Resume Extraction with NER Model using Spacy

Arda Cem Ozmen ardaozmen3@gmail.com

Buse Küçükçoban buse_kucukcoban@hotmail.com

May 2021

1 Abstract

To design a model this can parse information from unstructured resumes and transform it to a structured JSON format. Also, present the extracted resumes to the employer based on the job description.

2 Introduction

Corporate companies and recruitment agencies process numerous resumes circadianly. This is no task for humans. An automated perspicacious system is required which can take out all the vital information from the unstructured resumes and transform all of them to a mundane structured format which can then be ranked for a concrete job position. Parsed information include designation, electronic mail address, gregarious pro_les, personal websites, years of work experience, work experiences, years of edi_cation, edi_cation experiences, publications, certi_cations, volunteer experiences, keywords and determinately the cluster of the curriculum vitae (ex: computer science, human resource, etc.) The features in various cv types can be made easier to follow by using classi_cation to make them more understandable.

3 Quick Guide

3.1 Data

"Resume.json" file in the project was used, the data structure is as follows:

```
Dataset Attributes
{
'content' : resume content or free text
'label' : Annotated Tagged Entities
 }
```

3.2 Named Entity Recognition (NER)

Named-entity apperception (NER) (additionally kenned as entity identi_cation, entity chunking and entity extraction) is a sub-task of information extraction that seeks to locate and relegate designated entities in text into pre-de_ned categories such as the designations of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. NER systems have been engendered that use linguistic grammar-predicated techniques as well as statistical models such as machine learning. Hand-crafted grammar-predicated systems typically obtain better precision, but at the cost of lower recall and months of work by experienced computational linguists. Statistical NER systems typically require a substantial quantity of manually annotated training data. Semisupervised approaches have been suggested to evade part of the annotation effort.

3.3 Convert Json to Spacy accepted Format

Being a free and an open-source library, spaCy has made advanced Natural Language Processing (NLP) much simpler in Python. spaCy provides an exceptionally e_cient statistical system for named entity recognition in python, which can assign labels to groups of tokens which are contiguous. It provides a default model which can recognize a wide range of named or numerical entities, which include company-name, location, organization, product-name, etc to name a few. Apart from these default entities, spaCy enables the addition of arbitrary classes to the entity-recognition model, by training the model to update it with newer trained examples.

3.4 Model Training

We use python's spaCy module for training the NER model. spaCy's models are statistical and every "decision" they make — for example, which part-of-speech tag to assign, or whether a word is a named entity — is a prediction. This prediction is based on the examples the model has seen during training. We used 15 resumes of various structures to train our model.

3.5 Model Performance Evaluation

We calculated the accuracy values in 15 resumes we used for the model.
It is observed that the results obtained have been predicted with a commendable accuracy.

3.6 Saving Model and Loading Model

Training is an iterative process in which the model's predictions are compared against the reference annotations in order to estimate the **gradient of the loss**. The gradient of the loss is then used to calculate the gradient of the weights through backpropagation. The gradients indicate how the weight values should be changed so that the model's predictions become more similar to the reference labels over time.

When training a model, we don't just want it to memorize our examples – we want it to come up with a theory that can be generalized across unseen data. After all, we don't just want the model to learn that this one instance of "Amazon" right here is a company – we want it to learn that "Amazon", in contexts like this, is most likely a company. That's why the training data should always be representative of the data we want to process. A model trained on Wikipedia, where sentences in the first person are extremely rare, will likely perform badly on Twitter. Similarly, a model trained on romantic novels will likely perform badly on legal text.

This also means that in order to know how the model is performing, and whether it's learning the right things, you don't only need training data – you'll also need evaluation data. If you only test the model with the data it was trained on, you'll have no idea how well it's generalizing. If you want to train a model from scratch, you usually need at least a few hundred examples for both training and evaluation.[1] In image 1, you can examine diagram of training model.

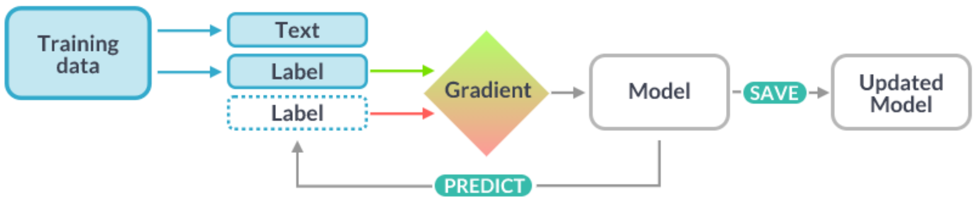

Image 1: Training diagram

4 Discussion

We may not get the results we want due to the nature of creating some resumes.

5 Conclusion

We transformed different types of unstructured resumes into data that we can obtain in a more organized and concrete way using Named-entity recognition classi_cation. Thus, more effective evaluation can be made and data can be made more organized.

6 References

A Two-Step Resume Information Extraction Algorithm, May 2018 (Jie Chen,Chunxia
Zhang, Zhendong Niu)

[1] Training Pipelines & Models (https://spacy.io/usage/training)