# Deep learning-based methods for technology skill named entity recognition

Tirthankar Mittra, Abulitibu Tuguluke

University of Colorado Boulder

**Abstract.** Several studies have reported that the US will face a tech shortage in the near future, which makes identifying technical skills from text data an ideal way for talent identification and recognition. In this project, we will introduce the Technologist Skill Named Entity Recognition (TSNER) task, where we apply rule-free and regex-free deep learning algorithms and traditional machine learning approaches to recognize technology-related named entities like skills, job headings, and company names. To the best of our knowledge, there are no models built specifically for our newly defined named entity recognition task, so we built three state-of-the-art models(Linear chain CRF, BiLSTM-CRF, BERT-CRF) that have previously been used for other named entity recognition(NER) tasks. The above models were used to conduct experiments and finally, we compared the efficacy of these models using precision, recall, and F1 scores. Hypothetically, humans can come up with a huge number of new NER tasks every day, our experiments suggest that to create a fast, lightweight, high-accuracy model for new NER tasks linear chain CRF with part of speech(POS) tagging should be used.

## 1 Introduction

The US, and the world in general, is facing a tech-talent shortage, which can be translated into revenue decrease[1]. One way to hire talent is through job postings, yet spotting the skills in a job still heavily depends on text parsing. A study done by Korn Ferry[2] predicted there will be roughly 85 million jobs unfilled by the year 2030. The shortages in the US can be amplified by insufficient CS, and STEM professionals. Named entity recognition (NER) is an information extraction technique that seeks to locate and classify named entities, such as(in our case) company name, job skills, etc. It will play an enormous role in talent spotting and recognition in the near future. Hence, in this project, we propose a new named entity recognition task: Technologist Skill Named Entity Recognition (TSNER), where we apply algorithms used for previous NER tasks, we use these algorithms to identify which organization requires what type of technology skill sets for a particular job title from a lengthy job description text.

A technologist is an expert in a particular field of technology, skills are tech words associated with that field of technology. For example, engineers and applied scientists are technologists, unfortunately, a musician or a lawyer is currently not considered in our definitions. As the name suggested, the tech skill

set is the skill that is associated with the technologist's job. We only consider certain words associated with the job title as skills, for example, SQL, Python, Ruby and omit the less relevant soft skills like writing and public speaking.

Traditional NER solutions involved dictionary look up, and expression rules, the disadvantage of these methods is that their performance heavily relies on human-designed rules which may not be able to fully generalize for edge cases, it also makes the task complex. Furthermore, there is still no developed method specifically targeted for the technologist skill set named entity recognition especially using deep learning models. Thus, we are proposing new models for our Technologist Skill Name Entity Recognition (TSNER) task, we will focus on a deeper understanding of NER application on technologist skill while comparing different deep learning and machine learning algorithms.

This problem is challenging due to three reasons. First of all, there are no pretrained models and word embedding for tech words or job descriptions. Therefore we need to transfer word embedding knowledge from BERT case-based models trained in the English language to our technologist skill name entity recognition task. Computation and time are also a constraint for working with BERT-based models. Second, we do not have labeled data for our supervised training and test set, so we need to come up with a good scheme for tagging, and be cautious about which word has an I-tech and other tags. Finally, although we have roughly 1.4 million job posting data, the majority are in HTML and poorly formatted, so we need a good parsing tool to get the job-relevant full text.

## 2 Related Work

### 2.1 Named Entity Recognition

Paper [3] gives us a clear picture of nine models that the authors used for natural hazard named entity recognition, BERT-CRF, ALBERT-CRF, XLNet-CRF, BERT-BiLSTM, BERT-BiLSTM-CRF, ALBERT-BiLSTM-CRF, XLNet-BiLSTM-CRF, BERT-BiGRU-CRF. They then proposed a new method, named XLNet-BiLSTM-CRF model, which had the highest precision (92.80%), recall (91.74%), and F1 score (92.27%). This may not be the case in our application, their model was specifically tailored for natural hazard-named entities, and our problem is focused on technologist skills. In our experiments, we will compare several models and pick the best-performing model. Second, the paper only defined 10 hazard categories with fewer words, and even fewer sample data (Wan-fang Database with 12,387 scientific papers), in our problem, we have multiple tech skills, and no categories are defined. Third, their tagging has more information such that the model has the following variables: $Haz\_sentenceModel = (origin, Location, Hazard, Method)$, we do not have location and method variables, which gives us fewer resources, which is another challenge.

For text information retrieval, word embedding is key to representing the data input. The second paper [4] proposed a new study DSpERT (Deep Span Encoder

Representations from Transformers), where they divide the word representation into two parts: standard transformer and span transformer (they then combine the two), first one is the traditional QKV representation, while the latter uses low layered span (deep semantic), where there is less span in a sentence, hence fewer layer, with a new transformer block along with the standard one. Our work will mimic this approach, but there may be a different word representation, since this one seems to only fit BERT based model, and we want a new scheme that works for both BERT and CRF.

### 2.2 Transfer Learning & Other Deep Neural Network Methods

Paper [5] talks about how recent advances in language representation have made it possible to transfer the learned internal states of a trained model to downstream natural language processing tasks, such as named entity recognition (NER) tasks. In this paper three distinct BERT models: Multilingual BERT-Base ($ML - BERT$), Portuguese BERT-Base ($PT - BERT_{BASE}$) and Portuguese BERT-Large ($PT - BERT_{LARGE}$) were used. The authors of the paper pre-trained ($PT - BERT_{BASE}$), ($PT - BERT_{LARGE}$) models from scratch using the largest open Portuguese corpus which consisted of 3.68 billion tokens from 3.53 million documents and it took around 4-7 days for them to train using a TPUv3-8 instance. Two transfer learning approaches: feature-based and fine-tuning were used. For the feature-based approach, the pre-trained BERT model weights were kept frozen only the classifier model and CRF layer were trained and for fine-tuning approach, the weights of the pre-trained BERT model were jointly updated. It was noticed that compared to previous non-BERT-based models the performance of BERT-based models improved significantly if the number of named entities to be identified also increased. We use the insights from this paper to design our model for example unlike the authors of the paper we don't plan to pre-train our BERT model from scratch nor do we plan to make our BERT layer trainable. One area of focus in this paper is how to create a lightweight quick model which gives high accuracy for our new NER task.

In this paper, [6], accuracy on NER datasets(using POS tagging and chunking) was compared for various non-BERT models like LSTM, bidirectional LSTM(BI-LSTM), LSTM with CRF layer(LSTM-CRF) and bidirectional LSTM with CRF layer(BI-LSTM-CRF). When spelling features, senna word embedding, and context features were fed to the Bi-LSTM-CRF model it produced the best result for all the tasks among all the models. In this paper, we plan to use this model as one of our neural network models for the technologist skill data set and compare its performance with the BERT-case-based model.

## 3 Methods

Named Entity Recognition(NER) models can be based on handcrafted rules or machine learning methods, neural network based NER has become popular because of minimal feature engineering, so in this paper, we plan to explore how

state-of-the-art neural networks for previous NER task performs on our new jobs dataset. The major challenge for this project will be labeling our huge unlabeled job posting data, this scenario is similar to real-life problems, where there are rarely any nicely labeled data. For all the models if padding was added to fix the length of a sentence the corresponding NER label for those padded entities was **UNK** in the output.

### 3.1 Data labeling

Our job data set is in JSON format[7], so in the first step we will be pre-processing the JSON file to extract the relevant data and then use regex to remove unnecessary characters, we then trim or pad the job description text to a fixed length **n** and then tokenize it. To solve the issue of labeling huge amounts of data, we plan to label the majority of the data using a "bag of words" model. Some of the data is also manually labeled, although this method will hinder the performance of our model as human annotations are much more accurate and relevant for our task. We initially use the "bag of words" model for labeling to just set up our model and then gradually feed more human-annotated data as it becomes available. For our project, we use three categories of named entities i.e. skill (**S**), job title (**J**), and company or organization (**C**), while the output for the rest of the inputs will be labeled as (**O**). For example,

| Google | wants | a | backend | developer | with | good | C++ | skill |
|---|---|---|---|---|---|---|---|---|
| C | O | O | J | J | O | O | S | O |

where all the job text from the first line, would be labeled as the second line. In future works, we plan to increase the number of named entities as more human labeling of data becomes available. For example, we plan to further divide skills into categories like soft v/s hard skills, relevant v/s irrelevant skills from a job description text. Previous works have shown that the performance of the neural network-based NER becomes significantly better as the number of named entities increases [5]. We will consider the labeled data set as ground truth, for evaluation purposes. Although we have millions of data points, we decided to train all our models on a smaller data set (we can always increase the training examples) mostly due to time and hardware constraints. We used no more than 20,000 randomly selected data to train and validate our models, the labeled data is split into training and validation sets.

We also use a non-neural network-based model i.e. linear chain CRF with part-of-speech (POS) tagging as our baseline for evaluating the progress made by our neural network-based models. Part of speech tagging tells us whether a word is a noun or a verb or an adjective, etc. For our task, we chose Bi-LSTM-CRF and BERT-CRF as our two neural network models. These models were chosen in particular because our survey of various NER tasks on different data set have shown that these models have given state-of-the-art performance. [5] [6]. Next, we will describe the high-level model architecture used for our NER task.

### 3.2 Linear chain (CRF)

In linear chain, CRF tag assignment for the present word,(denoted as $y_i$) depends only on the tag of just one previous word (denoted by $y_{i-1}$). For this, we need functions called feature functions that will assist in generating unique features. These features function can consider any logic (depending on the programmer). Linear chain CRF is a simple algorithm that can be summarized using the following sets of equations Eq[1].

$$p_\theta(y|x) = \frac{\exp(\sum_j w_j * F_j(x,y))}{\sum_{y'} exp(\sum_j w_j * F_j(x,y'))}, where$$
$$F_j(x,y) = \sum_{i=1}^{L} f_j(y_{i-1}, y_i, x, i) \tag{1}$$

Here, $p_\theta(y|x)$ is the probability of calculating a label sequence(y) given a sequence of words(x), and $F_j(x,y)$ is the summation of feature function $f_j()$ over the entire sequence of words. For our NER task, we choose the following feature functions, whether a word is a title, whether a word's first character is capitalized, whether all characters are capitalized, whether a word contains digits, etc. POS tagging was also used as one of our feature functions which made all the difference in producing high-accuracy results. The job of training in this model is to figure out the weights $w_j$ in Equation[1] so that the term $p_\theta(y|x)$ is maximized.

### 3.3 BERT-CRF

This model architecture is composed of a pre-trained case-based BERT model with a token-level classifier on top followed by a Linear-Chain CRF Fig [1], we choose to use case-based BERT model because named entities are case sensitive, also since BERT takes a lot of time to train we won't train the BERT layers again. To our BERT-CRF model we pass a string of words of a fixed length of 128, the BERT model we used here has its own tokenizer so passing string words were sufficient. For an input sequence of **n** tokens, BERT outputs an encoded token sequence with hidden dimension **H**. The classification model projects each token's encoded representation to the tag space, i.e. $R^H \rightarrow R^K$, where K is the number of tags, in our case we have four tags $\{S, J, C, O\}$. The output scores $Sc \rightarrow R^{nK}$ of the classification model are then fed to the CRF layer, whose parameters are a matrix of tag transitions $T \rightarrow R^{K \times K}$. The matrix $T$ is such that $T[i][j]$ represents the score of transitioning from $tag_i$ to $tag_j$. For our trainable layers, we used the 'rmsprop' optimizer to update model weights during backpropagation. Early stopping was used to prevent overfitting. In early stopping, we used patience of one, which means that training would stop as soon as the loss on the validation dataset increased compared to the last epoch. Our BERT-CRF model had 12 stacked encoder layer($L = 12$) with 768 hidden units($H = 768$) in the feed-forward layer of the BERT layer. In short, our model had 1 billion non-trainable parameters from the BERT layer and approximately 4000 trainable parameters from the CRF layer.

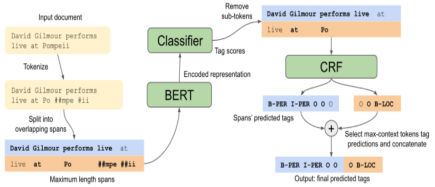



Fig. 1: BERT-CRF model.

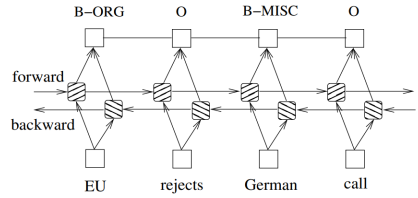



Fig. 2: Bi-LSTM-CRF model.

### 3.4 BI-LSTM-CRF

For our third model, we added a CRF network on top of a bidirectional LSTM network to get a Bi-LSTM-CRF network as shown in Fig. [2]. Unlike an LSTM-CRF model, a Bi-LSTM-CRF model can use the future input features so this model is superior for the purpose of our NER task and hence our choice to use it over the LSTM-CRF model. To the input of the BI-LSTM layer, we are feeding pre-trained GloVe embeddings of each word. The length of the sentence we are passing to our third model is also 128. Similar to our second model we are using the 'rmsprop' optimizer during back-propagation and early stopping with patience of one to prevent overfitting, which means that training would stop as soon as the loss on the validation dataset increased from the last epoch. In the BiLSTM layer of this model, we only used 64 hidden units in each direction followed by dense layers with 32 hidden units attached to these LSTMs for each time sequence, both the Dense and the Bi-LSTM layers used the 'relu' activation function. This model had approximately 100,000 trainable parameters.

## 4 Experimental Design

For our experiments we choose at most 20,000 randomly selected annotated data from our 1 million data set to train and validate our models, then trimmed each job description text to 128 characters, the reason for this is to standardize the model inputs. The labeled data was split into training and validation sets. For the base model CRF, we run the model on two sets of data, in the first one we used 1500(1.5k) training examples with 500(0.5k) validation examples, in the second one we used 15000(15k) training examples and 5000(5k) validation examples. We only trained our neural network models with the first set of data due to hardware limitations and memory issues. We then evaluate our models Bi-LSTM-CRF and BERT-CRF models against the CRF baseline model designed for NER, using the same data set. Finally, we document the recall, precision, and F1 scores for all three models on the test data set, we use these metrics specifically because it's a standard metric used for many other NER tasks and also because the distributions of the different named entities in a corpus are generally very skewed. A form of cross entropy loss on the validation data set was used as a metric to do early stopping and to solve over-fitting. Unfortunately,

we did not do much hyper-parameter tuning so the need for a separate test data set was not so imperative. The hardware we used was 32G memory/4G AMD graphic/ 2.6GHZ 6core CPU, along with Google Colab.

## 5 Experimental Results

Table 1 shows the evaluation metric for the CRF model for all 4 entities. We observed good results already with the first run of the model with 1.5k training examples validated on 0.5k validation examples, job title recognition has a better precision/recall/F1 score which is almost close to 1. Tech skill and company and also have above 0.99 scores, the only 'under' detect label is **O** (other). Once we run CRF on bigger training data 15k, and evaluate on 5k, even the **O** has precision/recall/F1 score most to 1. This is clearly due to better feature learning with bigger data. On top of context consideration, adding POS tagging into the model is one of the reasons this base model performed outstandingly. Another exciting observation is the quick training time for CRF related to our NER data, both pieces of training ran rather quickly and without any memory issues.

Table 2 shows the second part of our experiment results where we ran Bi-

| Train Size | Validation Size | Entity | Precision | Recall | F1-Score |
|---|---|---|---|---|---|
| 1.5k | 0.5k | C | 1.000 | 0.953 | 0.976 |
| | | J | 0.999 | 0.999 | 0.999 |
| | | O | 0.983 | 0.986 | 0.985 |
| | | S | 0.991 | 0.988 | 0.990 |
| 15k | 5k | C | 1.000 | 0.993 | 0.996 |
| | | J | 1.000 | 1.000 | 1.000 |
| | | O | 0.999 | 1.000 | 0.999 |
| | | S | 1.000 | 0.999 | 0.999 |

Table 1: Conditional random field (CRF) metric

LSTM-CRF along with the BERT-CRF model. Both models use bi-directional information to make predictions, yet due to the fact that both BiLSTM and BERT require more RAM for each layer, we ended up having memory overflow issues with 15k(15,000) training data. As a consequence, we used smaller 1.5k training samples for training and 0.5k for our validation set, we were not able to get good performance compared to our CRF model. The company name entity had zero scores, and the rest of the named entities also failed to give meaningful results on the validation set. From Table[2] it's clear that our pre-trained BERT CRF model performed much better than the BiLSTM-CRF model. The BiLSTM-CRF model only learned to classify words either as **UNK** or **J** also the validation loss increased after eight epochs for our BiLSTM-CRF model as can be seen from Fig[3] so it stopped learning. Deep Neural Network models

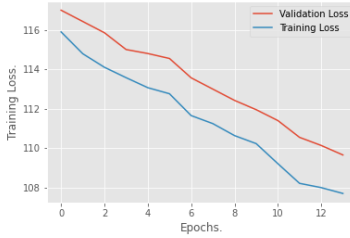
| Method | entity | precision | recall | f1-score |
|---|---|---|---|---|
| BiLSTM-CRF | C | 0.000 | 0.000 | 0.000 |
| | J | 0.545 | 0.992 | 0.704 |
| | S | 0.000 | 0.000 | 0.000 |
| | O | 0.000 | 0.000 | 0.000 |
| | UNK | 0.980 | 0.998 | 0.989 |
| BERT-CRF | C | 0.000 | 0.000 | 0.000 |
| | J | 0.581 | 0.870 | 0.697 |
| | S | 0.268 | 0.026 | 0.048 |
| | O | 0.341 | 0.105 | 0.161 |
| | UNK | 0.727 | 0.936 | 0.818 |

Table 2: Conditional random field (CRF) metric

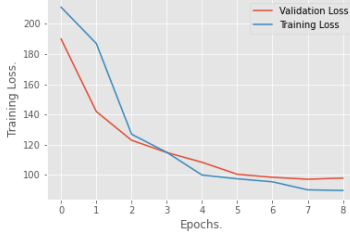
generally require huge amounts of data to train we believe with sufficient training examples our BiLSTM-CRF model would perform better, we also believe if we could incorporate POS tagging to the input to our BiLSTM model its performance would become better. For the BERT-CRF model from Fig[3], the loss curve shows it didn't stop learning till the maximum number of epochs had been reached, so training the model for more number of epochs with more examples would definitely yield better results, making the BERT layer trainable would also increase the performance but would make the training slower, hence we did not try this out. It was also noticed that using GPUs didn't significantly improve the training time for the Bi-LSTM-CRF model. One question we failed to answer is can more advanced models such as Bi-LSTM-CRF & BERT-CRF outperform CRF with bigger datasets and without memory issues.

## 6 Conclusions

In this paper, we introduced Technologist named entity recognition (TSNER) task. We evaluated precision, recall, and F1 scores on the job text with skill (S), job title (J), and organization (C) as our primary labels, for our different machine learning algorithms. Our results show the base model traditional linear chain CRF with POS tagging was able to accurately identify all four entities, including tech skills, organizations, and job titles with over 99 % accuracy, suggesting that it could be a useful tool in a new NER tagging application with little or no memory issues. Even though the BERT layer is defined as not trainable, it required extra GPU otherwise there was RAM overflow, making it less feasible than CRF. We have also learned that data collection and labeling are a big bottleneck for any supervised learning task. Our final takeaway is not every algorithm should be about Deep Learning, if there is a lightweight, practical easy-to-train model which gives better accuracy, e.g. CRF then there is no need for deep learning solutions which takes more resources and time to train.




BERT-CRF




BiLSTM-CRF

Fig. 3: Loss V/S Epoch Curves.

In addition, this paper only focused on high-demand STEM jobs from the job market, and paid no attention to the social science and skill related to that field, the ethical dilemma is all these studies will tilt toward engineering majors while ignoring the 'engineering of the mind' majors which play a key role in our society. There should not be any special treatment for any job just because of the market value and popularity, every harmonious society should accept the message that both social and technical knowledge are keys to success and happiness. The message we are not sending with this paper is that skills not related to tech are not important. In the future, we hope to replicate the same work for non-STEM job postings.

## References

1. Tuguluke, A.: Technologists data mining. Private project
2. Franzino, M.: The $8.5 trillion talent shortage. Korn Ferry Insight News
3. Sun, J., Liu, Y., Cui, J., He, H.: Deep learning-based methods for natural hazard named entity recognition. Scientific Reports **12**(1) (March 2022) 4598
4. Zhu, E., Liu, Y., Li, J.: Deep Span Representations for Named Entity Recognition (October 2022) arXiv:2210.04182 [cs].
5. Souza, F., Nogueira, R., Lotufo, R.: Portuguese named entity recognition using bert-crf. arXiv preprint arXiv:1909.10649 (2019)

6. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
7. Tuguluke, A.: job description json data dump, https://shorturl.at/eD137. Private data