Medical Abstract Classification Using Two Approaches

A Comparative Analysis of Statistical Machine Learning and Transformer Based Large Language Models for Text Classification

Jakob Berggren
TDDE16 / Text Mining
jakbe841@student.liu.se
github.com/jakeberggren

Abstract

Doctors are required to constantly update their knowledge within the medical field, which is largely done by reading medical papers and journals. Rapid increase in publications means it is becoming harder for doctors and medical researchers to filter out what is relevant to their specific work. This study presents a comparative analysis of machine learning (ML) techniques for text classification, aiming to limit the search space for doctors and medical researchers when skimming through medical abstracts. Two approaches will be analyzed: a statistical approach and a Large Language Model (LLM) approach, utilizing OpenAI's GPT-3.5 Turbo. The results of the study show that both approaches could be used in order to assist doctors and medical researchers, although higher accuracy may be required depending on their use. Further, while the LLM approach slightly outperforms, it is not currently worth the additional overhead regarding time and compute. However, great potential for LLMs is shown, and future progress in the LLM field will likely make this approach a more viable option. Other findings include that the difficulty in classifying different categories of abstracts varies, possibly because some categories are not as easily distinguishable linguistically.

1 Introduction

The day-to-day work of a doctor does not only include treating patients. Doctors must also constantly update their knowledge in their field. This in turn requires extensive reading of medical journals, where doctors wanting to update their knowledge spends about 4 hours a week on reading medical journals (Garba et al., 2010). Moreover, the volume of publications is rapidly increasing, making it impossible to keep up with the flow. Doctors must therefore be able to effectively pick out what journals are important for their own knowledge development and their current patients' needs. This often includes a quick glance at the title, introduction, and abstract to decide whether they should continue to read the journal, or rule it out (Garba et al., 2010). It is easy to conclude that limiting the search space would speed up this process, and allow doctors to spend more time on reading the journals that are actually important for their work. This is where medical NLP, and text classification, can play a role.

Text classification is a common task in Natural Language Processing (NLP) and is best described as the task of labeling or classifying a set of text into two or more specific classes. This can be done with a number of different techniques, and is most commonly done using supervised statistical machine learning (ML) techniques for classification, such as Naive Bayes, or Random Forest. These techniques requires labeled training data, typically consisting of examples of text paired with their corresponding class labels, in order to learn the patterns and characteristics of each class. But what if extensive labeled data is not available or hard to come by? In this case, one would typically turn to an unsupervised ML technique, such as K-means clustering. This project will however explore the use of Large Language Models (LLMs) and Generative AI, for the classification task.

Since the introduction of the transformer architecture, pioneered by Vaswani et al. (Vaswani et al., 2017), the advancements of LLMs and Generative AI has surged. It is now easy to see that the potential for this technology is huge, and LLMs have shown to have great abilities in understanding, reasoning and generating text, as well as an ability to perform downstream NLP tasks. While this often includes fine-tuning the model to the specific task, this project aims to explore the capabilities of a general pre-trained model, not fine-tuned to the specific task, and seeks to find out the performance and effectiveness of such models on the text classification task in a medical context. Moreover, this will be compared and evaluated against more traditional supervised ML techniques for the same task.

2 Theory

The theory chapter will present a relevant theoretical background to the project, explaining the main concepts employed in the method.

2.1 Statistical Machine Learning

Statistical machine learning is no different than it sounds like, and can be described as learning by data with the use of statistical algorithms. While there are others, the learning is often divided into two categories, *supervised* and *unsupervised*, where supervised learning is characterized by the presence of an outcome variable, whereas in unsupervised learning such a variable is not present (Hastie et al., 2009). The theory behind the supervised classifier algorithms used in this project will be briefly described below.

2.1.1 Multinomial Naive Bayes

Naive Bayes (NB) classifier is based on Bayes Theorem with the assumption that the features are conditionally independent. The multinomial NB classifier computes the probability of a document d belonging to class c by the probability of term t_k occurring in class c (see equation 1) (Manning et al., 2008). Manning et al. described this as the amount of evidence t_k contributes to c being the correct classification.

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c) \tag{1}$$

2.1.2 Support Vector Machine

A Support Vector Machine (SVM) is based on finding the hyperplane or decision boundary that best separates the classes in the feature space. This plane is maximally far away from any data point, a distance referred to as the *margin*. The position and orientation of the hyperplane is ultimately decided by the subset of data closest to the decision boundary. These points are called the *support vectors* (Manning et al., 2008).

2.1.3 Random Forest

Random forest is a decision tree based technique. Main ideas include combining several trees, where each tree is trained on random subsets of the data (also referred to as bagging). For classification, the class selected by most trees is then chosen (i.e., by majority voting) (Lindholm et al., 2022).

2.1.4 Gradient Boosting

Boosting can be described as learning several weak classifiers in sequence, where each classifier attempts to correct the errors made by its predecessors. Often, decision trees are used as the weak classifiers. In contrast to other boosting based techniques, gradient boosting uses gradient descent to minimize the loss function, where each added tree is fitted to the residual errors of the previous trees, reducing the overall error (Lindholm et al., 2022).

2.2 Large Language Models

Language modeling can be considered as a statistical model where the next word in a sequence of words can be predicted as a conditional probability of previous words, as described by Bengio et al. in equation 2:

$$\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t|w_1^{t-1}) \tag{2}$$

where w_t is the t:th word (Bengio et al., 2003). Radford et al. later showed that training a language model on large datasets could yield more general task solving capabilities across a multitude of Natural Language Processing (NLP) domains (Radford et al., 2019). Authors then introduce GPT-2, a model based on the transformer architecture, introduced by Vaswani et al. (Vaswani et al., 2017). Radford et al. further explain that through a process known as *"transfer learning"*, these models can learn to solve downstream tasks without needing task-specific data, leveraging the knowledge gained from their training data, also referred to as *"zero-shot learning"* (Radford et al., 2019).

2.2.1 Prompt Engineering

Large Language Models are often instructed or *"prompted"* by natural language that describes the task to be solved. Constructing effective prompts is often referred to as *"prompt engineering"*, where common techniques include zero-, one-, and few-shot prompting. While zero-shot learning leverages pre-existing knowledge from the training data (as explained above), one- and few-shot learning relies on including one or few examples, demonstrating how the task is to be solved (Brown et al., 2020). As shown by Brown et al. utilizing one- or few-shot prompting techniques can greatly improve the model's performance on a given task (Brown et al., 2020).

2.3 Additional Concepts

This subsection aims to describe additional concepts used in this study, beginning with introducing embeddings and similarity measures.

2.3.1 Embeddings and Similarity Measures

In NLP, an *embedding* refers to a vector representation of text. This can be either words, sentences or entire bodies of text, with the idea that embeddings with high similarity will be represented close together in vector space (Jurafsky and Martin, 2000). Measuring the similarity is often done using *"cosine similarity"*, the cosine of the angle between the two embeddings in vector space. As explained by Jurafsky et al. cosine similarity can be defined as in equation 3 (Jurafsky and Martin, 2000).

$$cosine(v,w) = \frac{v \cdot w}{|v||w|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}} \tag{3}$$

2.3.2 Undersampling

Class imbalance is a common problem in machine learning, meaning that there is an imbalance in the number of samples from each class. A common technique to counteract the imbalance problem is using *random undersampling*. This involves randomly reducing the number of instances or samples from the majority class, making the dataset more balanced (Mohammed et al., 2020).

3 Data

The dataset used in this project consists of about 14k medical abstracts describing different patient conditions, labeled in five different categories: 1. neoplasms, 2. digestive system diseases, 3. nervous system diseases, 4. cardiovascular diseases, and 5. general pathological conditions. The data was compiled by Schopf et al. in conjunction with the paper *"Evaluating Unsupervised Text Classification: Zero-Shot and Similarity-Based Approaches"* (Schopf et al., 2022), and pre-divided into training and testing data.

The final data used was then a pre-processed version of the original dataset. First, the test data was reduced to 50% of the original size. This was done since the later methods makes use of the OpenAI API[1] to access the OpenAI model GPT-3.5 Turbo[2], which is not free of charge. Using too large a dataset would increase the total cost of the project, which was preferred to be kept as low as possible. Reducing the dataset was done using scikit-learn's[3] (a common python package for data analysis) built-in function *"train_test_split"*, using the condition label as *"stratify"* to ensure the original class balance.

Second, looking at the structure of the training data, one could see that the classes were quite imbalanced. Since this could pose a problem for statistical methods for classification, the data was balanced by undersampling to the minority class. This of course also reduced the size of the training data set. Figure 1 and 2 shows the difference in the data structure before and after undersampling, respectively. Final training data contained 1200 examples of each class. No processing changing the balance of the test data was done.

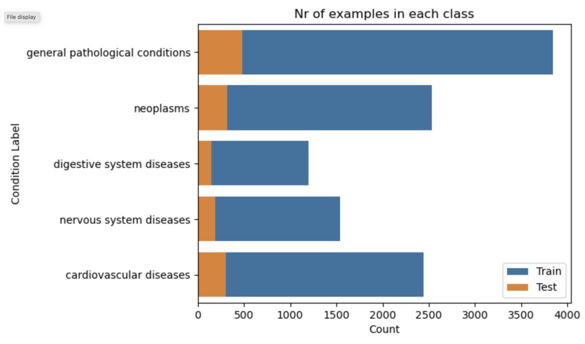

Figure 1: Structure of data pre undersample.

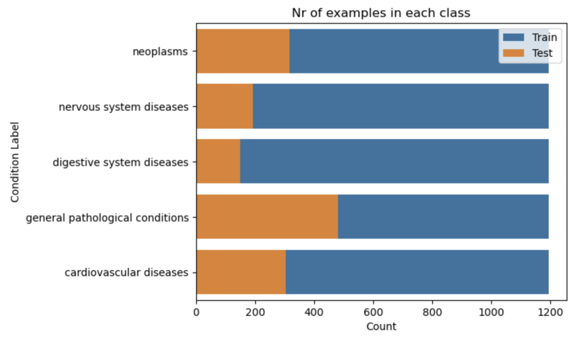

Figure 2: Structure of data post undersample.

4 Method

The method chapter presents the approach to the stated problem, and aims to give details for complete replicability. However, for full details, the

[1]https://platform.openai.com/overview
[2]https://platform.openai.com/docs/models/gpt-3-5-turbo
[3]https://scikit-learn.org/stable/

implementation can be found on GitHub[4]. The method is divided into three distinct parts, which will all be described in the subsequent subsections.

4.1 Loading and pre-preprocessing data

First, train and test data was loaded into separate pandas DataFrames, using the python package pandas[5]. As explained in section 3, 50% of the test data was first removed due to API constraints in the later methods (see section 4.3). The original data was labeled using numbers, and not the names of each condition, which is why one of the first steps was to translate each number into the corresponding labels. Correct translation was provided by the data set authors in conjunction with the data. Next, the data was explored by both looking at the structure (in terms of class count and balance), and creating word clouds for each class to get a better sense of common words, both across the classes, but also words that signify each class. Figure 3 shows the result of these word-clouds.

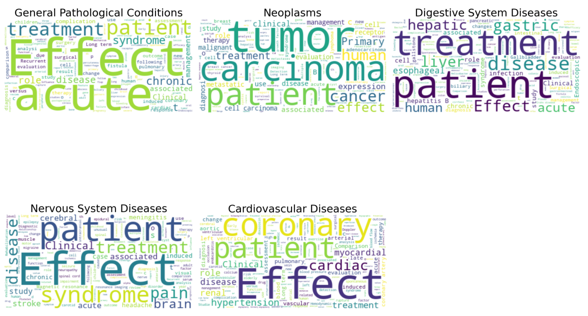

Figure 3: Word clouds showcasing common words in each class.

The data exploration resulted in two main insights. First, the training data was quite imbalanced, which resulted in additional pre-processing by undersampling to the minority class, as explained in section 3. Second, insights from the word-clouds resulted in the experimenting with medical stop-word removal to test if this could increase the performance of the statistical approach, which is further explained in subsection 4.2.

4.2 Statistical approach

The statistical approach to the problem implements four different techniques, all common in multi-class classification tasks: 1. Multinomial Naive Bayes, 2. Support Vector Machines (with linear kernel), 3. Random Forest, and 4. Gradient Boosting. These techniques were all implemented using count vectorization, meaning that the training documents are transformed into document-term matrices representing the frequency of each token in each document. Initially, no stop-word removal was applied.

Each technique was then fitted using the undersampled training data, and later tested on the test data. Metrics accuracy, precision, recall and f1-score were then computed by comparing predicted labels to the gold label in the test data. Results were saved into .csv files, and later visualized as confusion matrices and a bar plot for each metric and technique. The implementation was largely done using built-in packages from scikit-learn as well as visualized using Matplotlib[6] and Seaborn[7], two common statistical data visualization tools.

Following the analysis of the word clouds seen in figure 3, it was concluded that experimentation with stop-word removal could possibly increase the performance of the statistical approach. Some clear trends from the analysis could be seen. For example, *general pathological conditions* contain words like “chronic” and “acute” (also found in other categories but less frequently). *Neoplasms* include the words “tumor”, “cancer”, and “carcinoma”. *Digestive System Diseases* include “gastric” and “hepatic”, and *Cardiovascular Diseases* include “cardiac”. many of these are of course not surprising. However, some words are quite common across all the categories, such as “patient”, “effect” and “treatment”.

Stop-word removal was done in three different ways. First, common English stop-words were removed using Natural Language Toolkit, or NLTK[8] a python package commonly used for statistical NLP. Second, clinical stop-words were removed by adding stop-words used by Ganesan et al. in their work on discovering related clinical concepts using large amounts of clinical notes (Ganesan et al., 2016). The full stop-words list can be found on the authors’ GitHub.[9]. Third, three additional stop-words were manually added due to their frequency and lack of meaning in the context of medical abstracts: “patient”, “effect”, and “treatment”. Finally, any duplicates were removed from the stop-words list.

[4]https://github.com/jakeberggren/TDDE16-Text-Mining-Project/tree/main
[5]https://pandas.pydata.org

[6]https://matplotlib.org/stable/
[7]https://seaborn.pydata.org/#
[8]https://www.nltk.org
[9]https://github.com/kavgan/clinical-concepts

Previous experiments were then re-done, this time removing stop-words as explained above. Further, Results were computed, saved and visualized in the same way as previously explained.

4.3 LLM approach

The LLM approach, similar to the statistical approach, implements four different techniques separated into four experiments respectively: 1. Zero-shot classifier, 2. One-shot classifier, 3. Few-shot classifier, and 4. Similarity based classifier, all further explained in further detail in the subsequent subsections. Similarly to the statistical approach, results from each experiment were collected and computed in the form of accuracy, precision, recall and f1-score by comparing to the test data gold label. The results were then aggregated and visualized in the same way as in the statistical approach, in order for easy comparisons between the different approaches.

All four LLM approaches makes use of the OpenAI API for access to the model GPT-3.5 Turbo[10]. While not the latest model from OpenAI, GPT-3.5 Turbo is more cost-effective, while still offering very high complexity. For the similarity based classifier, OpenAI's embedding model was used[11]. Further, the python package LangChain[12] was employed to simplify the implementation for all techniques presented.

4.3.1 Zero-shot classifier

For the Zero-shot classifier, a simple zero-shot prompt was created, which can be seen in figure 4.

```
template = """Classify the given
medical abstract into one of these
5 medical conditions:

1. neoplasms
2. digestive system diseases
3. nervous system diseases
4. cardiovascular diseases
5. general pathological conditions

Medical abstract: {abstract}
Condition: """
```

Figure 4: Zero-shot prompt template. {abstract} acts as a placeholder for the actual medical abstract to classify.

[10]https://platform.openai.com/docs/models/gpt-3-5-turbo
[11]https://platform.openai.com/docs/models/embeddings
[12]https://www.langchain.com

Each medical abstract in the test set was then classified using GPT-3.5 Turbo and the prompt template in figure 4. The output data from the model was post processed to only include the actual prediction (in the case that the model produced more surrounding words other than one of the 5 medical conditions). Further, in cases where no prediction was made by the model, one of the 5 medical conditions were chosen at random.

4.3.2 One-shot classifier

The One-shot classifier was implemented in a similar manner. First, *one* example from each category was sampled from the training data at random. Next, the previous prompt was built upon, adding the following parts as shown in figure 5:

```
.
.
4. cardiovascular diseases
5. general pathological conditions

Base your answer on the following
examples of medical abstracts and
correctly labeled conditions:

Medical abstract: {example_abstract}
Condition: {example_condition}

Medical Abstract: {abstract}
Condition:""""
```

Figure 5: One- and few-shot prompt template.

Post-processing was made in the same way as previously explained.

4.3.3 Few-shot classifier

Using the same prompt as for the one-shot classifier, shown in figure 5, the few-shot classifier was implemented by including *three* examples (instead of one) of each category of medical abstracts from the training data to the model. No other changes were made to the implementation or prompt.

4.3.4 Similarity based classifier

The similarity based classifier was implemented with inspiration from Schopf et al. (Schopf et al., 2022), which proposed the original dataset used in this study. The method is based on comparing training data embeddings to the embedded input abstract, assigning it the same condition label as the most similar example from the training data. This is done by embedding the training data using the embedding model provided by OpenAI, and then storing the embeddings in a Chroma DB[13] vector database. Each example in the test data is then subsequently embedded and similarity matched to the training data in the vector database by cosine distance, and assigned to the label with the lowest similarity score (lower is better, or "more similar").

5 Results

The results chapter presents results from both the statistical, and LLM approach previously presented in section 4.

5.1 Results: statistical approach

Table 1 presents aggregated results from the statistical approach, both with and without (w/o) stop-word removal. Highlighted values showcase the best performing model in each category in this iteration.

Model/Method	**Accuracy**	**Precision**	**Recall**	**F1-score**
Naive Bayes	0.56	0.56	0.56	0.53
SVM	0.48	0.48	0.48	0.47
Random Forest	0.52	0.51	0.52	0.48
Gradient Boosting	0.59	0.59	0.59	0.58
Naive Bayes	0.57	0.57	0.57	0.54
SVM	0.50	0.50	0.50	0.49
Random Forest	0.52	0.52	0.52	0.49
Gradient Boosting	**0.60**	**0.60**	**0.60**	**0.59**

Table 1: Aggregated results from both statistical approaches. Models in *italic* showcase results with stop-words removed.

Gradient Boosting here displays the best performance across all metrics, regardless of stop-word removal. The general accuracy of the statistical approaches is around 48 to 60%, where again the gradient boosting classifier with stop word removal has the overall best accuracy. Further, figure 6 and 7 showcase confusion matrices for each method in this iteration. This shows that the category *general pathological conditions* is more difficult for all models to correctly classify compared to the other conditions. Again, gradient boosting outperforms in correctly labeling this condition. Further comparing the results, one can see that stop-word removal increases the predictive performance for *general pathological conditions* across all models, where for example the SVM classifier now predicts 137 of 481 correctly compared to 125 of 481 previously. Overall, the SVM classifier seems to benefit the most from stop-word removal.

[13]https://www.trychroma.com

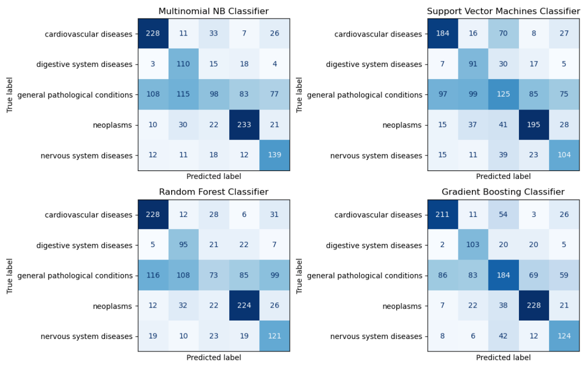

Figure 6: Confusion matrices for statistical approach w/o stop-word removal.

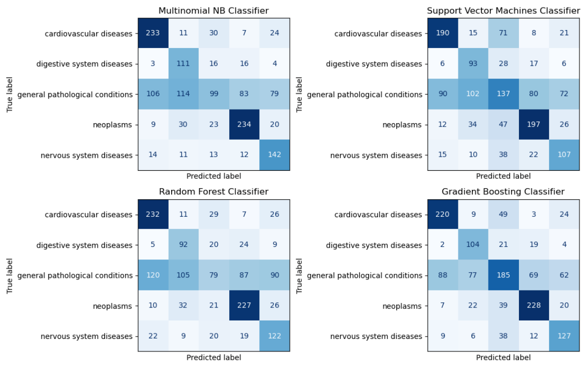

Figure 7: Confusion matrices for statistical approach with stop-word removal.

5.2 Results: LLM approach

Table 2 presents aggregated results from both the statistical approach, both with and w/o stop-word removal, as well as the LLM approach. Again, highlighted values showcase the best performing model in each category in this iteration.

Here, one can see that the Few-shot prompted GPT-3.5 Turbo model from the LLM approach performs the best out of all tested models, however very similar to gradient boosting in all metrics but one. The One-shot model also displays equally good performance in terms of F1-score. Figure 8 displays additional information to the results from the LLM approach in the form of confusion matrices for each tested method. Similar to the statistical approach, *general pathological conditions* seems to be the most difficult to correctly classify, especially for the zero-shot prompted model and the similarity classifier. However, significant improvement can be seen in correctly classifying this condition in the one- and few-shot prompted classifiers. At

Model/Method	Accuracy	Precision	Recall	F1-score
Naive Bayes	0.56	0.56	0.56	0.53
SVM	0.48	0.48	0.48	0.47
Random Forest	0.52	0.51	0.52	0.48
Gradient Boosting	0.59	0.59	0.59	0.58
Naive Bayes	0.57	0.57	0.57	0.54
SVM	0.50	0.50	0.50	0.49
Random Forest	0.52	0.52	0.52	0.49
Gradient Boosting	**0.60**	0.60	**0.60**	**0.59**
Zero-shot	0.55	0.59	0.55	0.52
One-shot	0.59	0.60	0.59	**0.59**
Few-shot	**0.60**	**0.61**	**0.60**	**0.59**
Similarity based	0.47	0.46	0.47	0.45

Table 2: Aggregated results from both the statistical and LLM approach. Models in *italic* showcase results with stop-words removed.

the same time, a slight decrease in performance compared to the zero-shot prompted model can be seen, for example in the categories digestive system diseases and neoplasms.

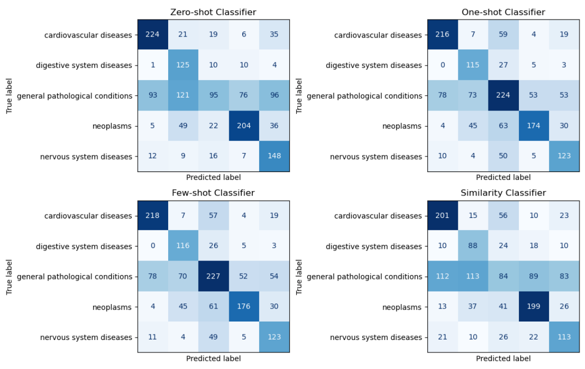

Figure 8: Confusion matrices for llm approach.

6 Discussion

The following chapter will analyze the results and discuss the possibilities and limitations of both approaches in the context of classifying medical abstracts. Moreover, the study will be compared to related work on medical text classification.

6.1 Results analysis

Results of the study are quite interesting and really show the potential for the use of LLMs in the task of text classification and how it can assist doctors and medical researchers. While not drastically outperforming the statistical methods, the results show that simply zero-shot prompting the model can achieve performance on par with most of the statistical approaches, and by including a few examples by few-shot prompting, the model slightly outperforms. What is very impressive here, is that this uses a highly generalized model, only pre-trained on vast amounts of data, which means that one could use this solution "out of the box" with no, or only a few labeled examples. In contrast, the statistical methods used in this study needs labeled data for the training process. An interesting future work could thus be to also fine-tune the large language model on the labeled medical abstract training data to see how this could affect and possible increase the performance further.

A clear trend that can be seen, is that the category *general pathological conditions* is the most difficult for almost all models and methods to classify. On the statistical side, gradient boosting handled this category the best, and slight improvement could be seen by removing applying stop-word removal. On the LLM side, a very clear improvement could be seen by applying one and few-shot prompting, compared to zero-shot prompting. Seeing this is such a clear trend, it is possible that the methods used, for example undersampling to the minority class, yields these results. Undersampling might lead to a loss of crucial information, making it more difficult for the statistical models to find good decision boundaries. However, seeing the same trend is seen on the zero-shot prompted LLM (which does not make use of the training data at all), it could also be the case, that the abstracts on general pathological conditions, do not have as clear linguistic features, making them harder to separate and tell apart from other abstracts.

An important thing to consider, is whether achieved accuracy is sufficient. An accuracy of about 60% means that many abstracts will be incorrectly classified. This could pose a problem dependent on use, possibly causing for example a doctor to miss an important abstract needed for a patient or for research. However, using these approaches as a helping tool, knowing that it will not categorize everything correctly can still be of great use, saving time as doctors will know what abstracts to look at first. A great addition to such a tool could be presenting a percentage point as to how "confident" the model is in it classification, and what other possible classes are considered.

6.2 Limitations

As stated above, one of benefits of the LLM approach is that the solution works "out of the box", and almost no labeled data is needed. However, this impressive performance comes at the cost of far more computational cost and time. Although not presented as part of the results, the runtime of the LLM based methods was far longer than the statistical methods, as well as computed in the cloud as compared to on a consumer laptop. This impacts the actual usefulness of the LLM approach, especially when the results presented here do not show any significant benefit in regard to classification performance. Also taking into account the amount of medical abstracts and that speed and computational costs are important factors, this makes the LLM approach less and less viable. However, mind that the these experiments were done on GPT-3.5 Turbo, which is not the currently most powerful model of this type. With the current rate of improvement, LLMs will likely outperform statistical methods on text classification, making them an eligible option.

A methodological limitation of this work is that it would have been interesting to compare the LLM based approach to *unsupervised* statistical approaches, rather than *supervised* ones. This could give further insights into what methods perform the best when labeled data is not available at all, rather than it being a separate benefit of the LLM approach. This will however be left as future work on the area.

6.3 Related work

Dernoncourt et al. presents PubMed 200k RCT (available on GitHub[14]), a dataset for sequential sentence classification in medical abstracts, focusing on randomized controlled trials (RCTs) (Dernoncourt and Lee, 2017). Similar to the work presented in this study, Dernoncourt et al. research aims to develop better tools for doctors and medical researchers to more quickly and effectively pick out important journals and papers from a large amount of abstracts.

Authors found, that many medical abstracts are unstructured, meaning they are not divided into semantic headings, such as "objective" and "results". They therefore focus their efforts on creating a dataset where each sentence is given a specific label, then classifying the different sentences (they call this *sequential sentence classification*), making it easier for doctors and researchers to locate the desired information. Authors also present several baseline classifiers for future researchers to build upon.

[14]https://github.com/Franck-Dernoncourt/pubmed-rct

7 Conclusion

Concluding the analysis, LLMs can be used for text classification and offer benefits for doctors and medical researchers picking out relevant medical abstracts. Furthermore, using one-shot or few-shot prompting with the LLM can enhance task performance, particularly for abstracts that may not have as clear linguistic features, even outperforming more traditional statistical methods. However, the current performance gap between the statistical approach and the LLM-based approach is not significant enough to justify the computational overhead and additional time required for using LLMs. This might change in the near future as LLMs are getting both better and computationally cheaper. In cases where there is a shortage of labeled data, LLMs can still be a viable option to consider due to their ability to perform the task with minimal examples.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. In *Journal of Machine Learning Research*, volume 3.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts.

Kavita Ganesan, Shane Lloyd, and Vikren Sarkar. 2016. Discovering Related Clinical Concepts Using Large Amounts of Clinical Notes. *Biomedical Engineering and Computational Biology*, 7s2.

Stephen Garba, Adamu Ahmed, Ahmed Mai, Geoffery Makama, and Vincent Odigie. 2010. Proliferations of scientific medical journals: A burden or a blessing. *Oman Medical Journal*, 25(4).

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning, Second Edition*, volume 27, page 2.

Dan Jurafsky and James H. Martin. 2000. *Speech and language processing: an introduction to natural language processing, computational linguistics,* *and speech recognition, 2nd Edition*, pages 105–110, 114–115.

Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. 2022. *Machine Learning - A First Course for Engineers and Scientists*, pages 164–165, 171–173, 182–187. Cambridge University Press.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*, pages 258–259, 320–321.

Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. 2020. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In *2020 11th International Conference on Information and Communication Systems (ICICS)*, pages 243–248.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Tim Schopf, Daniel Braun, and Florian Matthes. 2022. Evaluating Unsupervised Text Classification: Zero-shot and Similarity-based Approaches. In *ACM International Conference Proceeding Series*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 2017-December.