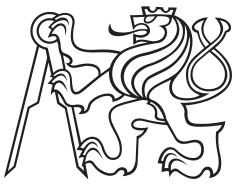
Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Multiple Instance Learning for Long Input NLP Models

Bc. Vojtěch Jeřábek

Supervisor: Ing. Jan Drchal, Ph.D.
Field of study: Open Informatics
Subfield: Data Science
May 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jeřábek** Jméno: **Vojtěch** Osobní číslo: **474470**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Datové vědy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Metody Multiple Instance Learning pro zpracování dlouhých vstupů modely NLP

Název diplomové práce anglicky:

Multiple Instance Learning for Long Input NLP Models

Pokyny pro vypracování:

Experiment with Multiple Instance Learning (MIL) methods aimed at processing long inputs of NLP models (> 1000 tokens). Focus on neural architectures such as Deep Sets and Set Transformers.
1) Explore the state-of-the-art MIL methods and methods related to Deep Sets and Set Transformers as well as modern neural text classifier architectures.
2) Select or develop an appropriate dataset(s) for textual classification.
3) Implement one or more MIL-based classifiers and assess their performance compared to the baseline non-MIL approach (such as the BERT classifier) - this would be possible for relatively short input texts, only.
4) Evaluate the best classifier architecture on Czech fact-checking datasets supplied by the supervisor.
5) Publish the code and data as open-source.

Seznam doporučené literatury:

[1] Zaheer, Manzil, et al. "Deep sets." Advances in neural information processing systems 30 (2017).
[2] Lee, Juho, et al. "Set transformer: A framework for attention-based permutation-invariant neural networks." International Conference on Machine Learning. PMLR, 2019.
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Minneapolis, Minnesota. Association for Computational Linguistics.
[4] Drchal, Jan, et al. "CsFEVER and CTKFacts: Czech Datasets for Fact Verification." arXiv preprint arXiv:2201.11115 (2022).

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Jan Drchal, Ph.D. centrum umělé inteligence FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **14.09.2022** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **19.02.2024**

Ing. Jan Drchal, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Acknowledgements

I would like to thank my supervisor Ing. Jan Drchal, Ph.D for the patient guidance, helpful consultations, and useful comments. He supported me throughout the whole course of writing the thesis and was always willing to help. In addition, I would like to thank my family and friends for their patience and constant support during my studies.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics" is also gratefully acknowledged.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 26. May 2023

Abstract

At present, models exploiting the Transformer architecture are reigning over challenges in the NLP field. These models, however, share a drawback - their quadratic computational complexity, which constrains the input size for the standard Transformer model between 512 and 1024 tokens, an amount inadequate for long inputs. The objective of our master's thesis is to employ Multiple Instance Learning techniques for processing long input Natural Language Processing models, while utilizing one of many pre-trained Transformer models. Multiple Instance Learning is a classification approach for instance sets. Our presented method relies on sequentially breaking down long inputs into shorter parts. Transformer-based language models process these shorter sections and collectively classify them using the MIL technique. In the experimental part, we tested the functionality of our model on three diverse datasets, two in English and one in Czech with input length up to 13,926 tokens. While the results revealed our model could handle even the longest inputs - a standard achievement language model-based methods fail to meet - we didn't observe a substantial contribution in enhanced accuracy compared to other models. In some instances, the results were even below the baseline. We discussed the results in more detail, described potential shortcomings, and proposed directions for further research.

Keywords: neural networks, natural language processing, multiple instance learning, classification, Transformers, long input processing

Supervisor: Ing. Jan Drchal, Ph.D.
Department of Computer Science,
Faculty of Electrical Engineering,
Czech Technical University in Prague

Abstrakt

V současných aplikacích v oblasti zpracování přirozeného jazyka převažují modely využívající Transformer architekturu, tyto modely spolu ovšem sdílí jednu nevýhodu. Tou je kvadratická výpočetní složitost, která omezuje velikost přijímaného vstupu pro standardní Transformer model na 512 až 1024 tokenů, což je v případě dlouhých vstupů nedostatečný počet. Cílem práce bylo využít metod Multiple Instance Learning pro zpracování dlouhých vstupů v modelech pro zpracování přirozeného jazyka. Díky tomu můžeme využívat široké škály již předtrénovaných Transformer modelů. Multiple Instance Learning je technika pro klasifikaci množin instancí. Naše metoda využívá sekvenčního zpracování dlouhého vstupu na kratší části. Tyto kratší části jsou zpracovány jazykovými modely založenými na Transformer architektuře a následně společně klasifikovány za použití Multiple Instance Learning techniky. V experimentální části jsme funkčnost modelu ověřili na 3 různých datových sadách, 2 v jazyce anglickém a 1 v českém, s nejdelším vstupem o velikosti 13926 tokenů. Ačkoliv výsledky ukázaly, že navržený model je schopen zpracovat i ty nejdelší vstupy, což standardní metody využívající jazykové modely nezvládly, tak nebyl zjištěn výrazný přínos v oblasti zvýšené přesnosti v porovnání s jinými modely. V některých případech byly dokonce výsledky pod úrovní standardu. Výsledky jsme blíže diskutovali, popsali možné nedostatky a navrhli případné směry pro další výzkum.

Klíčová slova: neuronové sítě, zpracování přirozeného jazyka, multiple instance learning, klasifikace, Transformers, zpracování dlouhého vstupu

Překlad názvu: Metody Multiple Instance Learning pro zpracování dlouhých vstupů modely NLP

Contents

Figures

Tables

Chapter 1

Introduction

Natural Language Processing (NLP) is a subfield of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. NLP has been an area of interest and research for several decades, with the earliest work dating back to the 1950s and 1960s [Hut04]. Since then, it has been an active area of research, with applications ranging from sentiment analysis to machine translation. However, new challenges arise as the demand for more sophisticated NLP models increases. One of those challenges is in handling long input texts.

Progress in NLP's ability brings new possibilities for its application. This work is part of a larger effort under Artificial Intelligence Centre (AIC) at Czech Technical University in Prague (CTU) to provide a solution for automated fact-checking. The spread of fake news during covid-19 pandemic showed us the need for a solution to counter this emerging threat. The fact-checking task can be generalised as a text classification task. The current state-of-the-art solution for such a task are Transformer models. However, when facing a very long input, these models are limited by quadratic computational complexity and memory requirements. The objective of this thesis is to investigate the effectiveness and ability of Multiple Instance Learning (MIL) to enhance the performance of long input NLP models. Specifically, we focus on the use of MIL for text classification tasks, where the input text is often lengthy and complex.

Multiple Instance Learning is a type of weakly supervised machine learning where the training data is composed of bags, each containing multiple instances. The goal is to classify the bag based on the properties of the instances it contains rather than the properties of individual instances. This makes MIL particularly suitable for NLP tasks involving long input text, where the MIL aggregation function summarises long inputs or where labelling each instance would be time-consuming.

The remainder of this thesis is organized as follows. In Chapter 2, we comprehensively explore key concepts underpinning the thesis's research approach. It delves into Multiple Instance Learning (MIL) [DLLP97] as a weakly supervised learning technique. The chapter then transitions into Natural Language Processing (NLP) [NOMC11], introducing the field, its development and its applications. Then focus on the Transformer model [VSP+17], its implementation in BERT [DCLT18], its adaptations for long sequence handling, and its adaptation in Set Transformer [LLK+19]. Lastly, it investigates Text Classification, linking its challenges and context to the research objectives.

Chapter 3 outlines the methodology designed to tackle the problem of processing long inputs in NLP models. It begins by addressing the issue and examining previous attempts. The chapter then presents the proposed solution, a text classification model that combines MIL and Transformer-based models to manage long sequences. It also details the three diverse datasets used for model evaluation, their classification tasks, and their relevance to the research.

In Chapter 4, we report the experimental results and analyze the performance of the MIL-based NLP models compared to the non-MIL models.

The Discussion in 5, summarizes our findings, discusses their implications and proposes future work. Finally, in Chapter 6, we conclude the thesis.

Chapter 2

Related Work

In this chapter, we explore critical concepts that inform and shape the approach used in our thesis. We begin with Multiple Instance Learning (MIL), a form of weakly supervised learning that allows for more efficient handling of labelling of data, a technique central to the processing of long inputs in our thesis.

Next, we delve into the realm of Natural Language Processing (NLP), an essential field which enables us to process and interpret text data. The discussion then turns to the Transformer model and its variations. First, focusing on its initial implementation in the BERT model and then on its variations managing long input sequences, a challenge directly related to the objectives of our thesis. We also examine the Set Transformer, which introduces new methodologies for set handling, a concept employed in our proposed solution.

Finally, we explore Text Classification, the primary goal of our thesis, offering a detailed understanding of its context, challenges, and its connection to our research goals.

2.1 Multiple Instance Learning

Multiple Instance Learning (MIL) is a form of supervised machine learning that offers a novel approach to classification problems. Introduced by Dietterich et al. in 1997 [DLLP97], MIL differs from traditional supervised learning in its labelling approach.

Instead of assigning labels to individual instances, labels are associated with a set [1] (or bag) of instances. More specifically, MIL is a weakly supervised form due to having only bag-level labels. A bag is classified as positive if at least one instance in the bag is positive and negative only if all instances are negative. This is a response to cases where the labelling of each instance in training data is not possible or too expensive. Example of bags with various labels can be seen in Figure 2.1.

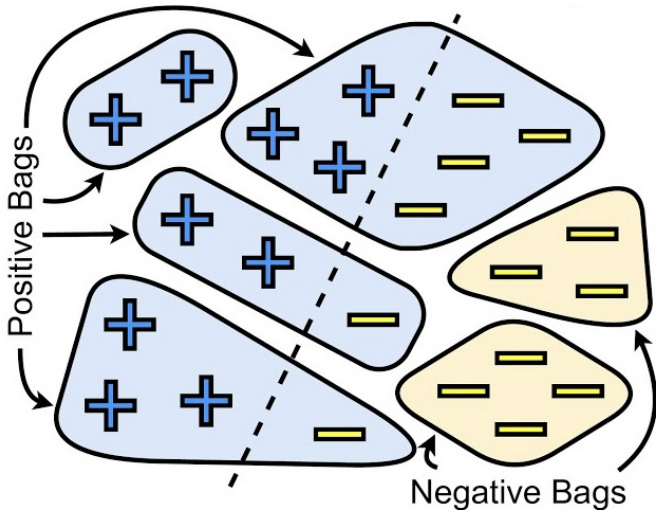


Figure 2.1: Visualization of multiple bags and their appropriate labels. + are positive instances and - are negative, note that bag level classification does misclassify some instances. Reprinted from [Min23].

This unique approach to classification has been applied to various domains, including computer vision [CZL+17], medical imaging [LXZ17], and text categorization [Zho04]. In computer vision, for instance, it has been utilized

[1] Throughout this thesis, we frequently refer to 'set'. However, it's important to note that it can technically be considered as 'multi-set'.

to identify objects within an image where the exact object location is unknown. In the context of text categorization, MIL has been applied to situations where only the document-level label is available but not the labels for individual sentences or phrases.

An important aspect of MIL is the choice of instance aggregation function, which determines how the bag-level representation is obtained from individual instances. These functions play a critical role in the performance of the MIL model.

Despite the strengths of MIL, there are certain trade-offs. The most notable one is that some information might be lost during the aggregation process due to the need to condense multiple instance representations into a single bag representation [Bab08].

In summary, Multiple Instance Learning provides a unique approach to handling tasks where ambiguity exists in the instance labels. Its flexibility makes it adaptable to various domains and tasks, and integrating it with deep learning techniques further enhances its potential.

2.1.1 Neural Multiple Instance Learning

The proposed approach in this work employs neural networks to address Multiple Instance Learning problems. Therefore, we must delve deeper into the structure of Neural Multiple Instance Learning models. The structure is visualized in Figure 2.2.

The first part of the architecture is a pre-processing layer. It is an instance-level neural network that transforms individual instances into a new feature space. This instance-level network is applied independently to each instance in a bag, essentially learning to extract meaningful features useful for the bag-level prediction task.

After the pre-processing layer has transformed all instances in a bag, an aggregation function is applied to these transformed instances to produce a

fixed-size bag-level representation. This function needs to be permutation-invariant, meaning that the order of the instances in a bag should not impact the resulting bag representation. Common choices for the aggregation function include the mean, max, or a learned function such as attention.

Finally, the bag-level representation is passed through post-processing. That is another neural network used to produce the final output, which is the prediction for the entire bag.

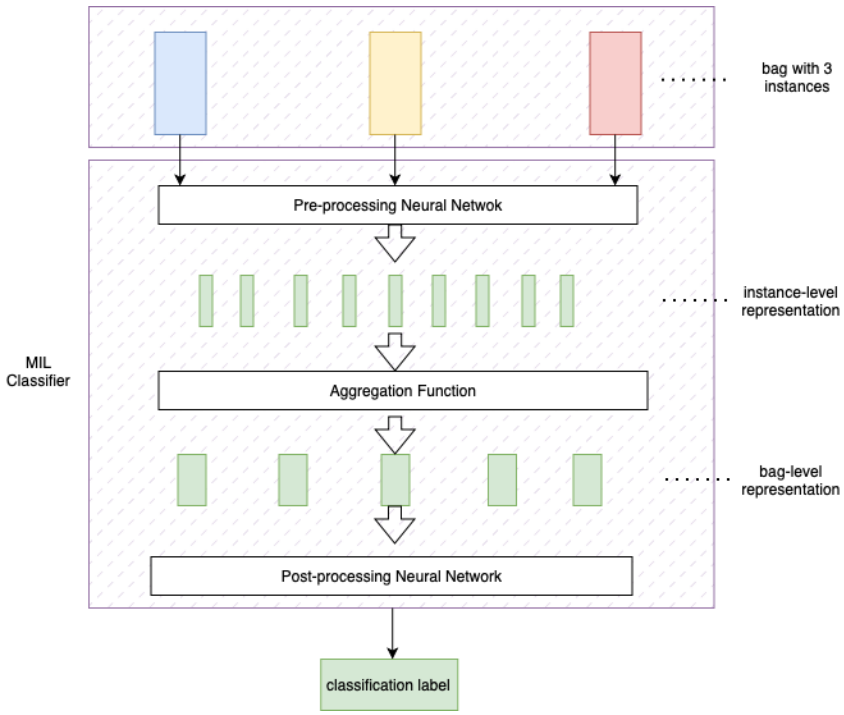


Figure 2.2: Visualization of described neural network Multiple Instance Learning model.

This architecture allows Neural MIL models to handle bags of varying sizes, learn to extract useful instance-level features, and make a prediction at the bag level, all of which are key requirements for multiple instance learning problems. Presented architecture is generalized approach inspired by [PS17].

2.1.2 Formal Notation for Multiple Instance Learning

In the context of Multiple Instance Learning, we consider a dataset $\mathcal{D}$ that consists of a set of bags. A single bag $B_i \in \mathcal{D}$, where $i = 1, \ldots, N$ and N is the total number of bags, is defined as a collection of instances $B_i = \{x_{i1}, x_{i2}, \ldots, x_{in_i}\}$, where x_{ij} denotes the j-th instance in the i-th bag and n_i is the number of instances in bag i. Each instance x_{ij} is a vector in $\mathbb{R}^d$, where d is the dimensionality of the instance space.

In the MIL framework, labels are associated with bags, not with individual instances. The bag label $Y_i \in \mathcal{Y} = \{0, 1\}$ indicates the class of bag i. In the binary classification scenario, $Y_i = 1$ indicates a positive bag (i.e., contains at least one positive instance), while $Y_i = 0$ indicates a negative bag (i.e., all instances are negative).

The objective is to construct a function $h(X) : \mathcal{X} \to \mathcal{Y}$, where X represents all bags, and $\mathcal{Y}$ is the set of corresponding bag labels. This function, h, is designed to predict the label of any new bag, based on the instances it comprises.

The hypothesis space of MIL, denoted as $\mathcal{H}$, includes all possible functions that map from the power set of $\mathbb{R}^d$ (encompassing all potential bags of instances) to the set of bag labels, $\mathcal{Y}$. The learning procedure in MIL is designed to identify a function $h \in \mathcal{H}$ that minimizes the expected risk, which is equivalent to minimizing the expected loss over the distribution of all potential bags and their labels.

It's worth noting that different MIL algorithms will employ various strategies to represent and learn from the bags of instances, such as different instance aggregation functions to form bag-level representations, different costs functions and methods for optimization of costs (gradient descent, heuristic search). These strategies can significantly influence the performance of the MIL model.

2.2 Natural Language Processing

The goal of Natural Language Processing (NLP) is to enable computers to understand, interpret, generate, and communicate with humans using natural language. NLP is a subfield of artificial intelligence (AI) and linguistics that focuses on developing algorithms, models, and techniques to analyze, process, and generate human language. Areas of utilization are text classification, text summarization, machine translation, sentiment analysis, speech recognition, natural language inference and many more. NLP aims to bridge the gap between human and machine communication, allowing for more natural and intuitive interactions with machines and applications.

2.2.1 Early NLP Development

During the early stages of Natural Language Processing, initial tasks focused primarily on understanding the structure of language and developing algorithms for parsing sentences. The field was heavily influenced by Chomsky's introduction of transformational-generative grammar, which laid the groundwork for parsing techniques [Cho57]. Early NLP tasks also included machine translation, which aimed to automatically translate text from one language to another, as demonstrated by the Georgetown-IBM experiment in the 1950s [Hut04]. However, mentioned experiment dealt with a limited vocabulary and a narrow set of grammar rules, which limited its adaptability and effectiveness in real-world translation tasks. Other early tasks involved keyword extraction and information retrieval, with methods relying on rule-based systems and basic statistical techniques, like Markov Models or Clustering, to accomplish these goals [MS99].

2.2.2 Current State

The current state of NLP has advanced significantly from its early stages, primarily due to breakthroughs in machine learning, deep learning, and the availability of large-scale datasets. State-of-the-art methods now rely on

neural network-based models, which have achieved impressive performance across a wide range of NLP tasks [NOMC11].

One of the most significant advancements in recent years is the development of Transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers) [DCLT18], GPT (Generative Pre-trained Transformer) [RNS+18], and RoBERTa (A Robustly Optimized BERT Pretraining Approach) [CKG+19]. These models have shown remarkable success in various NLP tasks, including sentiment analysis, named entity recognition, machine translation, text summarization, and question answering.

These models are pre-trained on large amounts of text data and can be fine-tuned for specific tasks using smaller labelled datasets. Large-scale datasets are typically derived from extensive and diverse sources, including web archives like Common Crawl, Wikipedia dumps, BooksCorpus, and WebText derived from Reddit submissions. This pre-training and fine-tuning approach has enabled these models to learn and generalize from vast amounts of linguistic knowledge, resulting in state-of-the-art performance across a wide range of NLP applications. Such applications are sentiment analysis, named entity recognition, question answering, text summarization, translation, and text generation. These tasks leverage fine-tuning of pre-trained Transformer models on specific, smaller labelled datasets, adapting their broad understanding of language to the particular requirements of each task. A more detailed discussion of Transformer-based models and their advantages can be found in the section below.

2.3 Transformers

This section focuses on the evolution of the Transformer model, an influential architecture in the field of Natural Language Processing. It was introduced in the seminal "Attention is All You Need" paper [VSP+17], where the Transformer model revolutionized NLP with its unique attention mechanism, enabling the model to capture long-range dependencies in text effectively. The BERT model, built on the foundational Transformer architecture, further propelled the field forward, achieving state-of-the-art results across a broad spectrum of NLP tasks through its innovative pre-training and fine-tuning

approach [DCLT18]. However, a significant limitation of these models is the processing of long input sequences. This led to the development of BERT derivatives, such as Longformer [BPC20] and Big Bird [ZGD+20], specifically designed to handle long input sequences more efficiently.

In our exploration of Transformer models, we also discuss a variant known as the Set Transformer. While it does not strictly align with the standard Transformer models commonly employed in NLP, it introduces novel methods for handling sets [LLK+19]. In particular, it's Pooling by Multihead Attention (PMA) function is employed in our classifier, making it relevant to our discussion. We discuss each mentioned solution, its mechanism and its implications in NLP.

2.3.1 Transformer Architecture

Paper Attention is All You Need [VSP+17] presents a new neural network architecture for working with NLP tasks called Transformer. The key feature of the Transformer is the attention mechanism, which allows the model to create context-aware representations of the input space. Attention mechanisms have become an integral part of effective sequence modelling and transduction models in various tasks, enabling the modelling of dependencies regardless of their distance in the input or output sequences. Unlike traditional recurrent networks (RNNs) [LQH16], the Transformer model does not contain any recurrent or convolutional layers. Instead, it relies solely on attention mechanisms to create dependencies between input and output. This design choice enables increased parallelization of computations and, therefore, faster computation times, making the Transformer a popular choice for large-scale NLP applications.

The self-attention mechanism computes value $\mathbf{v}$, key $\mathbf{k}$, and query $\mathbf{q}$ vectors for each input vector in a sequence. It calculates attention coefficients for every input vector based on the dot-product of its key and query vectors and applies the softmax function to calculated dot-products. The output is a context-aware representation of each input token, generated by summing the value vectors of all tokens, where the importance of each value vector is determined by the similarity between the current token's query vector and the

other tokens' key vectors. If we consider having multiple mentioned vectors in matrices $\mathbf{V}$, $\mathbf{K}$ and $\mathbf{Q}$, we can formally denote attention as following 2.1.

$$\text{Attention}\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right)=\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{d_{k}}}\right)\mathbf{V} \tag{2.1}$$

The context-aware representations produced by the self-attention mechanism serve as input to subsequent layers in the Transformer model. These representations capture the relationships between tokens in the input sequence, providing valuable information about the syntactic and semantic structure of the text. The computation and memory complexity of the self-attention mechanism is $\Theta(n^2)$, where n is the sequence length. As the self-attention mechanism is permutation invariant, positional encoding is needed to integrate information about the order of the sequences. Recurrent models like RNNs, LSTMs, and GRUs process sequences sequentially, making it harder for them to capture long-range dependencies [LBE15]. In contrast, self-attention allows the model to directly relate each token in the sequence to all other tokens, making it easier to capture relationships between distant tokens.

Multi-head attention is an extension of the attention mechanism that allows the model to learn and capture multiple types of relationships between tokens simultaneously. In multi-head attention, the attention mechanism is applied multiple times (multiple "heads") with different learned linear projections of the key, query, and value vectors. Each head computes its own set of attention coefficients and context-aware representations. The outputs of all heads are then concatenated and linearly transformed to produce the final output of the multi-head attention layer. It is formally denoted in 2.2, where $\mathbf{W}_{i}^{Q},\mathbf{W}_{i}^{K}\in\mathbb{R}^{d_{\text{model}}\times d_{k}}$, $\mathbf{W}_{i}^{V}\in\mathbb{R}^{d_{\text{model}}\times d_{v}}$, and $\mathbf{W}^{O}\in\mathbb{R}^{hd_{v}\times d_{\text{model}}}$ are the projections of parameter matrices. The dimension $d_{\text{model}},d_{v},d_{k}$ denotes dimensions of output, values and keys . The h denotes number of heads (parallel attention layers).

$$\begin{aligned}\text{MultiHead}\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right)&=\text{Concat}\left(\text{head}_{1},\ldots,\text{head}_{h}\right)\mathbf{W}^{O}\\ \text{head}_{i}&=\text{Attention}\left(\mathbf{Q}\mathbf{W}_{i}^{Q},\mathbf{K}\mathbf{W}_{i}^{K},\mathbf{V}\mathbf{W}_{i}^{V}\right)\end{aligned} \tag{2.2}$$

In complex sentences or scenarios where multiple relationships exist between tokens, a single attention mechanism might struggle to capture all these relationships effectively. The attention mechanism could potentially focus on one dominant relationship while missing or giving less importance to other important relationships in the sequence, while multi-head attention limits such a situation from happening. Described situation is pictured in Figure 2.3, where multi-head attention is used to capture all relationships of the word 'it'.

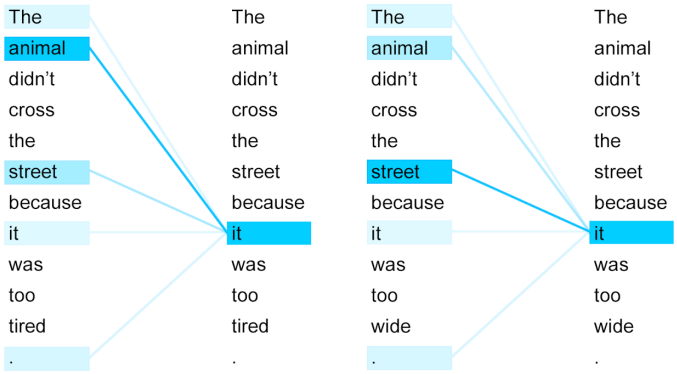


Figure 2.3: Visualization of attention scores for tokens in relationship with word 'it'. 'It' could reference to various words in text. The picture is reprinted from [Blo17].

The Transformer architecture uses a stacked encoder-decoder structure. The encoder processes the input sequence through multiple layers, each consisting of multi-head self-attention and position-wise feed-forward networks, creating a continuous representation of the input. The decoder generates the output sequence using multiple layers, each with multi-head self-attention (for output tokens), multi-head cross-attention (to incorporate information from the encoder), and position-wise feed-forward networks. The encoder captures the input's syntactic and semantic information, while the decoder combines this information with the output sequence context to generate the final output. The architecture is displayed in Figure 2.4.

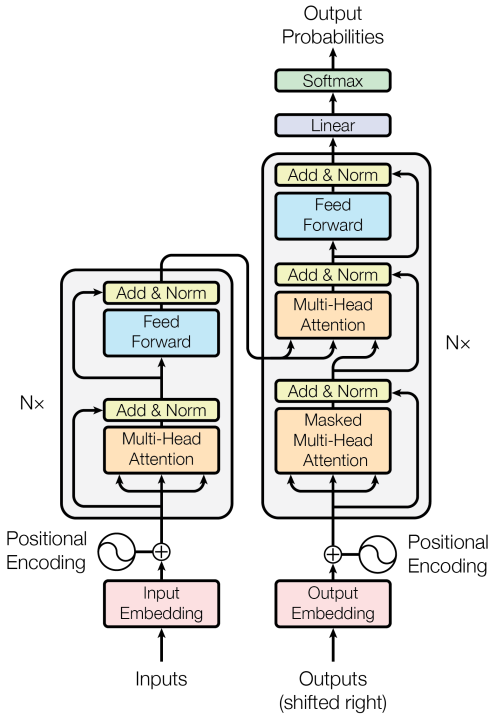


Figure 2.4: Visualization of the architecture of the Transformer model. On the left side is the encoder, which feeds its output to decoder on the right side. The picture is reprinted from [VSP+17].

2.3.2 BERT

A language representation model BERT is introduced in [DCLT18]. The acronym stands for Bidirectional Encoder Representations from Transformers. BERT's construction is based on the Transformer model proposed in [VSP+17]. The language models preceding BERT were based on unidirectional representation, unlike BERT, which is designed to learn bidirectional language representation. When the unidirectional language model processes a particular token, it can use only the tokens it has already processed, while the bidirectional language model works in both directions, so any token attends to all the remaining tokens. Thanks to that, the attention mechanism can capture the context of the entire input. However, as mentioned in the previous section, such a solution introduces additional computational and memory complexity, which is $\Theta(n^2)$ [DCLT18].

Working with BERT consists of two steps - pre-training and fine-tuning. Pre-training phase is done in two tasks. The first task is creating a masked language model that masks a part of tokens by [MASK] token in the input sequence. The model then has to use the complete context of the sequence to predict masked tokens correctly. This is known as Masked Language Modelling, a token-level prediction task. Thanks to the first task BERT is able to avoid the unidirectionality in token representation. In the second task, BERT tries to understand the relationships between two given sentences and determine whether one sentence directly follows another. This is the Next Sentence Prediction task, a sequence-level prediction task.

BERT is a versatile tool that can be easily fine-tuned with respect to specific tasks. The additional module can be "mounted" on top of it. The process of fine-tuning can be seen as a case of transfer learning, where the model uses the knowledge it already gained during its pre-training. The pre-trained model can be fine-tuned for various specific tasks, but it needs to be pre-trained only once. Compared with a situation where we would always have to train our model from scratch, this is an enormously time and money-saving step. There is a large amount of pre-trained models shared among the community, for example, on HuggingFace [WDS+20].

Thanks to the special tokens, it can be used for both sequence-level and token-level tasks mentioned above and many of their adaptations like question answering, natural language inference or sentiment analysis. Each sequence of tokens fed into BERT initiates and ends with a special [CLS] token. Typically, the final embedding of this token is utilized for tasks involving classification at the sequence level. Another special type of token is [SEP] token. If BERT processes as input two sequences simultaneously, each having a unique relevance to the task being performed. These two sequences are treated as a unified sequence, with a distinctive [SEP] token acting as a divider. This functionality proves to be particularly useful in scenarios such as question answering tasks.

BERT can work with sequences up to the length of 512 tokens due to the quadratic computational complexity of the self-attention mechanism. Longer inputs are too computationally complex.

2.3.3 Long Inputs in Transformers

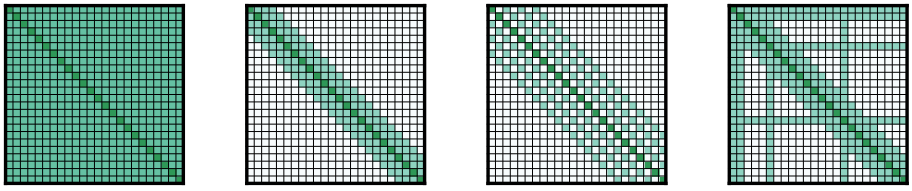
Despite the remarkable success of Transformer-based models in various NLP tasks, their ability to handle long input sequences remains a significant challenge due to their quadratic computational complexity. This limitation is caused by the self-attention mechanism, which computes pairwise attention scores for all tokens in the input sequence. As a result, the computational and memory requirements grow quadratically with the sequence length, rendering these models infeasible for long sequences.

To tackle this issue, several adaptations of the original Transformer architecture have been proposed, specifically designed to process long input sequences more efficiently. These models, including the Longformer [BPC20] and Big Bird [ZGD+20], employ various strategies to use a more computationally efficient sparse attention patterns.

LongFormer

The Longformer, proposed by Beltagy et al. in 2020 [BPC20], is a modification of the Transformer model, which has been designed to handle long input sequences more efficiently. Retaining the original architecture of the Transformer, Longformer strategically alters the self-attention mechanism to reduce the computational complexity and memory requirements from $\Theta(n^2)$ to $\Theta(n)$, where n is the sequence length. However, a reduction in complexity comes at the cost of loss of full context.

This is achieved by limiting the number of tokens that each token can attend to, instead of the standard practice where every token attends to all others in the sequence. It introduces different sparse attention patterns, such as the sliding window pattern and the dilated window pattern. The sliding window pattern restricts each token's attention to a constant-sized neighbourhood around it, while the dilated window pattern introduces gaps between attended tokens, providing more diversity in the local context. A comparison of the full attention pattern and patterns introduced in Longformer is visualized in Figure 2.5.



(a) : Full n^2 attention

(b) : Sliding window attention

(c) : Dilated sliding window

(d) : Global + sliding window

Figure 2.5: Visualization of the paterns of full attention mechanism and the paterns introduced in the Longformer [BPC20] paper. Reprinted from [BPC20].

Additionally, the Longformer designates a predefined number of tokens as 'global', which are allowed to attend to, and be attended by, all other tokens in the sequence. This ensures that important information is not missed due to the local attention restriction.

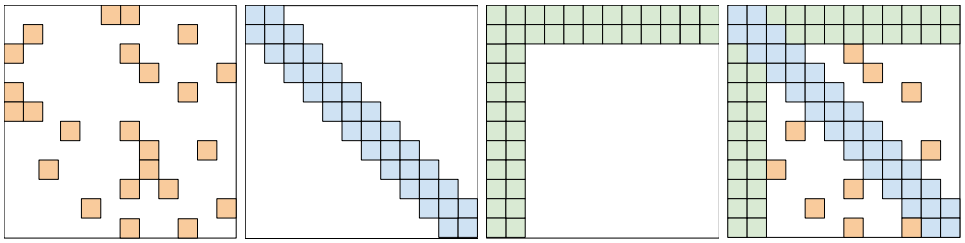
Despite the local attention patterns, the model can still capture a broader context from the input sequence by stacking multiple layers of its sparse self-attention mechanism and thus expand the receptive window of the tokens.

Mentioned alternation to attention patterns can be performed on any pre-trained Transformer-based language model.

Big Bird

The Big Bird model, introduced by Zaheer et al. in 2020 [ZGD+20], is another adaptation of the Transformer architecture designed to handle longer sequences. Authors reduced the time and memory complexity from quadratic $\Theta(n^2)$ to linear $\Theta(n)$ for sequence length n, thus enabling the processing of longer sequences. It achieves this by implementing a sparse attention mechanism similar to the one used in Longformer [BPC20], but with an additional feature. Besides the sliding window attention and global attention, Big Bird also introduces random attention, where each token attends to a random set of tokens outside its window.

This random attention mechanism, inspired by principles in graph theory, helps decrease the average lengths of the shortest paths between pairs of tokens, enabling faster propagation of information throughout different parts of the sequence. Using these attention patterns effectively makes Big Bird and similar models utilize graph theory, as the attention patterns could be viewed as graph adjacency matrices [ZGD+20, vac23].



(a) : Random attention **(b) :** Window attention **(c) :** Global attention **(d) :** Big Bird

Figure 2.6: Visualization of multiple patterns that together present Big Bird self-attention mechanism [ZGD+20]. Reprinted from [ZGD+20].

Zaheer et al. provided theoretical evidence demonstrating that the linear sparse attention mechanisms possess characteristics similar to the quadratic attention mechanism used in standard Transformers [ZGD+20, DCLT18]. Moreover, when suitably designed, they affirmed that these sparse attention mechanisms could serve as universal approximators of sequence functions. This strategy significantly augments the capability of Big Bird to process lengthier sequences, which can successfully scale up to 8x longer (4096 tokens) sequences than previous models like BERT while retaining competitive performance. The authors demonstrate that the model can be used effectively in genomics and other domains where long-sequence data is prevalent.

2.3.4 Set Transformer

The Set Transformer model, as proposed by Lee et al. in 2019, is an attention-based set-input neural network architecture [LLK+19]. It is an adaptation of the Transformer model specifically designed to process and model interactions among elements in the input set. The key feature is that it incorporates

self-attention mechanisms in a way that is permutation-invariant, i.e., the order of the input elements does not affect the output. This feature sets it apart from traditional Transformers, which are intended to handle sequences where the order of elements is significant.

The architecture of the Set Transformer is based on the same groundwork as the standard Transformer. Unlike basic pooling operations like mean or max, the paper introduces an aggregation function Pooling by Multihead Attention (PMA), which can be parametrized and learned to the problem at hand. Below is formal definition of all mentioned architecture subparts, definitions are generalized, for full context visit [LLK+19].

The encoder uses self-attention to concurrently encode the whole set X, thus giving us interactions among instances during the process. For this purpose, the Set Transformer uses the adaptation of a Multihead Attention Block (MAB) from the Transformer [VSP+17], utilizing the same multi-head attention (from Equation 2.2) but without positional encoding and dropout. To compute self-attention between the elements in the set, the authors define Set Attention Block (SAB). These blocks are stacked together to encode higher-order interactions. The rFF in equations is any row-wise feed-forward layer. Given the input matrices $\mathbb{X}, \mathbb{Y} \in \mathbb{R}^{n \times d}$.

$$\mathrm{MAB}(X, Y) = \mathrm{LayerNorm}(H + \mathrm{rFF}(H)) \tag{2.3}$$

where

$$H = \mathrm{LayerNorm}(X + \mathrm{Multihead}(X, Y, Y)) \tag{2.4}$$

$$\mathrm{Encoder}(X) = \mathrm{SAB}(\mathrm{SAB}(X)) \tag{2.5}$$

where

$$\mathrm{SAB}(X) = \mathrm{MAB}(X, X) \tag{2.6}$$

Features $\mathbb{Z} \in \mathbb{R}^{n \times d}$ from the encoder are then passed to the decoder, which feeds them to the feed-forward network to get the final outputs. As can

be seen from the Equation 2.8, the features are passed through the PMA aggregation function and SAB block before reaching the feed-forward network for the final output. Such construction is beneficial since the influence of each instance on the target is not necessarily equal.

The Pooling by Multihead Attention (PMA) is defined as applying multi-head attention on a learnable set of k seed vectors. PMA starts with a trainable matrix $S \in \mathbb{R}^{k \times d}$ of k seed vectors. The seed vectors k can be thought of as representing different 'points of view' on the input data, and the multi-head attention mechanism can focus on different parts of the input from each of these points of view. This results in a pooled output that captures more complex relationships in the data than simple averaging or max pooling would.

$$\text{Decoder}(Z) = rFF(\text{SAB}(PMA_k(Z))), \tag{2.7}$$

where

$$\text{PMA}_k(Z) = \text{MAB}(S, \text{rFF}(Z)) \tag{2.8}$$

Lee et al. prove that Set Transformer is a universal approximator for permutation invariant functions [LLK+19].

In some cases, the SAB quadratic complexity may be too expensive for large sets. The authors propose a more scalable solution called the Induced Set Attention Block (ISAB). It reduces complexity by introducing m inducing points forming a matrix of trainable parameters $I \in \mathbb{R}^{m \times d}$ for $X \in \mathbb{R}^{n \times d}$. These inducing points are essentially learnable parameters or "inducible vectors" that represent a subset of the input data. The idea is that inducing points can learn to represent the most important features of the input set and can thus provide a good approximation of the full attention mechanism with a fraction of the computational cost, which is $\Theta(mn)$, where m is the number of inducing points, typically smaller than n and n is the size of the input set $\mathbb{X} \in \mathbb{R}^{n \times d}$.

$$\mathrm{ISAB}(X) = \mathrm{MAB}(X, H) \tag{2.9}$$

where

$$H = \mathrm{MAB}(I, X) \tag{2.10}$$

The Set Transformer has been shown to perform well on various tasks, including point cloud classification, set anomaly detection, and amortized clustering, among others, demonstrating its versatility and effectiveness in handling set inputs. Unlike the already existing set-input models, which tend to lose some information about interactions between items of sets during encoding.

2.4 Text Classification

As the primary objective of our thesis is to develop a MIL-based text classifier, we aim to provide a more in-depth exploration of text classification to better understand the context and challenges associated with this task.

Textual classification is one of the disciplines of Natural Language Processing. The definition of textual classification is assigning a label or class to a given text. This task is essential to many real-world applications, such as fact-checking, sentiment analysis, spam detection, topic categorization, and document tagging. According to [MKC+21] text classification can be divided into 2 groups based on their approach to making classification decisions. Rule-based techniques categorize text by applying a predefined set of rules, requiring comprehensive domain knowledge. In contrast, machine learning-based approaches learn to classify text by analyzing data patterns. A machine learning algorithm discovers relationships between texts and their corresponding labels through training on pre-labelled examples [MKC+21].

Because we are developing a machine learning method for NLP models, we will further focus on the second group - machine learning approaches. They are further divided into 3 main categories: traditional machine learning, deep learning and hybrid methods.

Traditional machine learning methods for text classification combine specific text preprocessing, such as bag-of-words or TF-IDF [KJMH+19], with classic ML classifiers like SVM, Naïve Bayes, or Decision Trees. These methods are relatively simple, interpretable, and computationally efficient. However, they rely on handcrafted features and may struggle with capturing the complexities of natural language or handling large-scale datasets and complex NLP tasks that require context and semantic understanding.

Deep learning techniques for text classification employ neural networks, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Transformer-based models (e.g., BERT, GPT, RoBERTa) [MKC+21, KJMH+19]. These methods can automatically learn high-level features from the text data, reducing the need for manual feature engineering. Word embeddings, like Word2Vec, GloVe, and contextualized embeddings from Transformer-based models, are often used to represent text as dense vectors [KJMH+19]. We will dive deeper into the differences between mentioned methods in their respective chapters.

However, the primary difference is that deep learning models can automatically learn high-level, complex features from the data, while classic machine learning techniques typically rely on manually crafted features.

Hybrid methods combine elements from both classic machine learning and deep learning techniques. These methods often use deep learning models to extract features from the text and then feed those features into a traditional machine learning classifier.

Such methods, for example, involve combining Transformer-based models for encoding text with different types of classifiers, such as Support Vector Machines, Graph Neural Networks (GNN) [vac23], or, as is introduced in our thesis, Multiple Instance Learning (MIL).

They can leverage the strengths of both classic and deep learning techniques and potentially improve the performance compared to a single approach. However, such methods can be more complex to implement and fine-tune, which may require additional computational resources.

2.4.1 Fact Verification

One of the subtasks of our thesis is to evaluate our MIL classifier on a dataset and utilize fact-checking pipeline presented in the paper below. The 'CsFEVER and CTKFacts: Czech Datasets for Fact Verification' publication [DUR+22] aims to acquaint readers with the existence of Czech data sets for automatic fact-checking, as well as how these sets are used to automate fact-checking, including complex procedures for fact-checking. The solution chosen for the Czech language meets the problem of lack of data for training fact-checking models. This problem is common to all low-resource languages.

The first of the two new Czech fact-checking datasets is called CsFEVER, and it is based on Wikipedia articles. The CsFEVER is a version of the En-FEVER [TVCM18] dataset localized in the Czech language using a document alignment between Czech and English Wikipedia abstracts. The final dataset has 127 thousand machine-translated claims. The downside to this dataset is the fact that claims are translated from English and aligned with Czech claims. Due to this post-processing, the achieved precision is 66%. Lower precision might be a problem for models that are sensitive to noise.

The second of the new Czech fact-checking datasets is CTKFacts. In comparison with the previous dataset, the authors have chosen a completely different approach for creating this dataset. It is generated from 2 million news reports from Czech News Agency, which were manually anotated. The final cleaned dataset consists of 3097 claims. Further, the authors made available an open-source annotation platform. Despite being multiple times smaller than CsFEVER, this dataset offers language forms and types of articles we would not be able to get in CsFEVER.

A new approach for claim generation and claim labelling in documents with missing inter-document linking is presented. The method is based on document retrieval and clustering. Dictionaries are created from relevant and semantically diverse documents without interlinking them.

Furthermore, there are presented methods for the analysis of fact-checking datasets, such as inter-annotator agreement and spurious cue analysis. CTK-Facts and CsFEVER datasets are analyzed to determine whether or not they

contain annotation patterns leading to the overfitting of NLP models.

A full fact-checking pipeline is presented, utilizing document retrieval and natural language inference stages. In document retrieval, the stage is a comparison of four baseline models. Two of them are classical key-word approaches, DRQA [CFWB17] and Anserini [YFL17], and the remaining two (M-BERT [DCLT18] and ColBERT [KZ20]) are based on Transformer neural architectures. Then is the final natural language inference stage, where the veracity of a claim is classified based on the previously retrieved evidence. Several pre-trained Transformer models are examined. Specifically used models are Multilingual-BERT [DCLT18], SlavicBERT [ATKS19], Sentence M-BERT [RG20], original M-BERT [DCLT18], XLM-ROBERTA [CKG+19], FERNET-C5 [LS21] and ROBECZECH [SNSS21].

Chapter 3

Methodology

The following chapter outlines the framework and strategies employed to address the challenge of processing long input sequences in Natural Language Processing models.

The 'Problem Statement' section discusses the difficulties faced by current machine learning models in processing lengthy inputs and prior attempts at resolving these issues.

The 'Proposed Solution' section describes a text classification model that leverages the power of Multiple Instance Learning (MIL) and Transformer-based models. The model processes long input sequences by breaking them into smaller parts, which are treated as instances within a bag, with the bag representing the original, unsegmented long sequence. We also describe Alternative Graph Neural Network approach [vac23].

The 'Datasets' section describes the three distinct datasets we used to evaluate our model - MultiSource, Hyperpartisan News Detection, and CTK Facts. Each of these datasets provides a different challenge for our model, testing its versatility and effectiveness. We provide detailed descriptions of each dataset, explaining why they were chosen and how they will contribute to the research.

The 'Evaluation Metrics' section defines metrics used for evaluation of proposed model.

3.1 Problem Statement

Despite the rapid advancements in Natural Language Processing (NLP), tackling the challenge of processing long input sequences remains a significant hurdle. Standard Transformer-based models, which are currently heavily used in a plethora of NLP applications [VSP+17, DCLT18], are particularly affected by this limitation.

The crux of the problem lies in the quadratic computational complexity and memory requirements of the self-attention mechanism, which is at the heart of these models. This complexity, denoted as $\Theta\left(n^2\right)$, where n is the number of input tokens, renders the use of Transformers rapidly unfeasible for sequences longer than 512 tokens. Various solutions have been proposed to alleviate this issue 2. The most straightforward among them is truncating the input sequence to fit within the model's limits, either by retaining the first $N_{\max}$ tokens or concatenating key parts of the input sequence, such as its beginning and end. However, this method inevitably results in a loss of potentially crucial information. Other techniques aim to alter the attention mechanism itself to reduce its computational complexity, such as LongFormer [BPC20] and Big Bird [ZGD+20]. Both are direct adaptations of BERT, specifically devised to tackle long input sequences. A different approach is represented by Performer [CLD+20]. It approximates the full attention mechanism [CLD+20]. These methods can process significantly longer inputs, allowing them to capture a larger amount of information, which proves particularly beneficial for tasks such as question answering or summarization. They have been further introduced in section 2.3.3 in the Related Work chapter.

This problem statement sets the stage for our work, which aims to harness the capabilities of the MIL classifier in combination with pre-trained language models. This approach aims to address the classification of input sequences of long lengths, thereby extending the applicability and performance of NLP systems.

3.2 Proposed Solution

In this work, we propose a text classification model that harnesses the power of Multiple Instance Learning (MIL) and Transformer-based models to manage long input sequences that exceed typical Transformer size limitations. This methodology is adaptable and can be tailored to different text classification tasks such as sentiment analysis, document classification, fact-checking, etc.

Our classifier operates by segmenting the lengthy input sequence into smaller parts. Each of these parts is separately processed by a Transformer-based model, such as BERT, RoBERTa or one of its derivatives. The individually processed parts of the input, represented as embeddings, are treated as instances within a bag, with the bag representing the original, unsegmented long sequence.

The centre of our model lies in the MIL classifiers that process these bags of instances. The classifiers are designed to aggregate the instances within each bag using specific aggregation functions, including max, mean, or a pooling by multi-head attention. The outcome of this aggregation step is a representative feature vector for each bag, which encapsulates information from all instances in the bag, ready for the final classification stage. Intrinsic to our MIL classifiers are the preprocessing and postprocessing neural layers. The preprocessing layers take the Transformer-generated embeddings as input, transforming these into intermediate representations that are subsequently aggregated. After aggregation, the representative feature for each bag is fed into the postprocessing layer, which makes the final classification decision.

The classifier is by utilization of MIL becoming permutation invariant, which means we lose the information about the order in which instances within a bag are presented. Thus the order of instances does not affect the output of the model. What matters is the contents of the bag as a whole. In circumstances where instance order is crucial for classification, we need to incorporate additional positional encoding mechanisms similar to the ones in Transformer [VSP+17].

One trade-off of the Multiple Instance Learning (MIL) approach is that some information might be lost during the aggregation process [Bab08]. Each

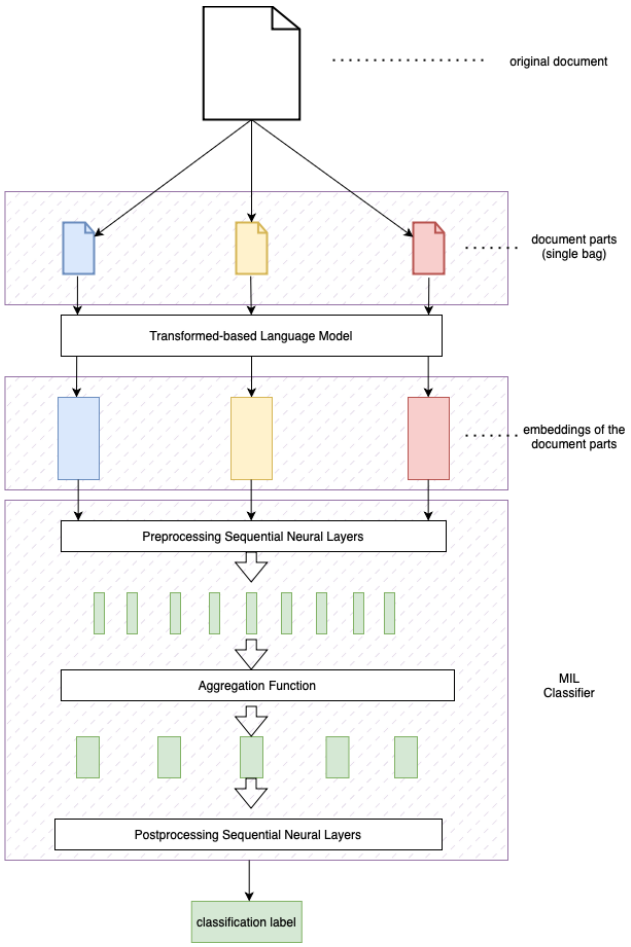


Figure 3.1: A diagram of the proposed model. Original input is split into multiple instances (parts), forming a single bag together. These instances (parts) are then embedded individually by the Transformer model. Created embeddings are then passed as a bag into the MIL classifier, where they go through pre-processing, aggregation and post-processing layers and finally output the classification label. The green rectangles represent different dimensions of features during pre-processing and post-processing.

instance in the bag is processed independently, and then these representations are aggregated to form a bag-level representation. Depending on the aggregation function used, this could result in a loss of specific details from individual instances. Even complex functions such as PMA may not perfectly preserve all information due to the need to condense multiple instance representations into a single bag representation. However, in theory, the Set Transformer, of which the PMA is a crucial part, is proven to be a universal approximator.

The model proposed in this thesis is highly flexible and can be adjusted to accommodate different text classification tasks. The segmentation of input into individual parts, the choice of aggregation function, and the design of the preprocessing and postprocessing layers can all be adapted according to the specific task requirements. However, there is the mentioned trade-off between the ability to process larger inputs by MIL classifier and the ability to maintain a richer, more detailed representation of the input data in the case of standard language models because they process the entire input as a whole without the need for aggregation.

Our objective is to establish a model that leverages the strengths of Transformer-based models and the flexibility of Multiple Instance Learning to handle long sequences in text classification tasks efficiently.

The implementation of the proposed model is publicly available at [1]. It shares portions of its codebase with the Graph Neural Networks (GNN) classifier introduced in 'Graph Neural Networks for Long Input NLP Models' [vac23]. Methods derived from the GNN classifier are used for dataset import/export, evaluation and baseline model. All affected methods are appropriately cited within the codebase.

3.2.1 Alternative Graph Neural Network Approach

In parallel to our exploration of Multiple Instance Learning for managing long inputs in NLP models, a separate thesis written by Vaclav Hlavac examined the potential of Graph Neural Networks (GNN) for the same challenge [vac23].

[1] https://github.com/aic-factcheck/long-input-mil

In the following section, we introduce the GNNs solution and compare it with ours.

Both the Multiple Instance Learning (MIL) and Graph Neural Network (GNN) models handle the challenge of lengthy input sequences by breaking them into manageable parts for initial processing via Transformer-based models. Each part is encoded into embeddings.

Despite these similarities, the two methods use fundamentally different approaches. MIL sees each part as an instance within a bag and then aggregates these instances into a bag-level representation. The initial task is then considered to be a task of bag classification. Conversely, GNN treats each encoded part as a node in a graph, utilizing the topology to model interactions between parts. Finally, text classification is seen as a graph classification task. An encoding of the entire graph is obtained through a pooling operation and then utilized to perform the final classification.

The GNN model considers two distinct graph topologies: the 'complete' topology and the 'local + global' topology. The 'complete' topology connects every pair of nodes, creating a permutation invariant network where all nodes can directly communicate. This topology could resemble the self-attention [VSP+17], they share the $\Theta(n^2)$ complexity. It is permutation invariant like our model presented above. The 'local + global' topology, in contrast, connects nodes corresponding to parts that appear sequentially in the input. It also introduces an additional latent node connected to all other nodes, enabling indirect communication. This configuration is sensitive to the input order, resulting in a graph that reflects the inherent sequence of the text and thus is not permutation invariant.

Graph Neural Networks are computationally more complex due to their need to model and process intricate relationships between nodes, which involves multiple computations across interconnected nodes in the graph. While Multiple Instance Learning may be more computationally efficient by simply aggregating over a bag of instances, they risk losing some information during the aggregation process. Graph Neural Networks can capture more complex relationships within data, potentially leading to better performance on tasks where these relationships are crucial.

3.3 Datasets

In this section, we explore the datasets used for testing classifiers used in our study. We leverage three datasets—MultiSource, Hyperpartisan News Detection, and CTK Facts—each with unique characteristics and challenges.

The MultiSource dataset initially featured in Vaclav Hlavac's thesis [vac23] is used to classify a set of input sequences based on their source. The Hyperpartisan News Detection dataset, sourced from various online news articles, assesses our classifier's ability to handle lengthy input sequences. Lastly, the CTK Facts dataset, a Czech fact-checking dataset, is utilized to explore fact-checking in the Czech language.

Each dataset is unique, presenting different challenges for long-input NLP classifiers, which we will delve into in the subsequent sections.

3.3.1 Multisource

The MultiSource task and its dataset were introduced as part of a broader initiative at AIC to develop Long Input NLP models. Due to the scarcity of text classification datasets with a substantial number of examples, we have created our synthetic dataset, allowing us to parametrize and fine-tune it to our desires. It was originally featured in a thesis by Vaclav Hlavac [vac23]. The MultiSource task involves classifying a set of input sequences based on their source. In our example, we treated sentences as individual instances, and the objective was to determine whether the set of sentences originated from a single document. A set is labelled as false if it comes from a single document and true if it consists of sentences from multiple documents.

The method for creating multisource datasets can be generalized to any collection of documents containing sufficiently numerous sequences. To compare classifiers in my thesis and thesis by Vaclav Hlavac [vac23], we utilized MultiSource datasets generated from an English Wikipedia snapshot.

Algorithm 1 Generating a part of a MultiSource dataset split. Reprinted from [vac23].

procedure GENERATEMULTISOURCESPLIT($\mathcal{D}$, N, S_n, S_{random})
  `examples` $\leftarrow \varnothing$
  for $i = 1$ **to** N **do**
    `example` $\leftarrow \varnothing$
    $d_{\text{orig}} \leftarrow$ `sample_document` ($\mathcal{D}$, `min_sentences` $= S_n - S_{\text{random}}$)
    `example` $\leftarrow$ `example` $\cup$ `sample_sentences` (d_{orig}, `n_sentences` $= S_n - S_{\text{random}}$)
    for $j = 0$ to S_{random} **do**
      $d_{\text{random}} \leftarrow$ `sample_document` ($\mathcal{D}$, `min_sentences` $= 1$)
      `example` $\leftarrow$ `example` $\cup$ `sample_sentences` (d_{random}, `n_sentences` $= 1$)
    end for
    `examples` $\leftarrow$ `examples` $\cup$ {`example`}
  end for
end procedure

The MultiSource dataset generation process involves treating documents as collections of sentences and creating train, validation, and test splits individually. Each split consists of multiple parts with defined parameters for the number of sentences (S_n), the number of sentences from different sources (S_{random}), and the number of examples (N). To create an example, an initial document is sampled, and a base set of sentences is selected. Additional sentences from other documents are added based on the size of S_{random}, and the resulting set forms the complete example. The label is determined by the number of random sentences, with positive labels assigned when at least one random sentence exists in the example. Finally, sentences are shuffled randomly to prompt the classifier to focus on comparing the text rather than identifying changes in the text's structure. Each sentence can be picked into the example only once. The number of positive and negative examples in each generated split and dataset is equal. Table 3.1 presents info about the dataset used as a baseline in MultiSource experiments. The complete process of generating dataset split is shown in pseudocode at Algorithm 1.

number of random sentences	train	validation	test
0	20000	1000	1000
2	5000	250	250
4	5000	250	250
6	5000	250	250
8	5000	250	250

Table 3.1: MultiSource Dataset with distribution of splits and instances, generated by similar sampling.

```
{
    "label": false,
    "ids": ["Adult_adoption", "Adult_adoption", "
        Adult_adoption", "Adult_adoption"],
    "sentences": [
        "In the United Kingdom, only children may be adopted
            .",
        "However, adoption of a person between the ages of 18
             and 20 (inclusive) transfers both inheritance
            rights and filiation.",
        "Adult adoption is a form of adoption between two or
            more adults in order to transfer inheritance
            rights and/or filiation.",
        "Depending on the laws of the jurisdiction, adult
            adoption may not be available as a legal option
            ."]
}
```


Figure 3.2: The example of a negative instance in MultiSource Dataset, where $N = 4$ and $S_{\text{random}} = 0$. The IDs represent articles of origin. In this case every sentence is from "Adult adoption" article on Wikipedia.

The selection of documents for the MultiSource dataset greatly influences its difficulty. Uniformly sampling documents creates easily classifiable examples, so an option to sample more similar documents was added, making the task more challenging. The Anserini toolkit [YFL17] is used to retrieve similar documents for the selection of random sentences into positive examples. By utilizing Anserini, the k most similar documents are found. From the top k documents found, random sentences are sampled and added to the original set of sentences in the example. This process allows control over the dataset's difficulty by adjusting the number of random sentences in positive examples and using similar document sampling. For instance, using a single sentence from a different yet similar source document can turn the task into a far more difficult than finding multiple random sentences. An example of a positive instance is shown at Figure 3.3, and a negative instance is shown at Figure 3.2. For further information about dataset generation, visit the relevant chapter at [vac23].

```
{
    "label": true,
    "ids": ["Isaac_Milner", "Barbara_Wilberforce", "
        Isaac_Milner", "William_Whiston"],
    "sentences": [
        "He was instrumental in the 1785 religious conversion
            of William Wilberforce and helped him through
            many trials and was a great supporter of the
            abolitionists' campaign against the slave trade,
            steeling Wilberforce with his assurance before
            the 1789 Parliamentary debate.",
        "Following her husband's death in 1833, Barbara Wilberforce
            spent her time with her sons, Robert and Samuel, or with
            her sister Ann Neale in Taplow in Buckinghamshire.",
        "Isaac Milner 11 January 1750      1 April 1820 was a
            mathematician, an inventor, the President of
            Queens College, Cambridge and Lucasian Professor
            of Mathematics.",
        "In 1710 he lost the professorship and was expelled from the
            university as a result of his unorthodox religious views."]
}
```


Figure 3.3: The example of a positive instance in MultiSource Dataset, where $N = 4$ and $S_{\text{random}} = 2$. The random sentences are highlighted. The IDs represent articles of origin.

In our preliminary experiments, we assessed both random document sampling and similar document sampling methods. However, we found that the uniform sampling of random documents made the task too straightforward for both our classifier and baseline methods. To achieve sufficient complexity, we decided to use datasets created through similar document sampling for all our experiments involving the MultiSource dataset.

3.3.2 Hyperpartisan News Detection

The Hyperpartisan News Detection dataset is compiled from online news articles sourced from various websites. This dataset focuses on the task of identifying articles that exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or a person [KMS+19], without offering opposing views or opinions. These articles tend to present information in a deceptive

manner and employ potent language to elicit emotional reactions from the reader.

The dataset was chosen as a benchmark for comparing methods of handling long input sequences in both my thesis and the thesis by Vaclav Hlavac, where he tackles the same problem with GNNs [vac23]. This dataset serves to assess whether our classifier can effectively manage extremely lengthy input sequences. The longest input sequence in the dataset contains 13,296 tokens, with a median token count of 863. Given that Transformer-based models typically have a limitation of 512 tokens, this dataset's instances exceed those constraints in most cases.

Split	positive	negative
train	164	281
validation	37	63
test	37	63

Table 3.2: Size of splits in Hyperpartisan News Detection with distribution of classes.

On the other hand, the dataset comprises just 645 examples, featuring an imbalanced class distribution as displayed in Table 3.2. The dataset is divided into training, validation, and testing subsets, created using stratified sampling to ensure the label distribution remains consistent with the full dataset.

The example of instance in dataset is displayed in Appendix A due to large size.

3.3.3 CTK Facts

The CTK Facts dataset, mentioned earlier in the Related Literature section 2.4.1, is output from Herbert et al. publications [DUR+22]. It is a Czech fact-checking dataset chosen for further experimentation in our thesis for two main reasons. First, we aim to explore a language other than English, potentially using different language models. Second, the dataset focuses on the subtask of text classification - fact-checking task, which involves classifying

a claim as true or false based on evidence obtained from a knowledge base. As the prevalence of fake news increases [VRA18], there is a growing need for methods to address this issue since manual fact-checking is a time-consuming process [GSV22].

Split	SUPPORTS	REFUTES	NEI
train	1104	556	723
validation	142	85	105
test	176	79	127

Table 3.3: Size of splits in CTK Facts dataset with distribution of classes.

The process of claim verification involves a fact-checking pipeline, which consists of two main components. The first component is document retrieval, which identifies a set of evidence from a knowledge base relevant to the given claim. The second component involves classifying the claim based on the provided evidence. Using Natural Language Inference, the claim-evidence pair is assessed and classified as either supporting or refuting the claim. In cases where the evidence does not clearly support or refute the claim, a "not enough information" (NEI) label is assigned.

The dataset is a result of a joint effort between the FactCheck group at AIC CTU and the Department of Journalism at the Faculty of Social Sciences, Charles Univerrsity. News reports from the Czech News Agency (CTK) serve as the data source for the claims. Journalism students reviewed and annotated each claim and its corresponding evidence. The dataset contains a total of 3,097 claims. An example of a single claim and its supporting evidence can be found in Figure 3.4. Table 3.3 displays the number of claims in each dataset split and the label distribution. Additional information about the dataset can be found in the Related Literature section 2.4.1 or the original paper by Herbert et al. [DUR+22].

3.4 Evaluation Metrics

In the process of evaluating the performance of the classifiers in our experiments, we utilize several metrics: accuracy, F1-score, precision, and recall.

Claim: Spojené státy americké hraničí s Mexikem. **EN:** The United States of America share borders with Mexico.
Verdict: SUPPORTS
Evidence 1: "Mexiko a USA sdílejí 3000 kilometrů dlouhou hranici, kterou ročně překročí tisíce Mexičanů v naději na lepší životní podmínky (. . .)" **EN:** *Mexico and the U.S. share a 3,000-kilometer border, crossed by thousands of Mexicans each year in hopes of better living conditions (. . .)* **Evidence 2:** "Mexiko také nelibě nese, že Spojené státy stále budují na vzájemné, několik tisíc kilometrů dlouhé hranici zeď, která má zabránit fyzickému ilegálnímu přechodu Mexičanů do USA (. . .)" **EN:** *Mexico is also uncomfortable with the fact that the United States is still building a wall on their mutual, several thousand kilometers long border, to prevent Mexicans from physically crossing illegally into the U.S. (. . .)*

Table 3.4: Showcase of one claim from CTKFacts together with its evidence. The example is from the original publications where the dataset was originally introduced [DUR+22].

Each of these metrics provides a unique perspective on the effectiveness of a classifier.

Accuracy measures the proportion of total predictions that a model gets right, serving as a straightforward and intuitive indicator of a model's general performance. However, accuracy can often provide a misleading picture in the context of imbalanced datasets, where the majority class could dominate the metric. We use it only in the case of the synthetic Multisource dataset, which has balanced classes.

To provide a comprehensive and nuanced evaluation for imbalanced datasets, we employ F1-score, precision, and recall. F1-score is a harmonic mean of precision and recall, effectively balancing the two in evaluating the accuracy of a classifier. It is beneficial in situations where both false positives and false negatives are important. A higher F1-Score indicates a more accurate and balanced classifier. Precision quantifies the proportion of positive class predictions that are actually correct. Recall measures the portion of actual

positive class examples that the classifier correctly identifies.

We calculate these metrics in both macro and micro settings. The micro setting computes the metric globally across all classes, while the macro setting calculates the metric for each class independently and then averages them. In the context of imbalanced datasets like ours, macro settings prove beneficial as they prevent the majority class from dominating the score, thereby ensuring a balanced evaluation.

Chapter 4

Experiments

We have conducted various experiments using the datasets introduced in the previous chapter. Initially, we focus on the MultiSource dataset to investigate the characteristics of our model. Next, we explore the Hyperpartisan News Detection dataset, which presents challenges due to lengthy inputs and a limited number of examples. Lastly, we assess the performance of our model on the CTKFacts dataset, which introduces a challenge for language models due to its Czech language localization.

4.1 Design of Experiments

In each experiment, we evaluate various settings of our MIL classifier. The best variant of the MIL classifier is then further compared with the baseline and GNN classifier to evaluate the effectiveness of our approach. Appropriate performance metrics are selected based on the specific characteristics of each dataset. These metrics are monitored throughout the training phase, and the model parameters yielding the best performance on the validation splits are chosen for use on the test splits.

All experimental results presented are original and were obtained through

our own execution, with the exception of the computations involving the GNN classifier. These specific results were reproduced from the thesis in which the GNN was first introduced, authored by Vaclav Hlavac [vac23].

4.1.1 Preliminary Experiments

In the initial stages of our research, we recognized the computational burden and time-intensive nature of hyperparameter tuning via traditional grid search methods. Given the complexity of our models and the scale of our data, this approach was not feasible. Instead, we carefully calibrated a subset of key parameters in our experiments. These settings, found to be robust, served as the consistent backbone for our subsequent experiments.

4.1.2 Setup of Experiments

- **Type of MIL classifier** - We utilize end-to-end classifier to enable fine-tuning of language model. With respect to the architecture introduce in chapter Methodology 3, we specify preprocessing and postprocessing layers used in experiments.
  - The **preprocessing layer** of our MIL model consists of a sequential arrangement of linear transformations and ReLU activation functions: it first transforms the input from its original dimension ($input\_size$) to 1268, then to 634, and finally to 256, with a ReLU activation function following each linear transformation. The output of this sequntial layer is then passed to aggregation function, which executes computations and then passes to postprocessing layer.
  - The **postprocessing layer** of our MIL model features a sequence of linear transformations and ReLU activations: it first reduces the dimensionality from 256 to 128, then to 32, and ultimately to the number of classes (k), with a ReLU activation function applied after each transformation, before the final class assignment.
- **Type of Utilized Language Model** - Based on a task we select one of the language models described below.

- **Type of Used Aggregation Function** - Max, mean or Pooling by Multihead Attention (PMA) over instances in a bag.
- **Used Optimizer** - We train model with AdamW optimizer [LH17].
- **Learning Rate** - We are using learning rate $\alpha = 2 \times 10^{-5}$, if not stated otherwise.
- **Weight Decay** - To prevent overfitting we use weight decay $\lambda = 0.01$.
- **Loss Function** - As a loss function, we utilize Cross Entropy Loss.
- **Seed** - Throughout experiments we are using seed set to 420.
- **Number of Epochs** - The number of epochs is set to 20, unless stated otherwise.
- **Batch Size** - The batch size is 40 for Multisource, and 32 for both Hyperpartisan News Detection and CTKFacts. When memory constraints imposed limitations on the classifier, we employed gradient accumulation to achieve the intended batch size.

For a comprehensive understanding of the architecture and its implementation details, please refer to the dedicated repository [1] which houses the complete codebase.

4.1.3 Used Models

Below is a list of models used throughout experiments. All of them are acquired through HuggingFace [WDS+20].

- **FERNET-C5** - Model trained exclusively on Czech data obtained from the common crawl [LS21].
- **RobeCzech** - RobeCzech is a Czech language model based on the RoBERTa architecture, trained on a large corpus of Czech text [SNSS21].

[1] https://github.com/aic-factcheck/long-input-mil

- **Multilingual BERT (M-BERT)** - The model is pre-trained on a large-scale multilingual corpus consisting of text from 104 languages using masked language modelling. Trained in a similar approach as the original BERT using data from Wikipedia [DCLT18].
- **XLM-RoBERTa** - It is a multilingual transformer-based model built upon the RoBERTa architecture, designed for cross-lingual natural language processing tasks. It is pre-trained on a large-scale multilingual corpus, including 100 languages from the Common Crawl dataset [CKG+19].
- **BERT** - It is a pre-trained transformer-based language model that learns deep contextual representations of words from large-scale unannotated text data. It is the first model introduced in the BERT family of transformer-based models. It laid the foundation for various other models that built upon and extended its architecture, such as RoBERTa, ALBERT, DistilBERT, and many more [DCLT18].
- **RoBERTa** - The 'Robustly optimized BERT approach' (RoBERTa) is a pre-trained language model based on the BERT architecture, which was introduced to address certain limitations and improve upon BERT's performance. It uses a larger training dataset, removes the next-sentence prediction objective, and modifies the training procedure with dynamic masking and longer training time, resulting in a more accurate and powerful language model [CKG+19].
- **LongFormer** - A transformer-based language model designed for long input sequences. It employs a sliding window self-attention mechanism and sparse global attention for computational efficiency and high performance on long NLP inputs [BPC20].

4.2 Multisource

The MultiSource task and its appropriate datasets are the first conducted experiments. The goal is to examine the behaviour of our model and test its different variations against the baseline and GNN classifier. As the baseline serves the BERT language model adjusted for classification tasks, the same language model is utilized as a layer for computing the input embeddings in conjunction with our classifier. Each dataset comprises examples with

10 sentences, but the specific arrangement of the examples may vary based on the intended experiment. Unless stated otherwise, we use a version of the Multisource dataset with 40,000 instances generated by similar sampling for increased complexity. The exact setup of dataset splits shows Table 3.1. During dataset generation, language model limitations are taken into account, ensuring that the language model can process each example in its entirety, and thus comparability of classifiers is guaranteed.

In the case of the following experiments, each example in a dataset is a set of N sentences, which are considered a bag. Each bag has its associated label depending on the source of sentences, as described in the section about the dataset. Our classifier aims to assign a negative label if sentences have the same source and a positive otherwise. In the end-to-end settings, the BERT language model embeds each sentence in a bag separately. Those embeddings in the form of [CLS] tokens are processed through the sequential neural layers before being passed to the aggregation function. This aggregation function combines values from each instance in a bag and forwards output to final neural layers for bag-level classification. We use max, mean, and PMA functions as aggregation functions.

4.2.1 Initial Experiments

In these experiments, we examine the performance of our MIL classifier utilizing different aggregation functions. Specifically, we will explore max, mean, and Pooling by Multihead Attention (PMA) aggregation functions. These functions will be compared with baseline BERT and GNN classifier presented by Vaclav Hlavac in his thesis [vac23]. Through experimenting with different aggregation functions, we can gain insights into which method is best suited for combining instance-level information in a given scenario.

The results in Table 4.1 indicate that the additional complexity of the PMA aggregation function does not yield better performance despite utilizing the attention mechanism, which in theory should be beneficial for classifying bag of sequences. The mean aggregation function does perform with comparable accuracy, while the max falls behind.

Types of classifiers	Accuracy
MIL-PMA	0.7635
MIL-MAX	0.704
MIL-MEAN	0.7625
Baseline BERT	**0.889**
GNN	0.808

Table 4.1: Accuracy of our MIL classifier compared against baseline and GNN solutions on test set.

As the results of baseline BERT classifier reveal, it outperforms MIL classifiers, with an accuracy of 88.9%. The GNN classifier reaches 80.8% in its finest settings [vac23]. The dominance of the BERT classifier is expected, as it is a state-of-the-art model for text classification, and sequences in the dataset used in this experiment falls within its input size limits, so our classifier could not benefit from its advantage in processing long inputs.

The difference in performance between the MIL and GNN classifiers is not substantial, and it can be attributed to the additional complexity of GNNs. GNNs are designed to model complex interactions between nodes in a graph structure, and this additional capacity to capture dependencies and relationships can lead to improved performance.

This finding highlights the importance of selecting a suitable aggregation function for effective MIL classification, such as PMA. However, the baseline BERT model still demonstrates superior performance.

4.2.2 Combinations of Aggregation Functions

In our subsequent experiments, we explored various combinations of aggregation functions to determine if they could yield any interesting results. By combining aggregation functions into pairs, we aimed to discover if any synergies or complementary effects could improve the overall performance of our MIL classifier.

The combination of aggregation functions is obtained through simple

concatenating outputs from two aggregation functions. We take features from the preprocessing layer of the MIL classifier, separately pass them to both aggregation functions and concatenate the outputs afterwards. Thus the dimension of the output of the aggregation step is twice the original size.

Types of Aggregation Function	Accuracy
PMA	0.7635
PMA+MAX	**0.769**
PMA+MEAN	0.752
MEAN+MAX	0.736

Table 4.2: Accuracy of MIL classifiers with combinations of aggregation functions into pairs on test set.

The experiment results are in Table 4.2. The introduction of combinations of aggregation functions did not yield significant improvements for any original function from the combination. The previously best-performing PMA aggregation function in end-to-end settings is surpassed only by 0.55%. As a result, we plan to continue experimenting with a single aggregation function. However, we hope this experimentation contributes to a better understanding of the nuances in MIL techniques and may inspire future research to discover more impactful combinations or approaches.

4.2.3 Number of Random Sentences

In the following experiments, we examine various versions of Multisource datasets which differ in the number of random sentences in positive examples. The appropriate numbers are 2,4,6 and 8. We expect the difficulty will get easier with the increasing number of random sentences. We aim to evaluate our MIL classifier in two embedding settings on mentioned datasets. The MIL classifers utilize either PMA or max aggregation function.

The first one is standard embedding, as we utilized it in previous examples. The second one is embedding using a sliding window over sentences in a bag. In this case, two sentences will be encoded as one instance and passed to the MIL classifier. We expect the classifier's performance to improve as it can simultaneously access multiple instances in a bag. However, further

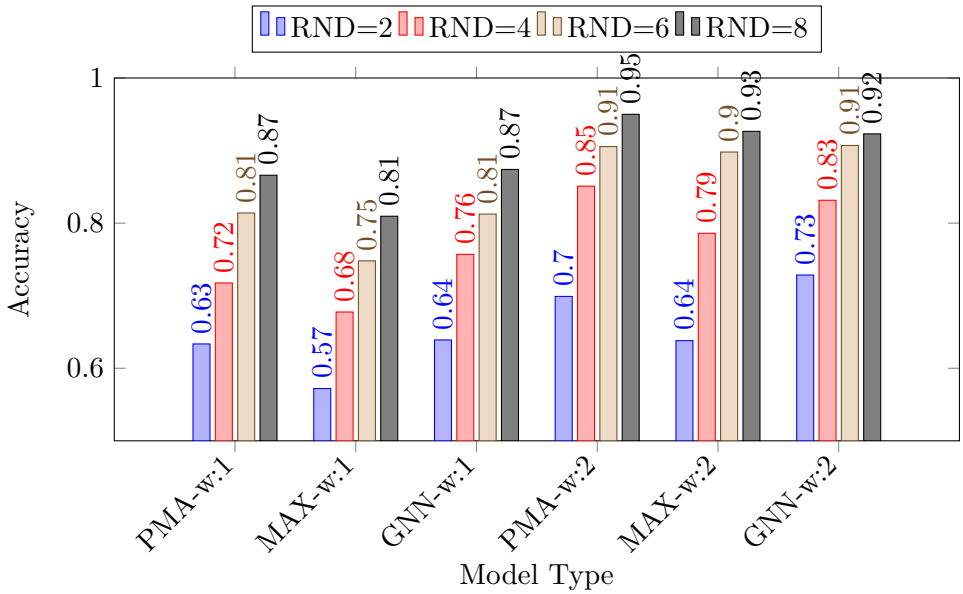


Figure 4.1: Accuracy of end-to-end classifiers with different size of sliding windows on datasets with changing number of random sentences *RND*. The 'w:N' represents size of window, where N is the number of instances in window. Original values are displayed in Table A.1.

experiments on the influence of multiple instance encoding are presented in the following subsection.

The results are presented in Figure 4.1, and the best-performing classifier for each window size is highlighted. As expected, the classifier performance decreases with decreasing number of random sentences. The classifier achieves better accuracy by encoding 2 sentences using a sliding window. The dominance of 2 sentence solution increases with the increasing amount of random sentences. Unlike in previous experiments, the MIL classifier shows comparable performance with GNN classifier with differences in terms of accuracy within 5%.

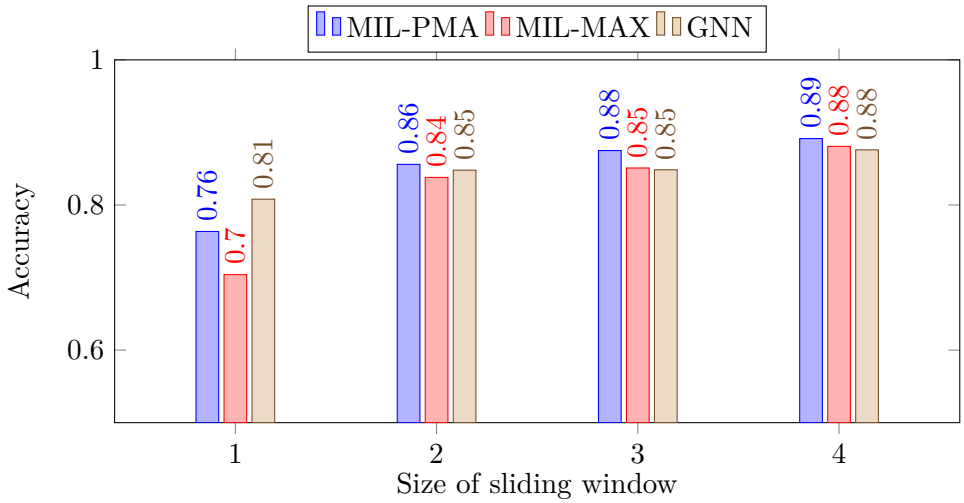


Figure 4.2: Accuracy of classifiers with different size of sliding windows. Original values are displayed in Table A.2.

Multiple Sentences as One Instance

In the last set of experiments on the Multisource dataset, we further inspect the influence of increased moving window over sentences in an example. We execute the following experiments with max and PMA aggregation functions, and the final results are compared with the GNN classifier by Vaclav Hlavac [vac23].

We expect that the task is partially passed down from classifier to encoder by encoding multiple sentences in Transformer. The encoder should be able to distinguish between sentences from different documents and thus help to recognize the positive examples by embedding this information for further processing by the MIL classifier. Nonetheless, this modification may diminish the significance of our classifier. The diminishing significance of our classifier can be seen in the levelling accuracy of max and PMA aggregation functions, which in single sentence settings in previous experiments shows dominance of PMA over max.

Presented opinion is supported by the outcome of experiments in Figure 4.2 where iterations with bigger sliding windows have better accuracy. The

best accuracy for each sliding window is achieved by the MIL classifier with the PMA aggregation function, which outperforms the same classifier with the max aggregation function and GNN classifier in complete setting [vac23]. Used GNNs utilize either local+global or complete settings, and these settings influence how many edges are between nodes in a graph. The complete setting connects every node. The GNN classifier is introduced closer in its respective section in Methodology 3.

4.3 Hyperpartisan News Detection

As stated in the section with the introduction to the Hyperpartisan News Detection dataset, it consists of long inputs. The median instance length is 863 tokens, which significantly exceeds the 512-token limit of standard language models. By conducting experiments on this dataset, we aim to assess our model's ability to handle long inputs up to 13,296 tokens and determine the overall performance advantages compared to classifiers constrained by language models.

The task is to classify news articles which tend to present information deceptively and employ potent language to elicit emotional reactions from the reader. In the following experiments, we utilize BERT and RoBERTa language models in our MIL classifier in different aggregation function settings. The second part compares best-performing MIL classifier versions against baseline solutions utilizing only language models and the GNN classifier presented in [vac23]. Apart from mentioned BERT and RoBERTa, we are going to experiment with LongFormer. The Longformer is a transformer-based model designed explicitly for processing long sequences of data, up to 4096 tokens, by using a self-attention mechanism known as "sliding window attention", which allows it to handle larger contexts without a significant increase in computational demands [BPC20].

The setup of experiments and MIL classifier is the same as in previous experiments. We are training our classifier on 20 epochs, batch size is 32 and the learning rate is $\alpha = 2 \times 10^{-5}$. The remaining hyperparameter settings are unchanged.

4.3.1 Initial Experiments

The goal of the initial experiments is to find the best-performing setup of the MIL classifier for applications with RoBERTa and BERT language models. Each language model is evaluated with Pooling by Multihead Attention and max aggregation functions.

The RoBERTa is trained on a more extensive dataset, which includes the BooksCorpus dataset and a more substantial portion of the English Wikipedia. Together with longer training time and training optimizations, like removing the next sentence prediction (NSP) task and using dynamic masking instead of static masking, the RoBERTa is considered to be an improved version of BERT [CKG+19]. Thus, we expect it to show comparable or better performance.

Types of language model	PMA	MAX
BERT	0.6754	0.7759
RoBERTa	**0.8444**	**0.8217**

Table 4.3: F1-scores for MIL classifiers utilizing BERT or RoBERTa language models with PMA and max aggregation functions. Performance is calculated on validation split.

We are evaluating the experiments on F1-Score, which is a harmonic mean of precision and recall, providing a single metric that balances both considerations. As our dataset has an imbalanced distribution of classes, we use macro computational settings to prevent the majority class from dominating the score.

As can be seen in the F1-score results of experiments in Table 4.3, the RoBERTa-equipped MIL classifier shows superior performance with both PMA and max aggregation functions. Due to its extensive and improved training, we expected the MIL classifier using RoBERTa to dominate. We use it as a benchmark for comparison with other baseline models. Nevertheless, for further experiments, we will continue to include classifiers based on BERT to observe any potential advantages provided by RoBERTa.

Our classifiers have successfully demonstrated their ability to manage ex-

tended inputs of up to 13,296 tokens, effectively addressing the challenge of lengthy sequences without compromising data integrity or requiring truncation.

4.3.2 Baseline Models Comparison

In these subsequent experiments, our goal is to evaluate the performance of different classifiers and language models to identify the most efficient classifier. We have classified our utilized classifiers into three categories. The first category includes our MIL classifiers, which use the BERT and RoBERTa language models and PMA as aggregation functions. The second category consists of baseline language models, such as LongFormer, RoBERTa, and BERT, adapted for classification tasks. LongFormer, in particular, is employed due to the distinct architecture that enables it to process up to 4096 token input sequences, making it ideal for handling extended texts [BPC20]. The third category comprises GNN-based classifiers in local + global configurations [vac23], which use the BERT language model for embedding creation.

We are going to be evaluating F1-score, precision and recall on test split in the macro computational settings.

The results presented in Figure 4.3 indicate a superior performance by the MIL-RoBERTa classifier, leading in all metrics, including F1-score, Precision, and Recall. This suggests that the MIL-RoBERTa classifier has achieved an effective balance between correctly identifying true positives and avoiding false positives or negatives. Interestingly, despite the underperformance of MIL-BERT on the validation split, it shows quite competitive performance on the test split, almost matching the score of MIL-RoBERTa. This may suggest that MIL-BERT is more resilient to the specific quirks of the training data and generalizes better to unseen data. The solid performance of language models, like BERT and RoBERTa, despite their input size limitation, may be attributed to the nature of the input data. News articles often summarize the key information in the initial part and then delve into details later in the text. Therefore, even if the whole text isn't processed, the key information necessary for accurate prediction might still be captured, leading to reasonable performance. The Longformer, which is explicitly designed to handle extended input sequences, has also shown robust performance. However, its metrics

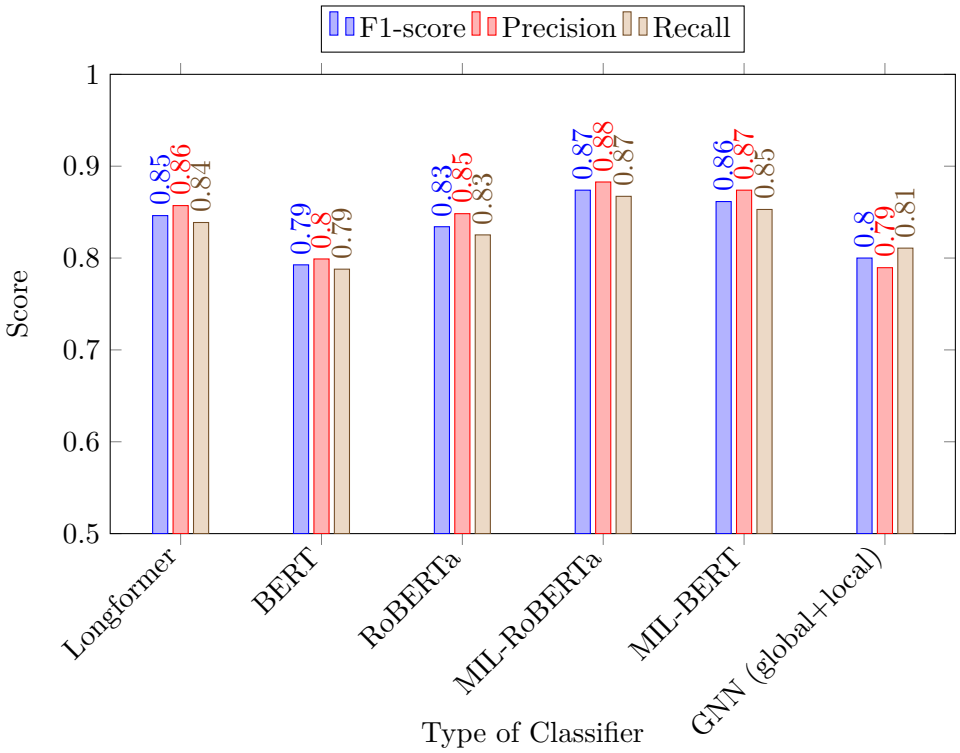


Figure 4.3: Comparison of performance of various classifiers. Performance is calculated on test split. Original values are displayed in Table A.3.

are still lower than those of the MIL-RoBERTa classifier. This suggests that the ability to process longer inputs does not automatically translate into superior performance in this context. It underscores the effectiveness of the MIL framework combined with the RoBERTa model in capturing essential information from the text. The MIL classifiers using BERT or RoBERTa outperforming the GNN classifier are intriguing. Despite GNN's complexity and superior performance on the MultiSource dataset, the MIL's straightforward approach paired with robust language models proves more effective in this context.

The experiments show that our MIL model outperforms Longformer in terms of performance, which reveals that combining our Multiple Instance Learning (MIL) model with pre-trained Transformer models limited to short inputs effectively meets one of the objectives of this work. This is to propose a solution comparable to long-input Transformer models such as the Longformer

for processing lengthy sequences. This means we can manage extensive inputs without the need to train dedicated long-input Transformer models from scratch, a process that is computationally expensive in comparison to our approach. Additionally, our methodology gains from the extensive selection of available pre-trained Transformer models.

4.4 CTKFacts

The CTKFacts dataset [DUR+22] introduces the fact-checking task, in which claims are classified based on provided evidence. This presents a challenge for language models, as the dataset consists exclusively of Czech data. As a result, we must explore alternative language models beyond BERT for document encoding. The BERT-based language models, trained on an extensive corpus of English data, are not well-suited for Czech due to grammar, vocabulary, and syntax disparities.

The language models chosen for this task can be categorized into two distinct groups based on the data used for pretraining. The first group contains multilingual models that have been trained on multiple languages, including Multilingual BERT [DCLT18], pre-trained on 104 languages, and XML-RoBERTa [CKG+19], which is finetuned on questions and answers based on Wikipedia articles, with the added challenge of unanswerable questions (SQUAD2) [RJL18]. As CTK-facts and SQUAD2 are datasets employed for natural language understanding, we deem this finetuning beneficial.

The second group of selected models are those pre-trained only on Czech data. These are FERNET-C5 and RobeCzech, which are further described in Design of Experiments section 4.1.3.

We adopt the claim and evidence processing approach presented in the thesis by Vaclav Hlavac [vac23] to enable a more precise comparison of the classifier performance. Instead of processing the input as a single, unified sequence, we handle them as pairs of input sequences separated by the specialized [SEP] token. This configuration allows for the straightforward extraction of evidence from claims within the language model, and subsequently, the encoded input

is passed to the Multiple Instance Learning (MIL) classifiers. For baseline methods, the input is truncated due to maximum length constraints. The predictions may be adversely impacted due to the MIL classifier's permutation invariance, potentially resulting in the loss of evidence significance after passing the aggregation function.

We employ a full pipeline setting by Herbert et al. [DUR+22] and retrieve evidence through a method for document retrieval. Consequently, the predictions may be affected by factors beyond the classifier's performance, such as the quality of the acquired evidence.

As a key performance metric, we selected micro and macro F1-score. The macro F1-score gives equal importance to all classes regardless of their frequency, while the micro gives equal weight to each instance, so it is more sensitive to class imbalances.

In the experiments below, we test the ability of our classifier to process long inputs by increasing the number of evidence passed together with a claim. However, each evidence piece is truncated to fit into the encoder in the language model utilized in our classifier. Our model takes as input claim which has assigned evidence from document retrieval. In case where we have evidence size 10, the MIL classifier processes the input as a bag. The bag consists of 10 instances. Each has an embedded claim, [SEP] token and one of the retrieved evidence. Together are then all evidence pieces evaluated in aggregation function, and the claim is labelled.

4.4.1 Initial Experiments

In the initial experiments, we examine the influence of employing various language models within the MIL classifier. The models are trained for 20 epochs, retrieving 10 semantically nearest pieces of evidence and utilizing either Pooling by Multihead Attention (PMA) or MAX aggregation functions. For comparative purposes, we have also established baseline classifiers that use solely language models while maintaining an identical experimental setup.

We use F1-micro and F1-macro scores to compare different models' per-

formances. The F1-micro score are displayed in Figures throughout the section, while the macro scores are in Tables in Appendix A. Results of initial experiments are in Figure 4.4. The best-performing language model is FERNET-C5, followed by XLM-RoBERTa, and the unofficial third place is shared by Multilingual BERT (M-BERT) and RobeCzech. Since FERNET-C5 and RobeCzech are the only models trained specifically on Czech datasets, we expected them to dominate. However, it's interesting to see multilingual language models like XLM-RoBERTa and M-BERT outperforming RobeCzech.

The reason for such an outcome could be that multilingual models are typically trained on vast amounts of data from multiple languages. This extensive training data enables them to learn robust representations of language features, which may generalize well to Czech, even if the model has not been explicitly fine-tuned on Czech data. Czech is a Slavic language and shares similarities with other Slavic languages in terms of grammar, vocabulary, and syntax. Multilingual models might benefit from these similarities, as they can transfer knowledge learned from one Slavic language to another.

The low performance of the original BERT model, which has been trained solely on English data, is anticipated, as it primarily serves to exemplify the potential limitations of applying a model ill-suited for the target task. Czech and English languages significantly differ in vocabulary, grammar, and syntax. It highlights the importance of employing language-specific or multilingual models when working with non-English datasets.

Additionally, numerous instances exhibit that the baseline language models yield marginally superior performance compared to our MIL classifier. Nevertheless, none of these instances demonstrate a substantial decline in the performance of the MIL classifier, except for 0.3891 F1-macro score for XML-ROBERTA in the PMA setting presented in Table A.5. Regarding the F1-score of the remaining, the difference does not exceed 0.03, constituting less than 10% of the value. This relatively minor discrepancy is reasonable, given that the MIL classifier inevitably loses some information during the aggregation step.

The classifier with the PMA aggregation function performs better on the Multisource dataset than the max aggregation function. In the case of the given task, the MAX aggregation function shows comparable performance. A

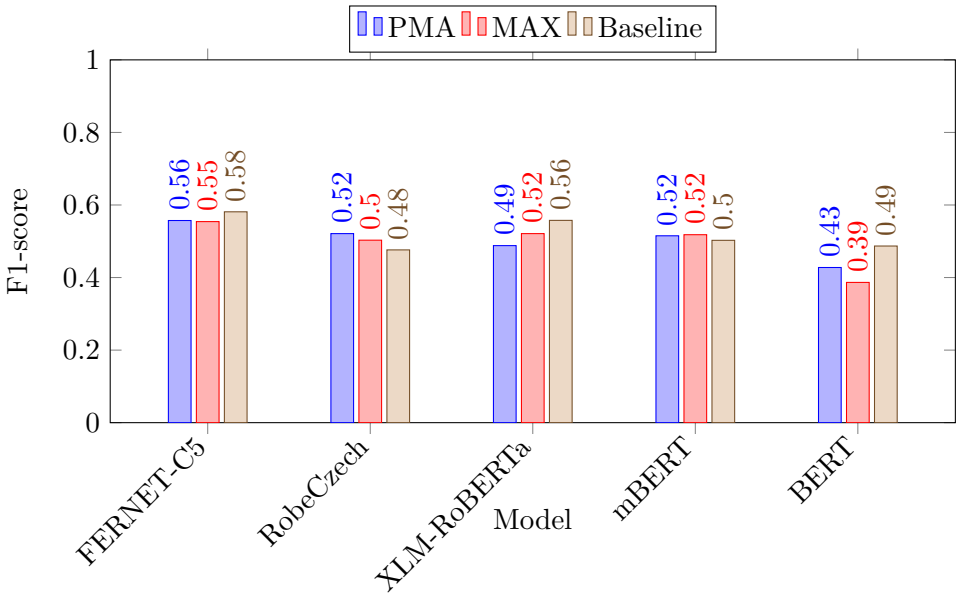


Figure 4.4: Results of F1-score (micro) on validation set for compared classifier utilizing various language models. The classifiers are MIL with PMA, MIL with max and standalone language model. Values are displayed in Table A.4.

possible explanation is the characteristics of the task at hand, which differs from the one in Multisource experiments and is not as suitable for PMA.

Because the MIL classifier with FERNET-C5 outperforms every other classifier in both micro and macro F1-scores, we will conduct further experiments only with the MIL classifier utilizing FERNET-C5.

In Figure 4.5, we compared the performance of our MIL classifier with FERNET-C5 against the baseline with FERNET-C5.

4.4.2 Size of Used Evidence

In the following experiments, we investigate the impact of varying evidence sizes on the performance of the Multiple Instance Learning (MIL) classifier. In prior experiments conducted on CTK-facts, the first 10 pieces of evidence

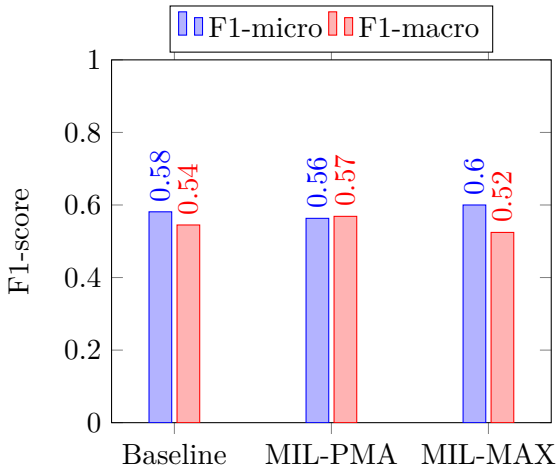


Figure 4.5: F1-scores of classifiers utilizing FERNET-C5 for test set. Values are displayed in Table A.8.

from available knowledge were used as a standard practice. However, now the number of experiments will be subject to change. The remaining experimental settings, identical to those in previous experiments, utilize the FERNET-C5 language model [LS21]for input processing in conjunction with the MIL classifier. In this case, we selected two aggregation functions in the MIL classifier. It's Pooling by Multihead Attention (PMA) and max aggregation function. For the purpose of comparative analysis and evaluation, we will employ the standalone FERNET-C5 model and the GNN classifier introduced by Vaclav Hlavac [vac23], which utilized the FERNET-C5 language model as well.

Reducing the number of evidence pieces supplied to the model might negatively affect its performance due to the limited information available. However, it is possible that using 10 pieces of evidence already introduces excessive noise into the model.

A similar issue might arise when increasing the evidence size, where additional evidence could contribute more noise to the model. Nevertheless, the model could benefit from the extra evidence if an effective aggregation function is employed.

We conducted experiments using 5, 10, 15, and 20 pieces of evidence

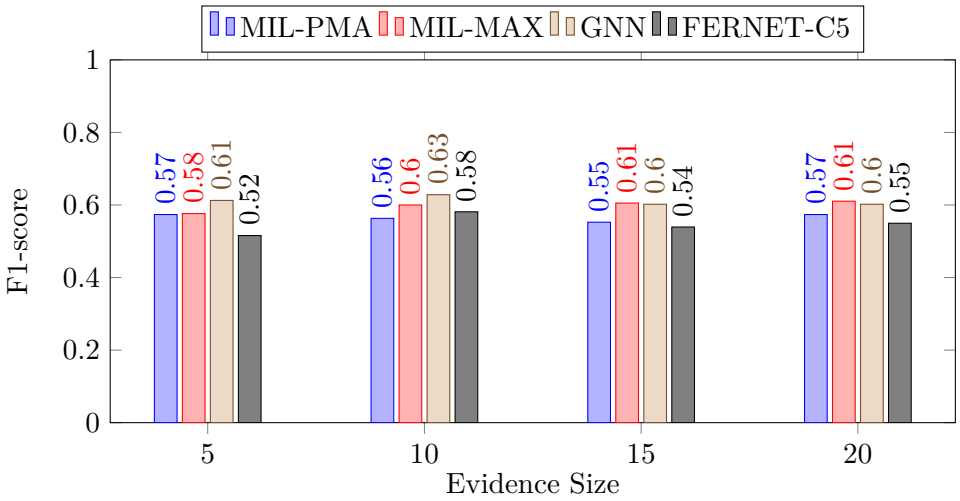


Figure 4.6: Results of F1-score (micro) on test set for compared classifiers. Values taken from Table A.6.

provided to the model. The average token counts for evidence sets are 415, 708, 1056 and 1416. The evidence was selected using the document retrieval method described in paper by Herbet et al. [DUR+22], ensuring the most semantical relevant evidence is provided. By analyzing the outcomes of these experiments, we aim to better understand the ideal balance between evidence size and model performance and gain insights into the pros and cons of different aggregation methods and alternative approaches.

The outcomes of the experiments are presented in Figure 4.6 and Table A.7, where we report both micro and macro F1-scores.

Upon comparing the two aggregation functions in our MIL classifier, the MAX function consistently outperforms the PMA function across all evidence settings. Moreover, the performance of the MAX aggregation function exhibits a slight improvement with each increase in evidence size. In contrast, the PMA function demonstrates the opposite trend, possibly due to its more complex structure being overwhelmed by the additional noise introduced by increased evidence.

Among the compared models, the best performing is GNN which stands above our MIL classifier and FERNET-C5 model. Our classifier has slightly

better performance with both aggregation functions than baseline FERNET-C5. Results show that changing the size of evidence brings only a small influence on the performance of models. Values for micro and macro F1-score correlate, and evidence settings that dominate in one metric also dominate in the other. So model's overall classification performance increases, and also it's becoming better at handling class imbalances and providing a more balanced performance across all classes.

Chapter 5

Discussion

The model we have proposed does not bring any significant improvements in terms of performance over the selected baseline methods. In some experiments, the baseline methods displayed superior performance. In this section, we discuss our solution, describe its drawbacks that might be the reason for its underperformance and suggest areas for potential improvements during further research.

We have pinpointed potential constraints that lie in embedding sections. The first one is the embedding of the input into a single vector. This constraint was already described in the GNN classifier proposed in the thesis by Vaclav Hlavac [vac23]. Our classifier shares with the presented GNN the same approach to input embeddings. This could be compared to the scenario with recurrent neural networks, where using a single vector to represent an entire input sequence had a limiting effect on their performance. However, this very feature was also pivotal in our model, enabling us to handle extended input sequences due to restricting their embeddings to only [CLS] tokens.

The second constraint of our approach is its independent handling of input sequences during the creation of their embeddings. We split the initial long input into shorter sequences and encode them individually. Because the language model does not have access to the whole initial sequence during embedding, it lacks the ability to capture the information in full context.

Our initial hypothesis expected that MIL would be able to extract important information for accurate classification from these embedded sequences and fine-tune the language model accordingly. However, the described restriction likely leads to the omission of certain information, as some input parts might only be deemed significant for the embedding once they are merged with the separated portion of the sequence. Consequently, this information might be discarded during the encoding phase.

The last constraint lies in the method for processing and combining the initially split sequences - the aggregation function. After embeddings are created, we need to aggregate over the entire set of shorter sequences (entire bag) to produce a classification of input. While experimenting with the model, we utilized different aggregation functions, namely mean, max and Pooling by Multihead Attention (PMA). Given the relative simplicity of mean and max, we anticipated that the mean and max functions might overlook certain details and fail to capture the full complexity of relationships between instances within a bag. The inclusion of the PMA function was envisaged to enhance the classifier's expressivity due to the utilization of the attention mechanism in PMA. However, the majority of experiments displayed PMA's performance as being on par with simpler alternatives, with only a handful of instances exhibiting marginal performance improvements. The underperformance of PMA may be attributed to the tasks' incompatibility or a deficiency in our PMA training process. Consequently, we suggest future work to focus on further experimentation with the PMA aggregation function.

Apart from comparing our classifier with baseline models, we also compared it against the GNN classifier introduced in the thesis by Vaclav Hlavac [vac23]. Our solutions share some similarities, and both have been developed concurrently under the supervision of Ing. Jan Drchal, Ph.D. The GNN solution has been closely introduced in its section in Methodology chapter 3. It shares the same approach to the initial division of long input and embeddings of its sequences, but the sequences are further passed to a graph neural network. While our solution surpassed in performance the GNN classifier on the Hyperpartisan News Detection dataset, their performance was comparable on the CTK Facts dataset. In contrast, on the Multisource dataset, the GNNs presented superior performance. One possible explanation is that they can capture more complex relationships and interdependencies between data points as they consider the entire topology of the graph, thus achieving better expressivity. The reason behind the alternating performance

between GNN and MIL can be explained by different characteristics of tasks and datasets, where the described advantage of GNN can not be exploited equally in each experiment.

On the other hand, we have demonstrated that by using our classifier, it is possible to utilize already pre-trained language models for short inputs to process long inputs. Such a feature can be exploited, for example, in the case of low-resource languages like Czech, where instead of training long input language models like Longformer [BPC20] from the ground up, we can utilize already existing Czech language models.

Chapter 6

Conclusion

In this thesis, our objective was to develop a sequence classification solution that could transcend the constraints of the Transformer-based architecture and therefore process inputs longer than 512 tokens.

The initial portion of our work involved a comprehensive review of related literature, focusing on pivotal areas such as Multiple Instance Learning, Natural Language Processing, Transformer models and their long input adaptations, and the overarching task of text classification. These concepts were integral to understanding and approaching the complex problem at hand.

From this theoretical base, we presented our problem statement, which grappled with the challenge of classifying long sequences that exceeded the constraints of traditional Transformer models. In response, we proposed a novel MIL-based classifier for processing long inputs. The initial input is split into smaller sequences to form a bag. The classifier then leverages pre-trained language models for individual sequence embedding and processes these embeddings using an aggregation function to predict a bag-level label representing the initial input.

To evaluate the performance of our proposed solution, we tested our

classifier on three distinct datasets, each representing a different task. The performance of our model was evaluated on different classifier setups and compared against baseline methods and a concurrent solution utilizing GNNs. The results exhibited the ability of our model to process long sequences of up to 13,296 tokens. However, we have not achieved significantly superior performance against mentioned methods. The experimental phase of our work was crucial for validating our approach and uncovering areas of strength and weakness within our model.

In the final stage of our work, we discussed our model's capabilities and limitations. This analysis aimed to provide a balanced perspective on the model, offering insights into the areas that could benefit from further research and optimization. Overall, our work presents a novel approach to sequence classification, with the potential for refinement and extended application in future studies.

Bibliography

[ATKS19] Mikhail Arkhipov, Maria Trofimova, Yuri Kuratov, and Alexey Sorokin, *Tuning multilingual transformers for language-specific named entity recognition*, Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing (Florence, Italy), Association for Computational Linguistics, August 2019, pp. 89–93.

[Bab08] Boris Babenko, *Multiple instance learning: algorithms and applications*, View Article PubMed/NCBI Google Scholar **19** (2008).

[Blo17] Google AI Blog, *Transformer: A novel neural network architecture for language understanding*, 2017, Accessed: 2023-05-17.

[BPC20] Iz Beltagy, Matthew E. Peters, and Arman Cohan, *Longformer: The long-document transformer*, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 8736–8746.

[CFWB17] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes, *Reading wikipedia to answer open-domain questions*, arXiv preprint arXiv:1704.00051 (2017).

[Cho57] Noam Chomsky, *Syntactic structures*, Mouton, 1957.

[CKG+19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov, *Unsupervised cross-lingual representation learning at scale*, 2019.

[CLD+20] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller, *Rethinking attention with performers*, ArXiv **abs/2009.14794** (2020).

[CZL+17] Zhe Chen, Hexiang Zhang, Yuxiao Lin, Liangjie Tao, Hanwang Zhang, Shaoting Luo, Yan Yan, and Yu-Gang Jiang, *Multi-level attention networks for visual question answering*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4187–4195.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018).

[DLLP97] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez, *Solving the multiple instance problem with axis-parallel rectangles*, Artificial intelligence **89** (1997), no. 1-2, 31–71.

[DUR+22] Jan Drchal, Herbert Ullrich, Martin Rỳpar, Hana Vincourová, and Václav Moravec, *Csfever and ctkfacts: Czech datasets for fact verification*, arXiv preprint arXiv:2201.11115 (2022).

[GSV22] Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos, *A survey on automated fact-checking*, Transactions of the Association for Computational Linguistics **10** (2022), 178–206.

[Hut04] W. John Hutchins, *The georgetown-ibm experiment demonstrated in 1954*, Machine Translation: From Real Users to Research, 2004, pp. 29–40.

[KJMH+19] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown, *Text classification algorithms: A survey*, Information **10** (2019), no. 4, 150.

[KMS+19] Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, David Corney, Payam Adineh, Benno Stein, and Martin Potthast, *Data for pan at semeval 2019 task 4: Hyperpartisan news detection.*

[KZ20] Omar Khattab and Matei Zaharia, *Colbert: Efficient and effective passage search via contextualized late interaction over bert*, Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 39–48.

[LBE15] Zachary C Lipton, John Berkowitz, and Charles Elkan, *A critical review of recurrent neural networks for sequence learning*, arXiv preprint arXiv:1506.00019 (2015).

[LH17] Ilya Loshchilov and Frank Hutter, *Decoupled weight decay regularization*, arXiv preprint arXiv:1711.05101 (2017).

[LLK+19] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh, *Set transformer: A framework for attention-based permutation-invariant neural networks*, International conference on machine learning, PMLR, 2019, pp. 3744–3753.

[LQH16] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang, *Recurrent neural network for text classification with multi-task learning*, arXiv preprint arXiv:1605.05101 (2016).

[LS21] Jan Lehecka and Jan Svec, *Comparison of czech transformers on text classification tasks*, CoRR **abs/2107.10042** (2021).

[LXZ17] Yiding Lu, Xiangyang Xue, and Lei Zhang, *Deep multiple instance learning for image classification and auto-annotation*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017), 2350–2358.

[Min23] Minds Mines, *pdmisvm.jl*, 2023, Accessed: 2023-05-17.

[MKC+21] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao, *Deep learning–based text classification: a comprehensive review*, ACM computing surveys (CSUR) **54** (2021), no. 3, 1–40.

[MS99] Christopher D. Manning and Hinrich Schütze, *Foundations of statistical natural language processing*, MIT Press, 1999.

[NOMC11] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman, *Natural language processing: an introduction*, Journal of the American Medical Informatics Association **18** (2011), no. 5, 544–551.

[PS17] Tomáš Pevnỳ and Petr Somol, *Using neural network formalism to solve multiple-instance problems*, Advances in Neural Networks-ISNN 2017: 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21–26, 2017, Proceedings, Part I 14, Springer, 2017, pp. 135–142.

[RG20] Nils Reimers and Iryna Gurevych, *Making monolingual sentence embeddings multilingual using knowledge distillation*, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 11 2020.

[RJL18] Pranav Rajpurkar, Robin Jia, and Percy Liang, *Know what you don't know: Unanswerable questions for squad*, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018, pp. 784–789.

[RNS+18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al., *Improving language understanding by generative pre-training.*

[SNSS21] Milan Straka, Jakub Naplava, Jana Strakova, and David Samuel, *RobeCzech: Czech RoBERTa, a Monolingual Contextualized Language Representation Model*, Text, Speech, and Dialogue (Cham) (Kamil Ekstein, Frantisek Partl, and Miloslav Konopik, eds.), Springer International Publishing, 2021, pp. 197–209.

[TVCM18] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal, *FEVER: a large-scale dataset for fact extraction and VERification*, Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (New Orleans, Louisiana), Association for Computational Linguistics, June 2018, pp. 809–819.

[vac23] *Metody grafovych neuronovych siti pro zpracovani dlouhych vstupu nlp modelů*, Master's thesis, Ceske vysoke uceni technicke v Praze. Vypocetni a informacni centrum, 2023.

[VRA18] Soroush Vosoughi, Deb Roy, and Sinan Aral, *The spread of true and false news online*, Science **359** (2018), no. 6380, 1146–1151.

[VSP+17] 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).

[WDS+20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew, *Transformers: State-of-the-art natural language processing*, `https://github.com/huggingface/transformers`, 2020.

[YFL17] Peilin Yang, Hui Fang, and Jimmy Lin, *Anserini: Enabling the use of lucene for information retrieval research*, Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, 2017, pp. 1253–1256.

[ZGD+20] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al., *Big bird: Transformers for longer sequences*, Advances in neural information processing systems **33** (2020), 17283–17297.

[Zho04] Zhi-Hua Zhou, *Multi-instance learning: A survey*, Department of Computer Science & Technology, Nanjing University, Tech. Rep **1** (2004).

Appendix A

Additional Results

Showcase of Hyperpartisan News Detection Dataset


```
{'text': '<p>Money ( <a href="https://farm8.static.flickr.com
    /7020/6551534889_9c8ae52997.jpg" type="external">Image</a
    > by <a href="https://www.flickr.com/people/68751915@N05
    /" type="external">401(K) 2013</a>) <a href="https://
    creativecommons.org/licenses/by-sa/2.0/" type="external">
    Permission</a> <a type="internal">Details</a> <a type="
    internal">DMCA</a></p> No Pill Can Stop Tinnitus, But
    This 1 Weird Trick Can \n<p>The walls are closing in on
    Congress.</p> \n<p>Terrifying walls of water from
    Hurricanes Harvey and Irma, which, when the damage is
    totaled, could rise to a half trillion dollars. The Walls
     of War: The multi-trillion dollar ongoing cost of
    Afghanistan, Iraq and other interventions. The crumbling
    walls of the U.S. infrastructure, which need at least $3
    trillion to be repaired or replaced. A wall of 11 million
     undocumented immigrants, whose deportation could easily
    cost $200 billion. The planned wall at the Mexican border
    , which some estimates place at $67 billion. Then there
    is the Wall of All, the $20 trillion national debt. The
    walls of debt are closing in.</p> \n<p>At moments of
    crisis in our nation, in addition to invoking the
    assistance of Higher powers, we can call upon the
    Constitution for guidance.</p> \n<p>Article I, Section 8,
     of the U.S. Constitution contains a long-forgotten
    provision, "the coinage clause," which empowered Congress
     "to coin (create) Money." The ability to create money to
     meet the needs of the nation is a sovereign power, which
     enables a nation to have control of its own destiny.</p>
     \n<p>The same article indicates the Founders anticipated
     having to borrow money on the full faith and credit of
    the United States. Enter the Funding , ....',
'title': 'Kucinich: Reclaiming the money power',
'hyperpartisan': True,
'url': 'https://www.opednews.com/articles/Kucinich-Reclaiming
    -the-m-by-Dennis-Kucinich-Banks_Debt_Funding_Money
    -170910-112.html',
'published_at': '2017-09-10'}
```


Figure A.1: Example of instance in Hyperpartisan News Detection dataset.

Multisource - Additional Results

Here are presented tables used for bar chart construction in Experiments 4.

RND	PMA-w:1	MAX-w:1	GNN-w:1	PMA-w:2	MAX-w:2	GNN-w:2
2	0.6335	0.572	**0.639**	0.699	0.638	**0.7285**
4	0.7175	0.6775	**0.7570**	**0.851**	0.786	0.8315
6	**0.814**	0.748	0.8125	0.9055	0.898	**0.907**
8	0.866	0.8095	**0.874**	**0.95**	0.9265	0.923

Table A.1: Accuracy of end-to-end classifiers with different size of sliding windows on datasets with changing number of random sentences. The 'w:N' represents size of window, where N is the number of instances in window.

Size of sliding window	MIL-PMA	MIL-MAX	GNN
1	0.7635	0.704	**0.808**
2	**0.856**	0.838	0.848
3	**0.875**	0.851	0.8485
4	**0.8915**	0.8808	0.876

Table A.2: Accuracy of end-to-end classifiers with different size of sliding windows.

Hyperpartisan News Detection - Additional Results

Here are presented tables used for bar chart construction in Experiments 4.

Type of classifier	F1-score	Precision	Recall
Longformer	0.8462	0.8571	0.8387
BERT	0.7926	0.799	0.7879
RoBERTa	0.8341	0.8483	0.8252
MIL-RoBERTa	**0.8739**	**0.8828**	**0.8672**
MIL-BERT	0.8615	0.8739	0.8529
GNN (global+local)	0.8	0.7895	0.8108

Table A.3: Comparison of performance of various classifiers. Performance is calculated on test split.

CTK Facts - Additional Results

Here are presented F1 micro and macro scores tables from experiments on CTK Facts dataset.

Model	PMA	MAX	Baseline
FERNET-C5	0.5572	0.5542	**0.5812**
RobeCzech	**0.5211**	0.503	0.4762
XLM-RoBERTa	0.488	0.5211	**0.5576**
mBERT	0.5151	**0.5181**	0.5026
BERT	0.4277	0.3866	**0.4869**

Table A.4: Results of F1-score (micro) on validation set for compared classifier utilizing various language models. The classifiers are MIL with PMA, MIL with max and standalone language model.

Model	PMA	MAX	Baseline
FERNET-C5	0.5521	**0.5524**	0.5375
RobeCzech	0.4637	0.4616	**0.4999**
XLM-RoBERTa	0.3891	0.5165	**0.5345**
mBERT	**0.4995**	**0.4955**	0.4622
BERT	0.3766	0.369	**0.3988**

Table A.5: Results of F1-score (macro) on validation set for compared classifier utilizing various language models. The classifiers are MIL with PMA, MIL with max and standalone language model.

Evidence Size	MIL-PMA	MIL-MAX	GNN	FERNET-C5
5	0.5737	0.5763	**0.6126**	0.5157
10	0.5632	0.6	**0.6283**	0.5812
15	0.5526	**0.6053**	0.6021	0.5393
20	0.5737	**0.6105**	0.6021	0.5497

Table A.6: Results of F1-score (micro) on test set for compared classifers

Evidence Size	MIL-PMA	MIL-MAX	GNN	FERNET-C5
5	0.5531	0.5454	**0.5762**	0.4794
10	0.5244	0.5687	**0.5959**	0.5449
15	0.5108	**0.5712**	0.5606	0.5019
20	0.5243	**0.5812**	0.5545	0.5213

Table A.7: Results of F1-score (macro) on test set for compared classifers

Model	F1-micro	F1-macro
Baseline	0.5812	0.5449
MIL-PMA	0.5632	**0.5687**
MIL-MAX	**0.6**	0.5244

Table A.8: F1-scores of classifiers utilizing FERNET-C5 for test set.