Group Project – Team ZeRoS: NLP: Resume Extraction Using NER

Team Members:

Reeka Hazarika (UAE) – Free Lancer
hazarika.reeka@gmail.com
Swapnil Vishwakarma (India) - SIES Graduate School of Technology
swapnilvishwakarma7@gmail.com
Zyad Al-Azazi (Lebanon) – Lebanese American University
zyadazazi@gmail.com

Submitted to Data Glacier on: 15/05/2021

GitHub Repo Link:
https://github.com/zyadalazazi/resume_extraction_team_zeros

Problem Description

When companies recruit for any position, they usually end up receiving thousands, if not millions, of resumes. Such a huge number of resumes makes the task of going over all these resumes an extremely difficult and tedious job for HR employees. This made a lot of companies opt for systems that take the necessary information from the candidate after they fill an application with all the required fields. The solution worked great for employers; nevertheless, candidates have always found it very illogical to spend tens of hours sharpening their CVs and cover letters only to find out that they must spend another hour or so re-entering all the information they have on their CVs in the designated fields.

Business Understanding

It is often observed by HR that the manual process of evaluation of resumes in bulk which are populated with excess information often becomes tedious and hectic. Therefore, we could automate this process by reading several formats of files (CVs). Then using some basic techniques of Natural Language Processing like word parsing, chunking, regex parser and/or Named Entity Recognition to easily capture information like name, email id, address, educational qualification, experience in seconds from a large number of documents.

Project Life Cycle

1. Data Understanding and Exploration.
2. Data Cleaning and Transformation.
3. Performing Exploratory Data Analysis and EDA Presentation
4. Model Selection and Building.
5. Model Deployment and Final Project Presentation.

Data Understanding

Our dataset is unstructured data of text in json format. When the data was imported using the Pandas data frame, we were able to know more details. The dataset contains two main columns: content and annotation. Content is the main text of the resume, whereas annotation is the labeling of the information provided in the content. It represents the resumes of 200 different people. These resumes include information about the applicants; this information is labeled into different categories: name, location, contact information (indeed account), university/college name, degree, graduation year, years of experience, companies of previous experience, designation, and skills.

Type of Data

Unstructured data (text) in json format.

Data Problems & Solutions

Since the type of our data is text, there are no outlier data points. Also, many of the issues related to quantitative data distribution, such as skew, or the need to statistically normalize the data are not applicable to our case. However, there was one record that was found to be replicated once in the dataset; we had to drop it. In addition to one more record that was not humanly readable and affected our analysis; hence, we had also to drop it.

Data Cleaning Techniques

- Lowering case of all characters.
- Expanding contracted words.
- Removing unnecessary characters (including full stops at the end of sentences).
- Removing stop words.
- Tokenizing according to spaces using the split() function.
- Lemmatizing words.

As for featurization, TF-IDF (Term Frequency - Inverse Document Frequency) was used with n-gram ranging between 1 and 3.

Exploratory Data Analysis

During the initial period of EDA, we were looking out for various ways to explore the text data since no one of us had any prior experience with exploring the text data. So before exploring the dataset itself we checked for few basic things like the number of resumes available with us and if it contained any duplicate resume (descriptive data analysis). While describing the whole dataset, we came to know that it had a duplicate value in it, so we dropped it from our dataset for further analyses.

We also noticed that the resumes contained some shorthand notations like "I'm" for "I am", "I've" for “I have" and many more for which we created a dictionary of common English contractions that we used for mapping the contractions to their expanded forms using a function which uses regular expressions to map the contractions in the text to their expanded forms from the dictionary. After this, we also removed the stop words from the dataset. Then, we created a function to get the average word length for each of the resumes to find out the average word length was around 7 characters. We also plotted word count distribution to find out that most of the resumes contained at most 500 words.

Next, we wanted to see the frequency of each word occurring in the dataset, so we created the distribution of Unigram words, Bigram words, and even Trigram words to analyze the most frequent words solely, in a pair and three words at-a-time, respectively. From these distributions, we came to know that “indeed” (the professional social media platform), “year” and “management” were the most frequent words when it comes to unigrams. “Work experience,” “email id,” and “additional information” were the top three bigrams. From the trigram distribution, we came to know that most of the candidates belong to or worked in Karnataka and were willing to relocate to work. Then, the distribution of Part of Speech was done to find out that most of the words were a singular noun. The next step was to plot the Word Cloud which is a visual representation of the frequency of different words present in a document. It gives importance to the more frequent words which are bigger in size compared to other less frequent words.

We were also able to come up with some interesting insights when looking at the 75 most frequent words. The most mentioned companies on the applicants’ resumes were Microsoft and Oracle.

Some of the keywords that the applicants emphasized on, in order, were “management,” “data,” “testing,” “customer,” “business,” “technical,” and “software.” If we assume that all the applicants were applying to the same vacancy, we can hypothesize that this job is a leadership role that requires business and customer-communication skills accompanied with technical skills.
After this, we tried exploring the dataset using the unsupervised machine learning technique using the technique called “K-means clustering.” When attempting to represent the 198 resumes using K-means clustering, it was clear that the graph fails to correctly represent and cluster all the points – resumes in our case. The reason behind this is the high number of dimensions that each vector possesses since we were using the Tf-Idf vectors to represent each resume.

After the failure of K-means clustering, we used Hierarchical clustering which groups data over a variety of scales by creating a clustering tree or a Dendrogram. The tree is not a single set of clusters, but rather a multi-level hierarchy, where clusters at one level are joined as clusters at the next level. The tree shows that we can split our data into 8 separate clusters. We applied the method of how unstructured text data for a specific field, namely, recruiting can be organized. With the right feature engineering (TF-iDF transformation) it is possible to split resumes into different groups (here we have 8 groups).

Model Building

NER Training Model with SpaCy

- We used Python’s SpaCy module for training the NER model.
- SpaCy’s models are statistical and every “decision” they make is a prediction. This prediction is based on the examples the model has seen during training.
- The model is then shown the unlabeled text and is expected to make a prediction.
- Because we know the correct answer, we can give the model feedback on its prediction in the form of an error gradient of the loss function that calculates the difference between the training example and the expected output. The greater the difference, the more significant the gradient and the updates to our model.
- At each iteration, the training data is shuffled to ensure the model does not make any generalizations based on the order of examples.

- The model is tested on and the predicted summarized resumes are stored in separate .txt files for each resume.

Model Deployment

As per our client’s request, we were able to create functions that would accept and feed different types of files, other than .txt, such as .pdf and .docx. This was an important preliminary step and have been proved to work (more on that in the limitations part). Unfortunately, we were not able to successfully deploy the app because of some issues we faced. Due to the lack in time, we were not able to fix these problems as some of them seemed to be concerning the way the model was imported. Nevertheless, the model works fine on the notebook and can be tested on some of the resume samples we provided in our repository.

Limitations

Aside from not being to deploy our model as an application, there was a problem in our model’s accuracy that we have come to notice. The model had several wrong predictions in some resumes, while not being to extract information in some others. The main two reasons we thought of were: the nature of resumes not having a specific format or even outline and the limited dataset that were used in out project. As for the first point, the huge variety in resumes and CVs outlines makes it very challenging to automate the task of extracting information even for humans, who need to read the whole document to be able to know the necessary information. As for the second reason, it is very challenging to try to generate one’s own data as it needs to be carefully, thoroughly, and accurately annotated. Other datasets that are concerned with this problem were not found.