Classification of Job Descriptions Using a TF-IDF Vectorizer and Linear SVC

Jens Lemmens
Antwerp University,
Student Computational Linguistics,
Jens.Lemmens2@student.uantwerpen.be

In this study, a model was designed for the 2019 COGAI shared task challenge. The purpose of this challenge was to develop a robust and innovative multiclass classification model that could predict the job domain of a job on the basis of its description. The final model consisted of a TF-IDF vectorizer and a linear SVC, and yielded an F-score score of .77 on the test set. All code is available at: https://github.com/JensLemmens/Shared_Task.

1. Introduction

What follows is a description and an evaluation of one of the models that participated in the 2019 Cognitive Artificial Intelligence (COGAI) challenge. This challenge was organized in the context of the COGAI course, which is part of the Master of Linguistics with profile Computational Psycholinguistics at the University of Antwerp. Included in this paper are the goal and the procedure of the challenge, a description of the data, the design of the model, its results, and suggestions for further improvements.

2. Goal and procedure of the challenge

The challenge was introduced on 01/04. Its goal was to develop a model that could accurately predict the job domain of a job on the basis of its description. The participators were asked to be creative and use techniques that were not discussed elaborately in class, such as a count vectorizer or a naive Bayes classifier. The training data was released a few hours after the introduction session. The participators then had time until 28/04 to develop a first model. On 29/04, elevator pitches took place, and feedback on the models was given. Afterwards, the models could still be improved until 06/05. On that day, the unlabelled test data was released, and it was no longer permitted to modify the models. The participators had time until 13/05 to predict the labels of the test data, and send these predictions to the teachers who led the challenge (Prof. Dr. Guy De Pauw and Stephan Tulkens). On 20/05, test set labels were released, so that the participators could calculate test set performance. Final presentations were held on 17/06.

3. Data

The training data used in this challenge comprised 24 468 labelled job applications. It was shared in a CSV file, in which the first column contained the indices of the job descriptions, the second column consisted of the job descriptions itself, and the third column held the labels, i.e. the domains. Before the data was used for model development, all duplicates (331) were filtered out of the file.

The test data consisted of 5 000 job applications. Similar to the training data, it was shared in a CSV file, but the labels were not included in order to prevent the participators from directly increasing test set performance. However, on 20/05 (after the predictions had been surrendered), a file which included the test labels was released, so that the participators could calculate test set scores. All job applications in both data sets were obtained from VDAB.

4. Methodology

The model was developed on a Windows 10 machine in Python 3.7 using the Anaconda environment (version 2019.03). Throughout the entire development period of the challenge, the general structure of the model remained constant: The data was first vectorized by a TF-IDF vectorizer, and then passed through a linear support vector classifier (LSVC).

The motivation behind the use of a TF-IDF vectorizer can be found within the structure of the type of text that needed to be classified: It can be expected that certain words and phrases such as "communicative", "motivated", "9-to-5-mentality", "profile", "experience", and "flexible", which do not directly indicate to what domain a certain job belongs, occur in many job applications. In contrast, other words or phrases such as "Python 3", "drill press", "bio-engineer", and "plasterer" occur in only a limited amount of job applications, and strongly indicate to what domain a certain job belongs. Therefore, a TF-IDF Vectorizer was preferred to other vectorizers, because it assigns high weights to features that occur in a small number of documents, and vice versa (see scikit-learn documentation).

The LSVC, on the other hand, was chosen out of personal interest. Support vector machines (SVMs) are namely implemented frequently and successfully in clinical information extraction (Li et al. 2010; Patrick and Li 2010; Doan et al. 2012), which is the field of my master's thesis. Therefore, this challenge was the perfect opportunity to test the performance of SVMs. For this study, the linear SVC was used, because it generally performs better than a regular SVC when processing large data sets (+10 000 documents), according to the scikit-learn documentation.

An interim model was designed before the elevator pitches, and this model was optimized further after the elevator pitches. Both models are described in the following subsections.

4.1 Interim model

Before the elevator pitches, the parameters of both components in the model were optimized via GridSearchCV (cv=10). Then, the model was fitted on 90% of the training data. The parameters that were optimized and their corresponding candidate values can be found in Table 1. In this table, the candidate value "stop_words" for the *stop_words* parameter of the vectorizer is a list constructed with the help of NLTK; It contains all Dutch, English, French, and German stop words. Further, the *strip_accents* parameter of the vectorizer was set to "ascii", and the *random_state* parameter of the classifier was set to 0. For all parameters that are not mentioned in Table 1 or above, the default settings were used. After the model was optimized, it was evaluated on the remaining 10% of the training data, which functioned as a development set. The optimal parameter settings can be found in Table 3, and the performance of the interim model on the development set can be found in Table 4.

Table 1
Grid search used to optimize parameters of interim model

Parameter	Component	Candidate values
max_df	vectorizer	[1.0, 0.9]
analyzer	vectorizer	['word', 'char']
ngram_range	vectorizer	[(1, 2), (1, 3), (1, 4), (4, 8)]
stop_words	vectorizer	[None, 'english', stop_words]

Originally, a preprocessing function that was passed to the *preprocessor* parameter of the TF-IDF vectorizer was included in the grid search (with the alternative value being *None*). The function translated all non-Dutch job applications to Dutch with the help of the Langdetect and the Translate module. However, the function resulted in a memory error, and was therefore discarded.

4.2 Final model

With the feedback of the elevator pitches in mind, a second grid search (cv=10) was executed to further optimize the parameters. In contrast with the first grid search, the training data was not split into seperate training and development sets. The parameters that were optimized and their corresponding candidate values can be found in Table 2. Similar to the first grid search, the *strip_accents* parameter of the vectorizer was set to "ascii", and the *random_state* parameter of the classifier was set to 0. Further, the *analyzer* parameter of the vectorizer was set to "char", since this yielded the best results in the first grid search (see Table 3). The *ngram_range* parameter of the vectorizer was optimized a second time to ensure the most accurate n-gram window. Further, another preprocessing function was designed to pass to the *preprocessor* parameter of the vectorizer. This preprocessor removes all non-alphanumeric characters at the start and/or end of a token and/or string, and tokens that consist of non-alphanumeric characters only. When using word n-grams, the vectorizer automatically ignores punctuation: it is always used as a token seperator. However, when using character n-grams, this is not the case. Therefore, it is the function of this preprocessor to deal with punctuation. Further, for all parameters not mentioned in Table 2 or above, the default settings were used. The optimal parameter settings can be found in Table 3, and the average results across the 10 folds can be found in Table 4. Finally, the *max_df* parameter was set to 0.5 after the grid search to reduce the number of features.

Table 2
Grid search used to optimize parameters of final model

Parameter	Component	Candidate values
ngram_range	vectorizer	[(2, 6), (3, 7), (4, 8)]
class_weight	classifier	[None, 'balanced']
C	classifier	[1.0, 5.0, 10.0]

5. Results and discussion

In Table 3, the results of the grid searches, i.e. the optimal parameter settings, can be found. Table 4 contains the scores of the models on the training data. For the interim model, the result was calculated on a development set, whereas the result for the final model was measured by averaging across the scores of the 10 folds of the GridSearchCV on the entire training set. Finally, the performance of the final model on the test set can be found in the form of a classification report in Table 5.

The final model consisted of a pipeline with a TF-IDF vectorizer and a LSVC. Their parameter settings can be found in Table 3. For the parameters not mentioned in that table, the default settings were used, except for the *max_df* parameter and the *strip_accents* parameter of the vectorizer, which were set to 0.5 and 'ascii' respectively. Surprisingly, the stop words and the preprocessor did not improve the performance of the model. Therefore, it may be interesting as a follow-up study to design a preprocessor that removes all (and not a selection of) punctuation, and translates non-Dutch applications to Dutch, but more efficiently than the preprocessor designed for the interim model.

The model yielded an average cross validation F-score of .81 on the training set, and an F-score of .77 on the test set. Since the macro average F-score (.67) was lower than the micro average F-score (.77), it can be expected that more mistakes were made against labels that occur infrequently than against labels that occur frequently. This can be observed in more detail in Table 5: Labels, such as "creatief", "management", "overheid", and "human recources", which occur in less than 1% of all instances, are generally predicted less accurately than labels that occur frequently, such as "administratie", "logistiek transport", "techniek", and "verkoop". On the other hand, there are labels that occur infrequently, but are nevertheless predicted with relatively high accuracy, such as "juridisch", "onderhoud", "onderwijs", and "onderzoek ontwikkeling". Presumably, job applications within these domains contain features that occur strictly in these domains only. Therefore, these features have high weights, which is why it is relatively easy for the classifier to predict the correct label. To conclude, the baseline F-score is .19, since the most frequent label ("techniek") occured in 19% of the test data. Therefore, the model designed in this study performed approximately four times better than the baseline.

Table 3
Optimal parameter settings for both grid searches (N/A: parameter is not optimized)

Parameter	Component	Grid search 1	Grid search 2
stop_words	vectorizer	None	N/A
max_df	vectorizer	1.0	N/A
analyzer	vectorizer	'char'	N/A
ngram_range	vectorizer	(4, 8)	(3, 7)
stop_words	vectorizer	None	N/A
preprocessor	vectorizer	N/A	None
C	classifier	N/A	1.0
class_weight	classifier	N/A	None

Table 4
Results of the models on the training set.

Evaluation metric	Interim model	Final model
F-score	.794	.809

Table 5
Classification report of final model on test set. For each label, its name, proportion of training data, proportion of test data, precision, recall and F-score are included (left to right).

Label	Train p (%)	Test p (%)	Precision	Recall	F-score
Aankoop	1.39	1.98	.82	.49	.59
Administratie	14.54	17.32	.70	.80	.75
Bouw	4.29	3.56	.77	.71	.74
Communicatie	0.17	0.82	.76	.46	.58
Creatief	0.11	0.54	.82	.33	.47
Dienstverlening	3.97	4.16	.71	.65	.68
Financieel	7.25	7.80	.82	.86	.84
Gezondheid	3.61	3.32	.86	.82	.84
Horeca & toerisme	1.89	2.24	.83	.66	.74
Human recources	0.15	0.12	.67	.33	.44
ICT	6.45	5.98	.85	.82	.83
Juridisch	0.48	0.62	.90	.61	.73
Land- & tuinbouw	0.44	0.74	.78	.38	.51
Logistiek & transport	12.70	11.20	.77	.80	.79
Management	0.45	0.56	.80	.14	.24
Onderhoud	1.40	1.64	.79	.66	.72
Onderwijs	1.17	0.94	.90	.81	.85
Onderzoek & ontw.	0.47	0.96	1.00	.52	.68
Overheid	0.41	0.32	.55	.38	.44
Productie	5.27	4.86	.77	.64	.70
Techniek	20.57	18.96	.78	.85	.81
Verkoop	11.63	11.36	.76	.82	.79
Macro average	100.00	100.00	.79	.61	.67
Micro average	100.00	100.00	.77	.77	.77

6. Conclusion

The goal of this work was to develop an innovative model in the context of COGAI 2019 shared task challenge that could accurately predict the domain of a job on the basis of its description. The final model consists of a TF-IDF vectorizer and a linear support vector classifier. The vectorizer uses character 3- to 7-grams (that occur in 50% or less of the job applications) as features, and performs basic character normalization. For the classifier, the *random_state* parameter was set to 0, and the default settings were used for the other parameters. The model yielded a micro average F-score of .77 on the test set, meaning that it performs approximately four times better than the baseline of .19.

The labels "financieel" (.84), "gezondheid" (.84) and "onderwijs" (.85) were predicted the most accurately, whereas the labels "human resources" (.44), "management" (.24), and "overheid" (.44) were the most challenging to predict.

Further, this model is innovative, because in class, we have not worked with SVMs, and especially not linear SVCs, we have only briefly talked about TF-IDF vectorizers, and we have not designed our own preprocessing functions.

In prospect, the model could be improved further by designing a preprocessing function that removes all punctuation, named entities (e.g. email adresses, location names, dates, ...), and translates all non-Dutch job applications to Dutch in a more efficient manner than attempted in this study.

References

Doan, Son, Nigel Collier, Hua Xu, Pham Hoang Duy, and Tu Minh Phuong. 2012. Recognition of medication information from discharge summaries using ensembles of classifiers. *BMC Medical Informatics Decision Making*, 12(1):1–10.

Li, Zuofeng, Feifan Liu, Lamont Antieau, Yonggang Cao, and Hong Yu. 2010. Lancet: a high precision medication event extraction system for clinical text. *Journal of the American Medical Informatics Association*, 17(5):563–567.

Patrick, Jon and Min Li. 2010. High accuracy information extraction of medication information from clinical notes: 2009 i2b2 medication extraction challenge. *Journal of the American Medical Informatics Association*, 17(5):524–527.