Project 1 – Text analytics Using spaCy

Natural Language Processing (NLP) is a way to programme computers to process and analyse human language. The NLP tool used in this project is called spaCy, which is an open-source software library for advanced NLP in Python, recommended by Dr Ng. spaCy is advertised as an "industrial strength" NLP tool, which means that it is one of the most powerful software libraries for computers to understand human language [1].

In the beginning, spaCy was proposed to be used alongside Excel to help identify similar text among the thousands of rows of data. By using the similarity function built into spaCy, one can calculate the cosine similarity between two text after running both text through its NLP pipeline as seen below [2].

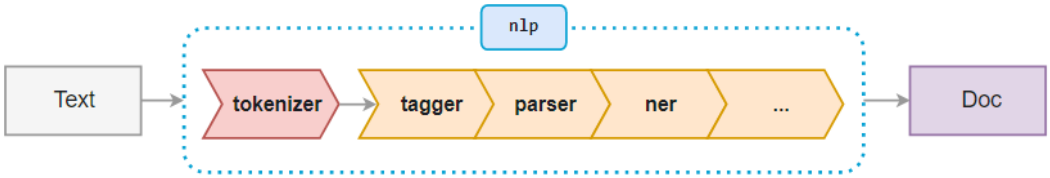


Figure 1: NLP pipeline of a default spaCy model

When "nlp" is called on a text, spaCy first tokenises them by segmenting words into tokens, to produce a Doc object. The NLP pipeline differs base on the type of model one loads into spaCy, and the pipeline seen in Figure 1 is using the default models (small, medium or large models) that spaCy provides. In the default pipeline, the "tagger" assign part-of-speech tags, the "parser" assign dependency labels and the "ner" detect and label named entities. Furthermore, users can also create and add custom components to the pipeline. Before finding out the similarity, each word will be assigned a vector that is generated using an algorithm called the Word2Vec. Each word vectors is a multi-dimensional meaning representations of a word as seen below that shows the word vectors of the word "Banana" [2].

Figure 2: Word vectors representation of "Banana" generated by Word2Vec

The first challenge I encountered was that the models doesn't work well with short forms and some local terms. This is because the models are trained on text from blogs, news and comments around the internet. Since having to train a new model for our use case was deemed too complex and possibly time consuming, pre-processing was done on text first, to let the Word2Vec algorithm perform better. This was done by first transforming each character into its lowercase form, remove any stop words, punctuations and then lemmatise each word. Furthermore, I was able to decide which model to use moving forward, based on the initial test results. The full analysis can be found in Appendix A, Figures A1 and A2. From Figures A1 and A2, we compare medium and large default models because the small

default model does not have enough information to calculate the similarity score between two text. The outliers found in Figure A2 refers to the test cases that spaCy computed to have low similarity scores, even though they are the same text with different wordings. Based on the data present in both Figures, there is an average difference of 1.552689015% between medium and large model's similarity score. On the other hand, there is an average difference of 42.28285579% between medium and large model's computation time. Therefore, all future spaCy programmes were done using the default large model due to its lower computational cost while having relatively the same output as the medium default model.

The second challenge I encountered was the software limitations of the Whole of Government (WOG) laptops. Due to security issues, I was unable to install Python directly into the laptops. Hence, I was tasked to try package my programme as an executable, as it might be easier to get approval. However, I was unable to do so even after two weeks of testing. Nonetheless, Dr Ng was still determined to find another use for spaCy and thus, it got repurposed as a standalone text analytics tool for news articles.

The purpose of the text analytics tool for news articles is to suggest to the user, how similar or different two news articles are. The main challenge here is to set accurate limits that separates the different categories of similarity. In the programme I made, similarity results were split into three categories; similar, some similarities and not similar. Which result appearing would be based on the similarity score output by spaCy. During initial testing of the programme, I found that my perception of similarity and a computer's version of similarity is different. This is because my perception of how similar two text are, is based on the semantic similarity between them, whereas spaCy perceive similarity based on the Word2Vec vectors between them. Hence, if two text have high amount of similar words but no contextual similarity, spaCy could output a high similarity score. However, after discussing with Dr Ng, we decided to go ahead and use the values I got from my initial testing. The test cases can be seen in Appendix A, Figures A3 to A5. The challenge of setting an accurate similarity score limit for the correct similarity result was mainly the lack of time to get enough and properly classifies test cases.

After getting the main code and limit done and approved, the next step is to create a Graphical User Interface (GUI) for users to load multiple news article URLs and even their own text documents to compare. Continuing the trend of using Python, the GUI was created using the PyQt5 library. Here, another challenge arises as PyQt uses object-oriented programming and their own design framework, which was new to me. However, with the help of Mr Goh, I was able to integrate spaCy into an engineering GUI seen below.

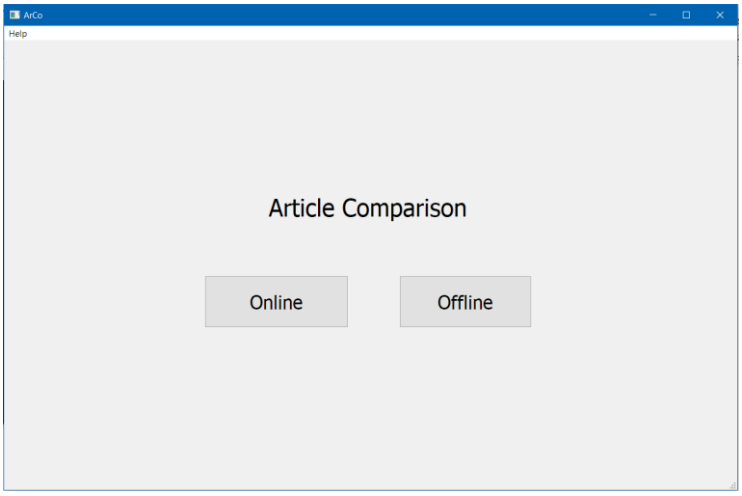


Figure 3: Main page of the engineering GUI with two selectable modes

As seen in Figure 3, the GUI has two main mode, online and offline. The online mode is able to take in URL inputs from news sites and then compare any selected article to the rest of the URL pool as seen in Figure 4 and 5 below.

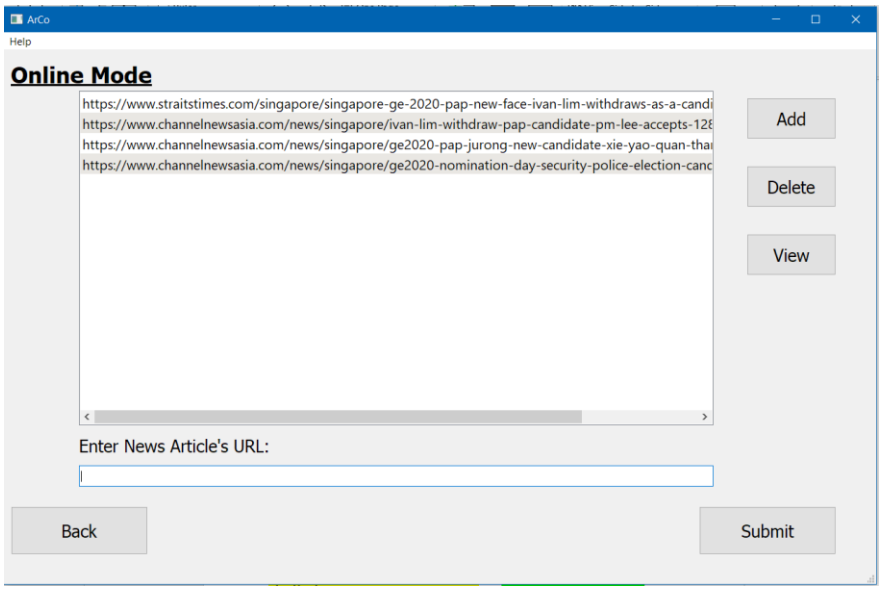


Figure 4: URL input page where users can either enter, delete or even view the URL that they entered

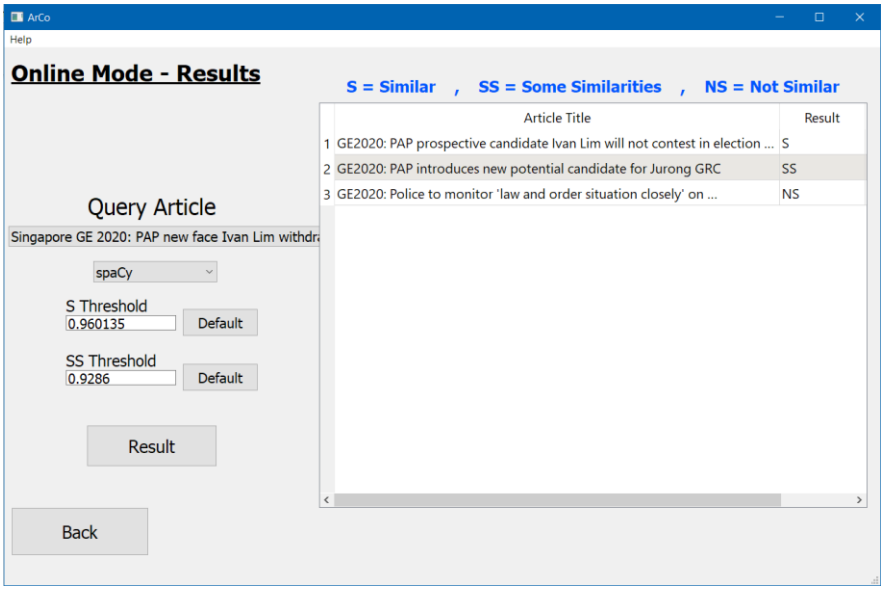


Figure 5: Online mode results page where users can choose the main article to compare against the rest of the pool on the left. Results are shown on the right.

As seen in Figure 4, users can enter URLs from news sites of their choosing. The input page also comes with a delete function and a view function if the user wishes to go to the URL in their browser. One caveat of this section is that it is not able to detect if the URL is from a news site, though the programme would still work. In Figure 5, users can select which article to test against in the dropdown box under the "Query Article" label. Clicking on the [Results] button would pass the scraped news article from the URL into spaCy, and the result would be shown on the right, under the "Results" column. As seen above in blue, there are three different outcomes; S, SS and NS. Not shown in Figure 5 is the similarity score that is to the right of the "Results" column. It was purposely hidden as the values would only

serve to confuse the user. Nonetheless, users can still scroll to the right to check the similarity score, and fine tune the results by changing the value under the “S” and “SS threshold” on the left.

The offline mode on the other hand, would require the user to import their own files from their computer directories, as seen below.

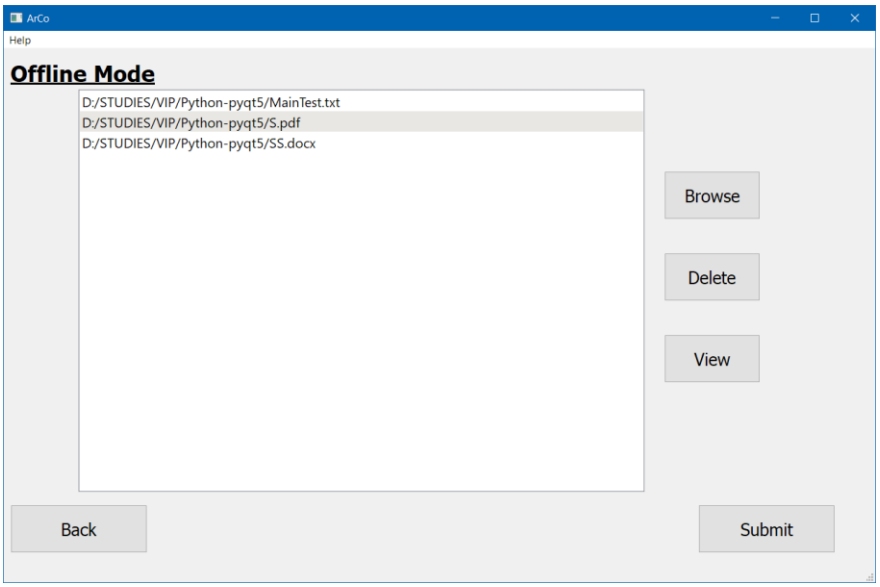


Figure 6: File import page where users can either upload, delete or view the files they uploaded

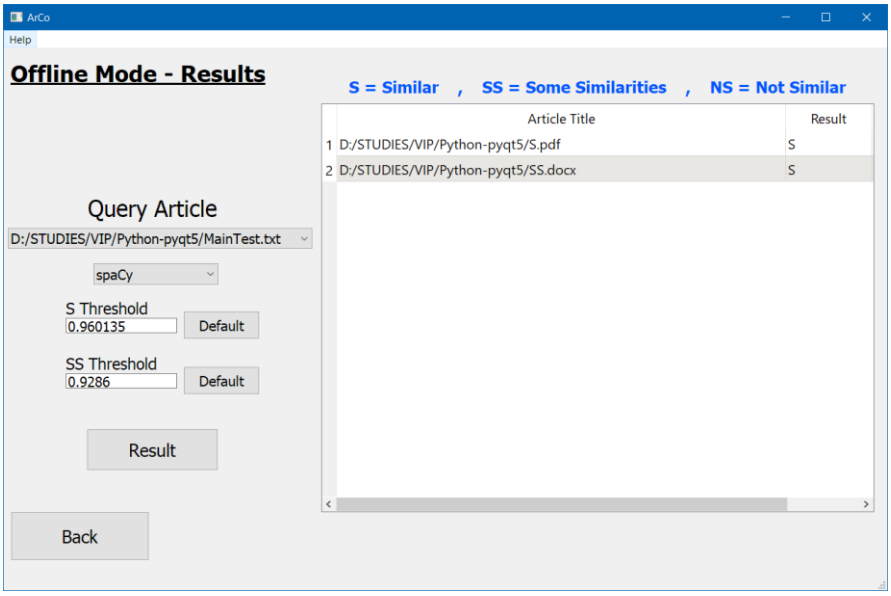


Figure 7: Offline results page where users can choose the main article to compare against the rest of the pool on the left. Results are shown on the right.

As seen in Figure 6, offline mode refers to users importing their own files to the programme. In this engineering GUI, it can accept text, pdf, and Microsoft word files as those are some of the most common document types. In Figure 7, the structure and the article uploaded is the same as in Figure 5. However, another caveat would be the difference in results when comparing the outcomes of articles from URLs and the same articles when it is uploaded. This is because in online mode, the news scraping library used to extract the articles, also included extra words like “advertisements” that served as placeholder for the actual advertisement one sees on the actual website. These words were another challenge for me,

as different news sites would have different layouts, which means the news scraping library would include different extra words for different news sites entered. Due to the lack of time, I was not able to figure out to remove those extra words.

Another feature of the engineering GUI is the option to choose between two models; spaCy's default large model and spaCy – BERT model as seen below.

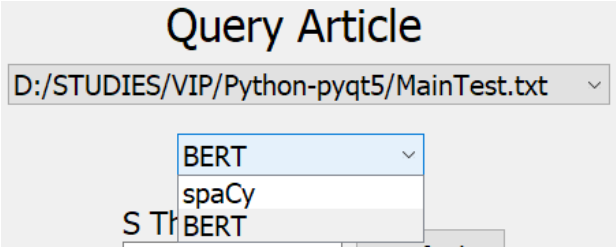


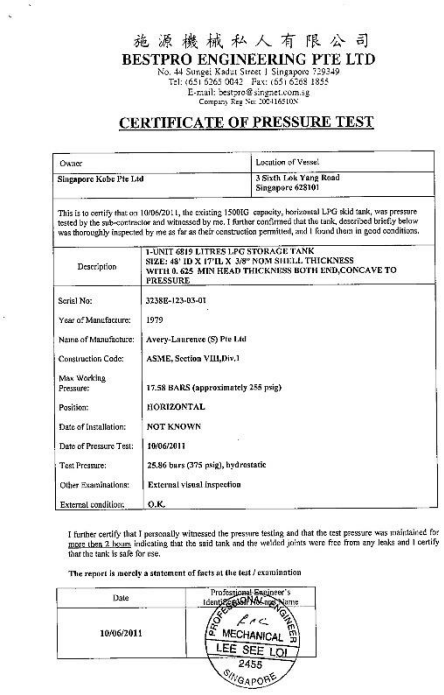
Figure 8: Option to choose which model to use when in the results page

BERT is an acronym for "Bidirectional Encoder Representations from Transformers". In summary, it is the state-of-the-art NLP language model that looks at both left and right side of a word when it was trained [3]. Luckily, spaCy published an open-source wrapper library called "spacy – transformers" that allowed me to use BERT in spaCy's framework. This library basically replaces the default NLP pipeline as seen in Figure 1 to its own pipeline as seen below.

Figure 9: Custom NLP pipeline when using spaCy -BERT model

As seen in Figure 9, after tokenising, it splits the text by sentences, then performs wordpiece pre-processing before running the transformer over the document with tok2vec [4]. Due to the "sentenciser", I remove the pre-processing function seen in the default model, since that would technically ruin BERT's understanding of each sentence. Initial testing of this model can be seen in Appendix A, Figures A3 to A5.

Another possible feature of this engineering GUI is to use it as a pdf document checker as proposed by Dr Ng. The idea is to extract the text in the scanned images of a pdf document, then check it contains the correct signatures, as seen below.



施源機械私人有限公司
BESTPRO ENGINEERING PTE LTD
No. 4d Sungei Kadut Street 1 Singapore 729549
Tel: (65) 6265 0042 Fax: (65) 6268 1855
E-mail: bestpro@singnet.com.sg
Company Reg No: 200416510N

CERTIFICATE OF PRESSURE TEST

Owner:	Location of Vessel:
Singapore Kobe Pte Ltd	3 Sixth Lok Yang Road Singapore 628101

This is to certify that on 10/06/2011, the existing 1500IG capacity, horizontal LPG skid tank, was pressure tested by the sub-contractor and witnessed by me. I further confirmed that the tank, described briefly below was thoroughly inspected by me as far as their construction permitted, and I found them in good conditions.

Description	1-UNIT 6819 LITRES LPG STORAGE TANK SIZE: 48' ID X 17'IL X 3/8" NOM SHELL THICKNESS WITH 0. 625 MIN HEAD THICKNESS BOTH END,CONCAVE TO PRESSURE
Serial No:	3238E-123-03-01
Year of Manufacture:	1979
Name of Manufacture:	Avery-Laurence (S) Pte Ltd
Construction Code:	ASME, Section VIII,Div.1
Max Working Pressure:	17.58 BARS (approximately 255 psig)
Position:	HORIZONTAL
Date of Installation:	NOT KNOWN
Date of Pressure Test:	10/06/2011
Test Pressure:	25.86 bars (375 psig), hydrostatic
Other Examinations:	External visual inspection
External condition:	O.K.

I further certify that I personally witnessed the pressure testing and that the test pressure was maintained for more then 2 hours indicating that the said tank and the welded joints were free from any leaks and I certify that the tank is safe for use.

The report is merely a statement of facts at the test / examination

Date	Professional Engineer's Identification Name
10/06/2011	PROFESSIONAL ENGINEER MECHANICAL LEE SEE LOI 2455 SINGAPORE

Figure 10: Example of a page of a scanned pdf document to be checked

As seen in Figure 10, it shows an example of a page to check. Using a Python computer vision library called “pytesseract”, I managed to extract out most of the text of page but not the stamp at the bottom, which is the most important part to check. Unfortunately, due to the complexity of computer vision pre-processing techniques and the lack of time, I was unable to optimise it further then a surface implementation. This includes implementing the technique for purely text pdf documents as well, in case of an event where both text and scanned images are present in the pdf document.