Homework-1 Report

Mohammad Rajiur Rahman
UHID 1407264
COSC6336 Spring 2020

Abstract

This report presents an implementation of a named entity recognizer to detect and classify named entities in noisy and unstructured text from social media (twitter) using Conditional Random Fields (CRFs). Using sklearn-crfsuit package, my best model could achieve 94.59% accuracy and 33.22% f1-score (micro) over the given dev set data.

1 Introduction

Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entity mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, etc (Wikipedia, 2020).

Twitter receives more than 100 million tweets every day. It creates an enormous corpus of 140 characters long noisy informal message. Many times this collective data is more informative and inclusive than news articles. However, tweets, like any other social media data, are different from traditional texts from books, newspapers, magazines etc. They are full of informal language, misspellings, abbreviations, hashtags, @-mentions, URLs, and unreliable capitalization and punctuation. Specially hashtags and "@" play a major role in tweets. People talk about everything including new entities never used before. Moreover, due to twitter's 140 character limit, tweets often lack sufficient context to determine an entity's class or label. These factors present a challenge for general-purpose NER systems that were not designed for social media texts.

2 Related Work

There are significant contributions in NER for social media data. The present state of the art Flair (Akbik, 2018) reported 93.18% F1-measure on for NER on CoNLL-03 dataset, but their F1-measure drops to 49.49% F1-measure for WNUT-17 (tweeter) dataset. It outperforms previous best results of 92.22% F1-measure on CoNLL-03 (Peters, 2018) and 45.55% F1-measure on WNUT-17 (Guilar, 2017). Since our task was to use CRF for NER, I looked up related contributions (Jain, 2018; Carreras, 2003; Alan Ritter, 2011; Tkachenko, 2019; Sutton, 2012) for feature extraction methods.

3 Dataset

For this assignment, training and development sets come from WNUT corpus, and test data is from an anonymous social media platform (as per homework instruction). Table 1 shows the number of posts, token and types in the training and development sets.

Set	Total Post	Total Token	Total Types
Train	2395	46469	10586
Dev	1001	16261	6255

Table 1: Total posts, token & types

The datasets are not balanced. Training dataset has 5.30% named entity tokens and dev dataset has 6.94% named entity tokens. Class

distribution of NE tokens is shown in Figure 1. Dev dataset has more ratio of unique NEs if compared to the training set. Person, other and location comprise the major portion of the training dataset. Whereas the dev dataset has an equal proportion for five labels except for company and title.

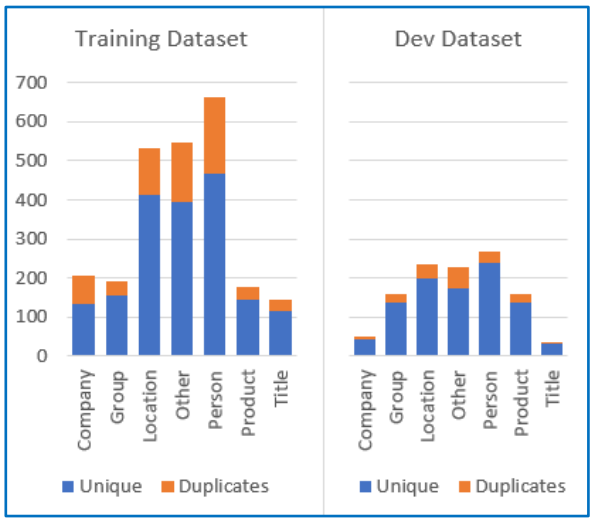

Figure 1: Number of NE in each class for training and dev dataset

4 Methodology

4.1 Preprocessing

I removed the duplicate posts in the training & dev datasets. I also removed the posts if that had any single punctuation character tagged as named entities. I took the decision depending on my models' prediction improvement. I believe I might have dropped named entities like U.S.A if it was split into five words: 'U', '.', 'S', '.', 'A' and '.' is labelled as 'I-location'

4.2 Features Selection

Features can be extracted from a token (word) being labelled and its surrounding context. Many features for the NER task are described in the class lecture (Solorio, 2020) and different previous work. I applied different subsets of the following features on a sequence of words in a sentence: w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2} where the current word in consideration is w_i. Please refer to Appendix A for more detail.

- **Lexical:** Word forms, their lemmas and stems.
- **Synthetic:** Parts of speech (POS) tags.
- **Orthographic:** Word properties with regard to how is it capitalized (initial-caps, all-caps), the kind of characters that form the word (contains-digits, all-digits, alphanumeric, roman-number), the presence of punctuation marks, membership of the word to a predefined pattern (URL).
- **Affixes**: The prefixes and suffixes of the word (up to 4 characters).
- **Capitalization**. A key feature for recognizing named entities is capitalization. However, capitalization in tweets is not a reliable feature for NE recognition. For example, non-entity words can be capitalized to emphasize. On the other hand, the entire tweet can be lowercased (including named entities). "Congratulations Tigers!", "tigers won the match" and "Save the Tiger from Extinction". First two tweets may refer to any sports team with nickname "Tiger". Whereas, the third tweet refers to the animal tiger. Sometimes the all the words of a post is capitalized to express yelling (Willingham, 2018). To address this issue, I tried to check if the entire post is capitalized. I couldn't look into the context to differentiate if a non-entity word capitalized or an entity is not capitalized.
- **Trigger Words**: Triggering properties of window words. An external list is used to determine whether a word may trigger a certain Named Entity (NE) class (e.g., "Mr" or "Dr" may trigger class PER). I collected a list of personal titles from Wikipedia (Wikipedia, List of titles, 2020).
- **Gazetteers**: A simple way to guess whether a particular phrase is a named

entity or not is to look it up in a gazetteer. It determines possible classes for each word. I have generated gazetteer features generated from several lexicons provided as supporting material. However, the gazetteer was had different labels other the one we are trying for. For example, Automotive Brand and Model were two examples gazetteer in the lexicon, but it wasn't clear whether to add these to product, company or maybe other class.

4.3 Model

I have used CRF model form sklearn-crfsuit library. The library implements five training algorithms for CRF:

- Gradient descent using the L-BFGS method (lbfgs)
- Stochastic Gradient Descent with L2 regularization term (l2sgd)
- Averaged Perceptron (ap)
- Passive Aggressive (PA)
- Adaptive Regularization Of Weight Vector (AROW)

I designed three models. I have used Gradient descent using the L-BFGS method (lbfgs) for all my models each having 100 iterations and varying coefficients for L1 & L2 regularization (c1 & c2). All the models have 'all_possible_transitions' set to True and 'all_possible_states' set to False. Features used for each model vary as below:

- Model 1 has all features described in section 4.2 except external sources like gazetteers and trigger-word list (c1=0.36, c2=0.08).
- Model 2 has all the features (c1=0.15, c2=0.2).
- Model 3 excludes lowercase form of the word itself from the features (c1=0.15, c2=0.2).

5 Results

I used seqeval library to measure the performance of my model. Following are the performance metrics of my models:

	Precision	Recall	F1-score
Person	0.475	0.345	0.400
Location	0.560	0.362	0.440
Group	0.333	0.020	0.038
Title	0.000	0.000	0.000
Other	0.281	0.137	0.185
Company	0.636	0.179	0.280
Product	0.333	0.065	0.108
Micro Avg	**0.458**	**0.220**	**0.297**
Macro Avg	**0.422**	**0.220**	**0.273**

Table 2: Performance metrics of Model-1

	Precision	Recall	F1-score
Person	0.400	0.065	0.111
Location	0.613	0.412	0.493
Group	1.000	0.020	0.039
Title	0.302	0.122	0.174
Other	0.576	0.376	0.455
Company	0.636	0.179	0.280
Product	0.200	0.062	0.095
Micro Avg	**0.534**	**0.239**	**0.330**
Macro Avg	**0.582**	**0.239**	**0.302**

Table 3: Performance metrics of Model-2

	Precision	Recall	F1-score
Person	0.400	0.065	0.111
Location	0.605	0.418	0.495
Group	0.750	0.030	0.058
Title	0.315	0.130	0.184
Other	0.559	0.369	0.444
Company	0.636	0.179	0.280
Product	0.200	0.062	0.095
Micro Avg	**0.528**	**0.242**	**0.332**
Macro Avg	**0.538**	**0.242**	**0.305**

Table 4: Performance metrics of Model-3

6 Discussion

I tried different combination of features and changed the window size from 1 to 3. Finally, I found window size 2 was working better for the dev dataset.

The accuracy of the model-1 was more than 93%, but the F1-measure was very poor compared to the provided baseline results. The training and dev datasets were not balanced (section 3). The dev dataset had less than 7% named entities. Any dumb classifier that returned 'O' tag for every token would get a little more than 93% accuracy. As a result. Without external features like gazetteers and list of trigger words, model-1 performs very good for person and location class as there are more person and location classes data in the training set compared to data for other classes.

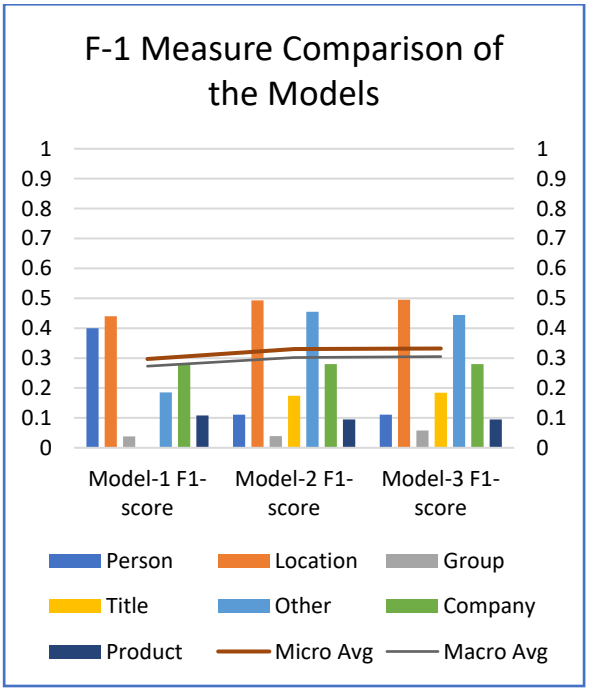

Figure 2: Performance Comparison of different Models

Inclusion of gazetteers and trigger word features in model-2 increased the overall F-1 measure (Figure 2). It reduces the recognition of person class and increases the recognition of title class named entities. I couldn't verify if gazetteers were helpful in generalizing the classifier. For example, "Sunday" is a surname (label: I-person) in one of the gazetteers whereas "Sunday" is also a day of the week (B-other).

After adding gazetteer and trigger words feature, I found the model has learned a few company names like Facebook, Twitter, Youtube, etc (Figure 3) from the lowercase feature of the word. I removed this feature in my third model as I wanted to the model to be more generalized.

Weight?	Feature
+0.307	word.lower():twitter
+0.196	word.lower():facebook
+0.108	word.lower():youtube
+0.107	word.lower():fb
+0.099	word.lower():ione
+0.092	word.lower():msn
+0.090	word.lower():yahoo
+0.090	word.lower():tumblr
+0.083	word.lower():walmart
+0.080	word.lower():ufc
... 83 more positive ...	
-0.034	-1:word.islower()
-0.037	+1:word.islower()

Figure 3: Top features for 'B-company' label the model has learned

Weight?	Feature
+1.181	-1:word.lower():in
+1.176	-2:word.lower():in
+0.861	+2:word.lower():where
+0.736	+1:word.lower():york
+0.720	-1:word.lower():at
... 383 more positive ...	
... 31 more negative ...	
-0.198	-1:word.lower():with
-0.207	+2:word.lower():,
-0.212	+2:word.lower():(
-0.226	+2:word.lower():after
-0.279	-2:word.lower():f

Figure 4: Top features for 'B-company' label the model has learned

Removing the feature lowercase of the word itself didn't change in recognition F-1 measure for 'company' labels. Moreover, there are no significant changes in other classes as well. Rather we got slightly higher f1-measure in model-3.

I tried to figure out what classifier learned. Features learned from location label (Figure 4)

shows that in the pre-window of size to "B-location" labels has "in", "where" and "at". If we look into features with negative weights, we can see that if any token is followed by "with" it's unlikely to be a "B-location".

Here are some more example over other class labels:

Top positive:	
4.201385 O	word.ispunctuation
3.258766 O	BOS
3.046755 O	EOS
2.584555 O	isHashTagUserName(word)
2.270356 B-company	company.gazetteer
2.161867 O	shape(word):xx
1.856841 O	shape(word):xxx
1.854384 B-product	word[:2]:iP
Top negative:	
-0.933433 O	-2:word.lower():happy
-0.933649 B-location	shape(word):Xxx
-0.950819 B-product	-1:word.istitle()
-0.966924 O	word[-2:]:ex
-0.974401 O	word[-2:]:ro
-0.975828 O	word[-2:]:za
-0.977214 B-person	-1:word.istitle()
-0.980358 B-location	-1:word.istitle()
-0.982512 B-company	person.gazetteer

Figure 5: Top positive and top-negative features

"4.201385 O word.ispunctuation" shows that if there's any punctuation mark as a word it must be labeled as 'O'

"3.258766 O BOS, 3.046755 O EOS" indicates that the beginning and ending words of a post are mostly labeled 'O'. This might have happened because I didn't remove '#' and '@' to read the word. Many of tweets begin with a word beginning with @ and ends with a word with hashtags ('#' symbol).

"0.933649 B-location shape (word): Xxx" shows that the model has learned that any place doesn't have a three-character word (Xxx).

These weights are learned by the model from the training set depending on the features we select.

7 Critical Analysis

I this section I will describe some limitations of my feature selection and experiments, which I believe I could look into given more time.

7.1 Features not used

I didn't experiment with the following features:

- Lemmas & stems of words in the window
- Word position in the window (e.g., W(1)="street").
- Chunk tags.
- Single character patterns (lonely initial, punctuation-mark, single-char), or the membership of the word to a predefined class.
- Type pattern of consecutive words in the context. The type of a word is either functional (f), capitalized (C), lowercased (l), punctuation mark (.), quote (') or other (x). For instance, the word type pattern for the phrase "John Smith paid 3 euros" would be CClxl.
- Gazetteer information for other words in the window.
- Emojis were not handled. The training and dev dataset had emojis converted to the text description. However, the test data had emojis as Unicode characters. I assumed any unseen character would be tagged as 'O', seen my model has never seen any single character emojis. I checked the test file after results were published and found that 26 emojis were predicted as 'person', 'location' and 'other' class labels.
- Number generalization. I just considered if the word was numeric or if there were any digits in the word

only. I could also consider masks of numbers, for example, 10-12-1996 → *DD*-*DD*-*DDDD*, 1999 → *DDDD*

- Preprocessed text with a truecaser, which attempts to automatically figure out what the correct capitalization of the text should be.
- Brown clusters
- Word2Vec clusters

7.2 Lack of Experiments

I spent quite an amount of time on gazetteers, which didn't improve the performance of my model. I needed to experiment with other algorithms inside the sklearn-crfsuit library. I tried to improve quality try to select regularization parameters using randomized search and 3-fold cross-validation. However, the results didn't improve.

7.3 Lack of Observing Classifiers

I tried to look into what the model is learning, but I think that wasn't sufficient. I could have looked in much more detail using different tools available and select a better set of features for the given task.

8 Conclusion

In this homework, we studied and experimented the different feature extractions techniques and implementation of models based on Conditional Random Fields (CRFs). We learned how different feature sets contribute to the training of the models that result in different levels of performance measurements (precision, recall and F-1 measures).

9 References

Akbik, A. a. (2018). Contextual String Embeddings for Sequence Labeling. *International Conference on Computational Linguistics*, (pp. 1638--1649).

Alan Ritter, S. C. (2011). Named Entity Recognition in Tweets: An Experimental Study. *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (pp. 1524-1534). Ritter, Alan & Clark, Sam & Mausam, Mausam & Etzioni, Oren.

Carreras, X. &. (2003). A Simple Named Entity Extractor using AdaBoost. *Seventh Conference on Natural Language Learning at HLT-NAACL*, (pp. 152–155).

Guilar, G. &.-M. (2017). A Multi-task Approach for Named Entity Recognition in Social Media Data. *3rd Workshop on Noisy User-generated Text*, (pp. 148-153).

Jain, D. &. (2018). Simple Features for Strong Performance on Named Entity Recognition in Code-Switched Twitter Data. *hird Workshop on Computational Approaches to Linguistic Code-Switching*, (pp. 103-109).

Peters, M. &. (2018). Deep contextualized word representations. . *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

sklearn-crfsuit. (2020). *sklearn-crfsuit*. Retrieved from readthedocs.io: https://sklearn-crfsuite.readthedocs.io/en/latest/index.html

Solorio, T. (2020, Feb 3). Class Lecture on Conditional Random Fields for Named Entity Recognition. COSC 6336 (Fall 2020).

Sutton, C. a. (2012). An Introduction to Conditional Random Fields. *Found. Trends Mach. Learn.*, 267–373.

Tkachenko, M. a. (2019). Named entity recognition: Exploring features. *Conference on Natural Language Processing (KONVENS)*, (pp. 118-127).

Wikipedia. (2020, Feb 4). *List of titles*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/List_of_titles

Wikipedia. (2020, Feb 9). *Named Entity Recognition*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Named-entity_recognition

Willingham, A. (2018, July 23). *CNN*. Retrieved from Why typing in all-caps looks like you're yelling (A brief history): https://www.cnn.com/2018/07/23/us/all-caps-typography-history-tweets-trnd/index.html

A Appendices

Specific Features Used

Given a sequence of words in a sentence: w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2} and the current word in consideration is w_i, I used the following features:

- The lower-case version of w_i
- Prefixes and Suffixes of length 2 of w_i
- Prefixes and Suffixes of length 3 of w_i
- Prefixes and Suffixes of length 4 of w_i
- Length of the word w_i
- If w_i is at the end of the sentence
- If w_i is at the beginning of the sentence
- If w_i is all capitalized
- If w_i is a title (first character capital letter)
- If w_i contains digits only
- If w_i has any digits
- If w_i is alphabetic
- If w_i has '#' or '@' as the first character
- If w_i belongs to trigger word list
- The shape of the word w_i
- If w_i is URL
- If w_i is the Roman numeral
- Lemma of w_i
- The stem of w_i (Portar algorithm)
- The stem of w_i (Lancaster algorithm)
- If w_k belongs to person gazetteer
- If w_k belongs to company gazetteer
- If w_k belongs to group gazetteer
- If w_k belongs to location gazetteer
- If w_k belongs to product gazetteer
- If w_k belongs to 'other' gazetteer

I looked following features considering the entire post

- If the post has all words in upper case
- Use NTLK to get parts of speech (POS) tags for all the words

For each of the words w_k (i=k-1 and i=k+1) in the context window of 1, I used the following features:

- Prefixes and Suffixes of length 2 of w_i
- Prefixes and Suffixes of length 3 of w_i
- Prefixes and Suffixes of length 4 of w_i
- Length of the word w_i
- The shape of the word w_i
- If w_i is URL

For each of the words w_k (i=k-2, i=k-1, i=k+1 and i=k+2) in the context window of 2, I used the following features:

- The lower-case version of w_k
- If w_k is all capitalized
- If w_k is a title (first character capital letter)
- If w_k contains digits only
- If w_k is alphabetic
- POS tag of w_k
- POS Prefix of length 2 of w_k

For the previous two words w_k (w_{i-1} and w_{i-2}), I used the following features:

- If w_k belongs to trigger word list