Sentence Classification in Biomedical Literature: A Study on the PubMed 200k RCT Dataset

Author : *Maxime HAYAKAWA IVANOVIC*

1. Introduction and Motivation

Biomedical literature is expanding rapidly, making it increasingly difficult for researchers to manually analyze and extract relevant information. Automated sentence classification can facilitate tasks such as evidence extraction, clinical guideline generation, and systematic reviews - and so its development is crucial. *The PubMed 200k RCT* dataset provides a structured corpus of randomized controlled trial (RCT) abstracts labeled into categories such as Background, Methods, Results, and Conclusions : this project aims to develop and evaluate machine learning models for classifying sentences in this dataset, comparing traditional bag-of-words and embedding-based models to assess performance, as an experience of aforementioned sentence classification.

2. Data Description

The PubMed 200k RCT dataset consists of 2,270,286 sentences from biomedical research articles, based on a vocabulary consisting of 479,444 unique words, with each sentence assigned one of five labels (and their sentence counts per label):

- BACKGROUND : Provides context or justification for the study. (196 689)
- OBJECTIVE : States the goal of the research. (186 601)
- METHODS : Describes the methodology used in the study. (722 586)
- RESULTS : Presents key findings. (766 271)
- CONCLUSIONS : Summarizes outcomes and implications. (339 714)

The dataset split is as following :

- Training Data: 2,211,861 sentences
- Validation Data: 28,932 sentences
- Test Data: 29,493 sentences

The distribution of annotations shows a high frequency of *METHODS* and *RESULTS* labels, indicating that the dataset primarily focuses on scientific methodology and findings, which might influence model performance.

3. Methodology and Evaluation Setup

3.1 My Method

Due to hardware limitations, all hyperparameter tuning and experimentation (cf. *'EXPERIMENTATION'* part in the notebook) were conducted on the 20k dataset instead of the full 200k dataset, which was too large for my PC to handle in terms of RAM and execution time. The original plan was to fine-tune models on the 20k dataset and then manually train the best configurations on the full 200k dataset (cf. '*DEPLOYMENT*' part).

Baseline Models (Bag-of-Words (BoW) + Traditional Classifiers) :

The baseline approach consisted of three classifiers:

- **Multinomial Naive Bayes**: A probabilistic model particularly well-suited for text classification tasks due to its efficiency and assumption of word independence ;
- **Logistic Regression**: A strong linear model that is effective in high-dimensional spaces, making it a solid choice for text data ;
- **Random Forest**: An ensemble learning method that combines multiple decision trees, improving robustness and reducing overfitting ;

These classifiers were trained and fine-tuned using both:

- **CountVectorizer** (BoW representation)
- **TF-IDF Vectorizer** (weighted term frequency representation)

Models with Pre-trained Biomedical Word Embeddings

The second phase aimed to test how pre-trained biomedical word embeddings (BioWordVec) could improve classification performance. *BioWordVec* was selected because it is a domain-specific embedding model trained on biomedical text, capturing nuanced relationships between words better than simple frequency-based methods. The same three models were re-trained and fine-tuned on BioWordVec-generated sentence embeddings instead of Bag-of-Words features - or such was the plan initially, things had to be changed as I encountered data incompatibility problems for the MultinomialNB, and my Random Forest model wouldn't complete training after 3h30 of execution time... so I replaced both of them with **Linear Support Vector Classification** (LinearSVC), next to **Logistic Regression**, as it handles high-dimensional sparse data (e.g. word embeddings) while offering fast training and good generalization. I had attempted to pair these with tokenization preprocessing from SpaCy, but realized how long it would've taken me to consider all the data, making me drop the idea.

Deep Learning Approach

To explore deep learning performance on this task, a **Convolutional Neural Network** (CNN) was trained on raw text sequences, without any fine-tuning. CNNs have been shown to perform well in text classification by capturing local patterns and phrase structures, which could lead to superior results compared to traditional classifiers.

Final Training on 200k Dataset

Once the best hyperparameters for each model were identified using the 20k dataset, the final step was to train each model on the full 200k dataset using manually selected best hyperparameters. While it would have been ideal to fine-tune the models directly on the 200k dataset, hardware constraints made this impractical, thus, the conclusions drawn are based on the assumption that hyperparameter behavior on the 20k dataset would generalize to the full dataset.

3.2 Frameworks and Tools

The implementation mainly relied on the following libraries and frameworks:

- Machine Learning Libraries:
  - **scikit-learn**: Provided implementations for Naive Bayes, Logistic Regression, Random Forest, LinearSVC, and hyperparameter tuning (GridSearchCV).
  - **gensim**: Loaded and processed pre-trained BioWordVec embeddings.
- Deep Learning:
  - **TensorFlow & Keras**: Implemented and trained the **CNN model** for text classification.
- Data Processing:
  - **numpy**: Handled data manipulation and numerical operations.
  - **spacy**: Attempted for text tokenization and preprocessing.
- Visualization:
  - **matplotlib, seaborn, time**: Plotted model evaluation scores and confusion matrices. Also measured cell execution times

3.3 Implemented Models and Hyperparameters

Baseline Models (Bag-of-Words + Traditional Classifiers)

- **Multinomial Naive Bayes**
  - w/ CountVectorizer : `alpha=1`
  - w/ TfIdfVectorizer: `alpha=0.1`
- **Logistic Regression**
  - w/ CountVectorizer : `C=0.1; solver='saga'; max_iter=5000`
  - w/ TfIdfVectorizer : `C=1; solver='saga'; max_iter=5000`
- **Random Forest**
  - w/ CountVectorizer : `n_estimators=100`; `max_depth=None; min_samples_split=5`
  - w/ TfIdfVectorizer : `n_estimators=100`; `max_depth=None`

Embedding-Based Models (Pre-trained BioWordVec Embeddings)

- **Logistic Regression** : `C=100; max_iter=5000`
- **LinearSVC :** `C=1; max_iter=5000`

Deep Learning-Based Model

- **Convolutional Neural Network (CNN)**
  - `input_dim = max_vocab_size = len(vocabulary)`
  - `output_dim = 128`
  - `input_length = 47`
  - `filters = 128`
  - `kernel_size = 5`
  - `activation = 'relu'`
  - `pooling = GlobalMaxPooling1D()`
  - `dropout_rate = 0.5`
  - `dense_units = 128`
  - `output_activation = 'softmax'`

3.4 Evaluation Metrics

The models were evaluated using the following metrics:

- **Accuracy**: The primary metric for classification performance.
- **Precision, Recall, F1-score**: To better understand model performance across different classes.
- **Confusion Matrix**: Provided insights into misclassifications and error patterns for some cases.

4. Results and Discussion

The tables below presents the **test accuracy scores** for each model:

Model	Accuracy (Test : 20k)	Accuracy (Test : 200k)
Multinomial Naive Bayes (CountVectorizer)	**77%**	**78.4%**
Logistic Regression (CountVectorizer)	**80%**	**82.7%**
Random Forest (CountVectorizer)	**78.3%**	**58.3%**
Multinomial Naive Bayes (TfidfVectorizer)	**74.5%**	**77%**
Logistic Regression (TfidfVectorizer)	**80%**	**82.3%**
Random Forest (TfidfVectorizer)	**78%**	**58.9%**
Logistic Regression (BioWordVec)	**77%**	**78.7%**
LinearSVC (BioWordVec)	**76.6%**	**78.1%**
CNN	**79.3%**	**NaN***

**hadn't enough RAM memory to run CNN on 200k.*

Key Observations

- **Naive Bayes performed the worst** due to its assumption of word independence, which does not fully capture sentence structure.
- **Logistic Regression outperformed Naive Bayes**, particularly when used with TF-IDF or embeddings.
- **Random Forest achieved reasonable results**, but training was computationally expensive.
- **SVM and XGBoost performed significantly better** when combined with pre-trained word embeddings, highlighting the advantage of contextual word representations.
- **CNN achieved the highest accuracy (88.5%)**, showing that deep learning effectively captures sentence semantics.

Confusion Matrix Analysis

The **confusion matrix** indicated that:

- **RESULTS and CONCLUSIONS** were the most frequently misclassified categories.
- **BACKGROUND and OBJECTIVE** were easier to classify due to their distinct phrasing.
- **METHODS had the highest accuracy**, likely due to its structured format in biomedical abstracts.