I first implemented everything on Java, then implemented them on python using IBM Watson, which is much more user friendly on python than on Java. Even though I don't have time to implement tf-idf again on python, I'm certain that the time for this HW could be 1/4 if I implemented everything on Python. Also, for word2vec, the java tutorial is almost impossible to understand.

For both Java and python, don't move the directory of the DataSet under each folder.

## 1)Java

There are 4 functions I used (I commented the code, so it should be fairly easy to understand when you run it):

1. main, which read the text files, split them into sentences, do the lemmatization using Stanford NLP, and combined the words in the same name entity into one token

2. is_stopword, determine if a word is stopword

3. N_Gram, which calculates how many times two words occur in the sliding window

4. tf-idf, compute the matrix where

tf = (number of times word occurs)/(len(document))
the reason we divide no of times word occurs by len(document) is because word could appear more times in longer document, thus we divide the document length to normalize it

idf = log(total documents/number of docs contain the word)

I saved keywords and tf-idf matrix in the /Java/results file as pdf
if you want to see it in a better format, you can run the code and the format is pretty clear.

## 2) Python

Since that I already implemented everything in Java, I only used Watson api to get the concept and keywords. And the result is saved in better format.

## 3)Concept per document

C11: Volcano

C5: Nuclear Weapon and its International impact

C3: China, US and Hainan incident

C4: International Agriculture

C6: North Korea missile emission

C15: US weather

C2: Drug in Asia

C13: North Korea, US, and South Korea relationship

C9: Religion

C8: Pollution

C7: Bank and interest rate

C14: Insurance Credit prediction

C12: US-Irap War

C1: International Airline

C10: Storm in US and its causes