Resume Named Entities Recognition with SpaCy

Qiwen Zhang
Halıcıoğlu Data Science Institute
University of California San Diego
La Jolla, CA 92093
q2zhang@ucsd.edu

Yufei Zhou
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
yuz051@ucsd.edu

Judy Yang
Halıcıoğlu Data Science Institute
University of California San Diego
La Jolla, CA 92093
s7yang@ucsd.ed

Haibo Li
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
hal011@ucsd.edu

Abstract

A common issue among recruiters in big companies during their resume reviewing is that they often have to spend large amounts of time identifying important information on the applicants' resumes. This paper uses Named Entity Recognition models to help identify and highlight core information on a resume, so recruiters can narrow down their focus and review resumes efficiently. To achieve this goal, we implemented spaCy module as the foundation architecture and trained the token to vector, RoBERTa-based, and transition-based NER models within the pipeline. After rounds of model fine-tuning, we are able to achieve a 0.426 F1 score in spaCy model. To further improve model performance, we can consider using a more robust labeling technique and gathering diverse datasets. The model code can be found at https://github.com/QiwenZz/resume_NER. The website code can be found at https://github.com/Haibo3939/NER-Web-App.

1 Introduction

Named Entity Recognition (NER) has always been important in downstream tasks such as building a knowledge graph, user internet modeling, question answering, dialogue systems, etc. as it allows us to extract real-world entities from text and label them using predefined types.

After getting familiar with different NER frameworks and replicating methodologies to combat issues like messy labels for open-domain NER tasks due to the lack of restrictions, we want to put the theoretical framework into practice and build a useful application using the knowledge and skills we obtained from quarter one.

As soon-to-be graduates, we can easily get frustrated about job applications and concerned about resumes not going through screening. There is no doubt that a resume is the most important stepping stone to secure a job. Therefore, we want to focus on creating an automatic resume information detector, where we implement Named Entity Recognition models to automatically highlight information on the resume; this will help people compare resume content with the job description in order to create a more refined and targeted resume.

From the HR side, they are reviewing hundreds of resumes just for one competitive position. Because of the strong competition in the job market, 75% of the resumes are rejected before they reach the hiring manager. Most resumes do not even get past the applicant tracking system (ATS), meaning they are not seen by human eyes. This raises a serious issue for both the recruiters and applicants. Recruiters may miss out on outstanding and qualified candidates. Applicants are wasting time on submitting resumes without a clear target. Technology is making it easier to submit applications, but it is also easier to reject candidates. Our intention is to find a sweet spot in the middle by using data science techniques to produce a precise resume information detection tool that will be extremely helpful to both recruiters and applicants.

In order to train a NER model on the resume data, we want to locate and classify named entities into predefined labels. With manually annotated tags, we can map each token in the sentence to a corresponding label. For resume data, in particular, these labels could be information extracted from resumes and often generalize a candidate's contact information, education and work background, etc. In this project, we are particularly interested in entities including Name, Email Address, College Name, Location, Designation, and Skills. The exact model we implemented comes from the module spaCy. The pre-trained model has high efficiency compared to previous models we studied, and it has the ability to maintain accuracy fairly well. SpaCy models can get updated by the training data repeatedly in small batches, and with the help of dropout rate, it will reduce variance and better generalize unseen data.

2 Datasets

Four source datasets were included in the current project. Three of which were downloaded from Kaggle, GitHub and livecareer.com and the last one was web-scraped from postjobfree.com. 220 resumes were downloaded from Github[1] and they were annotated already. 3418 raw resumes were collected from the other three datasets in which 30, 197, 120 resumes were selected and annotated from Kaggle, postjobfree.com, livecareer.com datasets respectively. Hence A total of 547 resume data were used in the current study.

An open-source annotator has been used to manually label the entities[2]. 15439 named entities were included from these resumes, which come from 10 categories. The categories are college name, companies worked at, degree, designation, email address, graduation year, location, name, skills, and years of experience.

Pre-processing of the named entities include: 1. remove UNKNOWN category from the Github repo data[1] 2. remove overlapping named entities 3. trim white spaces on two ends of named entities 4. trim special characters on two ends of named entities

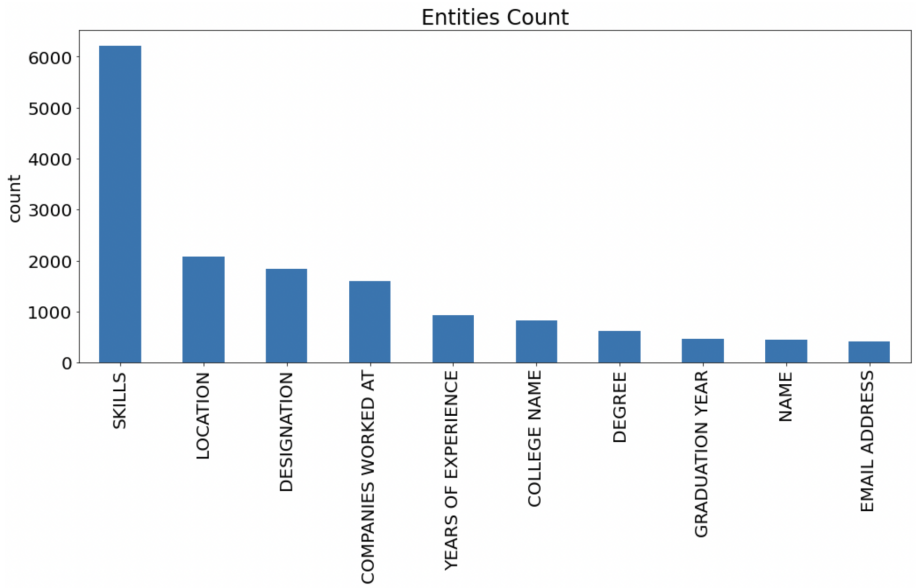

Figure 1: Named Entities Barplot By Category

In figure 1, we can see that skills are the most seen named entities over the 10 categories, which constitute almost half of all the entities. It also makes sense to observe that degree, graduation year, name and email address are the least frequent named entities as they often appear only once per resume.

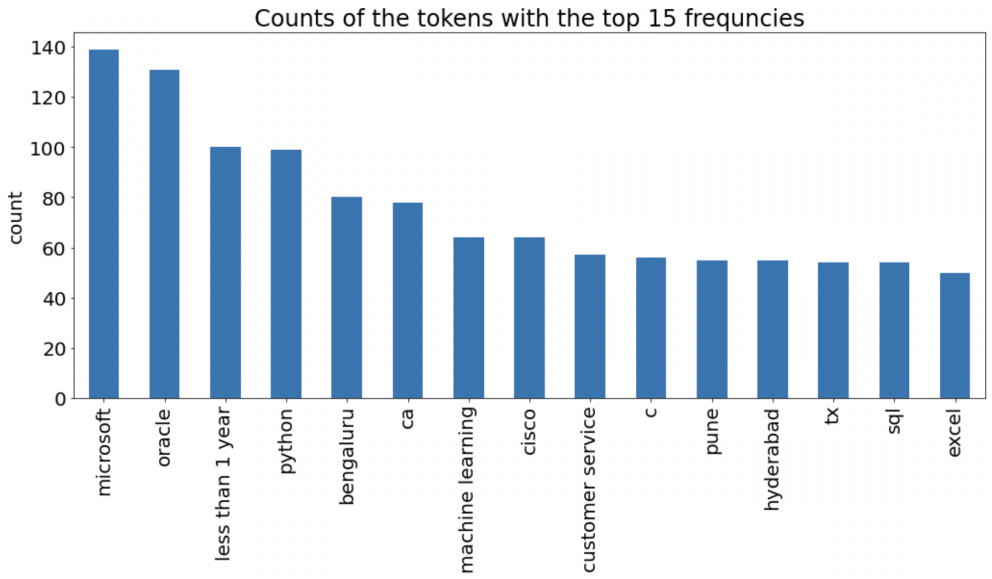

Figure 2: Barplot of top 15 frequent entities

For the visualization in figure 2, the following preprocessing has been done with the help from the library of nltk: 1. lowercase all entities 2. delete special characters such as '!', '?'... 3. remove stop words 4. remove blank entities. After these preprocessing tasks, we can see that lots of resumes contain information in the IT field such as a high occurrence of Microsoft, Oracle, Python, C, etc. Less than 1 year is often seen as a supplemental attribute to the skills. A high occurrence of this entity can imply that most of the skills on the resumes are used for less than a year. An interesting finding is that frequent location entities such as Bengaluru, Pune, and Hyderabad are all in India, implying that the resumes in the current data are largely from the Indian population and might be a factor that leads to bias during training.

3 Method

Named-entity recognition (NER) is a sub-task of information extraction that seeks to locate and classify named entities in text into predefined categories such as the names of persons, organizations, and locations. NER systems have been created that use linguistic grammar-based techniques as well as statistical models such as machine learning. In order to implement NER tasks, we need a large amount of manually annotated training data.

In order to conduct a Named Entity Recognition task on resume data, we need to locate and classify named entities into predefined categories. If we set tokens in the target sentence to be $X = [x_1, x_2, \dots, x_n]$, and each entity contains tokens space $s = [x_i, x_j, \dots, x_n]$, our task would be map labels $Y = [y_1, y_2, \dots, y_n]$ to X so each token in the sentence would be assigned to corresponding labels. For resume data, in particular, these labels are often key information extracted from resumes and often generalize a candidate's contact information, education and work background, etc. Therefore, it is a significant improvement to a higher level resume processing if we apply a NER model to the resume dataset. In this project, we are particularly interested in entities including Name, Email Address, College Name, Location, Designation, and

Skills. For model evaluation, F1 score, Precision, and Recall are the major metrics we refer to in order to identify the best model training results.

3.1 SpaCy

We use python's spaCy module as a foundation architecture for training the NER model. The spaCy module is commonly used as a production-based NLP developer tool and it also offers access to larger word vectors that are easier to customize, which is closely relevant to our NER application. spaCy's models are statistical and every "decision" they make is a prediction. The model is then shown the unlabelled text and will 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 (Figure 3). When training a model, we don't simply want it to memorize our examples , where we want it to come up with a theory that can be generalized across other examples. In order to tune the accuracy, we process our training examples in batches, and experiment with minibatch size and dropout rates.

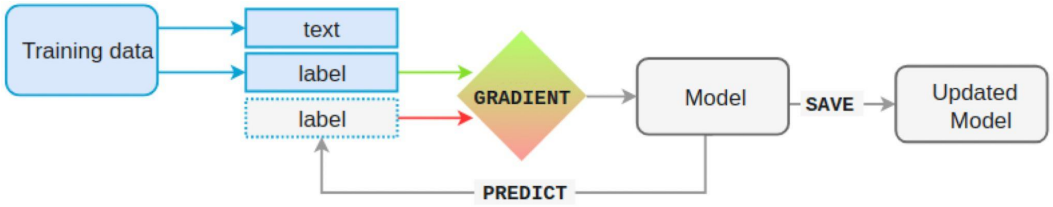

Figure 3: SpaCy model training mechanism

3.2 Models

In order to compare model performance and derive the most optimal model, we trained three models, the baseline model, the RoBERTa fine-tuned model, and the spaCy fine-tuned model. In addition, there are three submodels we used in the process of each of the three model's pipeline: Token-to-vector model, RoBERTa model, and transition-based named entities recognition model.

Token-to-vector model (spaCy.Tok2Vec):

This model embeds tokens into vectors that contain contextual information. It uses hash embedding to convert tokens to vectors which are context-independent and then uses convolution with maxout activation function to learn context information.

RoBERTa pre-trained model[4]:

This is a transformer model loaded from the HuggingFace library and has pre-trained weight and a PyTorch implementation. The **RoBERTa** model maps tokens into vectors with contextual information. The model uses a similar architecture to BERT while still different in a few ways: 1. RoBERTa was trained on 10x more data than BERT 2. It uses a dynamic masking pattern instead of a static masking pattern 3. It's trained on Longer Sequences. The attention mechanism in transformers enables the model to learn broader and more accurate contextual information for each token in a sequence**[5]:**

It uses a transition-based parsing to predict the named entities using structural information. The model relies on a stack data structure to incrementally construct chunks of the input. This model can only predict discontinuous named entities so any overlapping entities have been removed from the training sequences.

3.2.1 Baseline Model

The baseline model includes a token-to-vector model to first preprocess the resume data and a transition-based named entity recognition model to further predict the predefined entities using the resume vectors.

3.2.2 RoBERTa Fine-tuned Model

The model is a combination of the RoBERTa Fine-tuned Model and a transition-based NER model. Labels and raw text are passed into a roBERTa-base model and then optimally trained to continuously update themselves.

3.2.3 SpaCy Fine-tuned Model

The model includes a pre-trained token-to-vector model from spaCy and a transition-based NER model. Similar to the RoBERTa fine-tuned model, spaCy fine-tuned model also required a constant update using raw texts and annotations.

4 Results

After fine-tuning our models and adjusting different hyperparameters to achieve the best model output, we found that the spaCy fine-tuned model has the highest F1 and precision, while the RoBERTa fine-tuned models have the highest recall. There is some slight tradeoff between the models. The models are interactively displayed on our website, through which users can input their resume sample in pdf format each time, and obtain a highlighted resume with texts regarding the key information labeled. Website was built based on an open source Github repository[3]. The website link can be found at https://dsc180nerweb.herokuapp.com.

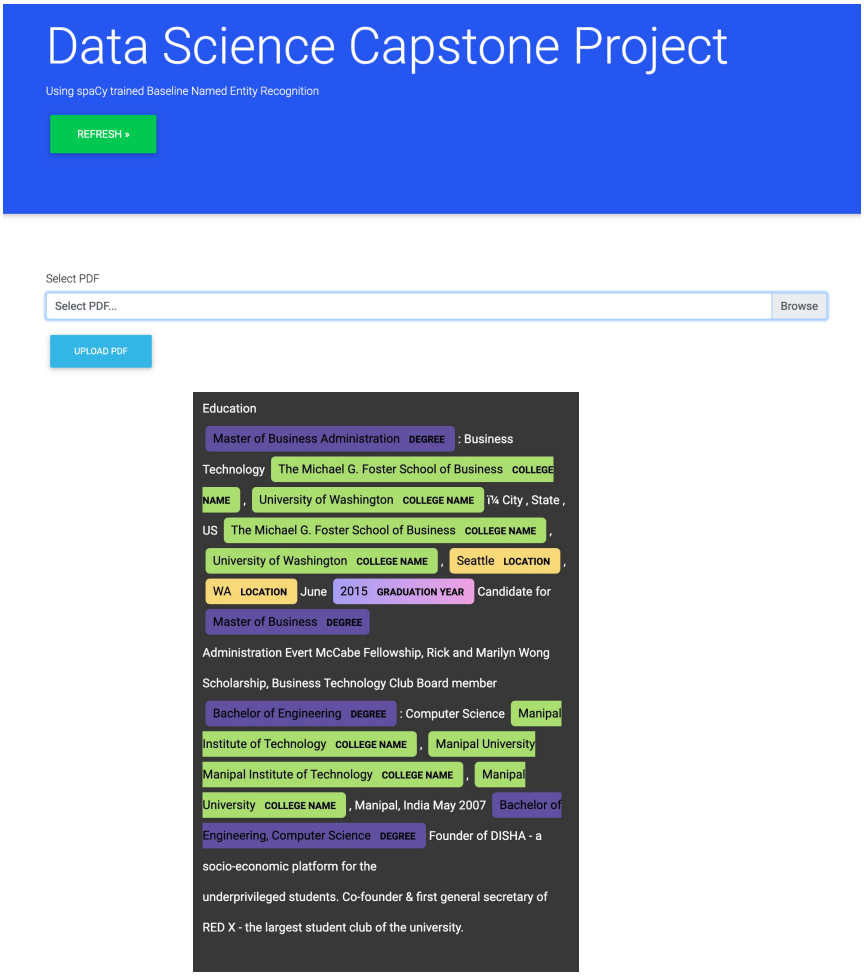

Figure 4, 5: Project interactive website and corresponding resume output

The above figure is an example output of our website. Words and phrases belonging to graduation year, companies worked at, email address, years of experience, college name, location, name, and skills would be recognized and highlighted in different colors. For user-input resumes, our model could achieve good accuracy in entity recognition and highlighting among various categories.

As per the exact model training and testing results, We compared the F1 score, precision, and recall of all our three models. In general, the spaCy fine-tuned model has the highest F1 and precision, while the RoBERTa fine-tuned models have the highest recall. Details regarding different models' metrics are explained below.

4.1 Model Performance

Table 1: Model performance summarization

Model	Evaluation Metric	Evaluation Score
Baseline	F1	0.368
	precision	0.609
	recall	0.264
Roberta fine tuned	F1	0.422
	precision	0.423
	recall	0.421
Spacy fine tuned	F1	0.426
	precision	0.649
	recall	0.316

Our model performs especially well in the education section like the table shown above. This might be because the entities about school names have relatively uniform formats. Most of the colleges have names with keywords such as "university", "school", "college", and most of the degree entities have the keywords such as "bachelor of" or the major name. Moreover, the contexts among education category entities are similar. People always describe their education level, major studied, school attended, and school attending year together, so it helps the model to find the education entity object nearby.

On the contrary, the entity recognition about the designation object is not working as well. The model might confuse the designation entity with some other categories, like the skills section or the degree type, since the latter two also contain the vocabularies similar to the designation entities'.

In terms of the model metrics, our best is selected as the rep-trained spaCy model. It can achieve an F1 score to be as low as 0.368, precision as 0.609, and recall as 0.264. Another model we implemented with close performance is the RoBERTa model. It has the F1 score to be 0.422, the precision to be 0.423, and the recall score to be 0.421. Considering the relatively training labeled dataset we applied, this precision is already quite high, suggesting our method is reasonable and

has the potential to be used on a larger scale.

4.2 Detailed Model Performance Comparison

Table 2: Model performance in evaluation metric

Evaluation on the small validation dataset			
Model	Evaluation Metric	Trained on Small Dataset	Trained on Large Dataset
Baseline	F1	0.582	0.506
	precision	0.652	0.573
	recall	0.525	0.453
RoBERTa fine-tuned	F1	0.663	0.390
	precision	0.714	0.286
	recall	0.619	0.613
spaCy pre-trained model fine-tuned	F1	0.625	0.609
	precision	0.730	0.637
	recall	0.547	0.583
Evaluation on the large validation dataset			
Model	Evaluation Metric	Trained on Small Dataset	Trained on Large Dataset
Baseline	F1	0.092	0.368
	precision	0.214	0.609
	recall	0.059	0.264
RoBERTa fine-tuned	F1	0.161	0.422
	precision	0.490	0.423
	recall	0.096	0.421
spaCy pre-trained model fine-tuned	F1	0.111	0.426
	precision	0.372	0.649
	recall	0.066	0.316

While tuning the model, we employed different metrics on models with various hyperparameters

and pre-trained methods. To improve the efficiency of model training and validating, we made use of small and large training and validation datasets and recorded different results based on F1, precision, and recall scores. The small dataset is the one that's downloaded from github directly where training data consists of 200 resumes and testing data consist of 20 resumes. The large dataset includes an addition of all the manually annotated resumes to both training and validation set. By evaluating on both datasets, we could see that incorporating more data for training increases the performance when evaluating on the large datasets. By comparing the results across the models, we could see that there is a large improvement with the fine-tuned RoBERTa and spaCy pre-trained models.

When using larger datasets as the training data, the models' performance gets better for all the models we are showing here. The reason can be attributed to the pre-trained property of these two models. With the limited dataset we are using, the pre-trained model layers allow us more flexibility to fit on the current model with reasonable model structures, and also ease our training process. Moreover, compared with the models trained from the scratch, pre-trained models can prejudice more reliable testing outputs as it requires less data for the training process. However, for the RoBERTa fine-tuned model, there is a slight decrease in its precision, which might be explained by the possibility that the small validation dataset can have a good representation of the data. This explanation may also be applied to other cases where the metrics between the small dataset and the large dataset group are small. On the contrary, for some other model metrics, like the precision and F1 score of spaCy pre-trained and fine-tuned model, there can be several multiples increments in the metrics. Therefore, the size of the data used for our resume entity recognition model could be a key variable. The ideal model needs to be trained and evaluated using relatively large and comprehensive datasets.

5 Discussion

One interesting finding from table 2 is that while the evaluating metrics on the bigger evaluation dataset is better for the models trained on the larger dataset, the evaluating metrics on the smaller evaluation dataset is worse for the models trained on the larger dataset. This inconsistency was investigated by checking the testing sample's output on the small validation dataset. The largest discrepancy in f1 score exists between the RoBERTa fine tuned models that are trained on the small and big dataset (f1 score of 0.663 vs 0.390) hence this particular model was investigated. It can be observed that the recall for both models are similar(0.619 vs 0.613) whereas the precision for the model that's trained on the larger dataset is much smaller than the model that's trained on the smaller dataset(0.286 vs 0.714). A significantly lower precision indicates that the model recognizes too many named entities that are not actually named entities. When I looked at a few resumes' NER outputs in the small validation dataset, I found why the precision is much lower for the model that's trained on the larger dataset. Most of the resumes in the small validation dataset tend to follow a pattern that a black dot is followed by a work experience description. Models trained using the small dataset tend to ignore these descriptions since they do not belong to any named entities. Models that are trained using the large dataset, on the other hand, tend to recognize the first word of the descriptions as the 'Companies Worked At' entity, hence generating a lot of false positives and lowering the precision(figure 6). This indicates that the small validation set is really biased since most resumes follow a format pattern. There are only 20 resumes in the small validation set hence this evaluation dataset is not very objective. Also, this observation explains why the f1 scores are extremely low for all three models that are trained using the small datasets when they are evaluated using the large dataset(0.09, 0.16, 0.11). Because the training data for the small dataset all follow a similar pattern, it makes the model trained on these data very hard to generalize to the unseen dataset(the added manually annotated data). Hence, incorporating new manually annotated dataset for training is crucial for the model to generalize to real world resumes that can be inconsistent in formatting.

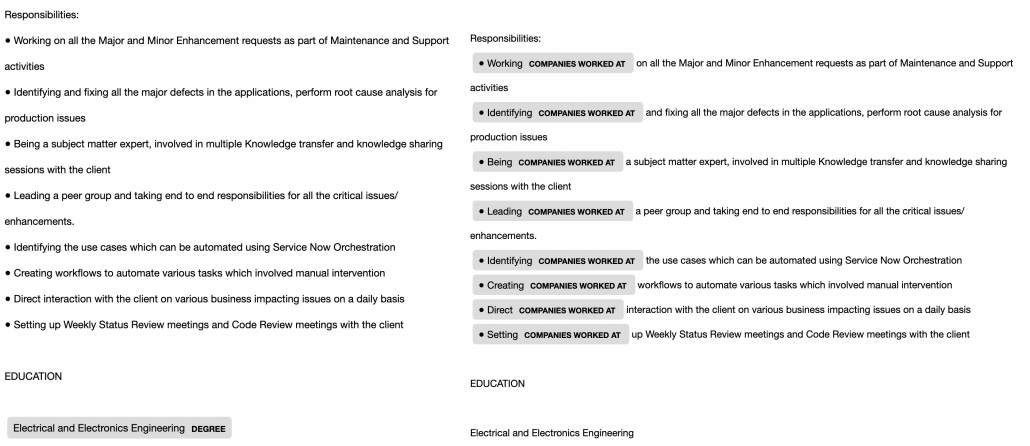

Figure 6: Sample outputs from model trained on the small dataset(left) and large dataset(right)

After studying our model performance results and the general output produced by the model, we noticed some potential limitations about our methods and the improvements we can make in the future. First, we manually labeled all the raw resume data in a short period of time. Although the labeling algorithm we used is the same among everyone, with manual labeling, it still has a higher chance to result in errors and inconsistency between people. Even though we were trying to adopt the same criterion for data labeling, there can still be individual differences regarding our labeling criteria, like the length of words we would like to label for certain entity subjects. Given the datasets we are using for model training and validation are not especially large, the difference can have a significant impact on our modeling outputs. Second, the training set we are using is not large enough. From our experiments, we see a clear better performance when using the larger datasets. The results are also more robust with bigger datasets. Therefore, if we have enough resources to expand the training set, the result can get even better. Lastly, we realized that our resume data sources are not diverse enough, where some of them have a similar pattern and format. We scraped and collected resumes from limited job hunting platforms, and most of the professions represented in these resumes are similar, so there might be bias when we use the same trained model on recognizing entities from non-common resumes. For instance, most of the training resumes have a separate section for skills. However, when skills are embedded between texts, the model has a hard time teasing them out. Although we have resumes from different occupations, the types of the resume can be skewed to specific geo-locations, where we see a fair amount of resumes containing cities in India. Without prior knowledge of these cities, during manual labeling, it is easy to miss out on them resulting in no detection on location during testing.

With these concerns in mind, we are able to come up with some solutions to help improve performance in the future. Instead of manually labeling the data, we should adopt a more robust annotating tool or algorithm to help us identify labels. When it comes to data collection, we need to pay closer attention to the diversity of resume types. Finally, we can make sure to use a wide selection of pre-trained models and fine-tuning the models with different hyperparameters as an attempt to improve the model performance.

6 Conclusion

We compared the F1 score, precision, and recall of all our three models. In general, the spaCy fine-tuned model has the highest F1 and Precision, while the RoBERTa fine-tuned model has the highest recall. The spaCy and RoBERTa fine-tuned models both used pre-trained models with more complex networks and flexibility to handle name entity recognition tasks and that is also the reason why they outperformed our baseline model. Our model performs surprisingly well on entity recognition especially on the "Education" section of resumes. It recognizes most of the relevant entities from the section with only a few left out. This can be explained by the fact that entities in the "Education" section are usually identified with relatively unique words, such as "Bachelor of", "Master of", "XXX College", etc. Other entities are usually identified with ambiguous words that require a lot more contextual information. For example, entities in the "Skills" section are especially hard for the model to pick out as they are hidden in larger paragraphs and contain grammatical variants.

The contribution of the current project is that we formed a pipeline that consists of data collection, data preprocessing, model selection and training, and website application. This pipeline allows people to easily train a better model with newly collected resume data and put them into practice with our website software.

Reference

[1] DataTurks-Engg. "DataTurks-Engg/Entity-Recognition-in-Resumes-Spacy: Automatic Summarization of Resumes with NER -> Evaluate Resumes at a Glance through Named Entity Recognition." *GitHub*, https://github.com/DataTurks-Engg/Entity-Recognition-In-Resumes-SpaCy.

[2]Tecoholic. "Tecoholic/NER-Annotator: Named Entity Recognition (NER) Annotation Tool for Spacy. Generates Traning Data as a JSON Which Can Be Readily Used." *GitHub*, https://github.com/tecoholic/ner-annotator.

[3] Jcharis. "Machine-Learning-Web-Apps -> Displacy_App-Using-Displacy-in-Flask." *GitHub*, https://github.com/Jcharis/Machine-Learning-Web-Apps/tree/master/DisplaCify_App-Using-Displacy-in-Flask

[4] Liu, Yinhan, et al. "Roberta: A Robustly Optimized Bert Pretraining Approach." *ArXiv.org*, 26 July 2019, https://arxiv.org/abs/1907.11692.

[5] Lample, Guillaume, et al. "Neural Architectures for Named Entity Recognition." *ArXiv.org*, 7 Apr. 2016, https://arxiv.org/abs/1603.01360.