# Summarization and Section Classification of Medical Abstracts Using SkimLit

## Introduction

Medical research papers, particularly abstracts, can frequently be difficult to interpret. For researchers and practitioners, carefully reviewing and extracting crucial information from each component (such as background, methodology, and findings) can be time-consuming. As the volume of scientific articles grows, there is a greater demand for an automated system that can break down and simplify abstracts while maintaining the original essence of the article.

This project proposes the development of an **Abstract summarization system** that automatically classifies sentences within medical abstracts and generates summaries based on different sections (for example, results, methods). This project was inspired by the need for more efficient and user-friendly access to medical research sources. The aim of this project seeks to make abstract reading easier for medical researchers.

## Summary of the Idea

This project aims to create a machine learning tool that categorizes medical abstract language into predetermined sections (e.g. Background, Methods, Results, Conclusions), summarizing each segment so that users may access information for specific portions of interest. The major objective is to use the SkimLit model to segment the abstract into sub-sections and then fine-tune pre-trained language models to provide abstractive summaries for each part.

## Input and Output Details

- **Input**:
    - **User-Provided Abstract**: Users will upload a medical abstract in text format.


```
[{'abstract': 'This RCT examined the efficacy of a manualized social intervention for children with HFASDs. Participants were randomly assigned to treatment or wait-list conditions. Treatment
included instruction and therapeutic activities targeting social skills, face-emotion recognition, interest expansion, and interpretation of non-literal language. A response-cost program was applied
to reduce problem behaviors and foster skills acquisition. Significant treatment effects were found for five of seven primary outcome measures (parent ratings and direct child measures). Secondary
measures based on staff ratings (treatment group only) corroborated gains reported by parents. High levels of parent, child and staff satisfaction were reported, along with high levels of treatment
fidelity. Standardized effect size estimates were primarily in the medium and large ranges and favored the treatment group.',
  'source': 'https://pubmed.ncbi.nlm.nih.gov/20232240/',
  'details': 'RCT of a manualized social treatment for high-functioning autism spectrum disorders'},
```


**Fig: Input Sample**

- **Output**:

- **Classified Abstract**: The model will classify the abstract's sentences into sections (background, objective, methods, results, conclusions).

| Section | Content |
|---|---|
| OBJECTIVE | This RCT examined the efficacy of a manualized social intervention for children with HFASDs. |
| METHODS | Participants were randomly assigned to treatment or wait-list conditions. |
| METHODS | Treatment included instruction and therapeutic activities targeting social skills, face-emotion recognition, interest expansion, and interpretation of non-literal language. |
| METHODS | A response-cost program was applied to reduce problem behaviors and foster skills acquisition. |
| METHODS | Significant treatment effects were found for five of seven primary outcome measures (parent ratings and direct child measures). |
| METHODS | Secondary measures based on staff ratings (treatment group only) corroborated gains reported by parents. |
| RESULTS | High levels of parent, child and staff satisfaction were reported, along with high levels of treatment fidelity. |
| RESULTS | Standardized effect size estimates were primarily in the medium and large ranges and favored the treatment group. |

**Fig: Output Sample Test Case from code**

## Dataset

The dataset for training and evaluation will consist of medical abstracts collected from **PubMed**, a popular biomedical research database. PubMed provides access to millions of abstracts across a variety of medical domains, making it an ideal source for extracting training data. Specifically:

- **Abstracts** will be raw textual data and will be pre-processed.
- **Labels** for sentence classification will be based on the typical sections found in scientific papers, such as background, objectives, methods, results, and conclusions.

The publicly available dataset of **PubMed 200k RCT (Randomized Controlled Trials)**, which already has labeled sections and abstracts that can be fine-tuned for specific section classification tasks will be used for training in this project.

PubMed 200k RCT is a PubMed-based dataset for sequential sentence categorization. The collection contains roughly 200,000 abstracts from randomized controlled trials, totaling 2.3 million sentences. This dataset is intended for sequential short-text categorization and the development of tools for skimming through literature (text segmentation).

Abstract sentences are classified as:

- Background

- Objective
- Method
- Result
- Conclusion

The dataset includes the original (PubMed 200k) and three subsets of the original. The raw data is converted into CSV files and is not part of the original dataset.

**Methodology**

1. **Data Preprocessing**:

   **A. Reading Abstract Text**: Each line in the abstract text file represents a labeled sentence.

   **B. Extracting Key Information**:

   - **Function**: extracts key information for each line, including the label (e.g., "BACKGROUND," "OBJECTIVE"), the sentence text, its position within the abstract, and the total number of lines in the abstract.
   - **Outputs**:
   - Label (e.g., "BACKGROUND," "OBJECTIVE")
   - Sentence text
   - Position within the abstract
   - Total number of lines in the abstract

   **C. Data Structuring**: Extracted data is converted into dataframes for further processing.

   **D. Preparing for Model Training**:

   - Text content is organized into lists for train, validation, and test sets.
   - Labels are one-hot encoded to create binary vectors for multi-class classification.
   - Labels are also encoded into integer format for numerical representation.

   **E. Class Information**:

   - Unique class names and total number of classes are determined to guide model output categories.

- Ensures compatibility between data format and model architecture.

**F. Outcome**: The preprocessing approach results in a uniformly structured dataset ready for training on labeled sentences from medical abstracts.

2. **Modeling and Evaluation**:
    - **Model 1: Naive Bayes Model**
        1. Processing Step: Text data is transformed into numerical features using TF-IDF vectorization.
        2. Model: Multinomial Naive Bayes classifier is applied for text classification.
        3. Evaluation Metrics: Model performance is assessed using accuracy, precision, recall, and F1-score.
    - **Deep Neural Modeling: Processing**
        1. Sentence Length Analysis: The average sentence length and distribution of sentence lengths are analyzed to determine a suitable sequence length for modeling, covering 95% of sentences.
        2. Text Vectorization: Sentences are transformed into token sequences using a text vectorizer, which maps each word to an integer, with the vocabulary capped at 68,000 tokens and sequence length standardized to 55 tokens.
        3. Token Embedding: Each token is embedded into a dense vector of 128 dimensions, enabling the model to learn semantic relationships between words. Padding and masking handle variable-length sequences for consistency.
        4. Dataset Preparation: Data is organized into TensorFlow datasets for training, validation, and testing, with each dataset batched and prefetched to improve processing efficiency during model training.
    - **Model 2: Custom token embedding with Conv1D**
        1. Processing Steps
            - Input (Text): The input layer accepts raw text data. It's a 1D shape (one text entry at a time) and is initially treated as a string.
            - Tokenization and Vectorization: Text is processed with a TextVectorization layer, which tokenizes the input text into a sequence of integers, each representing a specific token. This

vectorization layer generates sequences of fixed length (55 tokens per input in this case).

- Embedding: A learned embedding layer converts each token in the sequence into a 128-dimensional dense vector, which captures the semantic meaning of the words.
- 1D Convolutional Layers: This embedded vector sequence is passed through a 1D convolutional layer with 64 filters and a kernel size of 5. The convolutional layer extracts local features from each sequence, allowing the model to recognize patterns in word order and groupings.
- Global Pooling: A GlobalAveragePooling1D layer condenses the output of the convolutional layer into a single 64-dimensional vector, capturing the average value of each feature across the sequence.
- Output Layer: A fully connected (Dense) layer with a softmax activation function outputs probabilities for each class (5 in total), giving the likelihood of each label based on the input.

2. Model Compilation: The model is compiled using:
    - Loss Function: Categorical cross-entropy, suitable for multiclass classification tasks.
    - Optimizer: Adam optimizer, which adjusts learning rates for efficient convergence.
    - Metric: Accuracy, to track the model's performance during training.
3. Training: The model was trained over 5 epochs, with each epoch utilizing 10% of the training and validation datasets. This partial training speeds up the process, though the model is validated on the full validation set afterward.

- **Model 3: Character-Level Tokenizer and Conv1D Model**

This model focuses on character-level embeddings and uses a Conv1D network for text classification.

1. Character-Level Tokenizer: Each sentence is split into characters (as opposed to word tokens). A custom character-level vocabulary is built

using TextVectorization, covering letters, digits, and punctuation. Each character is then embedded into a 25-dimensional vector.

2. Conv1D Model: Inputs go through a 1D Convolutional layer, followed by global max-pooling to capture spatial features across the sequence. The final dense layer outputs class probabilities for text classification.
3. Training: The model is trained on character-level embeddings only, optimizing for categorical cross-entropy loss. This model captures fine-grained details in text but may struggle with longer dependencies due to the limited receptive field of Conv1D.

- **Model 4: Hybrid Token and Character Embeddings**
    1. This model combines both token-level embeddings and character-level embeddings, taking advantage of pretrained token embeddings while handling out-of-vocabulary (OOV) or unknown words at the character level.
    2. Token-Level Embeddings: Uses pretrained embeddings for token inputs, which are then passed through a dense layer.
    3. Character-Level Embeddings: A bi-directional LSTM is applied to character embeddings, capturing both forward and backward dependencies in character sequences.
    4. Concatenation: Token and character embeddings are concatenated to create a hybrid representation. Dropout layers are added to prevent overfitting, followed by a dense layer for further processing.
    5. Training: The hybrid model is trained on a dataset combining both token and character embeddings. This architecture is robust to OOV tokens while capturing both semantic (token-level) and orthographic (character-level) information.
- **Model 5: Transfer Learning with Token, Character, and Positional Embeddings**
    1. This advanced model incorporates positional embeddings alongside token and character embeddings, enabling it to understand the position of text segments within a document.
    2. Positional Embeddings: Adds contextual information about a token's position, useful for tasks that require sequence order awareness. Uses

one-hot encoding to represent the position in a document, like section headers or line numbers.

3. Combined Embeddings: Token embeddings, character embeddings, and positional embeddings are integrated to create a comprehensive representation of the input.
4. Transfer Learning: Pretrained embeddings are leveraged, allowing the model to benefit from large-scale pretraining on a wide range of language data. This approach improves generalization on unseen data and often leads to faster convergence.
5. Training: Trained on a dataset that includes the positional context of text, this model can better distinguish between different sections within a document, such as titles, abstracts, and content lines.

**Limitations**

- **Data Labeling:** While SkimLit is pre-trained for section categorization, some abstracts may require custom labels, especially if the abstract sections are different in structure.
- **Summarization Performance:** Abstractive summarization might be difficult to master in domain-specific situations such as medical literature. It is vital that the summaries stay factual and do not contain any misleading information.
- **Computational Resources:** Fine-tuning big models such as BART or GPT can be computationally intensive, hence adequate resources must be allocated for training.

**Evaluation**

- **Sentence Classification:** The accuracy of sentence classification into sections will be measured using metrics such as **precision, recall**, and **F1-score**.
    - Accuracy: Measures the proportion of correct predictions over total predictions.
    - Precision: Indicates the model's ability to avoid false positives.
    - Recall: Indicates the model's ability to capture true positives.
    - F1 Score: Harmonic mean of precision and recall, providing a balanced metric.

| | accuracy | precision | recall | f1 |
|---|---|---|---|---|
| baseline | 72.183238 | 0.718647 | 0.721832 | 0.698925 |
| custom_token_embed_conv1d | 80.110552 | 0.800858 | 0.801106 | 0.797837 |
| custom_char_embed_conv1d | 68.247716 | 0.683930 | 0.682477 | 0.678442 |
| hybrid_char_token_embed | 74.715345 | 0.744339 | 0.747153 | 0.741538 |
| tribrid_pos_char_token_embed | 84.261221 | 0.845472 | 0.842612 | 0.838721 |

**Fig: Model Evaluation as per code**

## Conclusion

This project will provide an efficient method for extracting and summarizing critical information from medical abstracts, allowing academics and medical practitioners to save time and focus on the most important facts. By utilizing the powerful SkimLit dataset and phrase classification models, the project aims to simplify how medical literature is interpreted.