SUP'COM

المدرسة العليا للمواصلات بتونس
ÉCOLE SUPÉRIEURE DES COMMUNICATIONS DE TUNIS

Report of tutored project : Development of a skills management platform for freelancers, ESNs and principals.

Realized by

Flihi Arij

Omrane Karim

Supervised by :

M. Riadh TEBOURBI

Company : Welyne Immeuble Le Palace, Tunis 1082, Tunisia

Academic year : 2022 - 2023

Ecole Supérieure des Communications de Tunis
SUP'COM
2083 Cité Technologique des Communications - Elghazala - Ariana - Tunisie
Tél. +216 70 240 900 – site web : www.supcom.mincom.tn

1 Abstract

This report summarizes the collaborative efforts of a team working on an academic project for the AIM (Application of Multimodal Information) option at SUP'COM and Welyne, spanning 14 weeks.

The goal of the project is to create an algorithm that detects various skills from a resume and categorizes them to streamline the recruitment process at the company.

Table of contents

2 Chapter 1 : Context

2.1 Introduction to project

The tutored project is a yearly project for final-year students of Sup'Com that takes place from September to January of the following year. It usually involves collaboration between Sup'Com and an industrial partner. Its goal is to expose students to the professional world and provide them with technical and personal development opportunities. This prepares them for their end-of-studies internships, particularly if they choose to pursue a career in industry rather than academia.

2.2 Introduction to host organism

Welyne was equipped in 2017 with a research and development center by young entrepreneurs precursors of web and mobile development and communication in Tunisia.

Since its inception, Welyne has relied on open-source technologies to give its customers the benefit of the shared experience of millions of specialists. This approach was motivated by the personal convictions of its founders, who wanted to build a near-perfect relationship with their clients, whom they considered as partners in their development.

2.3 Job description of the project

This project involves creating an algorithm that can categorize resumes based on the skills they possess, to improve the hiring process..

2.4 Conclusion

The aim of this project is to address a challenge faced by the industrial partner and provide an opportunity for Sup'Com students to gain valuable industry experience. By working on the proposed project, students will be able to bridge the gap between academic and professional worlds.

3 Chapter 2 : Requirements

3.1 Introduction to used technologies

The following is an overview of the technologies and tools used to complete the project.

3.1.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991.

Figure 1 : Python Logo

3.1.2 Jupyter

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Figure 2 : Jupyter Logo

3.1.3 Tika

Apache Tika is a python library that is used for document type detection and content extraction from various file formats.

Figure 3 : Apache Tika Logo

3.1.4 spaCy

SpaCy is a free, open-source library for NLP in Python written in Cython. spaCy is designed to make it easy to build systems for information extraction or general-purpose natural language processing.

Figure 4 : Spacy Logo

3.2 Dataset

We used a dataset from Kaggle that contains resumes and their corresponding categories. The resumes are in text format and the categories can include information such as job titles, skills, and education. This data can be used for tasks such as resume classification, job recommendation, and natural language processing.

4 Chapter 3 : Resume Parsing

The workflow of parsing a PDF resume using Tika and spaCy would involve the following steps :

4.1 Extract text from PDF

We used Tika for extracting the content as a text from the PDF resume.

4.2 Extract different informations

For extracting informations from the extracted content, we used Regular Expressions, also known as regex or regexp, which are a sequence of characters that define a search pattern, used to search and match informations.
We used regex to extract the email and the phone number.
Spacy was employed to extract the candidate's name, work experience, skills, languages, achievements and other parts from the text, as they have distinct and identifiable patterns.

4.3 Export a structured data

After parsing the resume and extracting various information, we created a structured dataframe containing the different parts of the resume. This

dataframe can be stored in a database and used later during the classification phase to identify suitable resumes for specific roles.

	0	1
0	E-mail	[karim.omrane@supcom.tn, karim.omrane@supcom.tn]
1	PhoneNumber	+216 [redacted]
2	Name	Karim OMRANE
3	education	education computer science engineering student...
4	experience	experiences software engineer intern, wemiam e...
5	projects	projects payment software, non-academic projec...
6	competitions	competitions 2nd place go data science 2022, i...
7	skills	skills programming c++, c, python, javascript,...
8	languages	languages arabic native language english b2 -...

Figure 5 : Exported structured dataframe

5 Chapter 4 : Classification of a resume

5.1 Introduction

In this part we will be detailing the creation of a machine learning model to predict the category of a resume written in english.

5.2 Data processing

The dataset created previously is now used to train and test our model. The dataset initially is composed of the following information :

The category
In this dataset we worked on 25 different categories whiwh are the following : Java Developer 84 Testing 70 DevOps Engineer 55 Python Developer 48 Web Designing 45 HR 44 Hadoop 42 Blockchain 40 ETL Developer 40 Operations Manager 40 Data Science 40 Sales 40 Mechanical Engineer 40 Arts 36 Database 33 Electrical Engineering 30 Health and fitness 30 PMO 30 Business Analyst 28 DotNet Developer 28 Automation Testing 26 Network Security Engineer 25 SAP Developer 24 Civil Engineer 24 Advocate

	Category	Resume
0	Data Science	Skills * Programming Languages: Python (pandas...
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
2	Data Science	Areas of Interest Deep Learning, Control Syste...
3	Data Science	Skills â□¢ R â□¢ Python â□¢ SAP HANA â□¢ Table...
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...

Figure 6 : dataset

Resume

As presented in the previous chapter it is simply all the words extracted from the pdf file. The next step is meant to transform the column Resume into a list of clean text. We will be removing any kind of special caracter, unwanted spaces and blanks, unnecessary symbols through apply the the following function :

```
[ ] import re
    def cleanResume(resumeText):
        resumeText = re.sub('http\S+\s*', ' ', resumeText)  # remove URLs
        resumeText = re.sub('RT|cc', ' ', resumeText)  # remove RT and cc
        resumeText = re.sub('#\S+', '', resumeText)  # remove hashtags
        resumeText = re.sub('@\S+', '  ', resumeText)  # remove mentions
        resumeText = re.sub('[%s]' % re.escape("""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""), ' ', resumeText)  # remove punctuations
        resumeText = re.sub(r'[^\x00-\x7f]',r' ', resumeText)
        resumeText = re.sub('\s+', ' ', resumeText)  # remove extra whitespace
        return resumeText
```

Figure 7 : Cleaning function

Converting label field to numbers

In order to simplify the work and get rid of the categories, we transformed them into simple numbers from 1 to 25.

Cleaned Resume	Labels
Skills Programming Languages Python pandas num...	6
Education Details May 2013 to May 2017 B E UIT...	6
Areas of Interest Deep Learning Control System...	6
Skills R Python SAP HANA Tableau SAP HANA SQL ...	6
Education Details MCA YMCAUST Faridabad Haryan...	6

Figure 8 : LED

5.2.1 TF-IDF method

— **Defenition**
TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a statistical measure used to evaluate the importance of a word in a document or a collection of documents (corpus). It is often used in natural language processing and information retrieval tasks such as text classification, information retrieval, and text mining. The TF-IDF score for a word in a document is calculated by multiplying its term frequency (TF) by its inverse document frequency (IDF). The TF is simply the number of times a word appears in the document, while the IDF is the logarithm of the total number of documents divided by the number of documents containing the word.

— **Vectorization**
Text vectorization is the process of converting a collection of text documents into numerical vectors of fixed size. The most common approach is called bag-of-words, where each document is represented as a vector with one dimension for each unique word in the vocabulary. The value of each dimension is the count of the number of times that word appears in the document. Another approach is to use the TF-IDF values as the feature representation.

```
from sklearn.feature_extraction.text import TfidfVectorizer

word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    use_idf = True,
    stop_words='english',
    max_features=1000)
```

Figure 9 : Vectorization

Multiclass classifier : One-vs-Rest
A multiclass classifier is a machine learning model that is trained to classify instances into more than two classes. For example, in image classification, a multiclass classifier can be used to classify an image as one of several different objects or animals, while a binary classifier can only be used to classify an image as one of two classes, such as "cat" or "not cat". One-vs-all (also known as one-vs-rest) - This method trains a separate binary classifier for each class, with the positive class being the class of interest, and the negative class being all the other classes combined. The class with the highest predicted probability is the final predicted class.

```
clf = OneVsRestClassifier(MultinomialNB()).fit(X_train, label_train)
prediction_mnb = clf.predict(X_test)
```

Figure 10 : One-VS-ALL

Performance

```
print('Accuracy of MultinomialNB Classifier on training set: {:.2f}'.format(clf.score(X_train, label_train)))
```

```
Accuracy of MultinomialNB Classifier on training set: 0.99
```

```
print('Accuracy of MultinomialNB Classifier on test set: {:.2f}'.format(clf.score(X_test, label_test)))
```

```
Accuracy of MultinomialNB Classifier on test set: 0.97
```

Figure 11 : One-VS-ALL performance

5.2.2 Word2Vec method

— **Defenition**
Word2vec is a method for learning vector representations of words, also known as word embeddings. These embeddings are learned in such a way that semantically similar words are represented by similar vectors. Word2vec is a neural network-based model that was developed by Google in 2013. The core idea behind word2vec is to use a neural network to predict a target word based on its context words (CBOW) or predict context words given a target word (Skip-Gram). The input to the neural network is a one-hot encoded representation of the words, and the output is a dense representation (vector) of the target word. The weights of the neural network are then used as the word embeddings.

—
The vocabulary of a word2vec model refers to the set of unique words that the model has been trained on. The vocabulary is determined by the dataset used to train the model and can be thought of as the "dictionary" of the model. The vocabulary is used to map the one-hot encoded representation of words to the dense vector representation (word embeddings) learned by the model.

```
from gensim.models import Word2Vec
# Train the word2vec model
w2v_model = Word2Vec(X_train, vector_size=1000, window=5, min_count=5, max_vocab_size=1000, epochs=25)
```

Figure 12 : Training the dataset

Performance

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf_model = rf.fit(X_train_vect_avg, y_train.values.ravel())
y_pred = rf_model.predict(X_test_vect_avg)
```

Figure 13 : Training

```
from sklearn.metrics import precision_score, recall_score
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='macro')
print('Precision: {} / Recall: {} / Accuracy: {}'.format(
    round(precision, 3), round(recall, 3), round((y_pred==y_test).sum()/len(y_pred), 3)))
```

```
Precision: 0.996 / Recall: 0.996 / Accuracy: 0.995
```

Figure 14 : performance

```
r resume: the top labels are: ['coordinating', 'file'], with percentages : {15: array([[0.5764959]], dtype=float
r resume: the top labels are: ['team', 'required'], with percentages : {20: array([[0.6069484]], dtype=float32),
r resume: the top labels are: ['through', 'skill'], with percentages : {12: array([[0.56408685]], dtype=float32)
r resume: the top labels are: ['at', 'till'], with percentages : {18: array([[0.64531267]], dtype=float32), 16:
r resume: the top labels are: ['june', 'requirements'], with percentages : {1: array([[0.57215333]], dtype=float
r resume: the top labels are: ['required', 'team'], with percentages : {24: array([[0.5072317]], dtype=float32),
r resume: the top labels are: ['learning', 'state'], with percentages : {6: array([[0.51465535]], dtype=float32)
r resume: the top labels are: ['automation', 'at'], with percentages : {17: array([[0.5088977]], dtype=float32),
r resume: the top labels are: ['up', 'on'], with percentages : {8: array([[0.5173892]], dtype=float32), 23: arra
r resume: the top labels are: ['team', 'on'], with percentages : {20: array([[0.50218964]], dtype=float32), 23:
r resume: the top labels are: ['on', 'team'], with percentages : {23: array([[0.63342285]], dtype=float32), 20:
r resume: the top labels are: ['at', 'skill'], with percentages : {18: array([[0.5570933]], dtype=float32), 21:
r resume: the top labels are: ['team', 'software'], with percentages : {20: array([[0.53800714]], dtype=float32)
r resume: the top labels are: ['up', 'at'], with percentages : {8: array([[0.50984275]], dtype=float32), 18: arr
r resume: the top labels are: ['at', 'coordinating'], with percentages : {18: array([[0.53621453]], dtype=float3
r resume: the top labels are: ['at', 'june'], with percentages : {18: array([[0.60265076]], dtype=float32), 1: a
r resume: the top labels are: ['team', 'on'], with percentages : {20: array([[0.5287711]], dtype=float32), 23: a
r resume: the top labels are: ['on', 'till'], with percentages : {23: array([[0.59821415]], dtype=float32), 16:
r resume: the top labels are: ['on', 'sql'], with percentages : {23: array([[0.52007824]], dtype=float32), 7: ar
r resume: the top labels are: ['required', 'june'], with percentages : {24: array([[0.5808345]], dtype=float32),
r resume: the top labels are: ['team', 'through'], with percentages : {20: array([[0.50880617]], dtype=float32),
```

Figure 15 : Results

6 Conclusion

In conclusion, this project has provided an overview of several important concepts in natural language processing, including dataset, data processing, vectorization, word embedding, and multiclass classification.

The dataset used in this project was preprocessed and vectorized using TF-IDF method to represent the text data numerically. Then, word embeddings were learned using the word2vec method. These embeddings were used as input for a multiclass classifier to train a model for a specific task.

The performance of the model was evaluated on a test set, and the results were used to make decisions about whether to continue using the model or make adjustments. The choice of a proper evaluation metric was also discussed.

Overall, this project has demonstrated the importance of preprocessing and vectorizing the data before training a model, as well as the advantages of using word embeddings in natural language processing tasks. The use of a multiclass classifier was also highlighted, as it allows for classification into more than two classes, which is often required in real-world applications.

7 Reference

— https ://spacy.io/models/en
— https ://www.tensorflow.org/tutorials/text/word2vec
— https ://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/
— https ://www.kaggle.com/datasets/gauravduttakiit/resume-dataset
— https ://tika.apache.org/
— https ://jupyter.org/
— https ://www.python.org/