Dwipam Katariya, Sanjana Agarwal and Krish Mahajan

REPORT:

This report engulfs the approach we have used for Assignment 3.
We used two novels for the training and test data, they are,

1) **DEVELOPMENT TEXT:** The Best American Human Short Stories
2) **TEST TEXT:** Jungle Book - Rudyard Kipling

The process we used to extract the relations of the form speak(X, Y) is as follows:

The annotators that we used from the Stanford CoreNLP module are: tokenize, parse, ner, pos, ssplit, dcoref, depparse, and lemma.
Certain annotators that we have used have a dependency on other annotators.

Annotators	Dependency
ner	tokenize, ssplit, pos, lemma
depparse	tokenize, ssplit, pos
dcoref	tokenize, ssplit, pos, lemma, ner, parse
lemma	tokenize, ssplit, pos

Because, most of the annotators that we need have a dependency on tokenize, ssplit and pos, we need to use those.
However, there are other reasons to use the annotators like ner, depparse, dcoref and lemma.

Why ner? We know that the Named Entity Recognition is one of the most important annotators for identifying name entity relationships in language processing. The ner annotator, basically gives the <PERSON> tag for the name of a person if it exists in their corpus of words. However, words like FRENCHMAN cannot be identified as a <PERSON>, hence we used <MISC> to identify such cases .

Why lemma? Lemmatization is the process where we determine the vocabulary or the morphological meaning of the word. Thus, it is important for words in different tenses, or the words that come under the same lemma, to be characterized as such. So, to identify sentences where conversation took place, root words like say, tell etc. can be the key identifiers. But say can be said, and tell can told. But lemmatization solves this problem by stemming the words with heuristic approach.

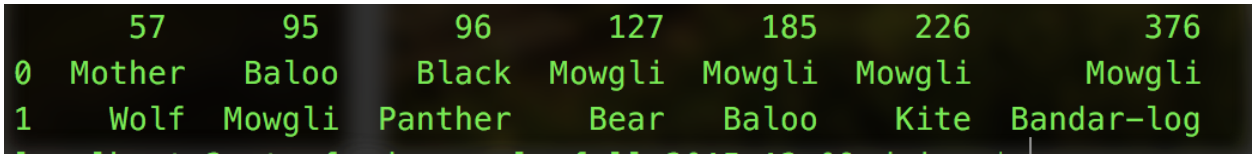
Why depparse? Dependency parsing was required because we needed to find the dependency or the grammatical relations amongst various words of a sentence. Thus, if X says something to Y, we require a certain dependency to be built between X and Y.

Why dcoref? We used the Deterministic Coref Annotator for the basic relation to be shown between a pronoun and its corresponding proper noun. Thus, if X says something to Y, and Y is mentioned again in the dialogue, but with a pronoun like he/she, we require that the pronoun be mapped to Y.

Based on manually viewing some files, we came across a few observations.

- If <Speaker> tag == 'PER*' in any of the tokens, then we have to consider those tokens for classification.
- Second, if the <NER> tag is 'PERSON' or 'MISC', the word is the name of a person. Or words like Oldman, Frenchman.
- In case of pronouns for the proper nouns, we have used the pos tag information, where if a tag has a PRP(Personal Pronoun), we determine the that if character's name is unavailable, then PRP like "He", "She" can be useful. Because this story starts with "*you*" as the conversation.

Thus, the frequencies that we got based on the conversations are on Test data:



	57	95	96	127	185	226	376
0	Mother	Baloo	Black	Mowgli	Mowgli	Mowgli	Mowgli
1	Wolf	Mowgli	Panther	Bear	Baloo	Kite	Bandar-log

Speaker X	Speaker Y	Frequency
Mother	Wolf	1
Black	Panther	1
Baloo	Mowgli	3
Mowgli	Bear	1
Mowgli	Kite	1

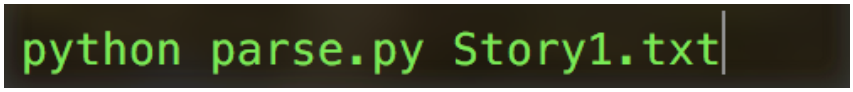
If we had more time to make the system better, we would:

- Check the dependencies between PRP words and Named entity words, that previously occurred or can occur.
- Sometimes, PRP words are used in the next sentence having no relation to previous sentence, we can check for such words that occur before the conversation happened and eventually replace them the the named entity.

- Using open information extraction we can detect the relationship between Name entity person and PRP words.
- We would also like to use Quote annotator, that would easily give us the sentences where conversation has happened
- There are names like “Shere Khan”, where it considers two different names. We can optimize this to one single name.

How to run the program (“parse.py”)
Pre-req : xmltodict, pandas
Place the file in the directory “stanford-corenlp-full-2015-12-09” (StanfordNLP folder that we download from the StanfordNLP website)
Execute:

Python parse.py <file_name.txt>
Eg: to run the program on Story1.txt



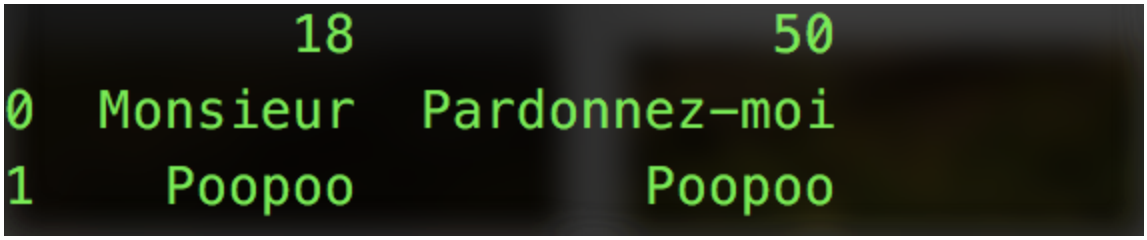
Output from the program on Train data :

	1	9	11	18	19	21	31	\
0	FRENCHMAN	them	Monsieur	Monsieur	Monsieur	them	Monsieur	
1	MORRIS	us	Poopoo	Poopoo	Poopoo	Monsieur	Poopoo	

	36	43	48	50	55	56	58	\
0	Monsieur	Monsieur	Monsieur	Pardonnez-moi	Monsieur	Poopoo	Poopoo	
1	Poopoo	Poopoo	Poopoo	Poopoo	Poopoo	it	Delmonico	

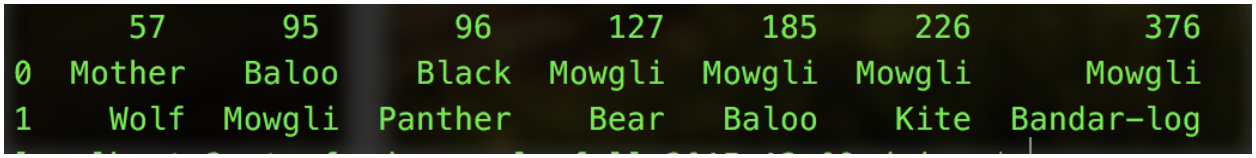
	63	79	82	96	122
0	Frenchman	Poopoo	Monsieur	Sare	Poor
1	he	he	you	I	Poopoo

Where row_names are the sentence number and column (0,1) is the (x,y), hence it can be interpreted as for sentence 1 Speak(Frenchman, Morris)
Now after using rule of quote for conversation and removing PRP words, we tested on the train again and here is the output:



	18	50
0	Monsieur	Pardonnez-moi
1	Poopoo	Poopoo

Output seems to remove PRP words, and sentences that does not have conversations.
After testing on the Test data:



	57	95	96	127	185	226	376
0	Mother	Baloo	Black	Mowgli	Mowgli	Mowgli	Mowgli
1	Wolf	Mowgli	Panther	Bear	Baloo	Kite	Bandar-log

Mother - Wolf -

We are submitting:
XML and .txt

This output seems to be reasonable, as there are less conversations happened in test data than train data.
We have sampled train data from a single Novel
files for Train(Story1.txt) and Test data(Story2.txt)

References:
https://cs.nyu.edu/grishman/jet/guide/PennPOS.html