machine learning models, the first challenge is vectorization which consists in transforming the textual data to introduce job offer into a new representation space which machine learning models can use. Thus, the collection of job offers, which can contain millions of words, is always represented in the new space by a restricted set of useful keywords to best describe each job offer without losing essential information.

Most job recommender systems that rely on machine learning are based on an automatic text classification model. Indeed, the common methodological scheme consists of a first step of text cleaning followed by feature extraction and vectorization and finally by the application of machine learning models generally for classification. And so, improved text classification models could lead to improved job offer recommender systems.

Concerning text classification models, previous works such as those of Anjuma et al. [1], and Wendland et al. [11] compare the impact on machine learning algorithms of feature extraction

and text vectorization techniques Count-vectorization and TF-IDF. The results they obtain show better performance when TF-IDF is used as a vectorization technique. Similarly, Singh et al. [8] compare the performance of text clustering applied following two vectorization techniques TF-IDF and Doc2Vec. They perform their experiments on a set of tweets and obtain better results with TF-IDF as a vectorization technique.

We note that previous work which compare vectorization techniques show that, performance of machine learning algorithms for text classification strongly depends on the results of vectorization. This confirms the relevance of work aimed at improving the results of vectorization. Moreover, in previous work on text classification, the step of features extraction and vectorization are carried out on all document collection without taking into account classes to which these documents correspond although we are in a case of supervised classification.

To overcome this limitation, it is important to propose text classification techniques in which a vectorization of class-oriented texts is done, following the example of Yu et al. [13]. Such a vectorization orientation could have an important positive impact on the quality of job offer recommender systems results, where job offers are documents and user profiles are classes.

III BACKGROUND : CLASSIC JOB RECOMMENDER SYSTEM BASED ON SUPERVISED MACHINE LEARNING MODELS

The steps of classic job recommender system based on supervised machine learning models are presented in figure 1. The first step is the cleaning step which consists in cutting the job descriptions into lists of words (tokenization) and then transforming these words into their root or radical, also called the stems (stemming) and finally remove all stopWords. The next step is the vectorization stage were someone can apply TF-IDF, Doc2Vec or Count-Vectorization. After that, the third step is to run a supervised machine learning text classification models to get top-N recommendation list for each user profile.

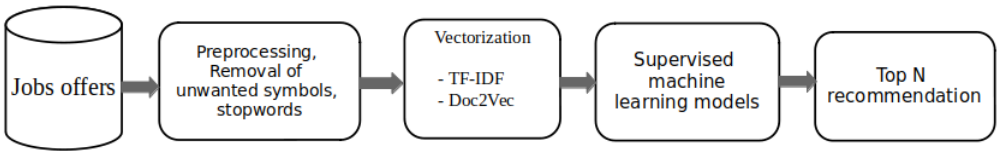

Figure 1: Steps of classic job recommender system based on supervised machine learning models.

3.1 Cleanning Job offers

This step consists of removing words that are devoid of information. In job recommender systems, at the tokenization step, each job offer is transformed into a list of words without separators according to Wisdom et al. [12]. After tokenization, for each word obtained, stemming method which consists in extracting the roots of a word, is apply according to Perkins et al. [7]. The new obtained list of words are the radical of the tokens, this radical is still called stem.

3.2 Vectorization

This is the step during which we transform the job offers from their textual representation to a new numerical representation in a smaller space. More explicitly, we search the corpus for keywords that will define the new space of representation of the job offers, and the coordinates of the offers in this new space are defined thanks to values that vary according to the vectorization technique used. Inspired by [8], the techniques used in this article are TF-IDF and Doc2Vec.

TF-IDF. which is a technique that gives each word its importance in proportion to the number of times it appears in the corpus, while limiting words that are too common to have any importance. We used the built-in scikit-learn library TfidfVectorizer [3] for TF-IDF.

Doc2Vec. which is an advanced technique used for word enbbeding, which uses a deep learning neural network (3 layers) and is based on the word2vec model. Doc2Vec takes into account the logical order of words. We train our own model and obtain Doc2Vec representations. We use the Gensim Doc2Vec library [4].

3.3 Supervised machine learning models and Top-N recommendation

To match profiles and offers and vice versa, we assimilated our problem to a classification problem, in which in a first step the classes were the profiles and in a second step the offers. We then used naive bayes to automatically detect the top best offers for each profile and the top best profiles for each offer. This automatic detection was done as follows: first we trained the naive bayes and decision tree models on our training data; Then we classify the test set and with the probabilities provided by the classification models on the test sets, we get the top N in each class as recommendation of the corresponding profiles.

Naïve Bayes. It is based on the probabilistic Bayes theorem. It is a method that builds supervised machine learning models. It is used to assign classes (profiles) to job offers. Inspired by [8] we used the implementation of naive bayes Gaussian available on scikit-learn[5].

Decisions Tree. Decision tree algorithm it is a method to build a model from a set of rules. We used the implementation of the ID3 version available in the scikit-learn python library[6].

IV VECTORIZATION ON CLASS-ORIENTED DOCUMENT COLLECTION

In the classic vectorization scheme, all job offer documents are directly passed to the vectorization technique and it tries to find the set of keywords that can best represent each job offer. In this paper, we propose to not pass the job offer documents to the vectorization technique, but to pass class-oriented documents; as list of documents where each one contains the words that best represent the associated class.

More explicitly, for each profile (class) we build a document that contains extracted words that best discriminate it from the others profiles. It is the collection of all these profile-oriented or class-oriented documents that will be passed to the vectorization technique. We justify this technique by the fact that the preliminary extraction of the class oriented keywords allows to better discriminate the classes and to hope for a more discriminating and representative vectorization. The figure 2 shows the steps of job recommender system based on supervised machine learning models with vectorization step apply on class-oriented document collection. In the figure 2 we can clearly see the additional step that does not appear in the figure 1 which is our contribution. This step is the extraction of the keywords in the class oriented document before vectorization.

In this paper, we propose three strategies to build class-oriented document collection: Occurrence count (OC), Zipf law (Zipf) and Occurrences weighted by the dispersion in the class(OWDC). The parameter K represents the number of keywords of class-oriented document. For a start we set the K value, since we had no methods to determine an optimal K depending on the corpus.

[3] https://scikit-learn.org/
[4] https://radimrehurek.com/gensim/models/doc2vec.html
[5] https://scikit-learn.org/stable/modules/naive_bayes.html
[6] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

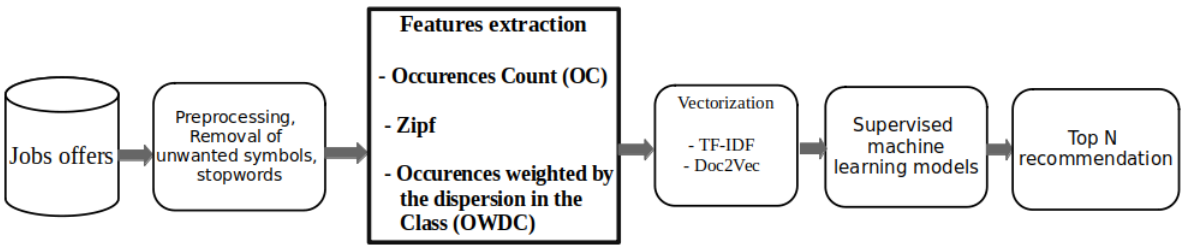

Figure 2: Steps of job recommender system based on supervised machine learning models with vectorization step apply on class-oriented document collection.

Occurrence count (OC). here the class-oriented document of each user profile contains the K words that are most frequent in the documents of job offers that match this user profile.

Zipf law (Zipf). Here the K words of class-oriented document are selected thanks to the zipf law which is a compromise between the number of occurrences of the word and its rank in the ranking of the most recurrent words [2].

Occurrences weighted by the dispersion in the class (OWDC). In this case, we rank all the words of each class using the coefficient $C(m)$ in equation 1 (where m is a word), and we select the K first words with highest values of $C(m)$. The coefficient $C(m)$ is a weighting of the number of occurrences of the word m, by the appearance frequency of m in job offers of the corresponding class.

$$C(m) = word\_occurrence\_in\_class * \frac{numbers\_of\_jobs\_containing\_word}{number\_of\_jobs\_in\_class} \quad (1)$$

V EXPERIMENTS AND RESULTS

This section first presents the datasets used, then the evaluation protocol of top-N job recommender systems considered. The section ends with results and comments.

5.1 Datasets used

We use two datasets. The first one which we call 'nigam' is available on kaggle[7] and was already used by Nigam et al. [6]. It contains 2 334 job offers for xxx user profiles. We have extracted job offers of the computer science and IT field, which corresponds to 790 job offers for 07 user profiles of IT specialists. The second one is a subset of 1 151 job offers for 11 user profiles in IT field extracted from minajobs platform[8].

5.2 Evaluation protocol

To assess the top-N job recommender systems considered, each dataset is split in two parts: 70% for training and 30% for testing. Recommender systems are built on the training set, and after applying them on test set, we have top-N recommendations which are evaluated using three top-N values (3, 5 and 10) and three evaluation metrics (Precision, MRR (Mean Reciprocal Rank) and MAP (Mean Average Precision)) [10].

[7] https://www.kaggle.com/jsrshivam/job-recommendation-case-study
[8] http://minajobs.net

The first evaluation metric Precision, is the ratio of the number of good recommendations on the total number of recommendations, when MRR is focused on the position of the first good recommendation. MAP is the mean of the means of precision's and it allows to evaluate the recommendation taking into account the position of good recommendations in the top-N list.

5.3 Results and comments

The table 1 presents all the results of job recommender systems following all the top-N values and evaluation metrics considered. For each dataset, we have 04 sections each of 09 blocks. In the first line of each section, there is a machine learning model (DT (Decision Tree) or NB (Naive bayes)) associated with a basic vectorization technique [BASIC - (TFIDF or Doc2Vec)]. In each block of each section, the job recommender system of the first line is compared to class-oriented vectorization variants (CO, ZIPF, OWDC) which are our contribution.

The green color in a cell of a block means that the job recommender system at line has the best performance according to the evaluation metric at column and is the best without ex-aequo. The other colors blue, red and white indicate the outcome of the comparison between job recommender system resulting from our contribution, and the classic job recommender system of the first line of the bloc (BASIC). Thus blue color indicates improvement, red one indicate deterioration and white indicate equality of performance.

NIGAM	Precision						MRR						MAP					
	P@3	imp.	P@5	imp.	P@10	imp.	R@3	imp.	R@5	imp.	R@10	imp.	M@3	imp.	M@5	imp.	M@10	imp.
DT-TFIDF-BASIC	58.33	-	60.0	-	63.75	-	56.25	-	61.25	-	62.81	-	57.29	-	62.86	-	68.55	-
DT-TFIDF-OC	79.17	35.7	82.5	37.5	75.0	17.6	87.5	55.6	87.5	42.9	87.5	39.3	88.54	54.5	85.49	36.0	84.45	23.2
DT-TFIDF-ZIPF	75.0	28.6	80.0	33.3	77.5	21.6	81.25	44.4	84.38	37.8	84.38	34.3	82.29	43.6	82.6	31.4	83.82	22.3
DT-TFIDF-OWDC	79.17	35.7	82.5	37.5	76.25	19.6	81.25	44.4	84.38	37.8	84.38	34.3	80.21	40.0	84.6	34.6	85.1	24.1
DT-D2V-BASIC	58.33	-	60.0	-	63.75	-	56.25	-	64.38	-	64.38	-	57.29	-	65.17	-	69.06	-
DT-D2V-OC	66.67	14.3	72.5	20.8	68.75	7.8	81.25	44.4	84.38	31.1	84.38	31.1	80.21	40.0	78.25	20.1	77.55	12.3
DT-D2V-ZIPF	75.0	28.6	80.0	33.3	77.5	21.6	81.25	44.4	84.38	31.1	84.38	31.1	82.29	43.6	82.6	26.7	83.82	21.4
DT-D2V-OWDC	75.0	28.6	77.5	29.2	72.5	13.7	81.25	44.4	84.38	31.1	84.38	31.1	80.21	40.0	82.93	27.3	82.16	19.0
NB-TFIDF-BASIC	75.0	-	75.0	-	77.5	-	81.25	-	81.25	-	83.04	-	81.25	-	79.22	-	81.04	-
NB-TFIDF-OC	79.17	5.6	77.5	3.3	80.0	3.2	79.17	-2.6	79.17	-2.6	79.17	-4.7	80.21	-1.3	81.39	2.7	81.56	0.6
NB-TFIDF-ZIPF	70.83	-5.6	72.5	-3.3	77.5	-	81.25	-	81.25	-	83.04	-	81.25	-	76.88	-3.0	79.24	-2.2
NB-TFIDF-OWDC	79.17	5.6	80.0	6.7	82.5	6.5	91.67	12.8	91.67	12.8	91.67	10.4	89.58	10.3	86.04	8.6	85.2	5.1
NB-D2V-BASIC	83.33	-	75.0	-	77.5	-	91.67	-	91.67	-	91.67	-	91.67	-	87.5	-	84.09	-
NB-D2V-OC	70.83	-15	70.0	-6.7	76.25	-1.6	79.17	-14	79.17	-14	79.17	-14	80.21	-13	77.64	-12	76.9	-8.6
NB-D2V-ZIPF	75.0	-10	70.0	-6.7	76.25	-1.6	87.5	-4.5	87.5	-4.5	87.5	-4.5	87.5	-4.5	83.12	-5.0	79.26	-5.7
NB-D2V-OWDC	87.5	5.0	85.0	13.3	85.0	9.7	100.0	9.1	100.0	9.1	100.0	9.1	93.75	2.3	92.07	5.2	89.61	6.6

MINAJOBS	Precision						MRR						MAP					
	P@3	imp.	P@5	imp.	P@10	imp.	R@3	imp.	R@5	imp.	R@10	imp.	M@3	imp.	M@5	imp.	M@10	imp.
DT-TFIDF-BASIC	57.58	-	56.36	-	60.0	-	59.09	-	62.73	-	63.86	-	59.85	-	62.23	-	65.06	-
DT-TFIDF-OC	72.73	26.3	74.55	32.3	70.91	18.2	80.3	35.9	82.58	31.6	82.58	29.3	81.06	35.4	79.75	28.2	79.36	22.0
DT-TFIDF-ZIPF	75.76	31.6	78.18	38.7	75.45	25.8	81.82	38.5	84.09	34.1	84.09	31.7	83.33	39.2	82.77	33.0	82.68	27.1
DT-TFIDF-OWDC	81.82	42.1	80.0	41.9	74.55	24.2	86.36	46.1	88.64	41.3	88.64	38.8	85.61	43.0	86.68	39.3	85.01	30.7
DT-D2V-BASIC	57.58	-	58.18	-	63.64	-	59.09	-	66.82	-	66.82	-	59.85	-	65.73	-	68.76	-
DT-D2V-OC	66.67	15.8	69.09	18.8	67.27	5.7	80.3	35.9	82.58	23.6	82.58	23.6	79.55	32.9	76.76	16.8	75.67	10.0
DT-D2V-ZIPF	75.76	31.6	78.18	34.4	75.45	18.6	86.36	46.1	88.64	32.7	88.64	32.7	85.61	43.0	83.9	27.6	83.55	21.5
DT-D2V-OWDC	78.79	36.8	76.36	31.2	71.82	12.9	86.36	46.1	88.64	32.7	88.64	32.7	85.61	43.0	85.47	30.0	82.88	20.5
NB-TFIDF-BASIC	75.76	-	72.73	-	77.27	-	80.3	-	80.3	-	81.6	-	80.3	-	78.48	-	80.18	-
NB-TFIDF-OC	78.79	4.0	74.55	2.5	79.09	2.4	84.85	5.7	84.85	5.7	84.85	4.0	84.09	4.7	82.58	5.2	81.29	1.4
NB-TFIDF-ZIPF	75.76	-	72.73	-	78.18	1.2	86.36	7.5	86.36	7.5	87.66	7.4	84.85	5.7	80.05	2.0	80.69	0.6
NB-TFIDF-OWDC	78.79	4.0	78.18	7.5	81.82	5.9	93.94	17.0	93.94	17.0	93.94	15.1	90.91	13.2	86.41	10.1	84.79	5.7
NB-D2V-BASIC	72.73	-	67.27	-	72.73	-	80.3	-	80.3	-	81.6	-	80.3	-	77.27	-	77.36	-
NB-D2V-OC	72.73	-	69.09	2.7	76.36	5.0	84.85	5.7	84.85	5.7	84.85	4.0	84.09	4.7	79.85	3.3	77.9	0.7
NB-D2V-ZIPF	78.79	8.3	70.91	5.4	77.27	6.2	90.91	13.2	90.91	13.2	90.91	11.4	89.39	11.3	84.6	9.5	80.71	4.3
NB-D2V-OWDC	84.85	16.7	80.0	18.9	82.73	13.7	100.0	24.5	100.0	24.5	100.0	22.5	93.94	17.0	90.34	16.9	87.14	12.6

Table 1: Results of recommendation on Minajobs and Nigam corpora.

5.3.1 *Best results table description*

In the table 1 we have 72 blocks of results where a classic job recommender system (BASIC) is compared to those resulting from our contribution (OC, ZIPF, OWDC). The table 2 contains 72 cells, each one associated to a block of the table 1 and contains the job recommender system that has the best performance in the associated block. The $imp.$ column indicates the improvement compared to the classic recommender system in the first line of the block (BASIC).

		Precision						MRR						MAP					
		P@3	imp.	P@5	imp.	P@10	imp.	R@3	imp.	R@5	imp.	R@10	imp.	M@3	imp.	M@5	imp.	M@10	imp.
NIGAM	DT-TFIDF	OC	35.7	OC	37.5	ZIPF	21.6	OC	55.6	OC	42.9	OC	39.3	OC	54.5	OC	36.0	OWDC	24.1
	DT-D2V	ZIPF	28.6	ZIPF	33.3	ZIPF	21.6	OC	44.4	OC	31.1	OC	31.1	ZIPF	43.6	OWDC	27.3	ZIPF	21.4
	NB-TFIDF	OC	5.6	OWDC	6.7	OWDC	6.5	OWDC	12.8	OWDC	12.8	OWDC	10.4	OWDC	10.3	OWDC	8.6	OWDC	5.1
	NB-D2V	OWDC	5.0	OWDC	13.3	OWDC	9.7	OWDC	9.1	OWDC	9.1	OWDC	9.1	OWDC	2.3	OWDC	5.2	OWDC	6.6
MINAJOBS	DT-TFIDF	OWDC	42.1	OWDC	41.9	ZIPF	25.8	OWDC	46.1	OWDC	41.3	OWDC	38.8	OWDC	43.0	OWDC	39.3	OWDC	30.7
	DT-D2V	OWDC	36.8	ZIPF	34.4	ZIPF	18.6	ZIPF	46.1	ZIPF	32.7	ZIPF	32.7	ZIPF	43.0	OWDC	30.0	ZIPF	21.5
	NB-TFIDF	OC	4.0	OWDC	7.5	OWDC	5.9	OWDC	17.0	OWDC	17.0	OWDC	15.1	OWDC	13.2	OWDC	10.1	OWDC	5.7
	NB-D2V	OWDC	16.7	OWDC	18.9	OWDC	13.7	OWDC	24.5	OWDC	24.5	OWDC	22.5	OWDC	17.0	OWDC	16.9	OWDC	12.6

Table 2: summary of results in table 1.

The blue color in the table 2 indicates that the best recommender system is the one using the class-oriented OC strategy, the green color indicates that it is the class-oriented ZIPF strategy and the yellow color is for the class-oriented OWDC strategy. Finally, the white color shows that our contribution did not do better than BASIC.

5.3.2 *Best results comments*

General comparisons with classic job recommender systems. When we look at the table 2 which contains the best recommender systems, we see that BASIC is never the best. Moreover, the OC strategy provides the best performance 12/72 (17%), ZIPF is the best 14/72 (19%) and OWDC is the best 46/72 (64%). We therefore conclude that in all the blocks, the recommender systems resulting from our contribution are better than the classic job recommender systems. This means that the positive impact of our contribution is undeniable.

Improvement rate in Nigam dataset. The improvement rates of the Decision Tree model (DT) according to Precision metric range from 21% to 37%, according to MRR from 31% to 55% and according to MAP from 21% to 54%. And those of the Naive bayes model (NB) range from 5% to 13% for Precision, 9% to 12% for MRR and 2% to 10% for MAP.

Improvement rate in Minajobs dataset. The improvement rates of the Decision Tree model (DT) according to the Precision metric range from 18% to 42%, according to MRR from 32% to 46% and according to MAP from 21% to 43%. And those of the Naive bayes model (NB) range from 4% to 18% for Precision, 15% to 24% for MRR and 5% to 17% for MAP.

Use of OC, ZIPF and OWDC according to the machine learning model. By carefully reading the results of the table 2, we can see that from the point of view of learning models decision tree (DT) and Naive Bayes (NB), OC (10/12) and ZIPF (14/14) work better in the case of decision tree, and OWDC (34/46) works better for the Naive Bayes model. Since Naive Bayes produces best results, we recommend the use of the combination NB-OWDC.

Use of OC, ZIPF and OWDC according to the vectorization technique. We note that from the point of view of vectorization techniques, OC (9/12) is better with TF-IDF and ZIPF (12/14) is better with Doc2Vec, while OWDC is stable whatever the vectorization technique used (24/46 for TF-IDF and 21/46 for Doc2Vec).

VI CONCLUSION

The goal in this work was the application of vectorization on class-oriented document collection to improve job recommender systems performances. To this end, we have proposed three strategies for building class-oriented documents. The first one is OC strategy, for a given class, its class-oriented document contains the most recurrent words in the job offers of this class. The second strategy is Zipf, which uses the Zipf's law to attenuate the penalty on the least recurrent words, and the last strategy is OWDC, which does not only consider the number of occurrences but makes sure that the word appears in the maximum job offers of the class.

Experiments are performed on two datasets (Nigam and Minajobs), using two supervised machine learning algorithms (Decision tree and Naive bayes), two vectorization techniques of text classification (TF-IDF and Doc2Vec), and considering three top-N recommendation evaluation metrics (Precision, MRR and MAP) and three top-N values (3, 5 and 10). Our results show that, whatever the case, the best performance is always reach by a job recommender system resulting from our contribution. Compared to classic job recommender systems, the improvement rates can go up to 13% and 24% for systems based on Naive Bayes and go up to 55% and 46% for those based on Decision tree, respectively in the Nigam and Minajobs datasets.

REFERENCES

[1] Nadia Anjuma and B. Srinivasu. "A Comparative Study on Classification Algorithms Using Different Feature Extraction And Vectorization Techniques For Text". In: ().

[2] Benoît Habert and Michèle Jardino. "R. Harald Baayen—Word Frequency Distributions. Text, Speech and Language Technology n° 18, Dordrecht". In: *Corpus* 2 (2003).

[3] Jonas Hjort and Jonas Poulsen. "The arrival of fast internet and employment in Africa". In: *American Economic Review* 109.3 (2019), pp. 1032–79.

[4] Miao Jiang et al. "User click prediction for personalized job recommendation". In: *World Wide Web* 22.1 (2019), pp. 325–345.

[5] Shyam P Joy et al. "A Job Recommendation System For Differently Abled Using Neural Networks". In: *TURCOMAT* 12.10 (2021), pp. 2751–2762.

[6] Amber Nigam et al. "Job recommendation through progression of job selection". In: IEEE. 2019, pp. 212–216.

[7] Jacob Perkins. *Python text processing with NLTK 2.0 cookbook*. PACKT publishing, 2010.

[8] Lovedeep Singh. "Clustering Text: A Comparison Between Available Text Vectorization Techniques". In: *Soft Computing and Signal Processing*. Springer, 2022, pp. 21–27.

[9] Farrukh Suvankulov, Marco Chi Keung Lau, and Frankie Ho Chi Chau. "Job search on the internet and its outcome". In: *Internet Research* (2012).

[10] Daniel Valcarce et al. "Assessing ranking metrics in top-N recommendation". In: *Information Retrieval Journal* 23.4 (2020), pp. 411–448.

[11] André Wendland, Marco Zenere, and Jörg Niemann. "Introduction to Text Classification: Impact of Stemming and Comparing TF-IDF and Count Vectorization as Feature Extraction Technique". In: Springer. 2021, pp. 289–300.

[12] Vivek Wisdom and Rajat Gupta. "An introduction to twitter data analysis in python". In: *Artigence Inc* (2016).

[13] Bowen Yu et al. "Fast Text Classification by Leveraging Class Feature Words". In: *Proceedings of 2021 Chinese Intelligent Automation Conference*. Springer. 2022, pp. 463–470.