An interpretable person-job fitting approach based on classification and ranking

Mohamed Amine Menacer[1], Fatma Ben Hamda[2], Ghada Mighri[2]

Sabeur Ben Hamidene[2] and Maxime Cariou[1]
[1]AYMAX Consulting, Tour Black Pearl, 14 Rue du Général Audran, 92400 Courbevoie, France
[2]AYMAX Consulting, Immeuble Youssef Towers-Avenue de Dinars, 1053 Tunis, Tunisia
{mohamedamine.menacer,fatma.hamda,ghada.mighri
sabeur.ben.hmidene,maxime.cariou}@aymax.fr

Abstract

While the development of online recruitment platforms has allowed companies to post their job offers and for job seekers to submit their resumes simply and efficiently, the recruitment process remains time and resources consuming. Accordingly, models are needed to support the automatic matching between candidate resumes and job offers, which is called person-job fit issue. Recent works have focused on modeling the matching between resumes and job requirements through deep learning techniques. However, due to the complex internal transformations that will subject to the input, these models suffer from the interpretability problem. Yet, in real deployment, it is necessary to explain why a candidate is accepted or rejected for a given job offer. To this end, we propose a hybrid approach that takes benefit from deep learning techniques while making the matching results human-readable. This was achieved by extracting several features from the resume and job description and use them to perform classification and ranking. The obtained results on French resumes dataset show an accuracy of 93.7% in the case of classification and 70% of accepted resumes were ranked on the top 5 candidates, and this in the case where the problem is processed as a ranking issue.

1 Introduction

Over the last few years, the number of job posts and resumes submitted to online recruitment platforms is evolving and growing at a rapid rate. In 2020, more than 40M people used LinkedIn to search for their dream job each week, which leads to an average of 3 people hired every minute on the same platform[1]. While these online platforms make the process of jobs posting and resumes submitting easy, the resumes analysis and the candidates selection still time and energy consuming. For example, at AYMAX consulting, a recruiter needs more than 19 days to process unsolicited applications.

The candidate selection is a process that consists of dividing all applicants to a job offer into two categories: those who we want to interview and those who are not accepted. A good candidates' selection process is the one, which saves the time and helps to find the suitable candidate to the job requirements. It should be as fast as possible because spending too much time looking for the best profile can drive talent away and we need therefore to start the process all over again. This process is based on a good balance between time optimization and quality. This could be achieved by designing effective models that perform job-resume matching automatically, which is called person-job fit issue.

Several approaches were proposed to deal with the person-job fit issue and this from several perspectives: the issue can be modeled as job recommendation issue (Patel and Vishwakarma, 2020), skills measuring (Gugnani and Misra, 2020), candidates matching (Jiang et al., 2020) and candidates ranking (Zaroor et al., 2017). In our case, we relied on this latter approach to deal with the person-job fit issue.

Candidate ranking consists on according a score to each submitted resume in such a way that the appropriate candidate will have the highest score. One way to handle with this problem is by modeling it as a classification or a regression issue where machine learning approaches are used. These approaches require a large amount of labeled data to achieve good results, which are not available because resumes and job posts data are sensitive and they are not shared publicly. An alternative to this approach is based on deep learning techniques. The idea is to learn representations (namely semantic features) from the free text of the job description

[1]Source.

and resume, then apply cosine similarity to the extracted representations to calculate the matching score. The semantic representation could be extracted by using shallow feed forward neural networks such as word2vec (Mikolov et al., 2013) and doc2vec (Le and Mikolov, 2014), or by using deeper and advanced neural network such as Convolutional Neural Network (CNN) (Lecun et al., 1998), Recurrent Neural Network (RNN) (Elman, 1990) and attention based model (Vaswani et al., 2017; Devlin et al., 2018). While training these models does not required labeled data, they suffer from the interpretation problem due to the complex internal transformations. However, in real deployment, it is necessary to explain why a candidate is accepted or rejected for a given job post.

To address the issues associated with the previously highlighted techniques, we propose in this work a hybrid approach that takes benefits from deep learning techniques to learn semantic representations from the job description and resumes, and to learn other features that improve the model performance. We summarize the contributions of our work as follows:

- Our proposed model performs on raw French data (often PDF files), we do not use at any time a form to get structured data. Thus, we propose a new approach to extract key contents form unstructured data.
- Besides learning representation from the free text in the resume and job description, our method extract several human-readable features that help to explain the matching results.
- Propose several techniques to fuse all the extracted features to get a final matching score for each resume.

2 Related work

A good person-job fit model must capture the semantic of the resume and the job description. This can be achieved via deep learning based models. Nowadays, significant improvements are observed in text representation using deep learning techniques (Devlin et al., 2018; Mikolov et al., 2013; Pennington et al., 2014). Such representations were used in several works to compute the similarity of the query and candidate documents. Authors in (Qin et al., 2018) proposed a word-level semantic representation for both job requirements and job seekers' experiences based on RNN. It was enhanced with four hierarchical ability-aware attention strategies that measure the different importance of job requirements for semantic representation, as well as measuring the different contribution of each job experience to a specific ability requirement. All these representations were combined and fed into a binary classification network. A similar work was done by (Zhu et al., 2018) where the authors used CNN to project both job postings and candidates resumes into a shared latent representation. The cosine similarity was applied against these representations to estimate the final matching score. These approaches perform on the free text in resumes and job descriptions; authors in (Jiang et al., 2020) proposed, in addition to the latent representation extracted from the raw text, to learn other representations from entities extracted from the whole resume and job description. In addition, they exploited the past actions to infer the preference or intention of candidates and recruiters.

The learning-to-rank models were also used to handle the person-job fit issue (Faliagka et al., 2012; Le et al., 2019). The idea is to combine predefined features (extracted from resumes and job descriptions) for ranking by means of supervised learning algorithms. Currently, ranking is transformed into a pairwise classification or regression problem where for each candidate we predict classes or scores depicting how well the candidate is in accordance with the job requirement. Training these models required labelled data of the form (resume-job features, score), which are unfortunately not available in most cases. This limits the use of these models to handle with the person-job fit issue.

In our work, we take advantage from the two approaches: the semantic based matching approach and the ranking based approach. The proposed model learn several representations from the free text in the resume and job description and from other entities extracted from the whole text. These representations are human-readable, which allows us to explain and to interpret the matching results. For each representation, scores are calculated to generate other feature that are combined according to several machine learning techniques to estimate a final score showing the relevance degree of the candidate.

3 Proposed approach

We divide the resume-job fit issue into three sub-tasks as it is shown in Figure 1:

Documents analysis. A resume or a job description could be a .PDF, a .DOCX or a .TXT file. In this stage, we extract the text in a structured format from these documents.

Features extraction. Extract semantic and human-readable information from resumes and job descriptions.

Score calculation. Use the extracted features to calculate scores and use them to perform classification, ranking or matching tasks.

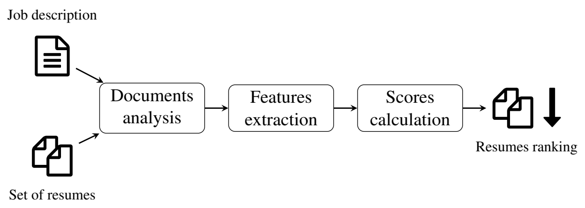
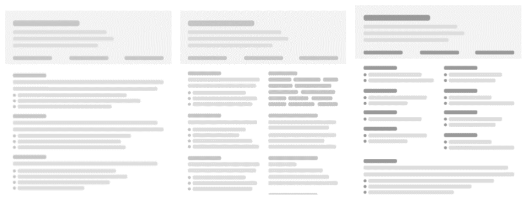


Figure 1: The architecture of proposed model.

3.1 Document analysis

While several tools were proposed to extract text from .PDF or .DOCX files[2], they failed to process files with complex structure. In fact, resumes layout is entirely at the discretion of the author, they can include list format (Figure 2a), double column format (Figure 2b) or combined format (Figure 2c) as shown in Figure 2.



(a) List (b) Column (c) Combined

Figure 2: Most used layouts in resumes.

The extracted text from resumes that include list format is spatially consistent, that means that the logical flow in the original document will match the physical flow extracted from the text. However, in the case where the resume includes double column or combine format, the logical flow will not be the same as the physical flow (starting from left (first column) to right (second column)). To handle with this issue, we built heuristic rules based on the spacing of the text on the page. In fact, for each text zone, we stored its x position and used it afterward to calculate gaps between the different zone texts. In the case where the gap is greater than a fixed threshold, we suppose then that the two zones belong to two different columns. This allows us to change the naive physical flow returned by the conversion tools to approximate the true logical flow of the document. An example of extracted text with and without our proposed heuristic rules is given in Figure 3.

Once the text is extracted from the original document, we still need to cluster it in order to detect the different sections mentioned in the resume (for example: experience, education, etc.). This is very useful to extract the experience years or the education level of the candidate. For this, we used metadata like font size, style, etc. to detect different headings from the original document and consider them as the existing sections. This is justified by the fact that heading properties remain the same in the whole resume. As the extracted text is logically ordered, we used these headings to divide the text into different sections.

By combining the consistent structure extracted from the original document and the available metadata, we were able to extract the full information contained within the resume while keeping the structure intact.

3.2 Features extraction

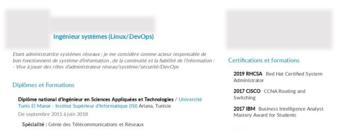
In the aim to make the matching results human explained, seven features were extracted from the resume and job description:

3.2.1 Count features

The count features aim to provide a simple way to tokenize both a resume and a job description, to build a vocabulary of known words and to encode combined documents (resume and job description) using that vocabulary. In order to enrich the representation of these features for our task and capture contextual information, they were calculated by using n-grams where $n \in \{1, 2, 3\}$.

The resulting features at this stage are multidimensional vectors that encode the frequency of each n-gram in the resume and the job description. We calculated afterward the cosine similarity between these vectors to measure how similar the resume and the job description are according to the

[2]Pdfminer, PyMuPDF and python-DOCX.



[redacted]

Ingénieur systèmes (Linux/DevOps)

Etant administratrice systèmes réseaux ; je me considère comme acteur responsable de bon fonctionnent de système d'information , de la continuité et la fiabilité de l'information :
- Vise à jouer des rôles d'administrateur réseau/système/sécurité/DevOps

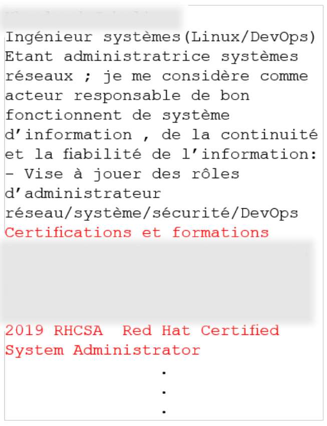
Diplômes et Formations

Diplôme national d'Ingénieur en Sciences Appliquées et Technologies / Université Tunis El Manar - Institut Supérieur d'Informatique (ISI) Ariana, Tunisie
De septembre 2015 à juin 2018
Spécialité : Génie des Télécommunications et Réseaux

Certifications et formations

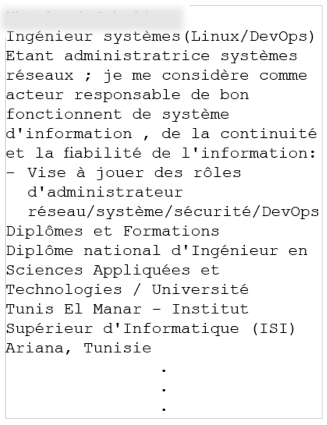
2019 RHCSA Red Hat Certified System Administrator
2017 CISCO CCNA Routing and Switching
2017 IBM Business Intelligence Analyst Mastery Award for Students

(a) A part of the original resume



```
[redacted]
Ingénieur systèmes(Linux/DevOps)
Etant administratrice systèmes
réseaux ; je me considère comme
acteur responsable de bon
fonctionnent de système
d’information , de la continuité
et la fiabilité de l’information:
- Vise à jouer des rôles
d’administrateur
réseau/système/sécurité/DevOps
Certifications et formations
[redacted]
2019 RHCSA  Red Hat Certified
System Administrator
                  .
                  .
                  .
```

(b) w/o heuristic rules



```
[redacted]
Ingénieur systèmes(Linux/DevOps)
Etant administratrice systèmes
réseaux ; je me considère comme
acteur responsable de bon
fonctionnent de système
d'information , de la continuité
et la fiabilité de l'information:
- Vise à jouer des rôles
  d'administrateur
  réseau/système/sécurité/DevOps
Diplômes et Formations
Diplôme national d'Ingénieur en
Sciences Appliquées et
Technologies / Université
Tunis El Manar - Institut
Supérieur d'Informatique (ISI)
Ariana, Tunisie
                  .
                  .
                  .
```

(c) With heuristic rules

Figure 3: Example of extracted text with and without heuristic rules. Red and blurred texts in the sub-figure 3b do not respect the logical flow of the original document. Personal information were blurred.

n-gram frequency.

3.2.2 TF-IDF features

The count features are very basic; for a token that has a large count (for example stop words), its corresponding value will not be very meaningful in the encoded vector. TF-IDF (Term Frequency-Inverse Document) features resolves this issue by integrating the inverse document term. This latter aims to downscale n-grams ($n \in \{1, 2, 3\}$) that appear a lot across documents. By using TF-IDF features, we were able to generate new multidimensional vectors that encode the resume and the job description. As it was done with count features, we calculated the cosine similarity between these vectors to obtain a new TF-IDF based feature.

3.2.3 Semantic features

The count and TF-IDF features are based on the token frequency in the document; they do not capture at any time the semantic relation between document tokens. Recently, deep learning based techniques were used for text representation where context and semantic of each token are captured efficiently. Examples for these: BERT (Devlin et al., 2018), attention based models (Vaswani et al., 2017), GPT (Brown et al., 2020) and ELMO (Peters et al., 2018). The advantage of these approaches is that their training is based on unlabeled textual data, but they require a large amount of data to success their training; for example, the initial BERT model was trained on more than 3 billion words. While pretrained models are available for free access and can be used in our case to generate semantic features, they are not adapted to our task where several technical terms could be found in the resume and job description. In order to prevent training a new model from scratch because only small amount of data are available in our case, we opted for the doc2vec technique (Le and Mikolov, 2014).

Doc2vec is based on a shallow neural network, which is trained to encode a document into a multidimensional vector in such a way that documents that share the same context will be closed in the multidimensional space. To train this model, we collected job descriptions from pole emploi website by using keywords like: *informatique (computer science)*, *développeur (developer)*, *data scientist*, etc. We extended this corpus with a collection of former candidate resumes at AYMAX consulting company. Statistics about the training corpus are given in Table 1.

Corpus	**#Documents**	**#Words**	**#Unique words**
Job descriptions	2242	645k	11k
Resumes	110	68k	4k

Table 1: Statistics about the doc2vec training corpora.

The resulted doc2vec model project the resume and the job description into 50-dimensional space. As the previous extracted features, we used cosine similarity to measure the semantic similarity between documents.

3.2.4 Skills feature

The human who is matching between job offer and candidates must consider the required skills and the candidate's skills, that is why it is necessary that the proposed model make explicit matching between the required and the possessed skills. This process starts with a skill extraction stage. Skills extraction from raw texts is challenging; if we take as an example Python and Java, they could be an animal and an island in Indonesia, respectively, or two programming languages. One way to handle with this issue is to use a Named Entity Recognition (NER) model, which is trained to extract skill entities from an input text. We used a pre-trained model[3] that is able to identify 2K technical skills. We extended this model to recognize more than 200 French skills.

Once skills are extracted from the resume and job description, the matching skills feature is calculated as shown in the Equation 1.

$$skillsFeature = \frac{|s_j \cap s_r|}{|s_j|} \tag{1}$$

where s_j is the set of skills extracted from the job description while s_r is the one extracted from the resume and $|x|$ function means the cardinality of the set x.

3.2.5 Experience years feature

Another important feature that could be used to enhance the matching between the resume and job description is the number of experience years. A candidate with the highest experience years is more likely to get the job than another with less experience years. Our approach to calculate experience years is to extract from the experience section all dates by using a rule based approach and calculate afterwards the total number of years. This latter was considered as experience year's feature.

3.2.6 Spelling errors feature

A candidate resume with several spelling errors could reflect that the candidate is not a fastidious person. For this reason, we decided to calculate the percentage of misspelling words in the resume by using a list of 320k French words extended with more than 22K technical words collected from LinkedIn plate-form. This latter stage is justified by the fact that resumes include several technical terms that will be considered as misspelled words in the case where only the French list of words is used. The percentage of misspelled words is calculated as shown in Equation 1. The obtained value was used as a spelling errors feature.

$$spellingErrorsFeature = \frac{|l_{French} \cap l_{resume}|}{|l_{resume}|} \tag{2}$$

where l_{French} is the list of French words, l_{resume} is a list of resume words and $|x|$ function means the cardinality of the set x.

3.2.7 Document layout feature

The resume is the first contact with the employer; that is why it matters how the candidate structures his resume and what information should include. The document layout feature aims to detect whether the resume is well-structured and included the minimum of required information. A good resume should include: 1. contact information (phone number or email address), 2. education section, 3. experience section, 4. skills section. And it should respect the following constraints: 1. the non-presence of personal information (marital status for example), 2. a number of pages less than 3 pages, 3. and the use of short sentences (less than 50 words).

Once the previous constraints were checked, the document layout feature is calculated according the Equation 3

$$documentLayoutFeature = \sum_{i=1}^{7} f(C_i) \tag{3}$$

with C_i are the constraints that the resume should include and $f(x)$ is defined as follow:

$$f(x) = \begin{cases} 1 & \text{if the resume respect the constraint x} \\ 0 & \text{otherwise} \end{cases}$$

The obtained value was used as document layout feature.

3.3 Score calculation

The features extracted and presented in previous sections were combined to calculate a final matching score depicting how well the candidate is in accordance with the job requirements. This issue was interpreted as:

Weighted score. The final score in this case was calculated as a weighted sum as shown in the Equation 4.

$$score = \sum_{i=1}^{7} w_i feature_i \tag{4}$$

[3] Source

where w_i are the weights associated to each $feature_i$. These weights could be fixed manually according to the recruiter's preference or estimated to maximize the precision on a validation corpus.

Machine learning based score. In this case, the extracted features were combined to perform a candidate classification. In fact, by using a labelled corpus (see below in section 4.1), we trained several machine learning based models while using as input our extracted features to classify resumes into two classes: accepted and rejected.

4 Experiment results

The evaluation of our proposed approach is based on the precision and f1-score calculation. For this, one need a labelled data where for each pair $< job, resume >$ a label as either 0 (not match) or 1 (match) should be provided.

4.1 Data sources

To the best of our knowledge, there is no standard large open source corpus for the person-job fit task, and even less so for French. Therefore, this paper uses an internal small dataset containing a set of 2054 candidate resumes that were applied for 121 jobs. The labelled corpus was split into three parts: 80% for the training (Train), 10% for the validation (Dev) and 10% for the test (Test). Figure 4 illustrates the total number of accepted/rejected candidates in this corpus.

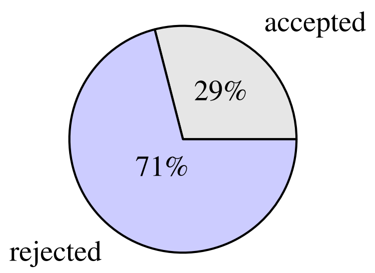


Figure 4: The percentage of accepted and rejected candidate in our corpus.

As the Figure 4 shows, our dataset is imbalanced. As a result, when we make the final decision for each candidate (accepted or rejected) by using machine learning approaches, the learning algorithm will make the decision rule biased towards the majority class. To deal with this issue, we: 1. upsample the minority class (Up), 2. downsample the majority class (Down), 3. and generate synthetic training examples by using Synthetic Minority Oversampling Technique (SMOTE) (Bowyer et al., 2011).

The number of samples after sampling the original Train dataset are given in Table 2.

Sampling	**Orig**	**Up**	**Down**	**SMOTE**
#Samples	1643	2554	732	2298

Table 2: Number of entries in the training dataset (Train) before and after applying the sampling techniques.

4.2 Results and discussion

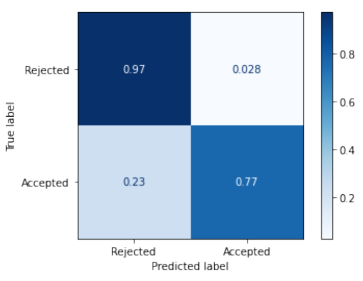
The final score of each candidate is calculated either by combining our proposed score features via a weighted sum (ranking issue) or via machine learning models (classification issue).

4.2.1 Weighted score

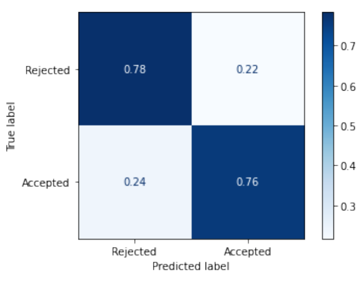
In the case where candidates are ranked according to the weighted features sum, the model evaluation requires a reference dataset including a set of resumes and job descriptions with scores, which are unfortunately not available. For this, the evaluation is carried out in this case by calculating the precision of the model to rank the accepted candidates among the top 5 and 10 candidates. We found that among the accepted candidates, 70% were classed on top 5 candidates according to our calculated relevant score and 88% were classed on top 10 candidates. This shows that ranking candidates by using the weighted score allows us to identify the potential candidates with a good accuracy. It should be noted that the features weights are estimated by maximizing the precision on the Dev corpus. We found that assigning a high weights to the features extracted from jointly the resume and the job description (count, TF-IDF, semantic and skills features) gives better results.

4.2.2 Machine learning based score

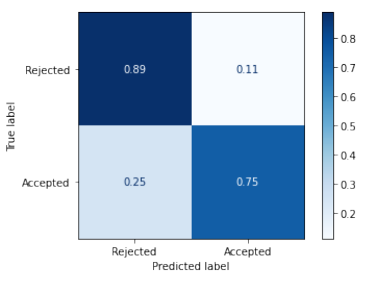
In the aim to evaluate the model on a large scale, we classified the candidate resumes into two classes (accepted or rejected). This was achieved by training several machine learning models while taking as input the different feature's scores. The trained models are: random forest (RF) (Ho, 1995), support vector machine (SVM) (Cortes and Vapnik, 1995), logistic regression (LR), K-Nearest Neigh-



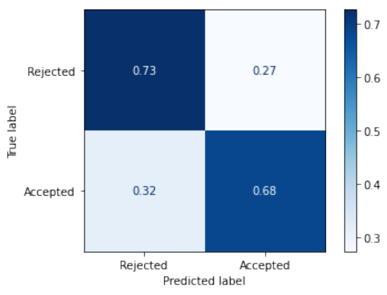
(a) RF



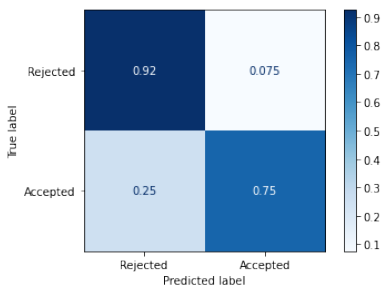
(b) SVM



(c) MLP



(d) LR



(e) KNN

Figure 5: Confusion matrices showing the classification precision for each class by using the different machine learning models.

bors (KNN) (Altman, 1992) and Multi-layer perceptron (MLP) (Popescu et al., 2009).

As the dataset is imbalanced, we carried out a comparative analysis between the sampling techniques: upsample the minority class, downsample the majority class and SMOTE. The obtained results (see Figure 6) shows that training the different models by upsamling the minority class gives better results in terms of f1-score. In should be noted that the evaluation was carried out on the Dev part by using cross validation on 10 folds.

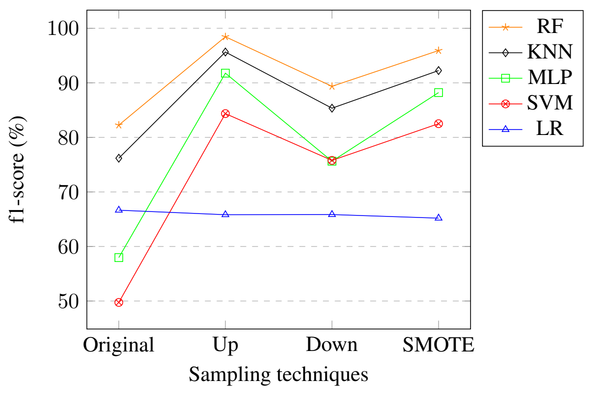


Figure 6: F1-score evolution with respect to the sampling techniques.

In the following experiments, all the models were trained by upsampling the minority class. The obtained results on the test part are illustrated in Table 3.

Model	**Precision**	**F1-score**
RF	93.7	84.3
KNN	88.6	74.6
MLP	85.6	70.1
SVM	77.9	60.6
LR	71.8	52.1

Table 3: Resumes classification results.

Results of Table 3 show that the best classification model is the one based on random forest algorithm with a precision of 93.7% and a f1-score of 84.3%. Since our dataset is imbalanced, it is more convenient to consider the f1-score rather than the precision. The gaps between the precision and the f1-score in all models is justified by the nature of our dataset. Class imbalance influences the learning algorithms during training by making the decision rule biased towards the majority class (rejected candidates). In the aim to highlight this point, Figure 5 sets forth the confusion matrices for each model.

It is clear that the model is biased towards predicting rejected candidates; the precision of these latter is more accurate than the accepted candidates class. However, the obtained precision on the accepted candidates remains acceptable for all models. In fact, the confusion matrix of the random forest based model shows that of all the candidates who are accepted, our algorithm identifies 77% of them accurately, which is not insignificant. This confirms that the extracted features from resumes and job description are informative enough, and the accuracy results should be improved by using a large and balanced dataset. In the same vein, we found that the most important features allowing us to obtain a precision of 93.7% with the random forest based model are those extracted from jointly the resume and job description, as it is shown in Figure 7. This confirms the obtained results in the previous section, where the features were weighted for the final score estimation and candidate ranking.

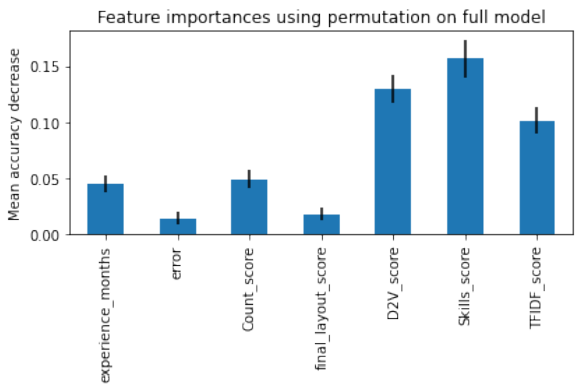
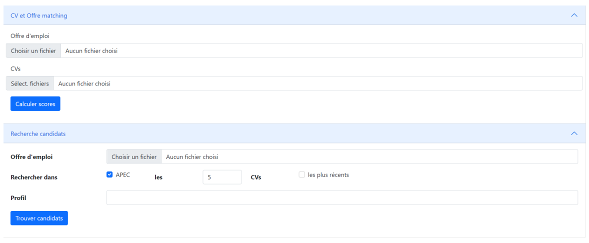


Figure 7: Features importances from Random Forest based model.

4.3 Prototype implementation

The proposed model was fully implemented as a web application where the recruiters have the possibility to select a job description and a set of candidate resumes locally or from the online recruitment platform APEC as it is shown in Figure 8.



CV et Offre matching

Offre d'emploi

Choisir un fichier | Aucun fichier choisi

CVs

Sélect. fichiers | Aucun fichier choisi

Calculer scores

Recherche candidats

Offre d'emploi: Choisir un fichier | Aucun fichier choisi

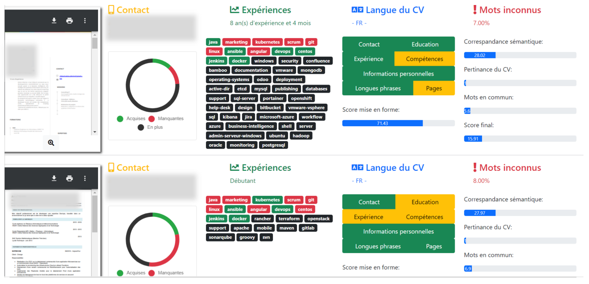
Rechercher dans: [✓ APEC] les 5 CVs ☐ les plus récents

Profil

Trouver candidats

Figure 8: Recruiter interface for the candidate resumes selection.

Once the recruiter selects the job description and a collection of resumes, and upon his request, the system estimates applicant's relevance scores and ranks them accordingly. This is achieved by calling our model via an API. The system provides more detailed information about the candidate (contact information, skills, resume layout information, etc.) as it is shown in Figure 9.



Contact

Expériences: 8 an(s) d'expérience et 4 mois

java, marketing, kubernetes, scrum, git, linux, ansible, angular, devops, centos, jenkins, docker, windows, security, confluence, bamboo, documentation, vmware, mongodb, operating-systems, odoo, deployment, active-dir, etcd, mysql, publishing, databases, support, sql-server, portainer, openshift, help-desk, design, bitbucket, vmware-vsphere, sql, kibana, jira, microsoft-azure, workflow, azure, business-intelligence, shell, server, admin-serveur-windows, ubuntu, hadoop, oracle, monitoring, postgresql

● Acquises ● Manquantes ● En plus

Langue du CV: - FR -

Contact, Education, Expérience, Compétences, Informations personnelles, Longues phrases, Pages

Score mise en forme: 71.43

Mots inconnus: 7.00%

Correspandance sémantique: 28.02

Pertinance du CV: 1

Mots en commun: 5.6

Score final: 15.91

Contact

Expériences: Débutant

java, marketing, kubernetes, scrum, git, linux, ansible, angular, devops, centos, jenkins, docker, rancher, terraform, openstack, support, apache, mobile, maven, gitlab, sonarqube, groovy, svn

● Acquises ● Manquantes

Langue du CV: - FR -

Contact, Education, Expérience, Compétences, Informations personnelles, Longues phrases, Pages

Score mise en forme:

Mots inconnus: 8.00%

Correspandance sémantique: 27.97

Pertinance du CV: 1

Mots en commun: 6.9

Figure 9: Candidate ranking results

5 Conclusion

In this work, we proposed a novel and an interpretable approach to deal with the person-job fit. It leverages on human-readable features extracted from the resume and job description that aim to make the matching results human explained. The proposed approach deals with raw data (.PDF or .DOCX files) with a complex structure. We proposed a heuristic rules based approach to extract structured information from such documents and used them to perform features extraction. Some features were extracted jointly from resumes and job descriptions (count, TF-IDF, semantic and skills features) and others were extracted only from resumes (experience years, spelling errors and document layout features). The extracted representations were fed to several machine learning models to perform resumes classification and ranking. The obtained results on an internal dataset showed a good precision and f1-score of 93.7% and 84.3% respectively. Unfortunately, the lack of models and French resumes data has not enabled us to compare our model with others. However, the obtained results demonstrated that the extracted features are informative enough to handle with the person-job fit issue. We found also that features extracted from jointly the resume and job description are more informative than the ones extracted exclusively from the resume. The proposed model was fully implemented as web application where resumes are ranked on the base of the obtained scores and an overview about each candidate was also displayed, which will save a lot of time and a lot of effort in the recruitment process.

References

N. S. Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. 2011. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Evanthia Faliagka, Kostas Ramantas, Athanasios Tsakalidis, and Giannis Tzimas. 2012. Application of machine learning algorithms to an online recruitment system.

Akshay Gugnani and Hemant Misra. 2020. Implicit skills extraction using document embedding and its use in job recommendation.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1.

Junshu Jiang, Songyun Ye, Wei Wang, Jingran Xu, and Xiaosheng Luo. 2020. Learning effective representations for person-job fit by feature fusion. *CoRR*, abs/2006.07017.

Quoc V. Le and Tomás Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Ran Le, Wenpeng hu, Yang Song, Tao Zhang, Dongyan Zhao, and Rui Yan. 2019. Towards effective and interpretable person-job fitting. pages 1883–1892.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Ranjana Patel and Santosh K. Vishwakarma. 2020. An efficient approach for job recommendation system based on collaborative filtering. In *ICT Systems and Sustainability*, pages 169–176, Singapore. Springer Singapore.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Trans. Cir. and Sys.*, 8(7):579–588.

Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Liang Jiang, Enhong Chen, and Hui Xiong. 2018. Enhancing person-job fit for talent recruitment: An ability-aware neural network approach. *CoRR*, abs/1812.08947.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Abeer Zaroor, Mohammed Maree, and Muath Sabha. 2017. A hybrid approach to conceptual classification and ranking of resumes and their corresponding job posts.

Chen Zhu, Hengshu Zhu, Hui Xiong, Chao Ma, Fang Xie, Pengliang Ding, and Pan Li. 2018. Person-job fit: Adapting the right talent for the right job with joint representation learning. *CoRR*, abs/1810.04040.