Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Matematyczny
specjalność: analiza danych

Dawid Sonor Kubkowski

Biomedical Text Classification with Pre-trained Language Models

Praca magisterska
napisana pod kierunkiem
dr. Michała Burdukiewicza i
dr. Grzegorza Jagielli

Wrocław 2025 r.

Contents

1 Introduction

Motivation

With more than 1 million articles published every year in the biomedical and life sciences domain [1], manual literature review, especially within the biomedical domain, became increasingly difficult for individual researchers or small teams. A 2023 analysis found that the total number of articles indexed in Scopus and Web of Science in 2022 was approximately 47% higher than in 2016 [2]. Curating datasets requires specialist knowledge, a lot of work and time, especially with such a growth of publications. A simple search provides us with many articles to review. As a result, researchers can overlook important publications or spend too much time analyzing materials of little value to their work.

For this reason, it is worth considering tools that would help relieve the burden on researchers and indicate publications that we might be potentially interested in based on previously labeled data.

Objective

Text classification in the biomedical domain creates particular challenges: the terminology is highly specialized and varied, the datasets are often imbalanced. The aim of this work is to compare classical machine learning methods [3] with transformer [4] based approaches on the binary biomedical text classification task.

This work focuses on automatizing the data curation process, which at the moment requires manual assessment and is time-consuming. We implement a binary classifier that leverages an article's abstract, title and journal name to predict whether the article is relevant for us. In addition, we develop a fetcher that retrieves these data from the Entrez API using the PubMed ID (PMID) of the article. The approach is mainly evaluated on a manually curated dataset that contains experimental research on the interaction between amyloids and antibodies.

The methods used in this work are based on classical machine learning and deep learning methods. In the case of classical methods, we deal with logistic regression, random forest and they will serve as a baseline. In terms of deep learning, we investigate BioMed BERT [5], SapBERT [6], BioMed RoBERTa [7], which represent specialized variants of the BERT [8] architecture adapted to biomedical domain.

An overview of the complete data processing and model training pipeline is shown in Figure 1.

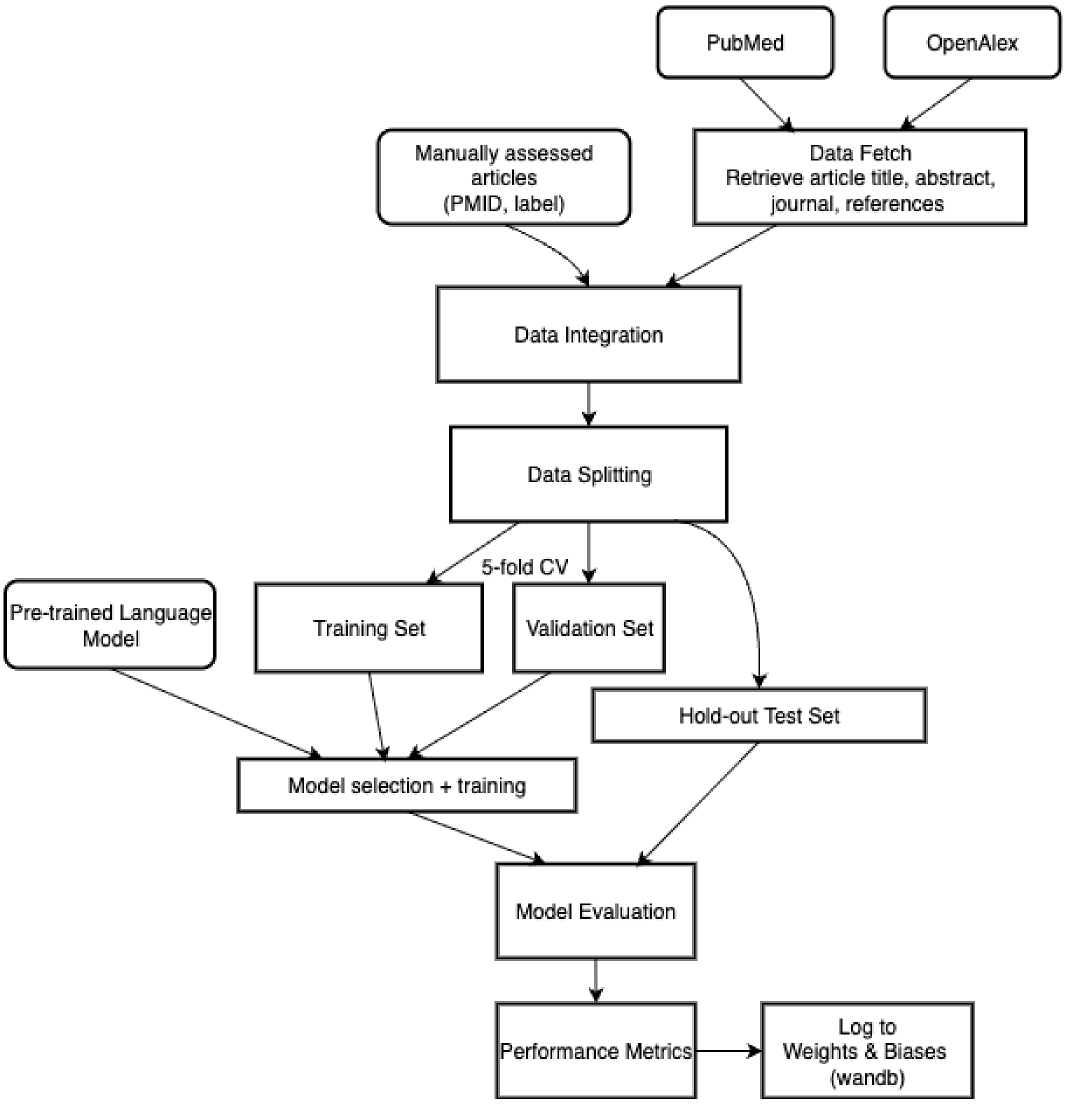

PubMed
OpenAlex
Manually assessed articles (PMID, label)
Data Fetch Retrieve article title, abstract, journal, references
Data Integration
Data Splitting
5-fold CV
Pre-trained Language Model
Training Set
Validation Set
Hold-out Test Set
Model selection + training
Model Evaluation
Performance Metrics
Log to Weights & Biases (wandb)

Figure 1: Diagram illustrating data integration, language model training and evaluation.

The diagram illustrates how the data are processed and the whole training pipeline procedure. The elements in oval shapes represent external inputs (raw data, pre-trained models), while the rest of the pipeline is designed and implemented as a part of this thesis work.

Structure and prerequisites

The thesis is divided into 6 sections. In section 1 we present motivations and objective of the work. Section 2 contains theoretical aspects of the used methods, the goal is to provide the background necessary to understand the experiments in later sections. We start with traditional approaches to represent the text, then we move to the transformer architecture and BERT. Section 3 describes the dataset used in this work. Section 4 explains the experiment procedures: data splitting, evaluation, hyperparameters. Section 5 describes alternative methods that were tried in parallel with the main approach. It presents the attempts to obtain missing reference data, construction of citation graph and experiments with reference model based on graph convolutional neural networks. Section 6 contains the experimental results for classical methods and transformer models (BioMed RoBERTa, SapBERT, BioMed BERT), analysis of hyperparameter influence, comparison of results, practical application and summary.

We assume that the reader has completed a basic course in machine learning. In particular, the reader should understand: different data splitting strategies, dividing into train/validation/test set, cross-validation, stratified sampling, concept of generalization, machine learning algorithms such as logistic regression, random forest, basic evaluation metrics (accuracy, precision, recall, F1-score). In addition, the reader should be familiar with deep learning, at least multi-layer perceptron, activation functions, loss functions, stochastic gradient descent and its variants, the role of the learning rate and weight decay.

2 Methods of text classification

2.1 Classical methods

Text data differ from numerical or image and need specialized NLP techniques for proper preprocessing. Traditional models typically rely on manually crafted features to represent the data, which are then fed into classical machine learning algorithms for classification. Therefore, the performance of these algorithms heavily relies on the feature engineering.

BOW

One of the most common approaches to represent text is the Bag-of-Words (BOW) model [9]. In BOW, each document is represented by a set of words (tokens), ignoring the order. For the corpus of N documents and dictionary C, the matrix $X \in \mathbb{R}^{N \times |C|}$ called Document-Term Matrix, is constructed. Each row in the matrix represents the document and columns correspond to words, each entry indicates the count of a term in a document.

TF-IDF

Another widely used representation is Term Frequency-Inverse Document Frequency [10]. This method is a weighted modification of BOW. TF-IDF is a measure of the significance of the term t in a document d in the context of whole corpus D. Mathematically it is defined as a product of two factors, TF and IDF.

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D) \tag{1}$$

$\text{TF}(t, d)$ is the count of a term t in a document d and

$$\text{IDF}(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|},$$

which is the log ratio of the total number of documents in corpus D to the number of documents containing term t. The rarer a word is in the corpus, the higher its IDF value. Frequent words like "a", "the", "and" get IDF close to 0. There are also different variants of TF and IDF.

BOW and TF-IDF can be extended with the usage of n-grams [11], which creates features as a sequence of words rather than one word.

2.2 Transformer

This section is based on the concepts presented in the paper “Attention Is All You Need” [4].

Moving on from classical methods, lately deep learning approaches, such as recurrent neural networks (RNNs) and transformers, have become popular for text classification. Compared to the sequence models (RNN, LSTM) that were consuming input word-by-word sequentially and had problems with long-distance dependencies, transformers allow parallel processing of input sequence, with the key aspect of the architecture being the usage of the attention mechanism and the *multi-head self-attention* that runs several attention layers in parallel, providing useful and rich representations of words.

The transformer architecture consists of two parts: encoder and decoder. The encoder is here for processing the input and creating continuous representations which then are fed into the decoder that outputs a sequence of final symbols autoregressively one element at a time. Both parts depend on the usage of self-attention, which allows the model to capture dependencies between words.

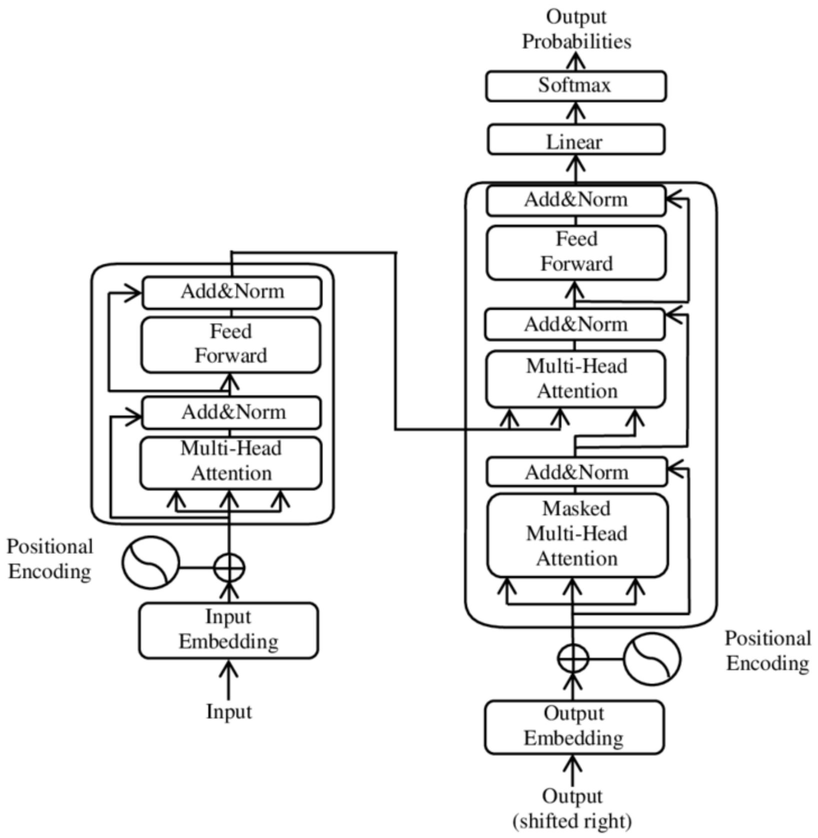


Figure 2: Transformer model architecture [12].

Tokenization

First of all, if we want to work with a transformer architecture, our text must be processed in some way. In the case of classical algorithms, we used bag-of-words text representations, while for neural networks, we need to convert the text into vector embeddings.

The text will first be tokenized into a sequence of words or sub-words from a fixed dictionary C. These tokens will be assigned their vector representation in a latent space. More precisely, given a sequence of tokens $\{w_1, w_2, \dots, w_n\}$. Each token w_i is mapped to an embedding vector

$$x_i = \text{Embed}(w_i) \in \mathbb{R}^d,$$

which can be written as the application of the embedding matrix $E \in \mathbb{R}^{d\times|C|}$ to the one-hot vector

$$x_i = E(\textit{one-hot}(w_i)).$$

The sequence of the obtained embeddings is then collected into a matrix

$$X = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^{n \times d}. \tag{2}$$

Positional Encoding

To take into account the position of the token in the sequence, we add a position vector $p_i \in \mathbb{R}^d$ to the embedding of x_i. In the original paper [4], they used sine-cosine patterns

$$\begin{aligned} p_{i,2k} &= \sin\left(\frac{i}{10000^{2k/d}}\right), \\ p_{i,2k+1} &= \cos\left(\frac{i}{10000^{2k/d}}\right), \end{aligned}$$

where $k = 0, 1, \dots, d/2 - 1$ and $p_{i,j}$ denotes the j-th coordinate of the position vector p_i. The matrix of embeddings augmented this way is then

$$Z = X + P, \text{ where } P = [p_1, p_2, \dots, p_n]^T. \tag{3}$$

2.2.1 Encoder

An encoder comprises N identical layers that are built from two specific sub-layers:

1. Multi-Head Attention (MHA),
2. Feed Forward Neural Network (FFN).

For each of these sub-layers, we add at the end residual connections and layer normalization. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is one of the two previously mentioned.

Scaled dot-product attention

The attention mechanism enables the model to create better word representations based on other words in the input sequence. Each word is represented according to its context by learning which other words are most relevant to it.

For each token we compute three vectors: a query, a key and a value. We obtain these by multiplying augmented input by $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_{\text{model}}}$, which are learned matrices during the training. These vectors typically has the same dimension d_{model}, but they can also be projected to smaller dimensions d_k and d_v. We define:

$$\begin{aligned} Q &= ZW^Q, \\ K &= ZW^K, \\ V &= ZW^V. \end{aligned}$$

To determine how much each word should attend to every other word, we measure the compatibility between queries and keys using the dot product (QK^T). The resulting scores are normalized with a softmax function. The new word representations are calculated as a weighted sum of values, using the softmax weights. In other words, the model asks (query) and looks at every word in the sequence. The words that are connected to the query are given more weight for the final representation.

The whole procedure can be performed with simple matrix multiplications and softmax.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4}$$

with the addition of a scaling factor $\frac{1}{\sqrt{d_k}}$ to prevent the gradients from being too small.

The softmax is applied row-wise

$$\text{softmax}\left(\begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} \end{bmatrix}\right) = \begin{bmatrix} \frac{e^{x_{11}}}{\sum_{j=1}^n e^{x_{1j}}} & \cdots & \frac{e^{x_{1n}}}{\sum_{j=1}^n e^{x_{1j}}} \\ \vdots & \ddots & \vdots \\ \frac{e^{x_{n1}}}{\sum_{j=1}^n e^{x_{nj}}} & \cdots & \frac{e^{x_{nn}}}{\sum_{j=1}^n e^{x_{nj}}} \end{bmatrix}.$$

Multi-head Attention

Instead of using just one attention mechanism, the model uses multiple heads (h heads). Each head i has its own learned linear projections W_i^Q, W_i^K, W_i^V for the queries, keys and values respectively. The whole Multi-head Attention can be described as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{5}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \quad i = 1, \ldots, h.$$

The projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.
In the original paper [4], $h = 8$, $d_k = d_v = d_{\text{model}}/h = 64$.

FFN

Feed Forward Neural Network is just two linear transformations with ReLU activation in between.

$$\text{FFN}(X) = \text{ReLU}\big(XW_1 + \mathbf{1}\, b_1^\top\big)\, W_2 + \mathbf{1}\, b_2^\top, \tag{6}$$

where $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}, b_1 \in \mathbb{R}^{d_{\text{ff}}}, W_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}, b_2 \in \mathbb{R}^{d_{\text{model}}}, \mathbf{1} \in \mathbb{R}^n$.

Encoder summary

To sum up, the encoder maps a sequence of token embeddings into contextualized representations using a stack of identical layers. The process consists of the following steps:

1. **Input:**

$$\mathbf{Z}^{(0)} = \big[\text{Embed}(w_1) + p_1,\ \ldots,\ \text{Embed}(w_n) + p_n\big]^T \in \mathbb{R}^{n \times d_{\text{model}}}.$$

2. **Stack of N Encoder Layers:**
   For $i = 1, 2, \ldots, N$:

$$\begin{aligned} \tilde{\mathbf{Z}}^{(i)} &= \text{LayerNorm}\big(\mathbf{Z}^{(i-1)} + \text{MHA}(\mathbf{Z}^{(i-1)})\big), \\ \mathbf{Z}^{(i)} &= \text{LayerNorm}\big(\tilde{\mathbf{Z}}^{(i)} + \text{FFN}(\tilde{\mathbf{Z}}^{(i)})\big). \end{aligned}$$

3. **Output:** $\mathbf{Z}^{(N)} \in \mathbb{R}^{n \times d_{\text{model}}}$.

2.2.2 Decoder

Decoder, similarly to the encoder is also built from N identical layers. Each of them has 3 sub-layers: Masked Multi-Head Attention, encoder-decoder Multi-Head Attention, FFN. We already know MHA, the above have some minor tweaks. Like before, after every sub-layer we add residual connections and layer normalization.

Masked MHA

The first sublayer of a decoder layer is Masked MHA. The decoder generates output tokens autoregressively, that is why we must mask the future. Specifically, generating the i-th token the model cannot attend to tokens from subsequent positions $j > i$. It means that we can only generate new tokens based on the previous known outputs.

$$\text{MaskedAttention}(Q, K, V, M) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V. \tag{7}$$

M is masking matrix with entries 0 for accessible positions and $-\infty$ for illegal future positions. This allows to give zero attention weights to the future outputs.

$$M_{ij} = \begin{cases} 0, & j \leq i, \\ -\infty, & j > i. \end{cases}$$

Encoder-decoder MHA

In encoder-decoder MHA the queries come from previous Masked MHA sub-layer and the keys and values come from the output of the encoder. The keys and values represent the context of the whole original input sentence. The decoder generates a query from the output and tries to find words that influence the question the most, based on that the output is generated.

Probability outputs

After passing the output through all N decoder layers, we obtain contextualized representation for each token in the output sequence. To generate next tokens, we apply a linear layer that maps it to the vocabulary dimension. Then we use softmax to get the probability distribution over all possible words. That way a single token is generated, it is appended to the existing output and next token is generated from the completed sequence.

2.3 BERT

This section is based on the concepts presented in the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [8].

BERT (Bidirectional Encoder Representations from Transformers) model, as the name suggests, uses only the encoder part from transformer architecture. It consists of N encoding layers. Unlike previously in the transformer's decoder we had unidirectional context (masking) when generating tokens, BERT is trained in a bidirectional way, taking context from left and right of each token. It is important to note that BERT does not have a decoder. During training, it uses prediction head on top of encoding layers, tailored to the specific task.

Pre-training

The training of BERT takes place in two steps, one of them is pre-training, where the model is trained on two unsupervised tasks. The objective of the pre-training is to familiarize the model with the language itself from large text corpora like English Wikipedia (2500M words).

Task 1 – Masked Language Modelling (MLM)

In order to take advantage of bidirectionality, the training is constructed by masking at random some percentage of the input sentence and predicting the missing words. Masked words are replaced by special token `[MASK]` but not always. Since the special token `[MASK]` does not appear in the second learning task and fine-tuning, the masking procedure can be explained by the following example.

- 80% of the time: Replace the word with the `[MASK]` token.
  I love green apples $\rightarrow$ I love green `[MASK]`.

- 10% of the time: Replace the word with a random word.
  I love green apples → I love green dog.
- 10% of the time: Keep the word unchanged.
  I love green apples → I love green apples.

Task 2 – Next Sentence Prediction (NSP)

The model receives two sentences: A, B and needs to decide whether the sentence B is the actual next sentence after A in the original text. The training is constructed so that 50% of time B is the next sentence and 50% of time B is a random sentence from the corpus. Through training, the model learns the relationship between sentences.

Fine-tuning

With general knowledge of the language, the model can be adapted to a specific task through a process called fine-tuning. The model can be trained on various downstream tasks:

- Text classification – sentiment analysis, spam detection.
- Token classification – named entity recognition.
- Question answering – extracting the answer span from a given context.
- Semantic similarity – comparing sentence embeddings for retrieval or clustering.

In each of those tasks, we use the same model with pre-trained parameters and switch the prediction head suited to the specific task.

3 Datasets acquisition and preparation

AmylogGraphAB dataset

AmyloGraphAB is an ongoing database of studies reporting the effects of antibodies on amyloid formation. Amyloids are fibrous protein aggregates involved in both biological functions and disease. Their β-sheet structure allows them to assemble into long unbranched fibrils. Accumulation of these fibrils in tissues and organs may impair normal function, leading to diseases known as amyloidosis.

The database is being curated under the guidance of Valentín Iglesias, Mariia Solovianova and Ronja Titel from the Medical University of Białystok, with the goal of collecting and systematizing data on the interaction between amyloids and antibodies.

The database was created by initially searching the PubMed database for articles using an appropriate query. The following search strings were used:

- `"amyloid"[Title/Abstract] AND "antibod*"[Title/Abstract]`,
- `"amyloid"[Title/Abstract] AND "nanobod*"[Title/Abstract]`.

Among the articles found, each was manually assessed and classified as useful or irrelevant, based on the following criteria.

Inclusion criteria:

1. Studies that report the effect of antibodies on amyloid formation.
2. Experimental data on amyloid fibrillation (AFM, PET, ThT, TEM).[1]

Exclusion criteria:

1. Pre-prints.
2. Duplicates (already included papers).
3. Non-English papers.
4. Reviews, commentaries, perspectives, editorials.

At the initial processing stage, the input data were in the form of PMID (PubMed ID) identifiers together with the classification decision (Useful or Rejected). For each paper, we fetched from the Entrez database:

[1]Detailed descriptions of these methods can be found under *Experimental methods for studying amyloid cross-interactions* [13]

- title,
- abstract,
- journal,
- references.

30% of the references was not available through the Entrez database, the missing references were retrieved from OpenAlex database.

During the experiment, we had **1853** labeled articles. **150** (**8%**) of them were labeled as positive (useful). This is a relatively small dataset, which could lead to overfitting to the training set and poor generalization to new data, as the model sees a small number of positive examples. The limited sample size also results in high variance across different data splits and model initializations.

4 Methodology

Every evaluated neural network based model was registered in Weights & Biases (wandb) project, where we can access various training artifacts such as history of training loss, score metrics, hyperparameters.

4.1 Data splitting and validation

For model evaluation, cross-validation with a hold-out test set was used. Hold-out test set is a part of data that the model does not see during the training. It serves as a final unbiased estimator of the model, like in a typical real-world scenario. The data were split into 85% train set and 15% hold-out test set. On the train data 5-fold cross-validation was performed. Cross-validation is not a typical way to train neural networks, but with a high variance dataset with only 8% positive articles among 1853 observations, a more stable method was necessary. Every splitting method used a stratified version to maintain the same class distribution. In each of the 5 splits, the data were divided into training and validation sets. The model was also evaluated on the hold-out test set. As a result, we obtained 5 validation results and 5 test results, providing a more reliable estimate of the model's performance.

4.2 Model training process

Models

We assume it is better to use models that were pre-trained on medical domain texts. These models usually give a better starting point than BERTs pre-trained on Wikipedia. The ones that are considered in this work:

1. `microsoft/BiomedNLP-BiomedBERT-base-uncased-abstract`: This model was pre-trained from scratch on abstracts from PubMed [5].

2. `cambridgeltl/SapBERT-from-PubMedBERT-fulltext`: The SapBERT model was developed by further pre-training BioMed BERT to recognize synonyms of the same entity using UMLS (Unified Medical Language System), a database of medical entities and their synonyms [6].

3. `allenai/biomed_roberta_base`: This model is based on RoBERTa-base [14], a modification of BERT with more robust pre-training precedure on more data. The model was adapted to the biomedical domain [7], pre-trained on The Semantic Scholar Open Research Corpus (S2ORC) [15].

Learning rate

With 110M parameters in a $\text{BERT}_{\text{base}}$ [4] a learning rate has to be carefully chosen not to disrupt pre-trained model. Typically used learning rates for fine-tuning are values from `1e-5` to `5e-5`. In this work we will test: `5e-6, 1e-5, 2e-5, 3e-5`.

Max length

This hyperparameter defines the length of the input sequence to the model after tokenization. In BERT, the context window is limited to 512 tokens. For every longer sequence, the text is truncated. For shorter sequences, the input is padded, missing tokens are `[PAD]` tokens.

In our work the abstracts are on average 230 words long, the longest having 796 words. Sometimes the whole word does not equal to 1 token, the word can be split into 2 tokens, we can accept that on average 1 word $\approx \frac{4}{3}$ tokens. Based on that, we will use mostly `max_length=350` and test `max_length=512` to capture most of the text.

Unfrozen layers

This setting controls the number of layers that are trainable during the fine-tuning. The BERT consists of 12 encoder layers. During the fine-tuning, all of the encoder layers are frozen, then a chosen number of the last layers are unfrozen. For small values, the parameters from the original model are mostly preserved and we train the classifier head. With `Unfreeze_last_k=12` all parameters are updated.

5 Other approaches

This section describes alternative methods that were tried in parallel with the main approach. It presents the attempts to obtain missing reference data, construction of citation graph and experiments with reference model based on graph convolutional neural networks [16].

Initial problem

In the data fetched from Entrez API about 30% of the observations were missing references (PubMed does not provide references). Because GCN uses references to build the graph, we tried to get the missing data.

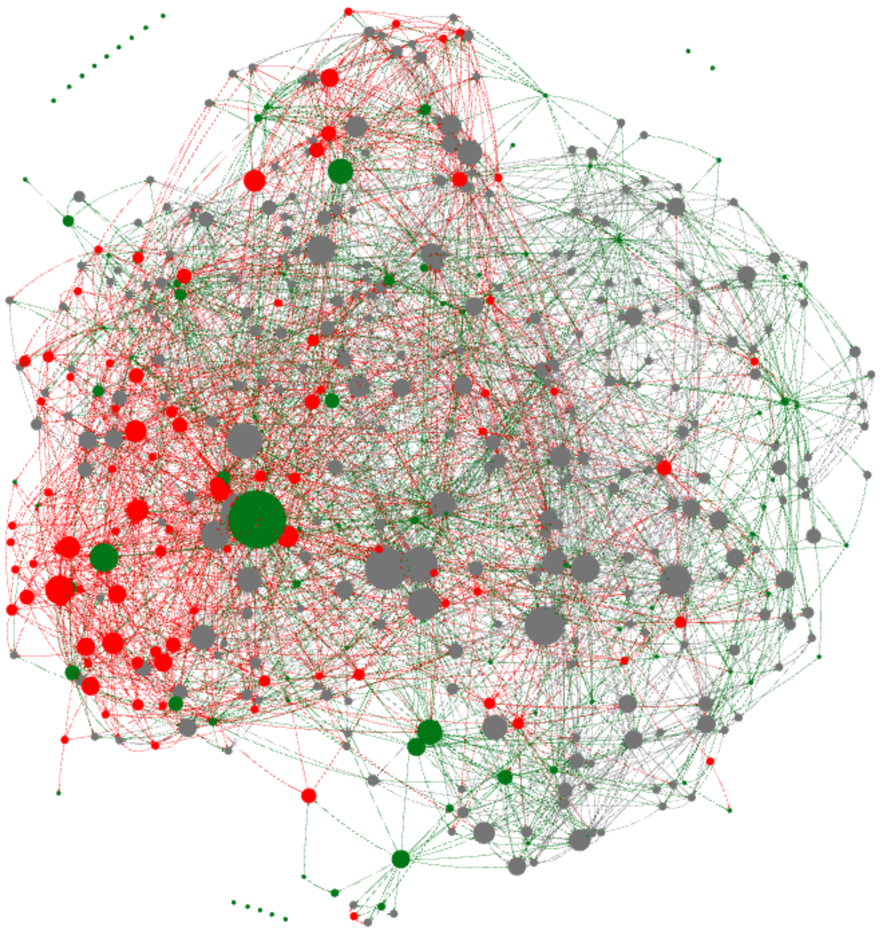
Alternative sources:

- Scopus – At first, we noticed that the missing data were available in the Scopus database. However, the Scopus API does not allow fetching references, so this source of information was rejected.
- OpenAlex – We used OpenAlex as an alternative and we obtained most of the missing data. The challenge with OpenAlex was that it was using

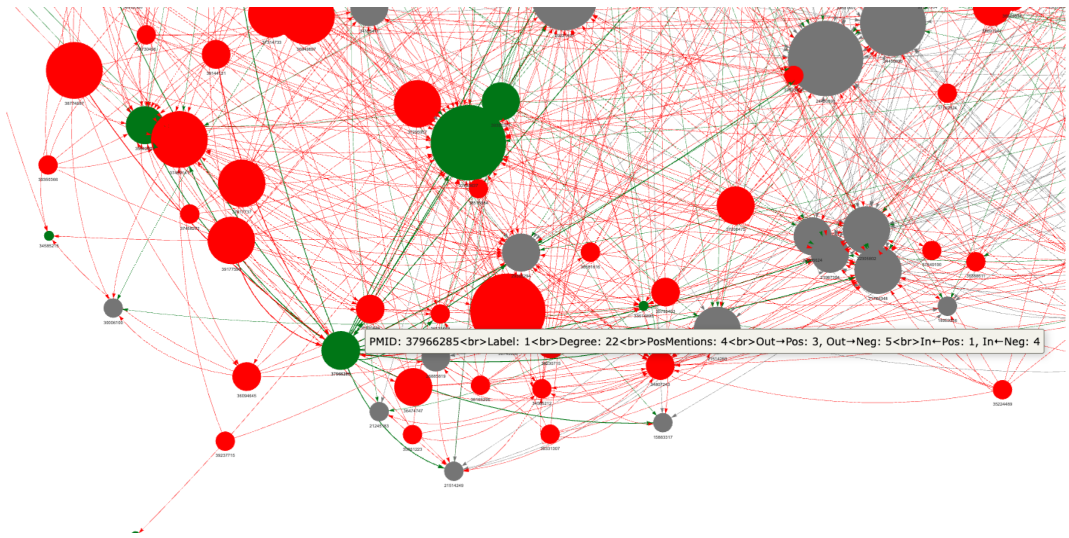
OpenAlex ID in references, so for all of the references we needed to once again use OpenAlex API to convert OpenAlex IDs into PMIDs. After this step, around 3% of the records lacked references. Those articles were not included in the later experiment.

Citation graph

Since the number of positively assessed articles was so small, we assumed that positive papers were more likely to cite other positives. We decided to create a citation graph that includes only documents referencing at least two articles that were previously labeled as positive.



(a) Citation graph.



PMID: 37966285
Label: 1
Degree: 22
PosMentions: 4
Out→Pos: 3, Out→Neg: 5
In←Pos: 1, In←Neg: 4

(b) Zoomed citation graph with interactive data.

Figure 3: Citation graph and zoomed section.

There are green nodes (positively labeled articles), red nodes (negatives) and gray nodes (unlabeled articles that are cited by labeled nodes). The size of a node depends on a number of positive references, the more an article has positively labeled articles in references, the larger the node. The zoomed part in Figure 3b displays information that appears after clicking on a node. For a node we get information about: PMID, label, degree, citation data.

5.1 Reference model

We tried using Graph Convolutional Network (GCN) [16] as an extension of the BERT model. GCN takes the input in the form of a graph, where nodes are articles (specifically: `[CLS]` embedding of the abstract reduced to 100 dimensions for memory efficiency) and edges are citation relations (citing $\rightarrow$ cited). During forward pass, each GCN layer updates every node by first aggregating the information from its neighbors and then applying a linear transformation with activation function. This setup enables the model to capture not only textual information, but add reference information from the neighbors.

While testing on a 60/20/20 train/validation/test split we did not see any improvements. The results on a validation set were similar to the BERT model but performance on the test set dropped.

6 Results

In the experiment we monitored standard evaluation metrics: accuracy, precision, recall, F1-score. The selection of the best model was based on the highest mean F1-score achieved on the validation dataset, however we also took into account the results of recall, since for researchers the goal was to identify as many relevant articles (true positives). Consequently, among models with comparable F1-score, those with higher recall were preferred.

6.1 Classical methods

For traditional machine learning methods, alongside different text representation, we tried various preprocessing techniques. We tried no preprocessing, processing that involved converting text to lowercase, removing special characters, digits, URLs, emails and extra whitespaces. Additionally, we applied lemmatization, which reduces words to their base forms.

Table 1: Classical models results.

Model	**Val F1**	**Std**	**Test F1**
Logistic Regression preprocess + lemmatize, TF-IDF	**0.49**	0.06	0.45
Logistic Regression no preprocessing, TF-IDF	0.48	0.05	0.43
Logistic Regression preprocess, TF-IDF	0.48	0.04	0.42
Logistic Regression preprocess + lemmatize, DTM	0.44	0.08	0.38
Random Forest preprocess, TF-IDF	0.43	0.02	0.41
Random Forest preprocess + lemmatize, TF-IDF	0.41	0.05	0.49

We found that the best results were achieved by logistic regression with TF-IDF representation of preprocessed, lemmatized text. Random Forest did not meet the expectations, but somehow performed the best on the test set. Logistic regression is interpretable directly through its coefficients; a positive coefficient increases the probability of the positive class. During training, we found that there are a few words strongly associated with a positive class.

Positive words:

- **scFv** (*single-chain variable fragment*),
- **mAb** (*monoclonal antibody*),
- **TTR** (*transthyretin*) – connected to the TTR amyloidosis,
- **hiAPP** (*human islet amyloid polypeptide*).

Also words like **a**β, **aggregate** and **antibody** had high positive coefficient. This finding gave a quick insight into which words we should pay attention to.

6.2 Language models

BioMed RoBERTa

Presented results come from fine-tuning the `allenai/biomed_roberta_base` model. The reported models are the best performing among different hyperparameter configurations. All models were trained with `max_length = 350`, updating the last six layers. For models with title added to abstract `max_length = 370` was set.

Table 2: BioMed RoBERTa results.

Model	**Val F1**	**Std**	**Test F1**	**Std**
lr: 2e-5, abstract	0.50	0.05	0.56	0.03
lr: 3e-5, abstract	0.50	0.05	0.56	0.05
lr: 2e-5, title+journal+abstract	0.45	0.04	0.52	0.03

Using only abstracts, the best hyperparameter configuration with a learning rate of `2e-5` achieved a mean F1-score of 0.50 on the validation sets (standard deviation = 0.05) and an F1-score of 0.56 on the test set (standard deviation = 0.03). Usually we should be happy when the results on the test set are better than on the validation set. It appears that other hyperparameter choices did not improve the performance and adding additional information such as title and journal name worsened the performance.
BioMed RoBERTa was originally trained on 2.68M biomedical full-text papers from S2ORC [7, 15]. These texts differ from PubMed abstracts, which are present in the fine-tuning, which might be the reason why this model has lower performance than the others that are pre-trained on PubMed texts.

SapBERT

Table 3: SapBERT results.

Model	**Val F1**	**Std**	**Test F1**	**Std**
lr: 3e-5, unfreeze: 6 seq_len: 350, abstract	0.55	0.06	0.45	0.03
lr: 2e-5, unfreeze: 12 seq_len: 512, title+journal+abstract	0.54	0.04	0.55	0.05
lr: 2e-5, unfreeze: 12 seq_len: 350, abstract	0.54	0.07	0.48	0.06

The SapBERT model was developed by further pre-training BioMed BERT, which was pre-trained on PubMed abstracts. That data match the data in our experiment and we immediately can see much higher F1-scores on validation sets. The best performing model achieves mean 0.55 F1-score on validation set and 0.45 on the test set. The best results are obtained with learning rate `3e-5` or `2e-5`. The second model operates on 512 token length input and has added title to the abstract, we can observe that it performs much better on the test set compared to other models. However, this might be random, since adding title information previously did not improve the score.

BioMed BERT

Table 4: BioMed BERT results.

Model	**Val F1**	**Std**	**Test F1**	**Std**
lr: 3e-5, unfreeze: 12, seq_len: 370, weight_decay: 0.1, title+journal+abstract	0.59	0.04	0.61	0.03
lr: 3e-5, unfreeze: 12 seq_len: 350, abstract	0.58	0.07	0.56	0.05
lr: 3e-5, unfreeze: 6 seq_len: 350, abstract	0.57	0.07	0.55	0.07

Using title, journal and abstract as input, the best hyperparameter configuration with a learning rate of `3e-5` and all 12 layers unfrozen achieved a mean F1-score of 0.59 on the validation sets (standard deviation = 0.04) and an F1-score of 0.61 on the test set (standard deviation = 0.03). Greater performance on the test set compared to the validation set is a positive indication of effective generalization.

Hyperparameters influence

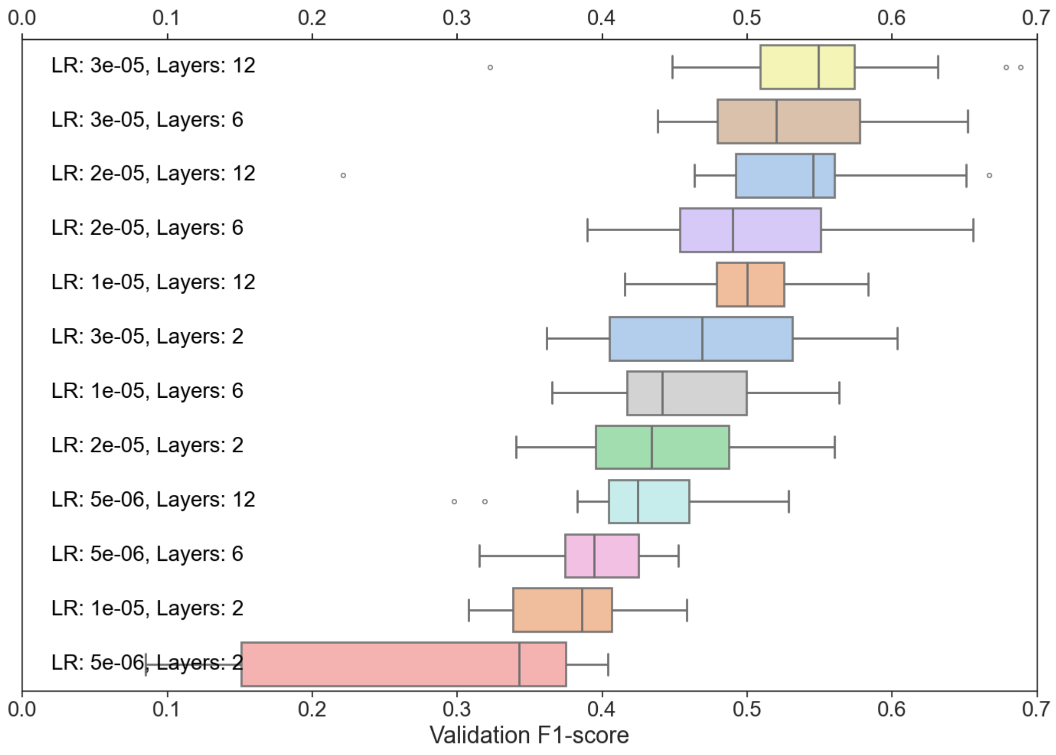


Figure 4: Validation F1-score results for different learning rates and unfrozen layers configuration for neural network models.

During the experiment, we mainly investigated the influence of learning rate, the number of unfrozen layers and the length of the input sequence. Increasing the `max_length` parameter from 350 to 512 did not show any improvements on validation data, but 2 out of 3 showed models achieved better results on the test set. Influence of learning rate with a number of layers significantly impacted performance of the models. As shown in 4, generally higher learning rates combined with greater number of trainable layers led to better results. Intuitively, we can assume that the lower layers of the encoder are responsible for word representations and dependencies between words, while higher are capturing the overall context of the sentence. If we unfreeze only higher layers, we are not fully fitting to the specific task. With more layers to train, we are adapting model to the domain.

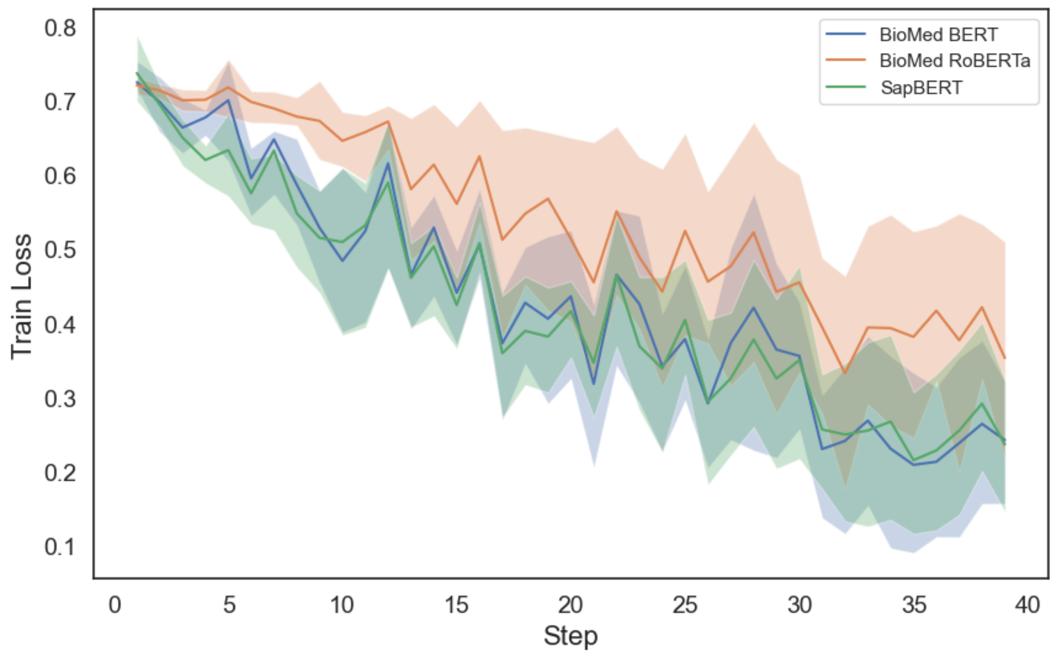


Figure 5: Train loss with Q1-Q3 Shading

We can also inspect the training history of each run across the three models. From the analysis, it is evident that BioMed RoBERTa is the worst, showing slower improvement during training and lower final scores compared to other models.

6.3 Comparison

Table 5

Model	Val Precision	Val Recall	Val F1	Test F1
BioMed BERT	0.54	0.65	**0.59**	**0.61**
SapBERT	0.51	0.60	0.54	0.55
BioMed RoBERTa	0.41	0.68	0.50	0.56
Logistic Regression	0.41	0.61	0.49	0.45
Random Forest	0.42	0.43	0.43	0.41

We evaluated classical machine learning methods (logistic regression and random forest) with different text representations, as well as several language models (BioMed BERT, SapBERT, BioMed RoBERTa). Logistic regression performed reasonably well, given the small dataset. This model is a good baseline and can be quickly implemented and deployed into production. It also offers high interpretability. One concern with classical methods is that they achieved poor performance on a test set, indicating limited generalization to unseen data.

The best perfmorming model turned out to be a BioMed BERT pre-trained on PubMed abstracts, achieving on a validation set an F1-score of 0.59 and 0.61 on a test set. All models have higher recall than precision, which aligns with the goal of finding as many positive examples as possible.

6.4 Other dataset results

The same experiment can be conducted on another dataset. One of them focuses on prediction tools for peptide activity [17]. The prepared dataset contains 256 labeled articles. 58 (23%) of them are labeled as positive. Records were assessed as positive, when the paper presented new tool or model for prediction of the peptide activity.

Table 6

Model	**Val Precision**	**Val Recall**	**Val F1**	**Test F1**
BioMed BERT	0.80	0.80	0.80	0.65
SapBERT	0.82	0.90	**0.85**	0.65
BioMed RoBERTa	0.75	0.8	0.75	**0.70**
Logistic Regression	0.66	0.71	0.69	0.63

This data immediately turns out to be much easier to classify as we see much better results. Logistic regression performed comparably to the language models on a test set. Looking at the validation set, language models outperformed logistic regression. The best results on the validation set were achieved by SapBERT, however on the test set the highest F1-score was obtained with BioMed RoBERTa. The difference between validation and test results means that the models overfitted to the validation data. It is not surprising, since the dataset contains only 256 records.

6.5 Practical application

Before conducting the experiment, with smaller dataset consisting of 1567 negatives and 117 positives (only 7% of positives), we trained a SapBERT model to help identifying new positives. The model achieved precision of 0.38, recall of 0.74 and F1-score of 0.5 on a validation set. We then generated predictions on 1477 unlabeled records and identified 201 positives. After manual assessment, 50 of them were confirmed as true positives, resulting in a real-world test precision of 0.25. Previously, the ratio of positives to negatives was approximately **1:13**. Thanks to SapBERT, we improved this ratio to **1:3**.

With new data (1747 negatives, 167 positives), we fine-tuned BioMed BERT that on the validation set achieved an F1-score of 0.67. We generated predictions on the same unlabeled dataset (without 201 identified articles by SapBERT) and this time we got 12 positives. Six of them turned out to be true positives, resulting in a ratio **1:1**. We should keep in mind that the previous model was trained with high recall to detect as many positives, so the new model might have had an easier job. The model also predicted new positives that the previous model did not.

6.6 Summary

By prioritizing potentially relevant publications, the approach can reduce manual curation effort and accelerate the discovery of meaningful research papers. Manual literature review and searching for works that are of interest to us can be greatly facilitated by using smart classification tools. We noticed that even a simple classifier based on logistic regression can speed up manual data curation. Using modern classifiers based on language models, we are able to speed up the finding of positives by 350%. The tool that has been implemented can be used as a filtering device or as a better search engine. We only need some data at the beginning to train the model. The proposed procedure for searching new relevant works would be to use the language model from the beginning and retrain it as the labeled data grow.

References

[1] Rita González-Márquez, Luca Schmidt, Benjamin M Schmidt, Philipp Berens, and Dmitry Kobak. The landscape of biomedical research. *Patterns*, 5(6), 2024.

[2] Mark A Hanson, Pablo Gómez Barreiro, Paolo Crosetto, and Dan Brockington. The strain on scientific publishing. *Quantitative Science Studies*, 5(4):823–843, 2024.

[3] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. A survey on text classification: From shallow to deep learning. *arXiv preprint arXiv:2008.00364*, 2020.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[5] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23, 2021.

[6] Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. Self-alignment pretraining for biomedical entity representations. *arXiv preprint arXiv:2010.11784*, 2020.

[7] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[9] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[10] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[11] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, volume 161175, page 14. Las Vegas, NV, 1994.

[12] Yuening Jia. Attention mechanism in machine translation. In *Journal of physics: conference series*, volume 1314, page 012186. IOP Publishing, 2019.

[13] Aleksandra Kalitnik, Anna Lassota, Oliwia Polańska, Marlena Gąsior-Głogowska, Monika Szefczyk, Agnieszka Barbach, Jarosław Chilimoniuk, Izabela Jęśkowiak-Kossakowska, Alicja W Wojciechowska, Jakub W Wojciechowski, et al. Experimental methods for studying amyloid cross-interactions. *Protein Science*, 34(6):e70151, 2025.

[14] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[15] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S Weld. S2orc: The semantic scholar open research corpus. *arXiv preprint arXiv:1911.02782*, 2019.

[16] TN Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] Oriol Bárcenas, Carlos Pintado-Grima, Katarzyna Sidorczuk, Felix Teufel, Henrik Nielsen, Salvador Ventura, and Michał Burdukiewicz. The dynamic landscape of peptide activity prediction. *Computational and Structural Biotechnology Journal*, 20:6526–6533, 2022.