Entity Extraction from Social Media using Machine Learning Approaches

Sombuddha Choudhury
Jadavpur University, India
sbc.cs73@gmail.com

Somnath Banerjee
Jadavpur University, India
sb.cse.ju@gmail.com

Sudip Kumar Naskar
Jadavpur University, India
sudip.naskar@cse.jdvu.ac.in

Paolo Rosso
UPV, Spain
prosso@dsic.upv.es

Sivaji Bandyopadhyay
Jadavpur University, India
sivaji_cse_ju@yahoo.com

ABSTRACT

In this work, we describe an automatic entity extraction system for social media content in English as part of our participation in the shared task on Entity Extraction from Social Media Text in Indian Languages (ESM-IL) organized by Forum for Information Retrieval Evaluation (FIRE) in 2015. Our method uses simple features such as window of words, capitalization, dictionary word, part of speech tags, hashtag, etc. The performance of the system has been evaluated against the testset released in the FIRE 2015 shared task on ESM-IL. Experimental results show encouraging performance in terms of precision, recall and F-measure.

CCS Concepts

•**Computing methodologies** → **Artificial intelligence;** *Natural language processing;* •**Information systems** → Information extraction;

Keywords

Entity extraction, named entity, social media

1. INTRODUCTION

Named entities refer to specific concepts which are not listed in the grammars or the lexicons. Automatic identification and classification of NEs benefit text processing due to their significant presence in the text documents. Recognition of named entity is a task that seeks to locate and classify NEs in a text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, etc. The NE recognition task has important significance in many NLP applications such as Machine Translation, Question-Answering, Automatic Summarization, Information Extraction, etc. On the other hand, with the advent of smart phones more people are using social media such as twitter, facebook to comment on people, products, services, organizations, goverments, etc. Thus, NE recognition on various social media data such as websites, blogs, tweets, emails, chats, social media posts has gained significance recently [7][1][4][5][3].

2. TASK DESCRIPTION

In this task, one has to identify the named entities (NE) from a collection of raw tweets and tag them with appropriate NE tags.

Input: A collection of tweets $T_1, T_2 \dots T_n$, where each tweet T_i is a set of words $(W_1, W_2, \dots W_m)$ and every word W_j is some standard or non-standard English word. By standard we mean that those words have their presence in an English dictionary. A set of named entity classes $\{C_1, C_2, \dots C_k\}$ were also provided by the task organizsers where each class determines a specific type of named entity like Person, Organization, Location, etc.

Output: For every tweet, the system has to identify which words classify as named entities and then tag them with a suitable named entity class. The format of the input and output is specified in the next section.

3. DATA

In this section, we describe the dataset provided to the shared task participants for the task. We were provided with two sets of data, namely training set and test set. For the training set, we were provided with two different files one of which contained a collection of tweets along with their tweet-id's and user-id's; the other was an annotation file that contained the named entities and their tags for the tweets in the raw file. The annotation file consisted of 6 columns separated by tabs: <Tweet ID User ID NETAG NE Index Length>

For example: Tweet ID:123456789012345678 User Id:1234567890 NETAG:ORGANIZATION NE:SonyTV Index:43 Length:6

The training corpus consists of 5941 tweets and 23483 unique tokens. The different NEs provided in the training annotation file and their corresponding counts are shown in table 1. The testset contains 9595 tweets and 39464 unique tokens.

4. SYSTEM DESCRIPTION

4.1 Pre-processing

For the raw training file, we first separated the tweet text from the user ids as they were redundant. The tweet ids were however preserved as they serve as keys to the tweet text as each tweet has a unique tweet id. In the same way, we removed the presence of all urls and hyperlinks from the tweet text. After this we applied a POS tagger on the new file generated. For POS tagger we used *ark-tweet-nlp-0.3.21*[1] [6] to generate pos tags in CoNLL format. From the annotation file, for each tweet id we get the list of words that

[1]http://www.ark.cs.cmu.edu/TweetNLP/

are named entities and their associated NE tags. We scan every word of every single tweet and assign that word its corresponding named entity tag. We used the BIO type of chunking for this purpose. If a sequence of words belongs to an NE with a particular NE tag, we marked the first word of the entity as NE tag_B(beginning) and the subsequent words as NE tag_I(intermediate). For words that are not NEs, we tag them as O(other). For example, if we have a tweet like *Chief Minister Arvind Kejriwal Wishes Luck to Special Olympics Participants* and the annotation file has entries like "NETAG:PERSON NE:Chief Minister Arvind Kejriwal", then the tagging of the tweet is done as: "Chief PERSON_B Minister PERSON_I Arvind PERSON_I Kejriwal PERSON_I Wishes_O Luck_O to_O Special_O Olympics_O Participants_O". The annotation file has a total of 22 classes/tags. By our format of encoding the total number of tagged classes becomes $2 * 22 + 1 = 45$ classes (2 tags, i.e. _B and _I, for each class and 1 for O). The same tagging format was applied for the test file.

4.2 Classification Features

We have used simple features for classification which are described in the next subsections.

4.2.1 Window of Words

The unique words in the corpus are mapped into integer vector, i.e., each unique word is assigned an integer value. Various work [2][8] on NER employed preceding or following words of the target word to determine its category. Therefore, we also employed a window of words approach which have the size 3. In our work, previous word and next word along with target word are considered to build the window.

4.2.2 Part of Speech (POS) Tag

The POS of the target word and surrounding words may be useful feature for NER. In the context of NER, noun tag is very useful because NEs are always noun phrases. We have used a POS tagger[6] specially developed for social media text.

4.2.3 Capitalization

Although this feature is not that effective for tweets or user generated content in social media, still a fairly large number of entities that are capitalized turn out to be named entities. Thus we included this feature as a binary feature that can be formally defined as:

$$Capitalization(word) = \begin{cases} 1 & \text{if } word \text{ starts with capital} \\ 0 & \text{otherwise} \end{cases}$$

4.2.4 Presence of Numeric Values

This feature is very helpful to identify time expression, measurement and numerical quantities (such as currencies). This could be used as a binary feature that can be defined as:

$$Is\_Numeric(word) = \begin{cases} 1 & \text{if } word \text{ has has numerals} \\ 0 & \text{otherwise} \end{cases}$$

4.2.5 Hashtag

A word having a hash('#') is very useful feature in tweets. Often a word with hash (such as #India, #ATK etc.) denotes a topic or trend in the tweets which turns out to be NEs. Therefore, we included this feature as a binary feature which is defined as:

$$starts\_with\_Hashtag(word) = \begin{cases} 1 & \text{if } word \text{ begins with \#} \\ 0 & \text{otherwise} \end{cases}$$

4.2.6 At the Rate

This is similar to the previous feature and can be defined as :

$$starts\_with\_attherate(word) = \begin{cases} 1 & \text{if } word \text{ begins with @} \\ 0 & \text{otherwise} \end{cases}$$

4.2.7 Dictionary Word

This feature checks whether a given word has its presence in the dictionary or not. We incorporated the English dictionary provided by *PyEnchant*[2] (an open source dictionary available for python). The main motivation behind using this feature is that words that appear in the dictionary have a fairly low probability of qualifying as a named entity. This is again a binary feature that can be described as:

$$is\_in\_Dictionary(word) = \begin{cases} 1 & \text{if } word \text{ in the Eng dict.} \\ 0 & \text{otherwise} \end{cases}$$

4.3 Classifiers

In this work, we have employed in total four different classifiers, namely Naïve Bayes (NB), Conditional Random Field (CRF), Margin Infused Relaxed algorithm (MIRA) and Decision Tree (DT). For Naïve Bayes and Decision Tree, we used the *WEKA toolkit*[3]. For CRF and MIRA we used the open source implementation of *CRF++ toolkit*[4] and *miralium*[5].

4.4 Output Generation

After the classifiers generated the corresponding NE tags, post-processing was done to convert the predicted tagged file into the same format as provided in the training annotation file. This was simply a reverse procedure of what we did for pre-processing of the training file. For every word that was tagged as one of the 45 named entity tags, we entered that word and the corresponding tweet-id, user-id, starting index and length of the entity into the output file. When we had a chunk of words (where a particular _B tag was followed by 1 or more _I tags) we simply clubbed those words together as a single named entity until we reached a word with O tag or B tag or the end of the tweet. For multiple word NEs, we considered the starting index of the first word (with _B tag) as Index entry and the total length of all the words in the NE (including blank spaces) as the length of that NE. Another very important part of the post-processing phase was proper identification of the tagged tweets since in the tagged file obtained from the classifiers there is no way to identify which tweet belongs to which tag. Again for this we maintained a line–tweet correspondence where the starting word of a tweet had a one-one correspondence to the line number in which it appeared in the file obtained from the classifier.

[2]https://pythonhosted.org/pyenchant/
[3]http://www.cs.waikato.ac.nz/ml/weka/
[4]https://taku910.github.io/crfpp/
[5]https://code.google.com/p/miralium/

5. EXPERIMENT

This section basically emphasizes on exemplifying the systematic steps performed in generating the training models using the four different classifiers as mentioned in Section 4.3 and then identifying the NEs and their corresponding NE tags in the given test file using the trained models generated from the training files.

5.1 Training the Classifiers

We performed pre-processing on the two training files provided to us and the detailed description of the pre-processing is discussed in Section 4.1. We have prepared four models with all of the features (discussed in Section 4.2) using the four classifiers, i.e., NB, DT, CRF and MIRA.

The descriptions of the models follow:

Model 1: Generated using the CRF classifier.

Model 2: Generated using the MIRA Classifier.

Model 3: Generated using the J-48 Classifier.

Model 4: Generated using the Naïve Bayes Classifier.

The NE tags together with their frequency of occurrence in the training data are shown in table 1.

Table 1: NE tags statictics in training file

NE Tag	Count
ENTERTAINMENT	486
LIVTHINGS	14
ARTIFACT	96
FACILITIES	124
MATERIALS	40
DATE	78
PLANTS	1
DAY	111
LOCATION	2806
ORGANIZATION	2470
LOCOMOTIVE	521
SDAY	7
COUNT	572
PERIOD	209
DISEASE	27
YEAR	111
DISTANCE	18
MONEY	95
MONTH	23
PERSON	3145
TIME	30
QUANTITY	19

5.2 Testing

Again the test file was made to undergo the same set of operations as the training phase where the raw test file was converted into a format suitable to be evaluated by the models generated. These set of operations were the pre-processing steps and the feature extraction steps. Then we ran our test file using each of the 4 models and generated 4 test runs where test run1 was generated using model1 (CRF), test run2 using model2 (MIRA), test run3 using model3 (J-48) and test run4 using model4 (Naïve Bayes). Finally output format preparation steps as mentioned in Section 4.4 were performed for each of the output test runs and converted into formats that are similar to the one specified in the training annotation file.

6. RESULTS

We have submitted four different runs using the approaches discussed in the previous section. In this section, we discuss about the performance of each of our submitted runs and our overall performance in comparison to the other participating teams. Standard Precision, Recall and F-Measure parameters were used for evaluation. The values of these metrics for the different runs that we submitted are shown in Table 2.

Table 2: Evaluation of the submitted runs

	Precision	Recall	F-Measure
Run 1	46.92	32.41	38.34
Run 2	58.09	31.85	41.15
Run 3	49.10	31.59	38.45
Run 4	46.50	30.20	36.61

In run1, run2, run3 and run4 the correctly detected and classified named entities are 11771, 8901, 11016 and 11122 respectively. We obtained a best F-measure of 41.15 for run2 using MIRA classifier which ranked third among all the runs submitted by the participating teams. CRF (run1) and J-48 (run3) perform almost at par while Naïve Bayes (run4) performs the worst among the four classifiers.

The training sample had a lot of words that were non-NEes and thus that somehow may affected the detection of NEes in the test set as the entire training was done on the training set. Some classes of NEes like Plants, Sday(Special Day), Distance etc. had a very less number of words tagged into them and thus their appearance in the classified outputs were also fairly low. J-48 classifier worked quite in detection of NEes but was unable to properly identify the appropriate NE tags for an entity. Similar was the case with Naïve Bayes for example *Tata's* in *Tata's narrow cars ...* was wrongly classified as Person instead of Organization and *iPhone* in *I really want the iPhone 6s Rose Gold* was misclassified as Person instead of Artifact. This was mainly caused due to the large disparity in number of the different NEes in the training set and lack of proper features for proper fine-graining. We also avoided the use of gazetteer lists in this task which might otherwise have helped us in detection of some special kinds of NEes. Another drawback of some of our systems was that when NEes involving more than one words were present, classifiers like J-48 and Naïve Bayes skipped some part of that entity for example if a named entity like *Mr. Narendra Modi* was present in some tweet text, these classifiers, in some instances, classified only *Mr. Narendra* as a Person. This error was less in case of CRF or MIRA as these classifiers are more suitable for sequence labelling operations. There were some other instances when a non-NE was misclassified as a NE.

7. CONCLUSIONS

In this paper, we have presented a brief overview of our machine learning based systems to address the automatic NE identification problem on social media. We have observed that the MIRA based approach provides better results than the systems which are developed using CRF, DT, NB classifiers. For our participation in ESM-IL subtask, we

have submitted four runs and the obtained results confirm that the overall accuracy of Run2 is more than almost 3% higher when compared to other runs, i.e. Run1. Run3 and Run4.

As future work, we would like to explore more sophisticated features to handle NE tags and apply post-processing heuristics to improve the performance of system. We also plan to incorporate more language specific feature in our future work to improve the accuracy of the system.

8. ACKNOWLEDGMENTS

We acknowledge the support of the Department of Electronics and Information Technology (DeitY), Government of India, through the project "CLIA System Phase III".

The research work of the second last author was carried out in the framework of WIQ-EI IRSES (Grant No. 269180) within the FP 7 Marie Curie, DIANA-APPLICATIONS (TIN2012-38603-C02-01) projects and the VLC/CAMPUS Microcluster on Multimodal Interaction in Intelligent Systems.

9. REFERENCES

[1] S. Ashwini and J. D. Choi. Targetable named entity recognition in social media. *arXiv preprint arXiv:1408.0782*, 2014.

[2] S. Banerjee, S. Naskar, and S. Bandyopadhyay. Bengali named entity recognition using margin infused relaxed algorithm. *Text, Speech and Dialogue*, pages 125–132, 2014.

[3] N. Dewdney. Named entity trends originating from social media. In *Workshop on Information Extraction and Entity Analytics on Social Media Data*, pages 1–16. COLING 2012, 2012.

[4] L. D. et al. Analysis of named entity recognition and linking for tweets. In *Inf. Process. Manage*, pages 32–49, 2015.

[5] W. Murnane. Improving accuracy of named entity recognition on social media data. *Master Thesis, University of Maryland*, 2010.

[6] O. Owoputi, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *NAACL*, 2013.

[7] A. Ritter, S. Clark, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1524–1534, 2011.

[8] S. K. Saha, S. Chatterji, S. Dantapat, S. Sarkar, and P. Mitra. A hybrid approach for named entity recognition in indian languages. In *NERSSEAL-IJCNLP-08*, pages 17–24, 2008.