

NLP APPLICATIONS

NER DETECTION IN CODE-MIXED DATA

SEMESTER PROJECT

MENTOR

Ponnurangam Kumaraguru

TEAM MEMBERS

Sourav Kumar, 20161072

Akshat Maheshwari, 20161024

TABLE OF CONTENTS

TABLE OF CONTENTS	1
ABSTRACT	2
Why NER Required for Code-Mixed DATA	2
INTRODUCTION	3
CURRENT STATE OF ART	4
General NER Detection	4
NER For Code-Mixed Data	5
DATASET	6
APPROACH	7
Model Pipeline	8
Features	8
EXPERIMENTAL SETTINGS	10
Why shifted to Neural Network from Statistical approach?	11
RESULTS	11
Statistical Approaches	11
Neural Network	14
ANALYSIS	15



ABSTRACT

People tweet more than 100 Million times daily, yielding a noisy, informal, but sometimes informative corpus messages which does not belong to one language that further mirrors the zeitgeist in an unprecedented manner. The performance of standard NLP tools is severely degraded on tweets. Named Entity Recognition (NER) is a major task in the field of Natural Language Processing (NLP), and also is a subtask of Information Extraction. In This project we have addressed this issue by re-building the NLP pipeline by extracting the sufficient features which do not require explicit language modelling. The challenge of NER for tweets lies in the insufficient information available in a tweet.

Why NER Required for Code-Mixed DATA

While growing code-mixed content on Online Social Networks (OSNs) provides a fertile ground for studying various aspects of code-mixing, the lack of automated text analysis tools render such studies challenging. To meet this challenge, a family of tools for analyzing code-mixed data such as language identifiers, parts-of-speech (POS) taggers, chunkers have been developed.



INTRODUCTION

Multilingual speakers often switch back and forth between languages when speaking or writing, mostly in informal settings. This language interchange involves complex grammar, and the terms “code-switching” and “code-mixing” are used to describe it.

Code-mixing refers to the use of linguistic units from different languages in a single utterance or sentence, whereas code-switching refers to the co-occurrence of speech extracts belonging to two different grammatical systems.

These are frequently seen in multilingual communities and is of interest to linguists due to its complex relationship with societal factors.

Some of the issues that we face in social media data are:

- The shortness of micro-blogs makes them hard to interpret. Consequently, ambiguity is a major problem since semantic annotation methods cannot easily make use of co-reference information.
- Micro-texts exhibit much more language variation, tend to be less grammatical than longer posts, contain unorthodox capitalization, and make frequent use of emoticons, abbreviations and hashtags, which can form an important part of the meaning.

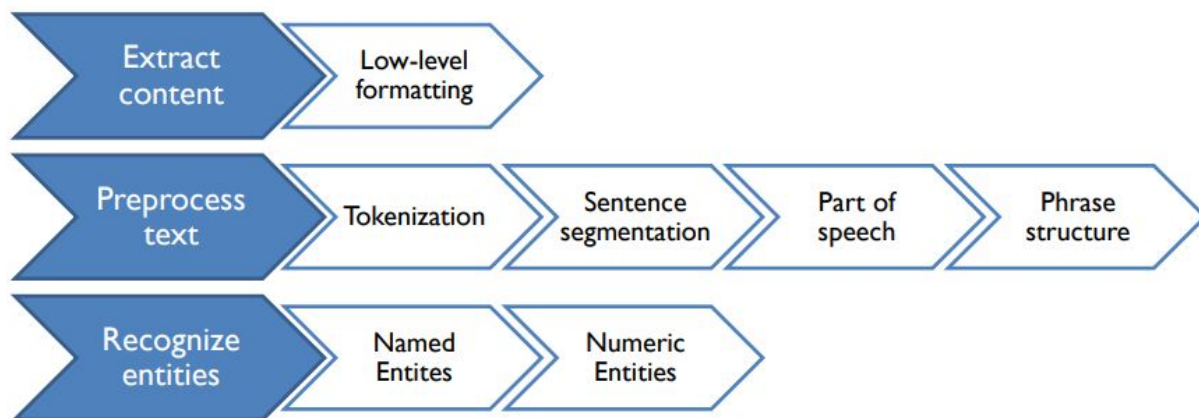
CURRENT STATE OF ART

Currently, there have been some researches related to language identification of code-mixed data. There also have been some researches on parts-of-speech (POS) tagging and transliteration.

General NER Detection

Background: Recently, deep learning-based approaches have been applied to social media named entity recognition and showed promising results. However, as deep learning approaches need an abundant amount of training data, a lack of data can hinder performance. Monolingual datasets are readily available but there is a scarcity for code-mixed resource.

Basic pipeline for Monolingual NER Detection



NER For Code-Mixed Data

In Named Entity Recognition there has been significant research done so far in English and other resource rich languages, but same cannot be said for code-mixed text due to lack of structured resources in this domain.

[One of the papers](#) has the following pipeline for the NER detection:

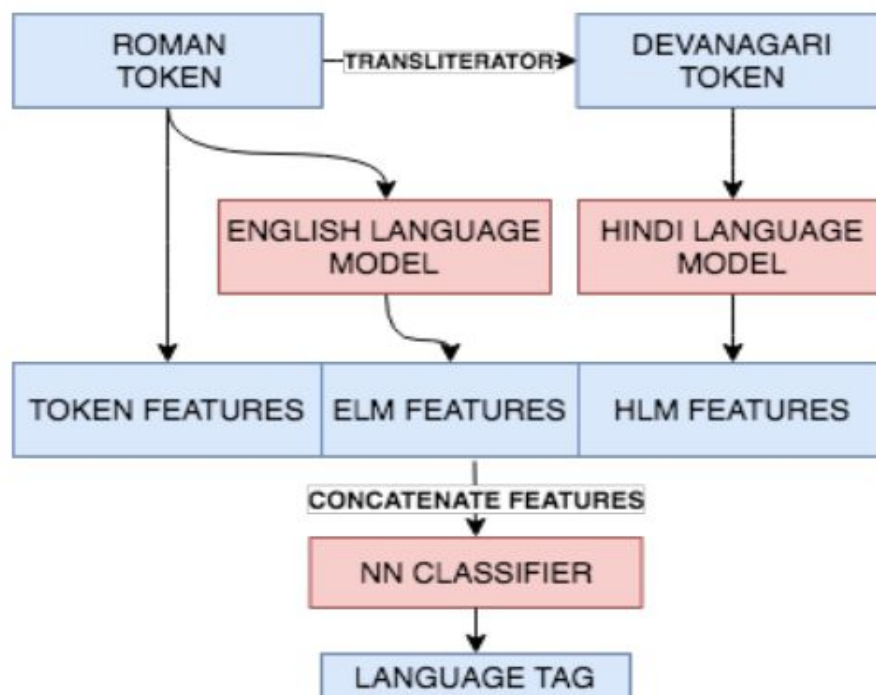



Figure 1: Different steps of our language identification pipeline. We extract features using language models trained on monolingual corpora, and train a classifier based on these features.



In this approach, the researchers have initially created 2 different language models, the English Language Model (ELM) and the Hindi Language Model (HLM) using transliteration. After the separation of the language models, they have done NER detection using feature extraction and application of Neural Networks on those extracted features for each language model.

The features that the researchers have taken into account are:


- Token based features
- Affixes
- Character based features
- Language based features
- Syntactic features
- Tweet capitalization features

In the neural network parts, they have used bidirectional RNN layers using LSTM cells and ReLU activation.

DATASET

For this project, we have used the Twitter dataset for NER detection. General approach we tried was to get the tweets using the Twitter API '**twitterscraper**' for scraping tweets from Twitter using certain key-words. After that, the raw data obtained had to be manually annotated, in which each word of every tweet had to be classified among the 3 categories, 'Person', 'Organization' or 'Location', which was a very tiresome and time-consuming process.

So, we used the already annotated Twitter data from the paper: [Link](#)



Tag	Count of Tokens
B-Loc	762
B-Org	1,432
B-Per	2,138
I-Loc	31
I-Org	90
I-Per	554
Total NE tokens	5,007

The above table shows tags and their counts in the corpus

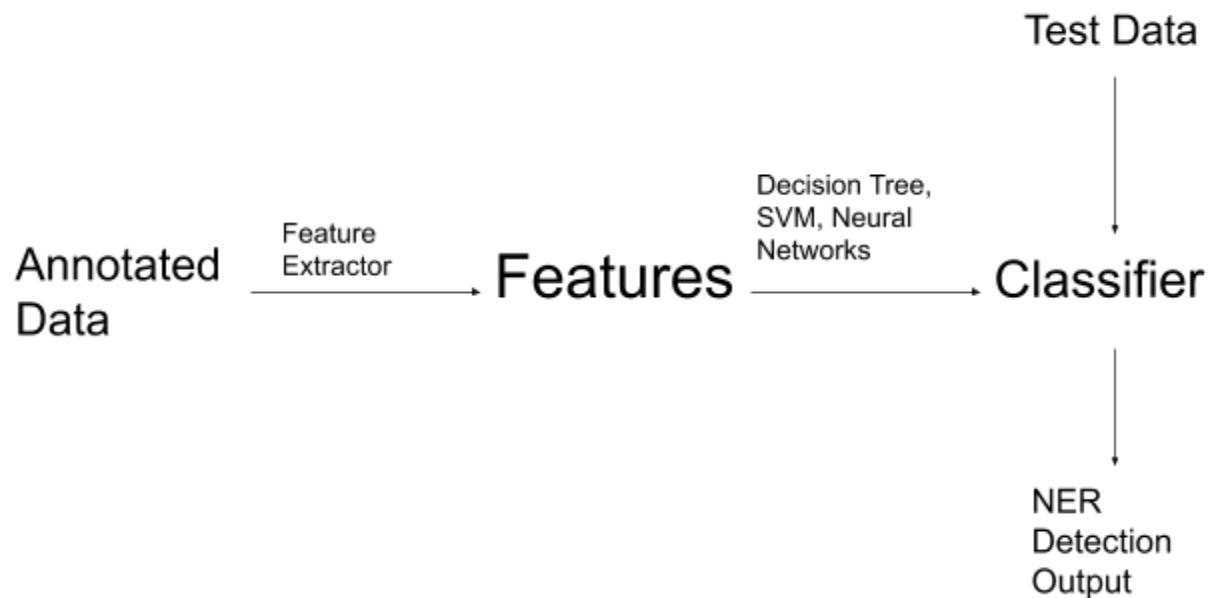
APPROACH

In our procedure, we are following something similar to the above paper only, but the main difference we are creating is that, we are **not separating the data into language models**. We are using more features, and those features themselves are responsible for some type of language differentiation instead of explicitly language model separation.

We found this method while browsing through the internet that instead of explicitly modelling different language models, we can extract some specific features so that we don't need that language modelling and our purpose is fulfilled with a better accuracy and efficiency.

After that, we are using classifier, in which we have tested Decision Tree, SVM and Neural Networks. We are mainly focusing on SVMs and Neural Networks for our model.

Model Pipeline



Features


The features we have taken into account consists of word, character and lexical level information like character-level N-Grams of Gram size 2 or 3 for suffixes, patterns for punctuation, emoticons, numbers, numbers inside strings, social media specific characters like '#', '@', and also previous tag information.

The features taken into account are:

- **Character N-Grams** : These are language independent and have proved to be very efficient for classifying text. These are also useful in situations when the text suffers from misspellings, which is a very common and basic problem in code-mixed data. Group of characters help in capturing the semantic meaning, especially in the code-mixed language where there is an

informal use of words, which vary significantly from the standard Hindi and English words.

- **Word N-Grams** : Bag of word features have been widely used in for NER tasks in languages other than English. Thus, we use N-Grams, where we used the previous and the next word as a feature vector to train our model. These are also called contextual features.
- **Capitalization** : It is a very general trend of writing any language in Roman script that people write the names of person, place or a things starting with capital letter or for aggression on someone/something use the capitalization of the entire entity name. This will make for two binary feature vectors one for starting with capital and other for the entire word capitalized.
- **Mentions and Hashtags** : It is observed that in twitter users generally tend to address other people or organization with their user names which starts with '@' and to emphasize on something or to make something notable they use '#' before that word. Hence presence of these two gives a good probability for the word being a named entity.
- **Numbers in String** : In social media content, users often express legitimate vocabulary words in alphanumeric form for saving typing effort, to shorten message length, or to express their style. Examples include abbreviated words like gr8' ('great'), b4' ('before'), etc. We observed by analyzing the corpus that alphanumeric words generally are not NEs. Therefore, this feature serves as a good indicator to recognize negative examples.
- **Previous Word Tags** : As mentioned in word N-Gram feature the context helps in deciding the tag for the current word, hence the previous tag will help in learning the tag of current word and all the I-Tags always come after the B-Tags.
- **Common Symbols** : It is observed that currency symbols as well as brackets like '(', '[', etc. symbols in general are followed by numbers or some mention



not of importance. Hence are a good indicator for the words following or before to not being an NE.

The tags that have to be used for NER Detection are:

- **B-Tags** : Beginning of a tag
- **I-Tag** : Intermediate of a tag

The classical entity types that we will be using:

- Person
- Location
- Organization

Now we will train our Classifier models on these features.

EXPERIMENTAL SETTINGS

We have experimented various classifiers for NER Detection namely,

Statistical Approach

- Decision Tree Classifier
- Naive Bayes Classifier
- SVM Classifier
- Random Forest Classifier
- CRF (Context random fields) Classifier

Neural Network Approach

- LSTM Classifier
- Bidirectional LSTM Classifier
- GRU Classifier
- Bidirectional Classifier

Why shifted to Neural Network from Statistical approach?

In a sentence there can be single word that can be repeated multiple times and has different named entities based on the context of different location. Since LSTM and GRU can handle previous memory/context in account so based on the context we can identify the correct named entity of that word at that position.

Let us consider an example,

{Aditya Birla} {started the} {Aditya Birla Group}.

In this, the First Phrase is the name of a person (Aditya Birla) whereas the last phrase is the name of a company (Aditya Birla Group). So here is the case where our general statistical approach can fail hence we are shifting to the use of RNN's.

RESULTS

Statistical Approaches

- Decision Tree Classifier

Tag	Precision	Recall	F1-Score
B-Loc	0.61	0.62	0.62
B-Org	0.67	0.65	0.66
B-Per	0.97	0.97	0.97
I-Loc	0.38	0.39	0.39
I-Org	0.42	0.50	0.45
I-Per	0.20	0.22	0.21
Other	0.51	0.52	0.52
Avg / Total	0.94	0.94	0.94

- Naive Bayes Classifier

Tag	Precision	Recall	F1-Score
B-Loc	0.16	0.19	0.17
B-Org	0.28	.48	0.35
B-Per	0.97	0.79	0.87
I-Loc	0.07	0.24	0.11
I-Org	0.00	0.00	0.00
I-Per	0.01	0.30	0.01
Other	0.07	0.35	0.11
Avg / Total	0.91	0.75	0.82

- SVM Classifier

Tag	Precision	Recall	F1-Score
B-Loc	0.16	0.19	0.17
B-Org	0.28	.48	0.35
B-Per	0.97	0.79	0.87
I-Loc	0.07	0.24	0.11
I-Org	0.00	0.00	0.00
I-Per	0.01	0.30	0.01
Other	0.07	0.35	0.11
Avg / Total	0.91	0.75	0.82

- Random Forest Classifier

Tag	Precision	Recall	F1-Score
B-Loc	0.90	0.26	0.41
B-Org	0.81	0.34	0.48
B-Per	0.94	1.00	0.97
I-Loc	1.00	0.01	0.01
I-Org	0.00	0.00	0.00
I-Per	0.00	0.00	0.00
Other	1.00	0.00	0.01
Avg / Total	0.93	0.94	0.92

- CRF Classifier

Tag	Precision	Recall	F1-Score
B-Loc	0.00	0.00	0.00
B-Org	0.00	0.00	0.00
B-Per	0.03	0.42	0.05
I-Loc	0.00	0.00	0.00
I-Org	0.00	0.00	0.00
I-Per	0.00	0.00	0.00
Other	0.91	0.50	0.65
Avg / Total	0.85	0.48	0.60

Neural Network

In this approach we have first separated the data into train and validation sets, and calculated the accuracy and loss instead of the Precision, Recall and F1-Score as we did in the statistical models.

- LSTM Model
 - **Loss:** Categorical Cross Entropy
 - **Optimizer:** Adam
 - **No. of Epochs:** 5
 - **Train Loss:** 0.3018
 - **Train Accuracy:** 0.9224
 - **Validation Loss:** 0.2719
 - **Validation Accuracy:** 0.9394

- Bidirectional LSTM Model
 - **Loss:** Categorical Cross Entropy
 - **Optimizer:** Adam
 - **No. of Epochs:** 5
 - **Train Loss:** 0.3029
 - **Train Accuracy:** 0.9226
 - **Validation Loss:** 0.2768
 - **Validation Accuracy:** 0.9392

- GRU Model
 - **Loss:** Categorical Cross Entropy
 - **Optimizer:** Adam
 - **No. of Epochs:** 5
 - **Train Loss:** 0.2941
 - **Train Accuracy:** 0.9246
 - **Validation Loss:** 0.2845
 - **Validation Accuracy:** 0.9324

- Bidirectional GRU Model
 - **Loss:** Categorical Cross Entropy
 - **Optimizer:** Adam
 - **No. of Epochs:** 5
 - **Train Loss:** 0.2958
 - **Train Accuracy:** 0.9241
 - **Validation Loss:** 0.2825
 - **Validation Accuracy:** 0.9153

ANALYSIS

- For Statistical models, Decision Tree performs best with F1-Score of 0.94.
- In Neural Network, the bidirectional models give less accuracy and the normal LSTM and GRU models given almost comparable accuracies.