Group Name: Argument Hackers

Sl.No	Matriculation Nr.	Name	Email address
1	6868664	Akshit Bhatia	abhatia@mail.uni-paderborn.de
2	6896703	Harshith Srinivas	harshith@campus.uni-paderborn.de
3	6852329	Suraj Manjunatha	surajm@mail.uni-paderborn.de
4	6866035	Vinaykumar Budanurmath	budanurm@mail.uni-paderborn.de

Documentation:

List of files required to execute the assignment tasks:

	File Name	Functionality	Input required	Output
1	assignment1_Json_generator.py	Python file contains the code to generate the unified "json output" file	Code folder consisting all the essays text and annotation files(described below)	Json file named: comp_arg_hackers.json (The json output file will be generated in the same folder where the python file is run)
2	assignment1_Statistics_generator.ipynb	python notebook file which contains code to generate the statistics	Json file that is produced after running the first python file.	Statistics

Some information related to python environment

Run the following four commands in the given order to install spacy properly.

1. python -m spacy download en_core_web_lg

2. python -m spacy download en_core_web_sm

3. python -m spacy download en

Version of Python: 3.6.10

Version of numpty: 1.16.0

Version of Spacy: 2.2.4

Instructions

1. unzip the file: Assignment1_CompArgsHackers.zip. Dont change the inner directory structure at all.
2. This folder contains the following contents
    1. **Code**: This folder contains one python file for generating the unified JSON file, one notebook which collects statistics from this json file, and one "essay_input_data" folder which contains all the input data required.
    2. **Data**: This contains unified JSON file
    3. **Documentation File:** A PDF file describing our approach and the steps for execution.
3. First Run the **assignment1_Json_generator.py .** This file requires "essay_input_data" folder in the directory in which the python file exist. (It is best not to touch the directory structure we have provided). This file will create a new Unified Json file**(comp_arg_hackers.json)** in this folder only.
4. Running the **assignment1_Statistics_generator.ipynb** in Jupiter Notebook will give the statistics from the JSON file, which was created in step 3.

Method used to compute the most specific words of each of the argument units:

We define the "most specific words" of a document as those words which can characterize a document's identity. We understand that TFIDF (Term Frequency, Inverse Document Frequency) will help us characterize a documents identity. TFIDF gives numerical statistics that can be used to reflect the importance of a word in a document. In a document, higher the TFIDF of a word, then that word is more important and more specific to that document.

APPROACH

We have implemented the TFIDF concept as described below.

Please note that we have not relied on sklearn package, instead we have written it ourselves.

For instance:

Consider the following lists which displays the tokens of argument units.

major_claim = {abc, abc, abc, def}

claim = {abc, abc, claim1, claim2}

permises = {abc, claim1, perm1}

Term-frequency of a word in a document:

	major_claim	claim	premise
Abc	3	2	1
Def	1	0	0
claim1	0	1	1
claim2	0	1	1
perm1	0	0	1

Inverse Document Frequency (IDF):

Inverse Document Frequency is the ratio which gives more importance to rare word. Basically, words which occurs only in one document will receive the maximum weightage. Below is the formula to compute IDF

Formulae:

$$IDF(word: = d) = log(N/N^d)$$

$N =$ no. of documents we have. We have 3 documents in total(major_claims, claims and premises).

Hence N = 3.

N^d = no. of documents containing the word d.

For the example we assumed above,

IDF(abc) = log(3/3) = log(1) = 0

IDF(def) = log(3/1) = log(3) = 1.58496250072

IDF(claim1) = log(3/2) = log(1.5) = 0.584962500721

IDF(claim2) = log(3/2) = log(1.5) = 0.584962500721

IDF(Perm1) = log(3/1) = log(3) = 1.58496250072

So looking at the above equations, we can say that the word abc which occurs in all the documents, receive the lowest weight. Meanwhile, perm1 and def have got the highest weights.

Now when we multiply these values with each word's frequency in a document, we can assure ourselves that the word specific to one document will get most importance than others.

$$TFIDF(w, doc) = N*IDF(w)$$

Here N is no. of times word w occurs in document 'doc'

This way we have calculated the most specific to a particular document.