Intent Based Search Engine

(Documentation / In Depth Explanation)

Running the Program Locally

(Including the approach used for reading files)

Setting Up:

Installing Dependencies:

For Reading Files:

- pip install pillow
- pip install pdf2image
- pip install PyPDF2
- pip install python-docx
- pip install striprtf

Working with Data:

- pip install numpy
- pip install scipy
- pip install pandas

For fetching data from an API:

- Requests Module to call the API: pip install requests
- In order to convert XML Received into usable data we have used XML ElementTree which is a part of the Python distribution itself.

Installing **gensim** for NLP Tasks:

- pip install gensim

Installing **spaCy's en_core_web_lg**:

- pip install en
- python -m spacy download en_core_web_lg

If BERT Model is being used similar to the one used in the solution which is a implementation by Hugging Face:

- Install PyTorch: https://pytorch.org/get-started/locally/
- Install sentence-transformers: pip install sentence-transformers

The BERT Model would be downloaded when the script is executed as the download has been included as a part of the script itself.

Running the Script:

1. Place the intent_search.py file within the Root Repository where the files lie.
2. Make sure Python is installed in the system.
3. Install the dependencies as mentioned earlier.
4. In Command Prompt, go to the directory and run

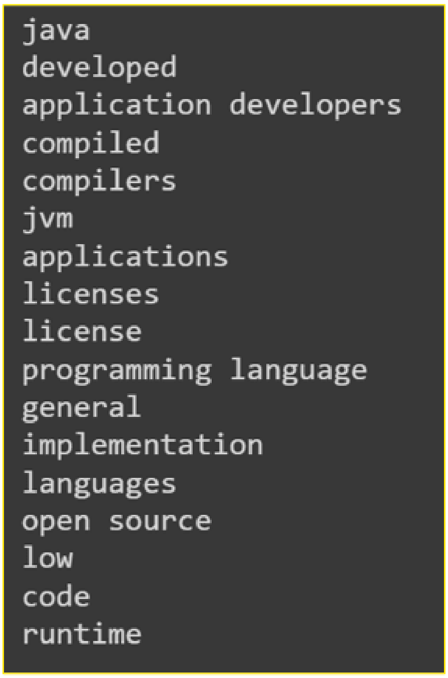
   " python intent_search.py "

Functions within the System

- getText(), read_pdf_data() are functions implemented to read data from docx and pdf respectively.
- find_files() function is implemented using os.walk in order to find all the files within the repository.
- In order to obtain Keywords from any given text: keywords_from_summary(summary) can be used. It is not necessary that the text used as an argument is the summary of file.
- clean_keywords(keywords) can be used to remove short / stop words from a set of keywords.
- find_keywords(filepaths) can be used alongside paths/to/files in order to retrieve the keywords as well as content of the files.
- get_summaries(documents) is used to extract the important sentences within content.
- LoadRSS(url) and parseXML(xmlfile) are used to fetch data in xml format and convert it into usable data respectively.
- sentence_similarity and keyword_similarity functions are the main parts of the program that calculate the similarities based on consine distances between 2 sentences or words respectively.
- deeper_solution and faster_solution functions handle the threshold values for the entire program.

Features:

1. Keyword Extraction:
    a. The first introductory passage of Java (Programming Language) was used to extract keywords and the results are as shown:



```
java
developed
application developers
compiled
compilers
jvm
applications
licenses
license
programming language
general
implementation
languages
open source
low
code
runtime
```

    b. As seen from the output most of the keywords almost all the words are related to Java either as its features or the way the language itself works.
    c. This helps us to get an overview about the article as a whole, while focusing on the necessary words rather than all the words within a file.

2. Word Similarity:
    a. In order to find the word similarity, we use Transfer Learning.
    b. Transfer Learning is storing knowledge gained while solving a certain problem and using it to solve different but different problem.
    c. The model used for word similarity: spaCy's en_core_web_lg
    d. en_core_web_lg:
        i. English multi-task CNN trained on OntoNotes, with GloVe vectors trained on Common Crawl:
        ii. OntoNotes:
            1. OntoNotes is a cumulative publications dataset which consists of over 2.9 million words.
            2. Data consists of text data sourced from conversations, newswire, newsgroups, weblogs and talk shows.
        iii. GloVe:

1. GloVe was released by Stanford and it consists of vector representation of words.
2. These words are represented in such a manner that the Euclidean Distance between two vectors provides an effective way of measuring the semantic or linguistic similarity between the words.
3. These GloVe Vectors are obtained by using words over Common Crawl which is a dataset created by web scraping.

iv. The nearest neighbors to **frog** in GloVe Vector Representation are:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. lizard
6. Eleutherodactylus

v. All of the above mentioned are closely related to frogs.

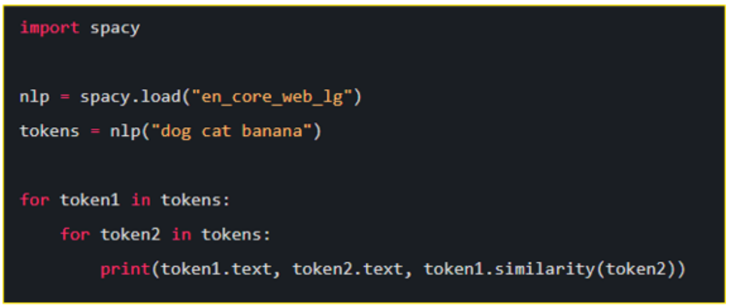
3. en_core_web_lg's Working:

a. It consists of 685k Unique Vectors with over 300 dimensions.

b. Example of how en_core_web_lg works:

i. Comparing the similarities between 3 words: dog, cat and banana

Code:

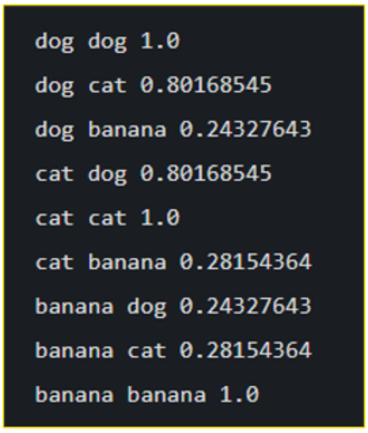


```
import spacy

nlp = spacy.load("en_core_web_lg")
tokens = nlp("dog cat banana")

for token1 in tokens:
    for token2 in tokens:
        print(token1.text, token2.text, token1.similarity(token2))
```

Output:


```
dog dog 1.0
dog cat 0.80168545
dog banana 0.24327643
cat dog 0.80168545
cat cat 1.0
cat banana 0.28154364
banana dog 0.24327643
banana cat 0.28154364
banana banana 1.0
```

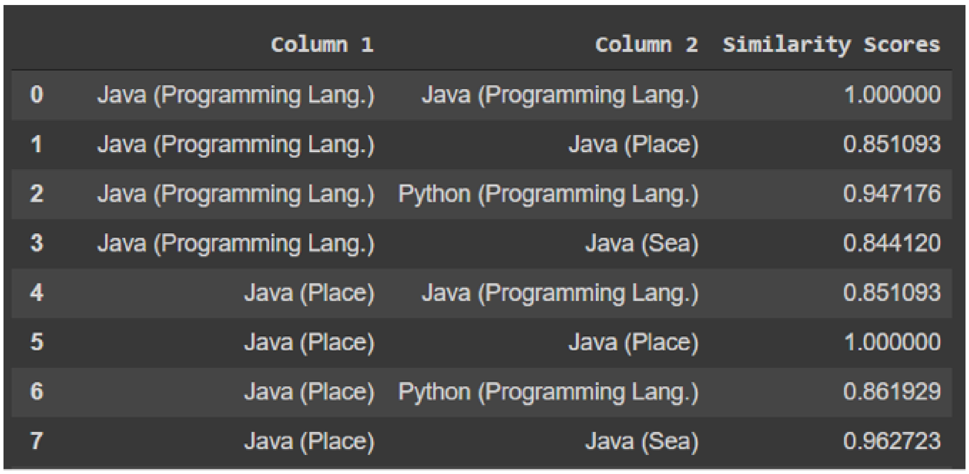
4. Sentence Extraction:
    a. Sentences capture more vital information than words themselves.
    b. For Example, the sentence: “He went to the prison cell with his cell phone to extract blood cell samples from inmates” has 3 different meanings of the word cell. The deeper approach is an attempt to capture this information.
    c. In order to extract the most important sentences from the text of the document again Text Rank Algorithm is used. However, instead of ranking the words themselves, sentences are ranked.
    d. This helps us extract the most important sentences from the documents.
    e. Example: When used with the text on Wikipedia about Java (Programming Language) the output was:
        i. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.
        ii. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses.
    f. These sentences give us a summary about the language itself.

5. Extracting Content based on Intents:
    a. In order to extract documents based on intents arXiv open access repository is used.
    b. ArXiv is an open access repository consisting of electronic print research consisting of 1,802,891 submissions related to topics such as mathematics, physics, mathematical finance, economics, etc. as of December 2, 2020.
    c. Our solution uses an open API provided by arXiv to fetch top articles based on the search term (intent).

d. This helps us retrieve relevant sentences based upon the intent entered which can be further used for sentence similarity with important sentences of the content of the files.
e. Example: Some of the documents retrieved for the intent 'Java' is as shown below:
    i. AIJ (ACL2 In Java) is a deep embedding in Java of an executable, side-effect-free, non-stop-accessing subset of the ACL2 language without guards. ATJ (ACL2 To Java) is a simple Java code generator that turns ACL2 functions into AIJ representations that are evaluated by the AIJ interpreter. AIJ and ATJ enable possibly verified ACL2 code to run as, and interoperate with, Java code, without much of the ACL2 framework or any of the Lisp runtime. The current speed of the resulting Java code may be adequate to some applications.
    ii. In recent years Java has matured to a stable easy-to-use language with the flexibility of an interpreter (for reflection etc.) but the performance and type checking of a compiled language. When we started using Java for astronomical applications around 1999 they were the first of their kind in astronomy. Now a great deal of astronomy software is written in Java as are many business applications.
f. Both documents talk about Java as a programming language and consist of sentences relating Java to software applications.
g. This gives us more context about the entered intent enabling us to use knowledge already available.

6. Sentence Similarity:

a. As earlier, Transfer Learning is used to find sentence similarities however, in order to understand the words based on the sentence Google's BERT (Bidirectional Encoder Representations from Transformers, 2018) Model is used.
b. The advantage this provides is that the vector BERT assigns to a word is a function of the entire sentence, therefore, a word can have different vectors based on the contexts.
c. Example: 4 Sentences were used to calculate similarity where the first sentence is a description of Java (Programming language), second Java (Place), third is a sentence concerned with Python (Programming Language) and fourth concerning Java (Sea). These sentences were:
    i. Java is class-based, object-oriented and is designed to have as few implementation dependencies as possible.
    ii. Java, lying between Sumatra and Bali, is a volcano-dotted island that’s at the geographic and economic center of Indonesia.
    iii. Python is an interpreted, high-level and general-purpose programming language.
    iv. The Java Sea is an extensive shallow sea on the Sunda Shelf. It is in the south of Java.



	Column 1	Column 2	Similarity Scores
0	Java (Programming Lang.)	Java (Programming Lang.)	1.000000
1	Java (Programming Lang.)	Java (Place)	0.851093
2	Java (Programming Lang.)	Python (Programming Lang.)	0.947176
3	Java (Programming Lang.)	Java (Sea)	0.844120
4	Java (Place)	Java (Programming Lang.)	0.851093
5	Java (Place)	Java (Place)	1.000000
6	Java (Place)	Python (Programming Lang.)	0.861929
7	Java (Place)	Java (Sea)	0.962723

d. Even though 3 of the 4 sentences consisted of Java, yet the sentence about Java (programming Language) was calculated to be most similar to Python (programming Language).
e. Similarly, Java (Place) was found to be most similar to Java (Sea)
f. Not that the current dataset only consisted of 4 sentences and the accuracy can be further improved when the actual dataset would be used.

Solution

The solution provided by us consists of 2 ways to search the files:

- A faster approach
- A deeper approach

The solution is an executable file which when executed within the root repository, will walk the entire repository to find files.

Based on the extensions of the file, different approaches are taken to read the contents of the file. If the file consists of graphical data which cannot be converted into text, then metadata of the file is used.

This data, in both the approaches, is then used with different similarity techniques to get the desired results.

Our approach to this problem is unsupervised and harnesses the power of transfer learning using pretrained models which means that there is no need to train or retrain the model itself.

Flow of the Faster Solution:

- Text within the document or the metadata is read based on the type of document.
- Keywords are then extracted from this text.
- Intent that is to be used for searching is entered by the user.
- For every keyword within the keywords found, en_core_web_lg is used to tokenize the keyword (obtaining vectors for the keyword) and then similarity (this is calculated based on the cosine distance of the words from each other) is calculated.
- The sum of the similarities is normalized by the count of the keywords (referred to as the score) which on average helps us to understand which set of keywords are closely related to the intent.
- The file with maximum score is returned along with files with a score of within 0.2 threshold value.

Flow of the Deeper Solution:

- Sentences would be extracted from the contents of the files within the repository.
- Intent would be taken as an input from the user.
- Articles would be fetched based on the intent using arXiv open repository.
- Sentence similarity using BERT Word Embedding is calculated between the sentences extracted from the files and the sentences related to the intents.
- Similarity will be normalized based upon the number of comparisons.
- Highest scoring file would be returned alongside other files that score within a threshold of 0.1 of the maximum file.

Comparing both the solutions:

- The time required for comparing texts based on keyword similarity for a set of 4 intents is about 19 seconds and the same task using sentence similarity takes about 240 seconds. Thus, for larger datasets the time difference would be significant.
- For repositories specific to a topic, keyword similarity would prove to be more efficient. For example, for repositories consisting of files only related to programming finding files for Java or Python would be more efficient using keyword similarity (faster solution).
- However, for repositories that might contain files consisting of diverse topics, using sentence similarity (deeper solution) would prove to be more efficient because of the inherent property of the solution to understand the texts itself.
- The entire system is similar to search and the advanced search options present in most of the popular search engines.
- Also, if a user does not have access to the internet, the faster solution would still work. The deeper solution requires internet to fetch articles related to the intents.

Advantages:

- No training or retraining required.
- Two different methods increases the use cases of the solution:
    - For Concise Datasets, Keyword Similarity would prove to be more efficient as it would not have to deal with ambiguous meanings.
    - For Diverse Datasets, Sentence Similarity would prove to be more efficient because a word can relate to different intents based on the context.
- In Offline Situations, Keyword Similarity would still work.
- It can be integrated with UiPath / Power Apps for better UI Experience.
- Feedback regarding the satisfaction of users with the results can be used to further adjust the threshold of the solution automatically.
- Model can be further fine-tuned to work on the kind of files used within EY.
- The solution can be used with any keyword without any issues.
- Because of the unsupervised nature of the solution, it can be easily scheduled to find and store files over a bunch of intents.

Porting Solution over to Azure:

- Both the methods use the Gensim package which is a very popular open-source NLP package. It is managed by RaRe-Technologies and is fast because it uses highly optimized Fortran/C under the hood.
- According to HG Insights, companies that use Gensim include RTX Travel, Grid Dynamics, LexisNexis and many more.
- Keyword Extraction can also be performed using the Text Analytics API provided by Microsoft Cognitive Services. Text Analytics API in its JSON response provides KeyPhrases which is similar to the Keyword Extraction implemented in our solution.
    - Link to API: https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/
- The BERT model used in the deeper solution is from sentence-transformers by Hugging Face, however it can be replaced by any model which provides us with BERT Word Embeddings.
    - Link to Repository: https://github.com/microsoft/AzureML-BERT
- Summaries from text can be obtained by using the Text Analytics API mentioned earlier to fetch KeyPhrases and then find sentences which contain those KeyPhrases.
- SpaCy's en_core_web_lg model is used within the solution. Spacy is a very popular open-source natural language processing library. According to HG Insights, some of the companies that use Spacy are: CVSHealth, Unitedhealth Group, U.S. Bank, Apple and many more.
- Other packages such as PyPDF, python-docx have been used to read data from the files which when ported can be implemented with methods already implemented within EY.
- Microsoft's NLP Recipes provides implementation of most of the techniques implemented in our solutions over Azure. References:
- Sentence Similarity: https://github.com/microsoft/nlp-recipes/blob/master/examples/sentence_similarity/bert_senteval.ipynb
- Text Summarization: https://github.com/microsoft/nlp-recipes/tree/master/examples/text_summarization

Project Links:

1. **Colab Notebook (To Run the Application in Colab):** https://colab.research.google.com/drive/1RzmGZj8h6GVPYABaF4TqrBfs_ztm8LJg?usp=sharing