COMP316 - Project Report
Vyasa Aiyer
222003978

Project Overview

The intended aim of this project is to extract medical terms from text and if this term is found on wikipedia then the first paragraph of its article is extracted and summarised. The medical term and the summary (if found) is then displayed to the user, in an almost glossary styled format.

This project incorporates two different areas of NLP, the first being Information Extraction which will be utilised to find/extract the medical terms in the provided text. The second more changing area involves Extractive Summarization used in the summarization of text from the wikipedia article.

Identification of Area of Application

My overall motivation for selecting this project revolves around my problem when reading medical based research papers, which is stopping each time I find a word , acronym or phrase that I don't fully comprehend to go and search for its meaning online for general understanding of the term so I can continue reading. I often found this using a great deal of time. In developing (and refining thereafter with pdf functionality) I hope to limit the time used for this repetitive task.

The intention of other individuals when using this project may include resolving similar problems I mentioned above. These individuals may include students that want to increase their understanding and remain productive or perhaps an individual with limited knowledge that aspires to tackle the terrains of medical text, without the hassle of continuous searches.

Information Extraction Section

For identification and hence the extraction of medical terms from text, a NER (Named - Entity Recognition) model was trained to identify all medical terms that fall under the entity labels of: SIGN_SYMPTOM, which covers symptoms of a disease or illness; DIAGNOSTIC_PROCEDURE, which includes procedures utilised in diagnosing a disease; BIOLOGICAL_STRUCTURE, which describes structures naturally found in organisms; DISEASE_DISORDER, which represents the name of a disease or illness; and MEDICATION, which covers the names of medications.

Extractive Text Summarization Section

For the text summarization of the Wikipedia article, I implemented an extractive approach. This incorporates the concept of extracting the most relevant sentences from the Wikipedia

article to form a summary.This process was facilitated through the use of a “Text Rank” algorithm that will only determine the output from the input text.

The text rank algorithm will parse the input/original text to find sentences, each of these sentences are assigned a score that will determine a rank in reference to the other sentences in the text. The abstractive summary is constructed by displaying the highest ranking sentences.

Choice of NLP/ML techniques

Information Extraction Section

Different approaches for developing NER models can be applied to this problem, some include the rule-based approach(utilises a pre-defined pattern-based and context-based rules for information extraction.) , dictionary-based approach (matches an entity in text to a vocabulary category based on data in a lexicon) and machine learning-based system approach.

For this project I utilised a a machine learning-based approach to develop the NER model, as both the dictionary-based am and rule-based approach suffer from the limited ability for identifying new or rarely utilised terms or phrases, additionally the handling cases of ambiguity is often a difficult issue to address in such approaches. In contrast, NER models developed using Machine Learning based approaches can often generalise rarer terms due to the training process , moreover during training of such models context cues and other patterns are found that have shown to massively decrease cases of ambiguity being issues.

I think that selecting a machine-learning approach is also applicable due to the ever fast paced of the medical world and hence the emergence of new terms. So in theory, a NER model developed machine-learning based strategies should better cope with newer terms and phrases (without revisions) in comparison to the other mentioned approaches. A dataset of medical case reports called MACCROBAT2020 was utilised in the training of the NER model.

The machine learning-approach I have taken involves four stages:

1. Annotation Data Preparation
    - Extraction of annotations from ann and txt files.
    - Only keeping annotations with relevance to the planned entities to recognize.
2. Training Data Preparation
    - Creating spans from valid annotations data.
    - Removing duplicate or empty spans
3. Training the Model
    - Usings spacy’s ner model pipeline with roberta transformer to train the model.
4. Testing and Evaluation
    - Evaluate models based on F1-Score, Precision and Recall.
    - Use NER on different sets of text and evaluate findings.

Extractive Text Summarization Section

There are many methods to approach creating a Text Rank algorithm, such approaches may include TF-IDF, Seq2Seq or graph-based methods. Term frequency-Inverse Document Frequency (TF-IDF) approach ranks sentences based on the frequency of words in text. Seq2Seq models make use of a neural network approach that incorporates encoder-decoder architecture, which can be used to generate abstract summaries. The graph-based approach views the text as a graph through which sentences are given rankings based upon the similarity of sentences in the input text; the top ranking sentences are used to produce a summary.

For this project, I opted to utilise a graph-based approach as in comparison to the other approaches that require training data this approach uses only the input text to determine the summary. The algorithm, through repetition, calculates the "importance" score for each sentence based on its connected sentences and similarity score. The repetition continues until the scores converge, this effectively ranks the sentences by their relevance. The top ranked sentences are then used to create a summary. Which in theory should provide a contextual summary that includes all the representative components of the original text.

This approach will encompass 4 stages:

1. Finding Sentence Similarity
    - Cleaning sentences.
    - Making vectors from unique words.
    - Calculating the cosine similarity.
2. Make Similarity Matrix
    - Iterate over all sentences.
    - Generate a similarity score for all "sentence pairs" that aren't the same.
3. Get Text Rank score
    - Iterate over similarity matrix.
    - Calculate news score.
    - Assign score.
    - Add Stop iteration mechanic.
4. Construct summary
    - Tokenize text to sentences.
    - Apply the above methods.
    - Get the desired amount of sentences.
    - Display summary.

These stages will be explored in greater detail in the following sections.

Designing and Modelling of the NLP Solution

Information Extraction Section

As mentioned above the development of the ner model will be constructed in four stages.

Annotation Preparation

In this stage of the project, the extraction of the needed data from the dataset is extracted.
The dataset used to train the NER model consisted of a collection of 200 text and annotation files, which provided a sufficient amount of training data.

The annotations file's contents utilised BRAT format which provided a wealth of information, however for the purposes of this project we are interested in extracting lines that begin with "T" (indicating it contains a token or entity) and contain one of the desired entity labels. The extraction of these lines were carried out by a single regex expression:

```
    tag_names    =    ["Sign_symptom"    ,    "Diagnostic_procedure",
"Biological_structure", "Disease_disorder","Medication"]
 regex = r'^T.*(?:' + '|'.join(tag_names) + r').*$'
```

Training Data Preparation

Part of the project involves the preparation of data for training the ner model, this done by first splitting the extracted lines from anonnations into its entity label, start index of the term and end index of the term and last lastly the actual term. An example of this is : `('DIAGNOSTIC_PROCEDURE', 138, 141, 'BMI')` .

Each annotation is then added to an "annotation group" which consists of the text file content and all its associated annotations. The general format of each annotation group is : `[text_file_content, (entity_label, start, end, term), ... (entity_label, start, end, term)]` ,each annotation - text file pair has an annotation group.

Thereafter, for each annotation group a spaCy Doc object called 'doc' is created using text_file_content. Spans created from the annotations,containing the indexes and entity label for each term, added to the 'doc' object, however all duplicate spans are removed using the "filter_spans". All valid spans are then added to a spaCy DocBin called 'doc_bin' which is loaded to 'train.model'.

Training the Model

During this phase the application of the prior data prepared is used to train the model, this is done through python's spaCy module. This stage consists of 2 commands.

The first command, `python -m spacy init config config.cfg --lang en --pipeline ner --gpu --force` initialises the configuration file, that declares the language,sets setting for training a NER model and to use the GPU (as recommended when training NER models). The "--force" is used to overwrite "config,cfg", where the configuration settings are saved.

The second command, `python -m spacy train config.cfg --output ./ --paths.train ./train.spacy --paths.dev ./train.spacy --gpu-id 0`
Is used to start the NER models training (./train config.cfg) and saves the output to the current directory (--output ./) The path to the training data is declared by the '--paths.train ./train.spacy', likewise '--paths.dev ./train.spacy' specifies the path to the development data. The '--gpu-id 0' portion declares that the training of the model must be done by the GPU.

Together, these lines initialised and thereafter began the training of the NER model. Finally, the trained NER model is loaded/saved to the disk for later use. This concludes the development of the NER model.

Extractive Text Summarization Section

1. Finding Sentence Similarity
    - Cleaning sentences.
    - Making vectors from unique words.
    - Calculating the cosine similarity.
2. Make Similarity Matrix
    - Iterate over all sentences.
    - Generate a similarity score for all “sentence pairs” that aren’t the same.
3. Get Text Rank score
    - Iterate over similarity matrix.
    - Calculate news score.
    - Assign score.
    - Add Stop iteration mechanic.
4. Construct summary
    - Tokenize text to sentences.
    - Apply the above methods.
    - Get the desired amount of sentences.
    - Display summary.

Finding Sentence Similarity

To get the similarity between 2 sentences, we put both sentences in lowercase and remove stopwords. This is done to make all words with different capitalization be treated the same, additionally the removal of stopwords is done so the finding the sentence similarity is only based on words that carry semantic meaning. The unique words from each sentence is used to create a vector for each sentence.

After processing the sentences, each vector is updated to contain frequencies of the unique words in each corresponding sentence. The cosine distance is calculated on the 2 vectors, this measures the similarity or similarity between the direction of two vectors hence this is used to determine the similarity of the 2 sentences. The statement `return 1-cosine_distance(vector_sentence1, vector_sentence2)` , ensures that only the similarity of the 2 sentences are returned,If the two sentences are identical, the cosine distance returns 0, which is then subtracted from 1. This subtraction leaves us with a similarity score of 1, indicating that the sentences are exactly the same.

Make Similarity Matrix

This stage of the project is carried out by a function that accepts an array of sentences, which count is used to create a square matrix. Now for each sentence pair (excluding the same sentence pairs aka the diagonal) a similarity score is determined using the `find_similarity` function and assigned to the respective matrix cell.

Get Text Rank Score

The sentence ranking process is implemented by a function that takes a similarity matrix. The outer loop iterates a maximum of 100 times. Within this loop, the inner loop calculates new scores for each sentence, considering its similarity with others. The damping factor, initialised at 0.85, controls the influence of prior scores on current scores, balancing the importance of existing rankings with the impact of new similarity information. Additionally, a threshold value named "epsilon" is included to ensure convergence. If the difference between updated and previous scores falls below epsilon, indicating minimal changes, the loop exits, optimising computational efficiency by stopping when the scores have converged to a stable solution.

Construct Summary

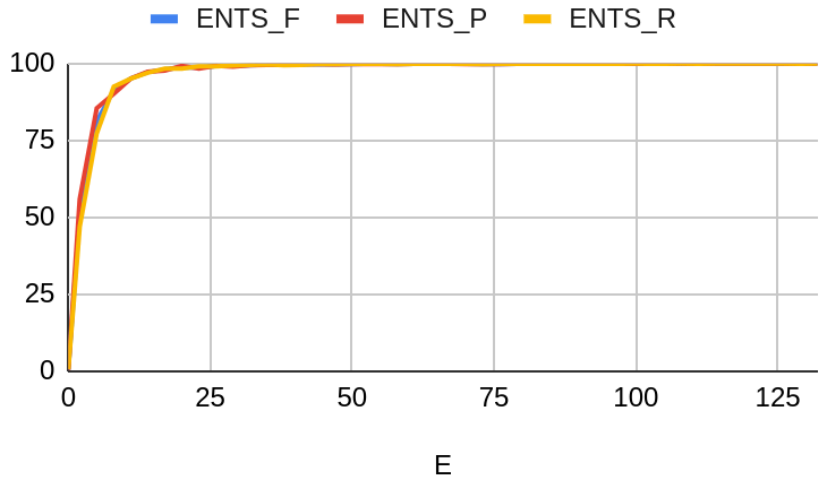
This part of the project utilises the above created functions. Like the above stages a function is used to construct the summary called `extract_summary` .The function accepts the original text to be summarised and the amount of desired sentences to act as the summary. In the function the stopwords are declared and sentences from the text are extracted. The sentences are then sent to the `construct_similarity_matrix` function to get a similarity matrix which is subsequent;y given to the `text_rank` function which obtains the ranking for each sentence. Thereafter the sentences are arranged in descending order of their score/rank, this is stored in `ranked_sentences`. Now the desired amount of sentences from `ranked_sentences` are added together using the join method, resulting in a summary.

Empirical evaluation of the NLP System

Information Extraction Section

For a complete evaluation of the NER model, we will first review the evaluation or performance metrics that were provided when the model's training began and later we shall run the model on test data and evaluate its performance from there.

For the sake of presentation and space conversation, I organised the data into charts. The following chart depicts the progression of the recall(ENTS_R), F1 score(ENTS_F) and precision(ENTS_P) of the model for each epoch or iteration (group):



The metrics mentioned earlier exhibit a logarithmic growth pattern approaching 100, but what does this mean? The precision of a model is determined by the amount of correctly identified named entities from all predicted entities thus far. Similarly, recall is determined by the proportion of correctly identified entities from actual entities. The F1 score is the harmonic mean of the recall and precision scores ,hence this metric is often one of the main indicators used to determine a model's expected performance. From this, we can predict that the model should be able to not only identify an entity but also correctly categorise the entity.

For the test data, we will be using 400 files derived from the first version of the annotation corpus from MACCROBAT2018, this differs from the 2020 annotation corpus used to train the model. So from this annotation corpus there are 13261 actual entities of interest, however when the model is executed 13288 entities are found leaving us with an excess of 27 identified terms, this could be an indication of overfitting. Focusing on the actual entities possible to be identified, 13063 entities were correctly identified (98.31%) with 198 entities incorrectly identified (1.49%). Despite the potential issue of overfitting, the model demonstrates outstanding performance.

Extractive Text Summarization Section

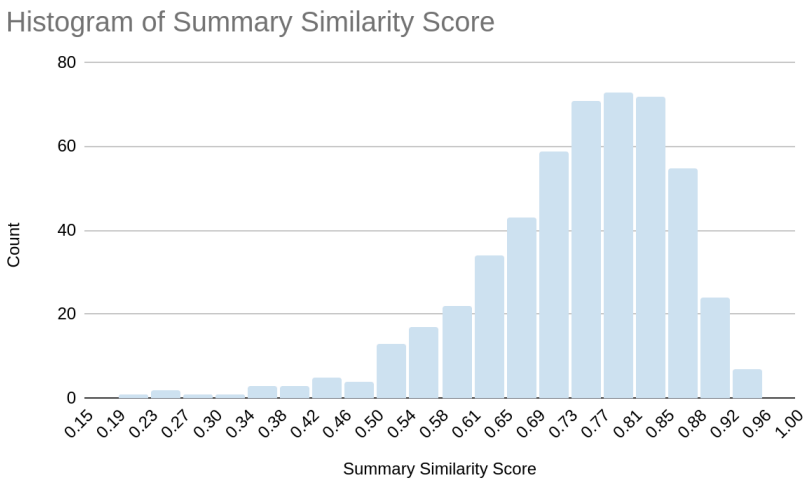
To determine the ability and limitation of the text rank algorithm, we can iterate over a selection of text files that have a corresponding summary that is not determined from my implementation of the text rank algorithm, for future reference this summary will be called the "baseline-summary". Then a comparison between the baseline-summary and one created from the Text Rank algorithm which will be used to determine the algorithm's suitability to summarise text. Note for these evaluations we assume that the "baseline-summary" acts as a good summary/representation of the original text.

Additionally for each summary created via my Text Rank algorithm must have a similar word count to the "baseline-summary". Without the control of the word count a fair similarity test between the extracted summary and the "baseline-summary" cannot be drawn as either summary might provide a reduced or extended context in relation solely based on the word count.

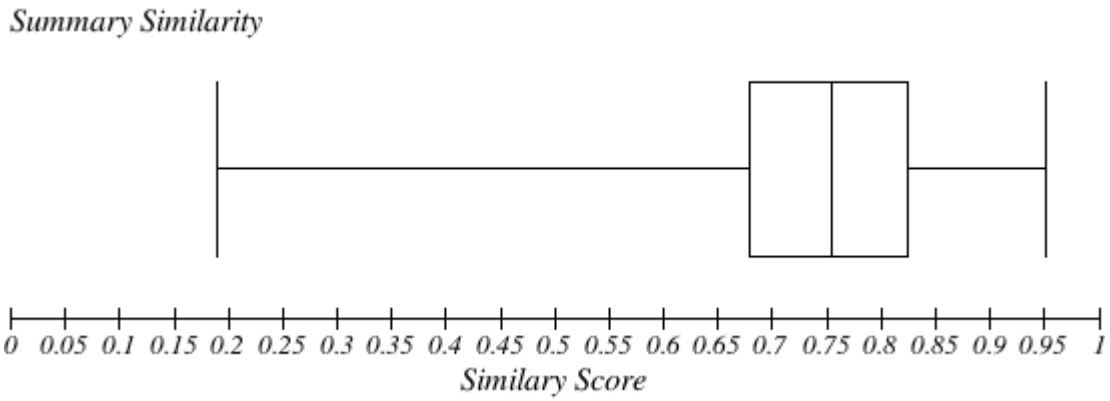
The baseline summary and original text used in this evaluation is found on BBC News Summary and only focuses on the business articles from the dataset. It is also important to note that this method of evaluation is valid due to the training data independence of the text rank approach utilised and its only dependance is the original text. Simply put, the algorithm in theory should be able to obtain a valid summary regardless of topic or even language as it fully relies on the occurrence of tokens in the original text.

The similar score between the 2 summaries is calculated by tokenizing both summaries by its words which is piped to the find_similarity method. This method processes the summaries into a vector format, these vectors are then used to calculate the cosine distance between the vectors which is used to find how similar the summaries are.

The results of the above process is graphed below:



From the graph above we can determine that most of the “baseline-summary” and extracted summaries share a high degree of content similarity. To further analyse this data a box plot can be produced.



From the box plot, we can draw the conclusion that 75% of the extracted summaries have a similarity of 0.68 with the “baseline-summary”. This infers that 75% have a similarity of 68% or higher. An average of 0.74 (74%) similarity between all summary pairs. The gathered metrics validate that text rank algorithm used,although simple acts a good summarization method.