CS 445 Natural Language Processing

Project 2: Named Entity Recognition

Due Date: 15 June 23:55

In this project, you are expected to develop a named entity recognition system for English news articles. In the first part, you will implement a gazetteer with regex and rules you created from Wikipedia pages.

In the second part, you will experiment with:

- *Conditional Random Fields (CRF)*
- *Recurrent Neural Networks (RNN)*

You will be given a Colab notebook and you are supposed to create a gazetteer and implement the above models for named entity recognition with the shared dataset. You will experiment with different features (such as POS tag, the start of a sentence, and so on) with the gazetteer you created for these models. You will write your findings, results, and interpretations at the end of the notebook.

You can find the notebook here:
https://colab.research.google.com/drive/1SvB7ryf2vgOfenMP1eUGuhYMgbJ5zqtN?usp=sharing

Please make a copy of this notebook and work on your own copy.

Dataset

The dataset provided to you is a collection of news articles from the Reuters Corpus. You can find the dataset here. More detail can be found in CoNLL 2003. The named entities in these articles are labeled based on the following named entity types:

- *Person*
- *Location*
- *Organization*
- *Miscellaneous*: named entities that are not Person, Location, or Organization.

You are provided with a train, validation, and test dataset. The datasets are shared in IOB format, i.e., the label of the first token of an entity starts with **B-** and is followed by the corresponding named entity type. The remaining tokens of the same entity start with I-

followed by the corresponding named entity type. If the named entity consists of only one token, it is labeled as **B-** and its named entity type. For example;

Elon	B-Person
Musk	I-PERSON
bought	O
Twitter	B-Corporation
.	O

The datasets are shared in the conll format, i.ie, data files contain four columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence. The first item on each line is a word, the second a part-of-speech (POS) tag, the third a syntactic chunk tag, and the fourth the named entity tag.

Implementation Part

In this project, you are expected to use Google Colab and you will do all your implementations on '***Project 2 Notebook.ipynb***' file and submit that file with the expected outputs. The notebook will contain relevant cells for gazetteer creation and token-level classification (CRF and RNN).

In creating a gazetteer, you will do the followings:

- You are given 20.000 Wikipedia pages to create your gazetteer. After loading the Wikipedia pages given in here, you are expected to extract interwiki links which are links to the Wikipedia pages of the phrase inside the URL link in HTML format. Since a phrase has its own web page indicates that it is most likely a named entity, we expect you to find these phrases on web pages with regex and create your gazetteer with these phrases. These interwiki links are founded in the *text* part of the pages as follows:

  Atatürk

  In this example, you can see that there are two different parts to texts. The text given on the right, *Atatürk,* is the anchor text. The text given for *href* section is the title of the page where the anchor text is referring.

- For example, the above sample is a link to the Mustafa Kemal Atatürk page on Wikipedia. However, the anchor text is *Atatürk* on the given Wikipedia page. In the end, you should extract the *Atatürk* and add it to your gazetteer. Additionally, you should add titles of all pages to your gazetteer as well. In the end, do not forget to keep only unique phrases in your gazetteer.
- However, sometimes non-special nouns have their own web pages like "*computer"* or "*book"*. To build a better gazetteer, you can develop rules to eliminate these nouns, such as checking if words start with a capital letter.

- After creating your gazetteer, you must display the size of your gazetteer.

In the second part, you will do the followings:

1. **Conditional Random Fields (CRF)**
    - Features:
      In the implementation of the CRF model, you will experiment with several features. Your experiments must contain at least the features listed below:
        1. Stem
        2. POS tag
        3. Chunk tag
        4. Start of the Sentence
        5. End of the Sentence
        6. Starts with an uppercase letter
        7. wi's shape
        8. wi's short word shape
        9. wi contains a number
        10. *wi* contains a hyphen
        11. *wi* is upper case and has a digit and a dash
        12. *wi* contains a particular prefix (from all prefixes of length 4)
        13. *wi* contains a particular suffix (from all suffixes of length 4)
        14. *wi* is all uppercase
        15. Whether a stopword (you can use the **nltk** library)
        16. Neighboring words
        17. Short word shape of neighboring words
        18. Word shape of neighboring words
        19. Presence of *wi* in **your gazetteer**

      You must apply all these features for each word in your dataset.

    - Model:
      You will use the **sklearn-crfsuite** library to implement the CRF model. Starting with the stem, train a CRF model. Then add features one by one and train a new model with each feature that you add. Calculate precision, recall, and F1 score for each model on the validation dataset and display your results in a table like the below:

	Features	Precision	Recall	F1 Score
0	Stem	-	-	-
1	+POS	-	-	-
2	+Chunk	-	-	-
3	+BOS	-	-	-
4	+EOS	-	-	-
5	+isUpper	-	-	-
6	+shape	-	-	-
7	...	-	-	-

Finally, after you find the best set of features, evaluate your best model with the best features on the test set. Please calculate the precision, recall, F1 score, and classification report.

2. Recurrent Neural Networks (RNN):

- Preprocessing:
  First, you need to map each unique label to a number (from 0 to the length of unique labels). E.g., B-Location: 0, I-Location: 1, …
  You need to pad your *labels* as well as your sentences. You should pad your label with *Other* label. You should be careful about eliminating the padding tokens when you will evaluate your model. You need to ignore/remove these tokens in the evaluation phase.

- Model:
  You can use the **Keras** library to implement a Recurrent Neural Network (RNN). You will try different hyper-parameters for word embeddings and the RNN algorithm. The base model should consist of an embedding layer followed by at least an RNN and a Dense layer wrapped in a TimeDistributed layer (since we are working on token-level).
  For the embeddings, you should at least try the following three embedding strategies;
  - Randomly initialized word embeddings
  - Word embeddings trained from scratch with gensim
  - Pretrained word embeddings from gensim.api (you can pick one)

  For the RNN layer, you should try at least RNN and LSTM layers with two different hidden layer sizes. For the Dense layer, you should try at least two different hidden layer sizes. All other details are dependent on your choice. Additional to the requirements mentioned, you can try different architectures and embedding strategies. There are much fewer details on the implementation of this part in

the notebook. You can find quite different ways of implementing the model. Please do not forget to add a reference to the pages where you have taken a piece of code, etc. Please report all your findings.

- *Evaluation*:
  You are going to report the F1, Precision, and Recall scores from the seqeval library. You will also print the confusion matrix from the same library. You cannot use the sklearn F1 score function since it calculates at the token level but you need to report at the *entity level*. Also, you should not include the padding token in the evaluation of the model. You will use these to discuss and compare the approaches in your report. Do not just state the obvious please elaborate on the results based on what you have learned in the class.

Make sure that your '***Project 2 Notebook.ipynb***' is well commented. After running all the cells, export the *.ipynb, .py, html* output of your notebook, and put a link to your Colab notebook in a txt file, upload that as well. In the end, you should submit 4 files.

You can use the popular/standard python packages. In case you are not sure of a particular library, please ask the instructor or TAs.

You are expected to implement this project on your own. Your scripts will be analyzed by using state-of-the-art tools for any type of plagiarism.

Report:

In your report, you are going to summarize your approach and your findings. Discuss what is working and what is not. You will write your findings at the end of the Colab, section named "My Report".

Your notebook will be evaluated with state-of-the-art anti-cheating programs.

All files should be under the same directory (named as your student ID, only the 5 digit numbers), which you will zip and submit.

Submission Instructions:

- You will submit this project via SUCourse.
- Please check the slides for the late submission policy.
- You can resubmit your project (until the deadline) if you need to.
- Please read this document again before submitting your solution.
- After submitting, you should download your submission to a different path to double-check whether everything is in order.
- Please do your assignment individually, do not copy from a friend or the Internet. Plagiarized assignments will receive -100.