Cristiano Mesquita Garcia

Concept Drift Adaptation for SentenceBERT Models in Text Stream Classification Settings

DOUTORADO EM
INFORMÁTICA
PUCPR

CURITIBA
2025

Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

Concept Drift Adaptation for SentenceBERT Models in Text Stream Classification Settings

Cristiano Mesquita Garcia

Supervisor

Jean Paul Barddal

Co-supervisor

Alceu de Souza Britto Jr

Curitiba
2025

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Concept Drift Adaptation for SentenceBERT Models in Text Stream Classification Settings

Cristiano Mesquita Garcia

Thesis presented to the *Programa de Pós-Graduação em Informática* as a partial requirement for the degree of Doctor in Informatics.

MAJOR FIELD: Computer Science

SUPERVISOR: JEAN PAUL BARDDAL

CO-SUPERVISOR: ALCEU DE SOUZA BRITTO JR

CURITIBA
2025

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Edilene de Oliveira dos Santos CRB 9 / 1636

G216c
2025

Garcia Cristiano Mesquita
Concept Drift Adaptation for SentenceBERT Models in Text Stream Classification Settings / Cristiano Mesquita Garcia ; supervisor: Jean Paul Barddal ; co-supervisor: Alceu de Souza Britto Jr. -- 2025
162 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2025.
Bibliografia: f. 140-147

1. Informática. 2. Tecnologia streaming (Telecomunicação). 3. Estrutura de dados (Computação). 4. Aprendizado do computador. I. Barddal,Jean Paul. II. Britto Junior, Alceu de Souza, 1966-. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 23 de janeiro de 2026.

01-2026

DECLARAÇÃO

Declaro para os devidos fins, que **CRISTIANO MESQUITA GARCIA** defendeu a tese de Doutorado intitulada "**Concept Drift Adaptation for SentenceBERT Models in Text Stream Classification Settings"**, na área de concentração Ciência da Computação no dia 27 de novembro de 2025, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

gov.br
Documento assinado digitalmente
JEAN PAUL BARDDAL
Data: 30/01/2026 14:00:22-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Jean Paul Barddal
Coordenador do Programa de Pós-Graduação em Informática

To God, my wife Thati and my son Miguel (whose little face I'm looking forward to see), my parents Rudimar and Luisa, my sister Marcela, my grandparents Manoel (Nelo) (in memoriam), Nenzinha, Lourival (in memoriam), and Diva (in memoriam), and my godchildren Maria Luisa and Pedro.

"Faith shows the reality of what we hope for; it is the evidence of things we cannot see."
– Hebrews 11:1

"We ourselves feel that what we are doing is just a drop in the ocean. But the ocean would be less because of that missing drop." – Mother Teresa of Kolkata

Abstract

Since its popularization, the Internet has been a rich data source. People use the Internet for activities including posting comments, reacting to news, and reviewing products and services. Such activities create textual streams, which can provide insights into public opinion and opinions about particular services, entities, products, and so on. This thesis focuses on the adaptation of SentenceBERT models for textual data stream classification, in particular, in the development of adaptive methods that provide updated vector representations in order to overcome the necessity of retraining models from scratch, while also detecting and adapting to changes in data distribution, a phenomenon named concept drift. In textual data streams, concept drift can appear in different forms, and deteriorate the performance of machine learning models, which makes these models lose their utility. Two methods are proposed in this thesis: (a) a method for adapting SBERT-based models to concept drift by enriching an external vocabulary with newly learned words, in which the representations are generated using the distributional hypothesis; and (b) a method based on a pool of SBERT pre-trained models that prevent abrupt updates in the language model representations, named Bi-SBERT. The proposed methods were evaluated in terms of Macro F1-Score and elapsed time, using six different datasets under four types of synthetic drifts. The ADWIN concept drift detector was applied in the streaming scenario to monitor changes in performance. The methods with the distributional-hypothesis-based method were statistically compared to their original version. Results suggest that the distributional-hypothesis-based method can provide better representations in scenarios with higher rates of unknown words, e.g., the complete word is not available as a single token in the language model's vocabulary. Besides, for Bi-SBERT, the results suggest an actual improvement in the Macro F1-Scores, reaching the best rank in around 42% of the 26 scenarios, compared to 10 methods, including AdaNEN, a state-of-the-art method.

Key-words: Concept drift adaptation, Text streaming scenarios, Pre-trained language models, Language model adaptation

Resumo

Desde sua popularização, a Internet tem sido uma rica fonte de dados. As pessoas usam a Internet para atividades que incluem postar comentários, reagir a notícias e avaliar produtos e serviços. Essas atividades criam fluxos textuais, que podem fornecer insights sobre a opinião pública e opiniões sobre determinados serviços, entidades, produtos e assim por diante. Esta tese foca na adaptação de modelos SentenceBERT para classificação de fluxos de dados textuais, em particular, no desenvolvimento de métodos adaptativos que fornecem representações vetoriais atualizadas para superar a necessidade de retreinar modelos do zero, ao mesmo tempo em que detectam e se adaptam a mudanças na distribuição de dados, um fenômeno denominado *concept drift*. *Concept drift* pode se manifestar de diferentes formas e deteriorar o desempenho dos modelos de aprendizado de máquina, o que faz com que esses modelos percam sua utilidade ao longo do tempo. Dois métodos são propostos: (a) um método para adaptar modelos baseados em SBERT à *concept drift*, enriquecendo um vocabulário externo com palavras recém-aprendidas, no qual as representações são geradas usando a hipótese distribucional; e (b) um método baseado em um conjunto de modelos SBERT pré-treinados que previnem atualizações abruptas nas representações do modelo de linguagem, denominado Bi-SBERT. Os métodos propostos foram avaliados em termos de Macro F1-Score e tempo de execução, usando seis diferentes conjuntos de dados sob quatro tipos de *drifts* sintéticos. O detector de *concept drift* ADWIN foi aplicado no cenário de *streaming* para monitorar mudanças no desempenho. Os classificadores com o método baseado em hipótese distribucional foram comparados estatisticamente com sua versão original. Os resultados sugerem que o método baseado em hipótese distribucional pode fornecer melhores representações em cenários com maiores taxas de palavras desconhecidas, e.g., palavras as quais o token completo não está disponível no vocabulário do modelo de linguagem. Além disso, para o Bi-SBERT, os resultados sugerem uma melhora real nos Macro F1-Scores, alcançando a melhor classificação em cerca de 42% dos 26 cenários, em comparação com 10 métodos incluindo AdaNEN, um método de estado da arte.

Key-words: Adaptação à *concept drift*, Cenário de *text streaming*, Modelos de linguagem pré-treinados

Acknowledgements

Primeiramente, agradeço a Deus, pelo dom da vida e pelas pessoas que Ele colocou no meu caminho até aqui;
- Agradeço à minha esposa Thati, pois o *projeto* de obtenção do título de doutor só foi possível também por conta de seu amor e sacrifício diário;
- Agradeço a meus pais Rudimar e Luisa e minha irmã Marcela, pelo incentivo incondicional, desde sempre; à Maria Luisa, minha afilhada, por mostrar um tipo de amor que eu ainda não conhecia; à minha avó Nenzinha, pelas orações;
- Agradeço a meus orientadores, profs. Jean e Alceu, que desde o primeiro contato, foram muito receptivos, solícitos, incentivadores, pacientes, nunca hesitando em compartilhar sabedoria, conhecimento, esperança e bom humor, e os quais considero amigos;
- Agradeço aos amigos que, de perto ou de longe, acompanham a construção desse caminho, em especial Ramon, Xande, Diego Catalano, Danilo, Will, Padre Clayton, Ademir e Paulo. Aos colegas do PPGIa, em especial Ricardo Martins, e ao Rômulo, que infelizmente não continuou no programa;
- Agradeço à Pontifícia Universidade Católica do Paraná (PUCPR), por proporcionar uma educação de qualidade e pela isenção das mensalidades;
- Agradeço à CAPES e à sociedade brasileira pela bolsa de doutorado sanduíche na ÉTS, Montréal, sob a orientação do prof. Alessandro Koerich, a quem também agradeço muito pela oportunidade;
- Agradeço aos demais professores do PPGIa, pelo conhecimento compartilhado, suporte e zelo com a docência e com os estudantes;
- Agradeço ao Instituto Federal de Santa Catarina (IFSC) pelo afastamento para pós-graduação com vencimentos integrais;
- Agradeço às minhas cachorrinhas, Neve e Sol, que apesar de me desconcentrarem algumas vezes ao longo do dia, foram companhias nas tardes e noites solitárias de trabalho.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AdaNEN	Adaptive Neural Ensemble Network
ADWIN	Adaptive Windowing
ARL	Average run length
BERT	Bidirectional Encoder Representations from Transformers
CBOW	Continuous bag-of-words
CNN	Convolutional Neural Network
CSV	Comma Separated Values
CUSUM	Cumulative sum
DHV	Distributional-Hypothesis-based vocabulary
EWMA	Exponentially Weighted Moving Average
HDDM	Hoeffding's bounds Drift Detection Method
IWV	Incremental Word-Vectors
LLM	Large Language Model
LSTM	Long short-term Memory
MDR	Missing detection rate
ML	Machine Learning
MLM	Masked language modeling
MTD	Mean time to detection

MTFA Mean time between False Alarms

MTR Mean time rate

NLP Natural Language Processing

NN Neural network

NSP Next sentence prediction

PMI Pointwise mutual information

PPMI Positive pointwise mutual information

RDDM Reactive Drift Detection Method

SBERT SentenceBERT

SPC Statistic Process Control

SSA Spave-saving Algorithm

SVM Support Vector Machine

TF-IDF Term frequency - Inverse of document frequency

VFDT Very Fast Decision Tree

1 Introduction

Data, the raw material for machine learning applications, have been continuously produced by humans and machines. People use, for example, social media platforms to post text, photos, and videos. In addition, social media users can express opinions, re-post and comment news. Some authors, including Suprem & Pu (2019), mention social media posts as *human sensors*. It is important to highlight that people generate data through product and service reviews, while machines can produce log files with performance information and collect data from sensors, for example. When learning from those types of data, machine learning algorithms can generate models for several tasks, such as classification.

Learning from textual data is a challenging task. It depends on several preprocessing steps, such as text cleansing, which may include special characters removal and reduction operations using *stemming* or *lemmatization*, and transformations to a text representation. Additionally, different machine learning algorithms may receive different vector representations of texts as input. Although some methods use words as input, text representation is crucial since most machine learning techniques can only receive numerical vectors as input. According to the systematic literature review listed in Section 1.5 (second bullet) (GARCIA et al., 2025), text representations are categorized into (i) frequency-based, (ii) embedding, and (iii) words. Embeddings, provided by pre-trained language models or large language models (LLMs), are considered the state-of-the-art in text vectorization methods.

Text-based intelligent systems are often regarded in batch scenarios. This means that, using a fixed textual dataset, a machine learning model is trained, leveraging the patterns present in the textual dataset but incapable of learning new patterns. Therefore, a static model is used for a given downstream task. However, this work is particularly concerned with evolving scenarios. According to Gama et al. (2014), data generally evolve over time. The data evolution regards changes in data distribution, called *concept drift*. Detecting and adapting to concept drifts is relevant since such changes may render models obsolete due to the fact that the concept previously learned does not represent reality once a change occurs. Concept drift detection/adaptation is a relevant challenge since a given trained model may not represent reality under the occurrence of concept drift.

Considering a data stream scenario, additional difficulties and constraints appear. Bifet et al. (2018) state that data streams are "an algorithmic abstraction to support real-time analytics". Some of the characteristics of data streams are: (i) data arrive sequentially and continuously, (ii) data arrive fast, and (iii) the data streams can be potentially infinite (GAMA et al., 2014; BIFET et al., 2018). Regarding these constraints, Gama et al. (2014) and Bifet et al. (2018) list requirements for machine learning models to properly function in data streaming scenarios, that include: (i) learning from data as it arrives, (ii) discarding data within a short period after learning from it, and (iii) performing single-pass operations. Furthermore, the machine learning system in this scenario must consume limited memory and storage. New challenges are added in a scenario of text streams, a specialization of data streams. For instance, vocabulary maintenance and text cleansing become challenging since they must be fast and occur in a single pass.

According to a systematic literature review performed as part of this study (GARCIA et al., 2025), most works in the recent literature focus on updating the machine learning model to address concept drift adaptation. The systematic literature review also pointed out that the approaches that resort to pre-trained language models maintained them without an update. Language is ever-evolving, and word meanings and contexts may change in a short period, such as weeks (STEWART et al., 2017) or even hours (GARCIA; Britto Jr; BARDDAL, 2023). Consequently, neglecting the phenomenon of concept drift may negatively impact the performance of machine learning models dependent on up-to-date text representations. Therefore, this thesis proposes two concept drift adaptation strategies for textual data streams scenarios. Particularly, this thesis proposal focuses on pre-trained SentenceBERT (SBERT) (REIMERS; GUREVYCH, 2019), a siamese architecture based on Bidirectional Encoder Representations from Transformers (BERT) (DEVLIN et al., 2018) models. BERT (DEVLIN et al., 2018) is a pre-trained deep-learning-based language model capable of learning contextual vector representations. In addition, BERT has a unified architecture, suitable for several downstream tasks with little modifications. BERT has been chosen since it is considered state-of-the-art for language modeling. In addition, pre-trained versions of BERT can provide good quality representations for text stream classification task, attested in benchmark comparisons (THUMA et al., 2023). However, fine-tuning a BERT model can be computationally costly, since it depends on a reasonable amount of data, and can take several hours to run (TAI et al., 2020). Therefore, to effectively resort to BERT models in text stream settings, new adaptation methods for BERT are necessary to avoid creating a bottleneck in the stream learning process.

Although BERT was released in 2018, BERT-based models are still competitive as of the moment of drafting this document (2025), especially considering specific tasks (JAMSHIDI et al., 2024; WANG et al., 2025), such as text classification. There is some discussion on the requirements for a language model to be regarded as *large*. Some research communities mention BERT as a large language model (LLM). In contrast, others consider LLMs as models generally constituted of billions of parameters and able to execute tasks based on prompts, sometimes in a zero-shot fashion (RAINA; LIUSIE; GALES, 2024; KOJIMA et al., 2022), thus excluding BERT.

In addition, it is acknowledged that using only SentenceBERT as a language model is a limitation of this thesis; however, it was not feasible to reproduce the experiments and test different language models due to both hardware and time limitations. On the other hand, extending the analyses performed in this thesis to other language models and LLMs is intended to be approached in future works. Another important aspect is that, although nowadays there are embeddings based on encoders and those based on decoders, embeddings based on encoders, such as BERT, are more suitable for generating semantically meaningful embeddings. Thus, to indeed leverage decoder-only to generate good quality embeddings for similarity or classification, additional steps may be required, such as those demonstrated by BehnamGhader et al. (2024), i.e., using bidirectional attention, fine-tuning procedure based on the masked next token prediction, and using unsupervised contrastive learning. Therefore, this thesis focuses on SentenceBERT, an encoder-based model, due to its intrinsic ability to generate semantically meaningful embeddings.

1.1 Problem Statement

As mentioned above, approaches generally focus on updating the machine learning model to overcome the concept drift problem in text stream scenarios. However, it is well-known that language is ever-changing: for instance, the way people communicated in the past differs from how people communicate nowadays, and words have their meanings changing over time (see Section 2.2). Different types of concept drift can exist in a textual data stream (HEUSINGER; RAAB; SCHLEIF, 2020b). As explored in Section 2.2, concept drifts can be categorized in terms of types and dynamics. In addition, a specific type of drift, i.e., semantic shift, emerges in textual data streams. Semantic shift regards the change of words' meanings over time. A number of papers address this problem of detecting changes in words' meaning over time (BELOTTI; BIANCHI; PALMONARI, 2020; RYZHOVA; RYZHOVA; SOCHENKOV, 2021), which generally comprise long time spans, i.e., decades and centuries. However, recent works (STEWART et al., 2017; BRAVO-MARQUEZ; KHANCHANDANI; PFAHRINGER, 2022a) demonstrated that these changes could occur in shorter periods, such as weeks. In addition, **paper #3** (GARCIA; Britto Jr; BARDDAL, 2023) in Section 1.5 suggests that positive/negative connotations can be acquired even faster, within a few hours. Overcoming or mitigating the effects of concept drift in textual data streams is crucial because it prevents a degradation in the vector representation quality. By doing this, it makes the intelligent system ready for the real world, where there are changes in social and cultural contexts, such as the appearance of new words and the emergence of new topics.

A simplified incremental learning setting using pre-trained language models in a textual stream classification task commonly works as in Fig. 1.1. Each arriving text t is preprocessed (becoming t') and converted to a vector representation X by a pre-trained language model. Next, the vector representation is applied to a machine learning model, that predicts a class y'. In this case, a concept drift detector that triggers the update of the machine learning model in case of concept drift is considered. The concept drift detector is updated using the values of y and y'. The language model may not be updated in this sort of scenario due to the computational cost of a fine-tuning process and the delay that it causes in the learning process. In Fig. 1.1, the yellow cubes, also indicated with a fire emoji, are updated, and the blue ones (with an ice cube emoji) are used statically. Although the language model is frequently static, it may undergo updates over time; this is the reason for the existence of the ice cube and fire emojis in the language model box. It is important to notice that, the scenario proposed and illustrated by Fig. 1.1 is supervised, i.e., it is label-dependent. In addition, it is assumed that the label is available as soon as the classifier performs a prediction. Furthermore, although the scenario is supervised, the concept drift detector operates unsupervisedly, measuring the average of hits/errors.

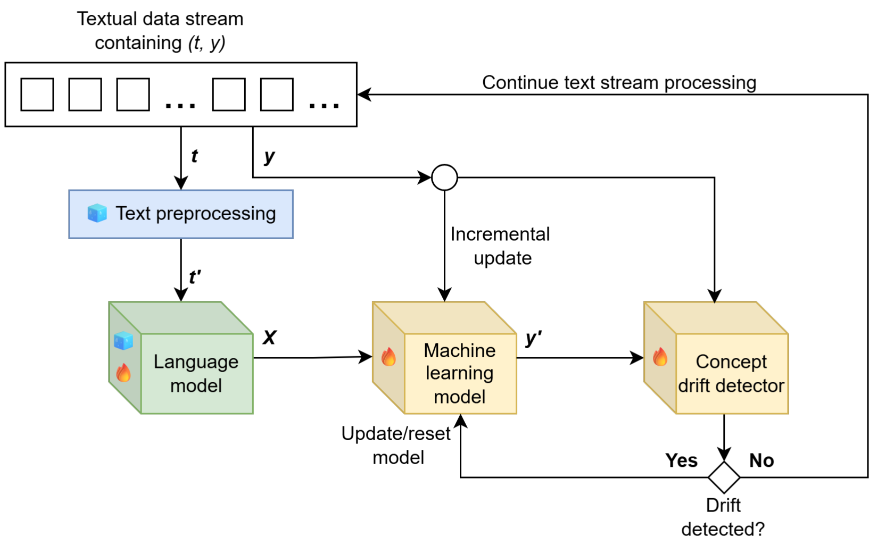

Figure 1.1: Incremental learning setting. Components marked with an ice cube are/can be static, while those marked with the fire emoji are adaptive. In this scenario, data arrive in the form of a stream, are processed, and vectorized by a language model. The vectorized text is used as input for the machine learning model. Whenever y is available, the concept drift detector is updated, considering also the machine learning model prediction. If the machine learning model is incremental, it is updated considering y' and y; however, in the case of concept drift detection, if the machine learning model is batch-based, it could be updated/reset.

As verified in the systematic review (GARCIA et al., 2025), the update of pre-trained language models is frequently overlooked. This thesis addresses concept drift adaptation for SBERT-based pre-trained models in text streaming environments. The main objective is to evaluate the impacts of the distributional hypothesis on the language model adaptation over time. The distributional hypothesis (HARRIS, 1954) expresses the following: "Words that occur in the same contexts tend to have similar meanings". Thus, this thesis presents a distributional-hypothesis-based method that generates vector representations based on the representations of the surrounding words. Maintaining updated representations of new words could help improve the performance of the machine learning model in the downstream task. In a language model, a way of representing unknown words is to combine wordpieces (WU et al., 2016), whose meanings could be sometimes conflicting.

At this point, changes in writing patterns, for example, can lead to changes in the surrounding words of a given word. Those changes can result in concept drift, semantic shift, and so on. Thus, considering the distributional hypothesis, those changes in a given word's surrounding words could be used as a proxy to suggest a [language] model update.

Therefore, the research question (RQ) that guided this work was: "Does the distributional hypothesis help reach concept drift adaptation for BERT-based language models in textual data stream classification scenarios?"

1.2 Objectives

The general objective is to develop a method that effectively updates or extends a pre-trained language model to generate up-to-date representations in textual data stream settings subject to concept drift. Particularly, this thesis focuses on pre-trained SBERT models. To reach the general objective, the proposed method leverages SBERT's representations as a basis to generate new word representations over time, tackling concept drift.

The specific objectives are:

- To perform a systematic literature review to understand the state-of-the-art on concept drift adaptation in textual data stream settings;
- To execute experiments using pre-trained and online learning models to determine a baseline regarding textual data stream classification;
- To develop methods based on distributional-hypothesis to complement a pre-trained model, and based on the combination of representations calculated over time in continuous processing of textual data stream settings;
- To evaluate the proposed methods and compare them against different methods, including AdaNEN (GHAHRAMANIAN et al., 2024), a state-of-the-art method.

The specific objectives have been developed and performed only for the text stream classification task.

1.3 Hypothesis

Considering: (a) the data stream characteristics, i.e., data arrive fast, sequentially, and the stream is potentially infinite; (b) the desired characteristics of machine learning models, e.g., learn data one-by-one or in small batches and perform single-pass operations (GAMA et al., 2014); and (c) the characteristics of language models of learning vector representations in an unsupervised way bound to random processes, this thesis evaluates the distributional hypothesis (HARRIS, 1954) as an approach to help update the language model and mitigate the effects of concept drift.

According to Harris (1954), the distributional hypothesis refers to "words that occur together in the same context have similar meaning". It can imply a few observations such as: (i) unseen words could be represented by the combination of the representations of the surrounding words; (ii) a word representation could be updated over time; and (iii) a given word appearing in new contexts over time, i.e., concept drift, could be directly verified, and its representation, updated.

Precisely, this thesis argues that leveraging pre-trained representations and the distributional hypothesis allows the generation of good quality representations for previously unseen words over time. This argument would result in concept drift adaptation by maintaining new word representations updated and domain adaptation of the pre-trained language model's vocabulary.

Therefore, the hypotheses for this thesis are:

Hypothesis #1: A distributional-hypothesis-based method overcomes the effects of concept drift by allowing higher quality representations;

Hypothesis #2: Combining a pool of two language models to generate vector representations mitigates the effects of concept drift by preventing abrupt model updates (using fine-tuning).

1.4 Contributions

The main contribution of this thesis regards the introduction of methods for concept drift adaptation in text streams, presented in Section 4. More specifically, this thesis provides methods for language model adaptation. These methods are suitable for text streaming scenarios. In addition, the development of this thesis brought the following contributions:

- Garcia et al. (2025): A systematic review of the literature (**Publication #2** in Section 1.5, and Section 3.1), that provides the understanding on the state-of-the-art regarding concept drift adaptation in text streams subject to concept drift;
- Garcia, Britto Jr & Barddal (2023): *Semantic shift* is frequently mentioned as a phenomenon regarding winning or losing of meaning over time, e.g., decades or centuries. **Publication #3** provides the sense that a word can gain a positive or negative connotation in a very short time, i.e., few hours, during a soccer match;
- Garcia et al. (2024): The difficulty of obtaining textual datasets with labeled drifts is frequently mentioned in the literature and justified in publication #2, where only two out of 48 papers had labeled drifts. Therefore, one contribution is the development of methods for generating concept drift in text streams (**Publication #6**);
- Thuma et al. (2023): Extrinsic analysis of embeddings' quality generated by language models (**Publication #1**), which allowed to conclude that leveraging SentenceBERT models as feature extractors are more useful to machine learning models in text streams than other methods, such as Word2vec (MIKOLOV et al., 2013a), and Incremental Word-Vectors (BRAVO-MARQUEZ; KHANCHANDANI; PFAHRINGER, 2022a);
- Garcia et al. (2025): Comparative analysis of loss functions for fine-tuning SentenceBERT models for text streams classification, combined with different sampling methods (**Publication #5**). This paper shows that our proposed sampling method, i.e., WordPieceToken ratio, led to the best Macro F1-scores in 7 out of 10 scenarios;
- Imai et al. (2024): Analysis of recurrent fine-tuning language model in a Brazilian Portuguese news posts stream (**Publication #7**). This paper shows that recurrent fine-tuning, e.g., yearly, is more beneficial for classification than using a general pre-trained model.

This thesis also provides byproduct contributions. In particular, two datasets were created. The first, called `hashtags.csv`[1], contains hashtags co-occurring

[1]Available at <https://github.com/cristianomg10/temporal-analysis-of-drifting-hashtags-in-textual-data-streams-a-graph-based-application>

with #mybodymychoice from tweets between 2018 and 2023, temporally ordered, and `sci-vs-colocolo.csv`[2], a dataset with tweets regarding a soccer match between Sport Club Internacional and Colo Colo in the Sudamericana Cup in 2022. This dataset contains labeled drifts, because the drifts are triggered by events in the match. More details on the papers, the datasets, and their contexts of use, please see Appendix A.

Finally, the software developed following the methods proposed in this thesis was formally registered in Brazil with the *Instituto Nacional de Propriedade Intelectual* (INPI), under registration number BR512026000324-5.

1.5 Publications

Formalizations, categorizations, and experiments presented in this thesis or related to its theme are published or submitted as follows:

1. Bruno S. Thuma, Pedro S. Vargas, Cristiano M. Garcia, Alceu S. de Britto Jr, Jean P. Barddal. **Benchmarking Feature Extraction Techniques for Textual Data Stream Classification**. *International Joint Conference on Neural Networks*. 2023. **(Qualis A1)**;

2. Cristiano M. Garcia, Ramon Abilio, Alessandro L. Koerich, Alceu S. de Britto Jr, Jean P. Barddal. **Concept Drift Adaptation in Text Stream Mining Settings: A Systematic Review.** *ACM Transactions on Intelligent Systems and Technology*. 2025. **(Qualis A1)**;

3. Cristiano M. Garcia, Alceu S. de Britto Jr, Jean P. Barddal. **Event-driven Sentiment Drift Analysis in Text Streams: An Application in a Soccer Match.** *International Conference on Machine Learning and Applications (ICMLA)*. 2023. **(Qualis A2)**;

4. Cristiano M. Garcia, Alceu S. de Britto Jr, Jean P. Barddal. **Temporal Analysis of Drifting Hashtags in Textual Data Streams: A Graph-Based Application.** *Expert Systems with Applications*. 2024. **(Qualis A1)**;

5. Cristiano M. Garcia, Alessandro L. Koerich, Alceu S. de Britto Jr, Jean P. Barddal. **Improving Sampling Methods for Fine-tuning SentenceBERT in Text Streams.** *International Conference on Pattern Recognition (ICPR)*. 2024. **(Qualis A1)**;

6. Cristiano M. Garcia, Alessandro L. Koerich, Alceu S. de Britto Jr, Jean P. Barddal. **Methods for Generating Concept Drift in the Classification of Text Streams.** *Under development*;

[2]Available at <https://github.com/cristianomg10/sentiment-drift-analysis-text-stream-football>

7. Bruno Y. L. Imai, Cristiano M. Garcia, Marcio V. Rocha, Alessandro L. Koerich, Alceu S. de Britto Jr, Jean P. Barddal. **Is it Fine to Tune? Evaluating SentenceBERT Fine-tuning for Brazilian Portuguese Text Stream Classification.** *IEEE International Conference on Big Data*. 2024. **(Qualis A1)**.

The papers above are listed respecting the order of development/writing rather than the publishing year, when available.

1.6 Financial Support

Pontifícia Universidade Católica do Paraná (PUCPR) and *Instituto Federal de Santa Catarina (IFSC)* supported this thesis by exempting the student's fees (CIP) and providing leave for study without impacting the student's salary, respectively. In addition, CAPES supported the student during the doctoral internship at the École de technologie supérieure (ÉTS), in Montréal, Québec, Canada. The student is deeply grateful for the opportunities provided by these institutions.

1.7 Overview

This thesis is organized as follows. Chapter 2 provides the background of this proposal, including concepts of data and text streams, concept drift in text streams, concept drift detectors, Bidirectional Encoder Representations from Transformers (BERT), SentenceBERT (SBERT), AdaNEN, the main state-of-the-art competitor, and the distributional hypothesis. Chapter 3 provides information about related works. Chapter 4 technically details the proposal. Chapter 5 explains the evaluation framework, including the dataset, concept drift simulation, drift detectors, sampling methods, metrics, statistical evaluation methods, and the experimental protocol. Chapter 6 shows the results obtained in the experiments, including the statistical evaluation and an ablation study. Finally, Chapter 7 draws conclusions, discusses the results, and proposes future works.

2 Background

This chapter presents the definition of concepts important for the understanding of this thesis. It introduces the definitions and characteristics of data and text streams. In addition, concept drift, a frequent phenomenon in data and text streams; and concept drift detectors are presented. Furthermore, information about the Bidirectional Encoder Representations from Transformers (BERT), the SentenceBERT (SBERT), a sentence encoder based on BERT, and the distributional hypothesis, an essential concept for this thesis, is provided since one of the proposed methods in Chapter 4 is based on it.

2.1 Data and Text Streams

According to Bifet et al. (2018), data streams are an abstraction that allows for real-time analytics. Different from batch learning, this setting has characteristics that include: (i) data arrive on an instance basis or in small batches, (ii) data arrive fast, and (iii) data can be potentially infinite. These characteristics hamper traditional machine learning methods since they rely on previously obtained static datasets.

Considering that data streams can be infinite, collecting all data from the stream aiming at learning from them becomes unfeasible regarding storage and time. This leads to some desirable characteristics of ML methods to work in data stream settings, which include (GAMA et al., 2014; BIFET et al., 2018): (i) learning from data as it arrives, (ii) discarding data quickly after learning from it, and (iii) performing single-pass operations. ML methods that address these desirable characteristics are generally referred to as incremental or adaptive methods. Although traditional machine learning methods can also be trained with small batches collected from data streams, it is relevant to highlight that such a model would be outdated earlier than incremental/adaptive machine learning methods. In addition, retraining a traditional ML method from scratch often is more costly compared to incremental/adaptive ML methods. Consequently, incremental/adaptive methods are the most suitable for streaming scenarios.

Text streams are a specialization of data streams. Text streams correspond to a potentially infinite and unbounded sequence $T = \{t_1, t_2, t_3, ...\}$ of texts t that arrive temporally ordered, sequentially, and fast. In addition to the desired characteristics of machine learning methods for streaming scenarios, new challenges emerge. For example, in natural language processing (NLP)-related tasks, the texts are often tokenized, cleaned, and are either lemmatized or stemmed, i.e., reduced to a root. Furthermore, NLP systems maintain a vocabulary with these tokens and, more recently, numeric vector representations of those tokens. These vector representations are known as *embeddings* depending on the language model. These embeddings can be combined and used as input for machine learning methods in several tasks, such as classification or clustering. Therefore, considering the input vectorization, it is assumed $X_i = f(t_i)$, where $f(\cdot)$ could be any text vectorization method. Thus, a text stream can be rearranged as $T = \{X_1, X_2, X_3, ...\}$, where X are numeric vectors. Generating and maintaining those structures of vocabulary and embeddings while respecting the data streams' constraints simultaneously is challenging. Thus, learning from text streams is even more challenging than regular data streams.

2.2 Concept Drift in Text Streams

The world is ever-changing. Collecting real-world data over time often brings pattern changes. Concept drift is a phenomenon that corresponds to changes in data distribution. Changes in data distribution may render machine learning models obsolete to the model's previously learned patterns' outdatedness.

Concept drift in text streams is formally described as follows. Let a text stream $T = \{X_1, X_2, X_3, ...\}$ be a potentially infinite sequence of vectorized input

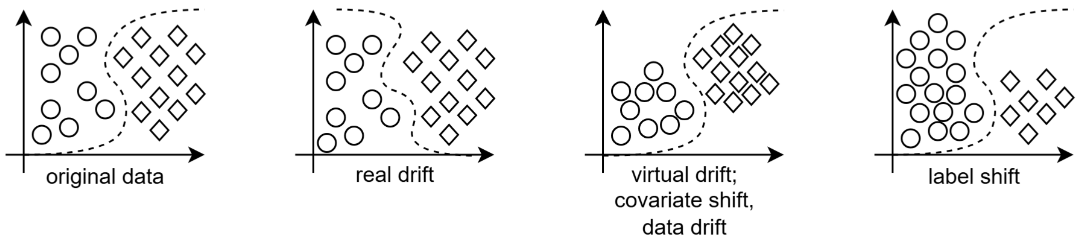

Figure 2.1: Types of concept drift (GAMA et al., 2014; GARCIA et al., 2025). The dashed lines represent the boundaries between classes, while circles and diamonds represent items of different classes.

texts X_i, where i is the vectorized text index, i.e., a timestamp. Assuming a textual data stream in a classification task, each text may be accompanied by its label y, thus becoming a sequence of pairs (X, y). Therefore, according to Gama et al. (2014), a concept drift is deemed to have occurred if

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y), \tag{2.1}$$

in which $p_{t_0}(X, y)$ is the joint distribution between X and the label y in a time t_0.

Regarding the concept drift types, a concept drift can be real or virtual. According to Gama et al. (2014), a *real concept drift* corresponds to the changes in $p(y|X)$, meaning that the classes' boundaries change. Conversely, *virtual concept drift* regards changes in $p(X)$ without necessarily affecting the classes' boundaries. Across scientific and industry communities, virtual drifts may also be referred to as *covariate shift* (FENG et al., 2024), or *data drift* (RABINOVICH et al., 2023). Another type of drift is the *label shift*, which corresponds to changes in label distribution, compared to reference data (ZHAO et al., 2021). Fig. 2.1 visually displays the types mentioned above of concept drift. The items, i.e., circles and diamonds, correspond to instances, whose shapes correspond to their respective classes, and the dashed lines correspond to the boundaries between classes.

Furthermore, concept drifts can be analyzed considering their dynamics. Gama et al. (2014) mentioned four types of concept drift dynamics over time: (a) *abrupt/sudden*, (b) *incremental*, (c) *gradual*, and (d) *reoccurring*. Fig. 2.2 depicts these dynamics. In *abrupt/sudden* changes, the data distribution changes from t_i to t_{i+1}. In *incremental* changes, the data distribution changes from t_i to $t_{i+\Delta}$, where $\Delta > 1$. In *gradual* changes, the data distribution switches between different means until remaining in the last distribution. At last, in *reoccurring* changes, the data distribution changes and later switches back to the first observed data distribution.

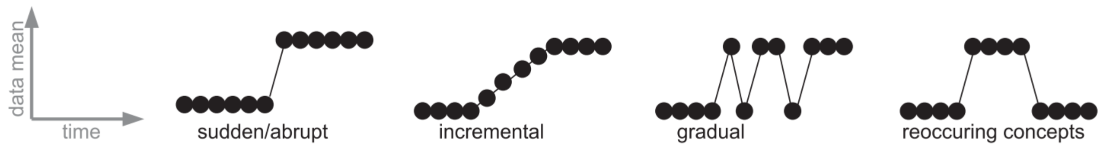

Figure 2.2: Concept drift dynamics over time (GAMA et al., 2014).

Different types of drift may emerge in text stream scenarios. For instance, words aggregating or losing meanings over time is the object of study of *semantic shift*, sometimes referred to as *semantic change* (SÁ; SILVEIRA; PRUSKI, 2024). Semantic shift regards the "evolution of word meaning over time" (KUTUZOV et al., 2018). Several studies on this topic use *diachronic* datasets, which contain texts from different time slices. Resorting to Word2Vec representations and t-SNE for dimensionality reduction, Hamilton, Leskovec & Jurafsky (2016b) plotted Fig. 2.3. Considering Fig. 2.3(c), in the 1850s, *awful* had a positive connotation. Its surrounding words, e.g., *majestic* and *solemn*, confirm the previous statement. Later, in the 1900s, the word *awful* shifted to a negative connotation due to its proximity to the words *terrible* and *horrible*.

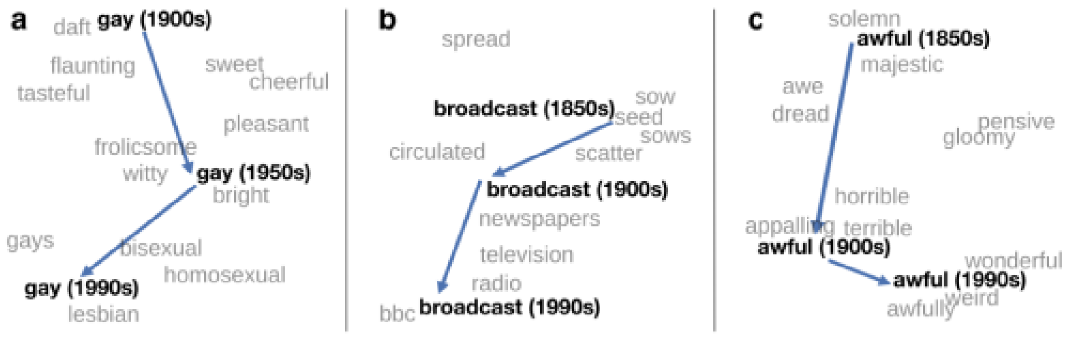

Figure 2.3: Semantic shift over decades, regarding the words *gay*, *broadcast*, and *awful*, respectively (KUTUZOV et al., 2018).

Sá, Silveira & Pruski (2024) overviewed the subject and characterized semantic changes considering the aspects of *dimension*, *relation*, and *orientation*. In the case of dimension, Sá, Silveira & Pruski (2024) considered broadening, i.e., gaining new meanings, and narrowing, i.e., becoming more specific or losing previous meanings. Considering the relation, Sá, Silveira & Pruski (2024) mentioned metaphorization and metonymization, in which the latter occurs, according to the authors, "when a word takes on a new meaning that, to some extent, inherits qualities from its original meaning through a figurative relationship the speaker aims to convey". Finally, changes in orientation regard the connotation of a new meaning, i.e., towards positive (amelioration) or negative (pejoration).

Several works describe methods to measure a word's meaning evolution over time (BELOTTI; BIANCHI; PALMONARI, 2020; RYZHOVA; RYZHOVA; SOCHENKOV, 2021). Some approaches measure the cosine distance between word embeddings in a period and the word embeddings of the same words in a previous period (HOMBAIAH et al., 2021). If the distance exceeds a certain threshold, it is deemed a semantic shift to have occurred. Other approaches may use embedding alignment across time slices, such as orthogonal Procrustes (HAMILTON; LESKOVEC; JURAFSKY, 2016a) and compass alignment (BELOTTI; BIANCHI; PALMONARI, 2020). However, semantic shift detection is mostly addressed by traditional machine learning methods, thus disregarding any constraints on processing and storage.

Handling text streaming becomes a crucial feature in a world with huge amounts of data produced every second. In addition, despite works that depict semantic shifts over long periods, works such as Stewart et al. (2017) demonstrate that semantic shifts may occur in a shorter period, e.g., weeks. Therefore, this thesis proposal focuses exclusively on text stream scenarios.

To conclude, considering streaming scenarios, maintaining the model updated is essential. In addition, in order to prevent outdated text representations and the performance degradation of the stream mining task, it is crucial for the language model to be updated. A manner of performing so is to use concept drift detectors to monitor a performance metric over time, and trigger a model update mechanism to recover from the concept drift when necessary.

2.3 Concept Drift Detectors

Concept drift detectors are methods tailored for detecting changes in data distribution, and they can be beneficial in performing both concept drift and semantic shift detection. Concept drift detectors were initially developed in statistics. However, there is no guarantee that such methods would work specifically in streaming scenarios because some may not work in a one-pass fashion

(BIFET et al., 2018).

Gama et al. (2014) categorized concept drift detection methods into four classes: (i) *sequential analysis*; (ii) *control charts*; (iii) *monitoring two distributions*; and (iv) *context-based methods*, which are also called *heuristic methods*. *Sequential analysis* corresponds to a scenario in which two subsets of data are generated sequentially by processes bound to different unknown distributions, e.g., P_0 and P_1. According to Gama et al. (2014), "when the underlying distribution changes from P_0 to P_1 at point w, the probability of observing certain subsequences under P_1 is expected to be *significantly* higher than that under P_0". It means that a statistical test, for example, may detect this change above. Two representatives of this category are the cumulative sum (CUSUM) test (PAGE, 1954) and the Page-Hinkley test (PAGE, 1954), which is a variant of the CUSUM test (GAMA et al., 2014; BIFET et al., 2018).

The second category proposed by Gama et al. (2014) is *control charts*, also known as *statistical process control* (SPC). Control charts correspond to "standard statistical techniques to monitor and control the quality of a product during continuous manufacturing" (GAMA et al., 2014). In this case, the model receives the data over time, and the concept drift detector uses the model's error as a proxy to determine the system states. The system states are as follows: (i) *in-control*, which indicates that the system is stable; (ii) *drift detection*, which signifies the error increased significantly, compared to the historical error; and (iii) *warning*, which indicates the error increased but insufficiently to raise a detection. Generally, *drift* and *warning* are associated with a statistical confidence of 99% and 95%, respectively. An example of this category is the exponentially weighted moving average (EWMA) (ROSS et al., 2012).

The third category regards *monitoring two distributions*. Methods in this category, according to Gama et al. (2014), "typically use a fixed reference window that summarizes the past information and a sliding detection window over the most recent examples". In this scenario, it is considered a drift to have occurred if the distributions of the windows are statistically different. An example of a method that embeds a concept drift detector from this category is the Very Fast Decision Tree (VFDT) (GAMA; FERNANDES; ROCHA, 2006). An actual concept drift detector that fits this category is the Adaptive Windowing (ADWIN) (BIFET; GAVALDA, 2007). ADWIN is a distribution-free concept drift detector that detects drifts in real-valued or bits streams (BIFET et al., 2018). It maintains a window with the most recent items, from which subwindows are compared. If these subwindows exhibit different means above a threshold based on Hoeffding's bounds, a drift is flagged (GAMA et al., 2014). ADWIN is computationally more expensive in time and memory than *sequential analysis* detectors; however, it is simpler to use because the user does not need to specify a cutoff parameter (GAMA et al., 2014; BIFET et al., 2018). In addition, ADWIN provides more precise change points (GAMA et al., 2014). Nonetheless, Barros & Santos (2018) suggested that ADWIN triggers a high false positive rate. On the other hand, Hoeffding's bounds Drift Detection Method (HDDM) (FRIAS-BLANCO et al., 2014), in its averaged version, reached the best false positive rate in a number of scenarios. In addition, compared to ADWIN, the Reactive Drift Detection Method (RDDM) (BARROS et al., 2017) triggered considerably lower false positive rates.

The last category, i.e., *context-based*, regards specific approaches that use characteristics intrinsic to machine learning methods to perform drift detection or adaptation. For example, in Leite et al. (2012) and Garcia, Leite & Škrjanc (2019), the authors proposed a method that balances incremental learning and forgetting using fuzzy granular computation. Whenever a new instance arrives, the existing granules, i.e., groups that share similar properties, have their (either complete or partial, whenever there are missing attributes in the new instance) similarity with the newly seen instance calculated. The new instance is assigned to the chosen granule if the similarity exceeds a certain threshold. However, if no granule matches the newly seen instance, i.e., a drift occurs, a new granule is created to accommodate the new instance. In addition, a pre-defined parameter controls the periods of verifying stale granules, which can be deleted to maintain the model's conciseness.

The traditional metrics used to evaluate and compare concept drift detection methods, according to Bifet et al. (2018), are as follows: (i) mean time between false alarms (*MTFA*), which assesses the frequency with which a method raises false alarms; (ii) false alarms rate, calculated as $FAR = \frac{1}{MTFA}$; (iii) mean time to detection (*MTD*), which assesses how quickly the method detects and responds to drift once it occurs; (iv) missing detection rate (*MDR*), which determines how frequently the method fails to warn when drift occurs; and (v) average run length (*ARL*), which is the time it takes to raise the alarm once a drift occurs (BIFET et al., 2018). *ARL* integrates *MTD* and *MTFA* (BIFET et al., 2018). Additional metrics, such as Mean Time Rate (*MTR*) (WARES; ISAACS; ELYAN, 2019; BIFET, 2017), may emerge in the literature; however, their primary focus is on missing drifts, hits, time/iterations until detecting an actual drift, or a combination of such factors. *MTR*, for instance, is analogous to *ARL* (WARES; ISAACS; ELYAN, 2019).

Typically, concept drift detectors are coupled to either traditional or online machine learning systems by receiving the hits and errors of prediction. These concept drift detectors have two levels of alarms: *warning* and *drift*. The most straightforward strategy of using a concept drift detector coupled to a classifier is: whenever a warning alarm is issued, either the input data are buffered, or a new machine learning model is trained from scratch. Therefore, when the drift alert occurs, the new model (trained using data from the buffer) replaces the outdated one. This learning strategy is called *background learning* (GOMES et al., 2017). Thus, the goal is to maintain an updated model based on the most recent/frequent data.

2.4 Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) (DEVLIN et al., 2018) is a pre-trained deep-learning-based language model that learns contextual vector representations. Contextual vector representations are extracted using context-sensitive features learned from left-to-right and right-to-left language models (PETERS et al., 2017; DEVLIN et al., 2018). It has a unified architecture, suitable for several downstream tasks. According to Devlin et al. (2018), "BERT's model is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al. (2017)".

Transformer (VASWANI et al., 2017) is an architecture for language modeling that uses the attention mechanism. The attention mechanism allows for extrapolating the surrounding words, finding intrinsic relations, and deciding which are the important parts of the integral text in an unsupervised way. Fig. 2.4 displays a visualization of attention weights considering the sentence "Attention is a mechanism present in Transformers". Precisely, in this example, the weights were collected from layer number 0 and attention head number 6. The library BERTViz[1] was used to generate the visualization. In Fig. 2.4, the darker

[1]https://github.com/jessevig/bertviz

the line, the bigger the weight. Please notice that in the text, according to the attention weights, the terminology *attention* has bigger weights to *mechanism*, *present*, and *transformers*. Interestingly, these words are related to *attention*, and these relations are learned unsupervised.

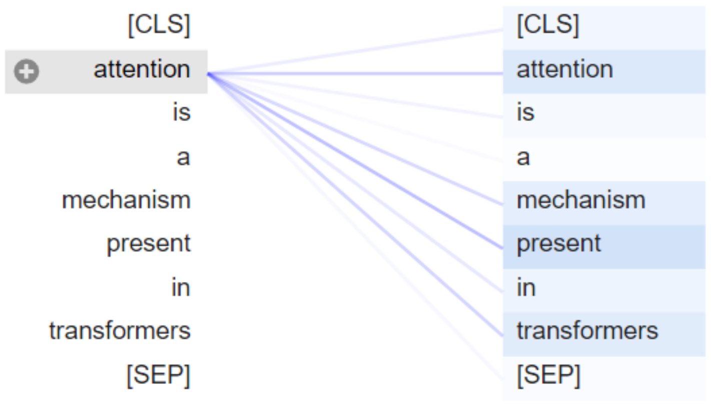

Figure 2.4: Example of the attention mechanism working in a simple sentence. [CLS] and [SEP] are specific tokens in Transformers that denote a vectorial summary of the text sequence and a sentence separator, respectively.

The Transformer architecture has two big parts: the encoder and the decoder. Fig. 2.5 displays the visual representation of Transformer architecture, provided by Vaswani et al. (2017).

Fig. 2.5 shows a two-part structure, in which on the left, there is the encoder, while on the right, the decoder. According to Vaswani et al. (2017), the encoder can have several stacked layers, i.e., N in Fig. 2.5. In addition, each layer has a multi-head attention mechanism and a fully connected feed-forward network, around which a residual connection is utilized, summed, and normalized. Since BERT leverages only the encoder, this thesis proposal does not discuss the decoder.

Devlin et al. (2018) argued that most techniques for language modeling are unidirectional, which limits the power of pre-trained representations. Therefore, BERT leverages the bidirectionality to improve the capacity of encoding representations. BERT pre-training uses two unsupervised tasks simultaneously: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM is a task in which an arbitrary number of tokens are masked, e.g., 15% according to Devlin et al. (2018), so BERT predicts the masked tokens based on the surrounding words. NSP is a task that receives a pair of sentences, and BERT outputs a label. This label expresses whether the pair of sentences are

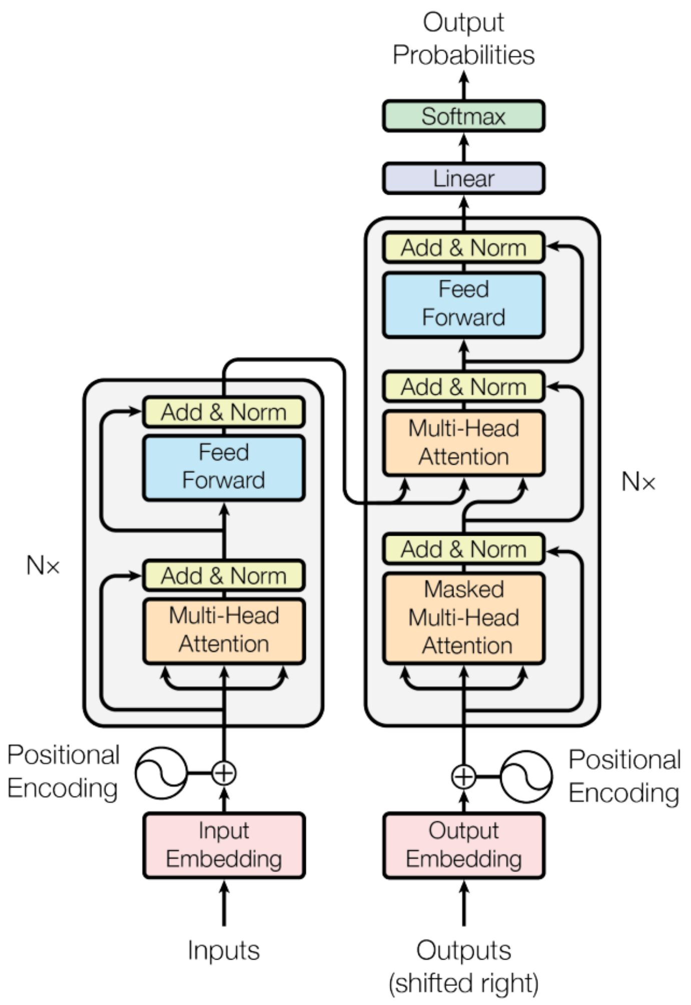

Figure 2.5: Transformer architecture proposed by Vaswani et al. (2017). In the original paper, $N = 6$.

related and whether one is followed by the other. BERT can be used in several downstream tasks, such as classification (GAO et al., 2019; THUMA et al., 2023), sentiment analysis (HOANG; BIHORAC; ROUCES, 2019), and clustering (SUBAKTI; MURFI; HARIADI, 2022). For example, the downstream task can be managed by dense neural network layers coupled at the end of BERT or by another method in a transfer learning fashion.

Devlin et al. (2018) presented two model sizes: (a) *base*, with 12 layers, hidden size of 768, and 12 attention heads; and (b) *large*, with 24 layers, hidden size of 1024, and 16 attention heads. These values directly impact the number of parameters: roughly 110 million for base, and 340 million for the large model. According to Reimers & Gurevych (2019), BERT has an important drawback: the sentence embeddings are not calculated independently due to its structure. In addition, the authors pointed out that two ways of overcoming the aforementioned problem are: (a) averaging the outputs, or (b) using the special token `[CLS]`, which denotes the beginning of a sentence. To generate more representative embeddings for sentences, Reimers & Gurevych (2019) later proposed an alternative architecture called SentenceBERT.

2.5 SentenceBERT

SentenceBERT (SBERT) is an architecture that leverages pre-trained BERT models (REIMERS; GUREVYCH, 2019). Particularly, the authors took advantage of pre-trained BERT (DEVLIN et al., 2018) and RoBERTa (LIU et al., 2019) models. The authors created siamese and triplet networks to fine-tune the model and generate semantically meaningful representations that could be compared using cosine similarity.

Original BERT uses Cross-Encoders to determine the similarity between sentences (DEVLIN et al., 2018). However, it creates a computation overhead, which leads to 65 hours to find the most similar sentence pair among 10,000 sentences (DEVLIN et al., 2018). On the other hand, SBERT uses siamese architecture with Bi-Encoder for tasks such as semantic textual similarity, reducing the aforementioned computation to 5 seconds. While a Cross-Encoder produces an output directly with the similarity between 0 and 1, the Bi-Encoder produces a sentence embedding. Therefore, the sentence embedding can have their similarity computed. Fig. 2.6 shows the difference between Bi-Encoder and Cross-Encoder in SBERT and BERT, respectively.

As seen in Fig. 2.6, on the Bi-Encoder side, two distinct sentences, i.e., Sentences A and B, are inputted to each BERT of the siamese architecture. After encoding, a pooling layer is used to average each contextualized representation produced by BERT, resulting in two sentence embeddings, i.e., u and v. From these embeddings, the cosine similarity can be calculated. On the Cross-Encoder side, Sentences A and B are inputted simultaneously to BERT, which produces an output that indicates the similarity between the two sentences. Reimers & Gurevych (2019) point out that Cross-Encoders generally produce better results but are impractical, since no embedding is generated.

The authors tested architectures with a siamese network structure for clas-

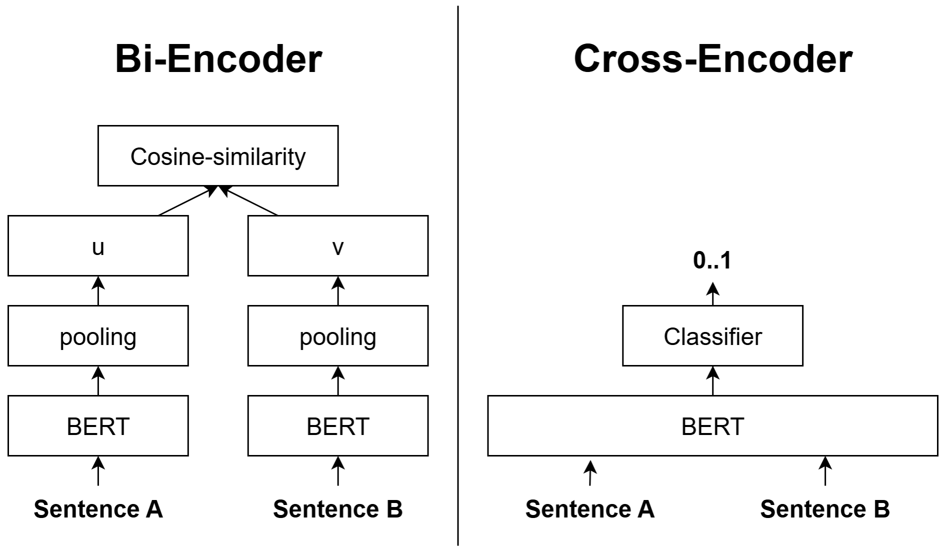

Figure 2.6: Bi-Encoder and Cross-Encoder (SBERT.NET, 2022).

sification and regression objective function, respectively depicted in Figs. 2.7 (a) and 2.7 (b). The classification objective function considers the two input sentences, and their element-wise difference, multiplied by a trainable parameter to learn relations between the input and labels. The regression function is used when the cosine similarity between the input sentences is calculated. The architecture displayed in Fig. 2.7 (a) was used by Reimers & Gurevych (2019) for fine-tuning using a classification dataset, i.e., SNLI (BOWMAN et al., 2015). This dataset contains 570,000 annotated sentence pairs distributed across three labels: contradiction, entailment, and neutral. Another important piece of information is that, originally, the SBERT fine-tuning is applied to all layers, following the directions in Devlin et al. (2018) (REIMERS; GUREVYCH, 2019). Fine-tuning all layers generally reaches good levels of metrics. Lee, Tang & Lin (2019) reached the best levels for classification metrics by fine-tuning all layers while fine-tuning partially, i.e., 25% of the layers, could reach 90% of accuracy.

The authors considered a reasonable amount of data to fine-tune SBERT: (a) SNLI (BOWMAN et al., 2015), with 570,000 annotated sentence pairs, and (b) MultiNLI (WILLIAMS; NANGIA; BOWMAN, 2017), with 430,000 sentence pairs.

The authors evaluated SBERT primarily in semantic textual similarity. Interestingly, Reimers & Gurevych (2019) highlighted that "the purpose of SBERT sentence embeddings are not to be used for transfer learning for other tasks". However, they also pointed out that one experiment has shown the embed-

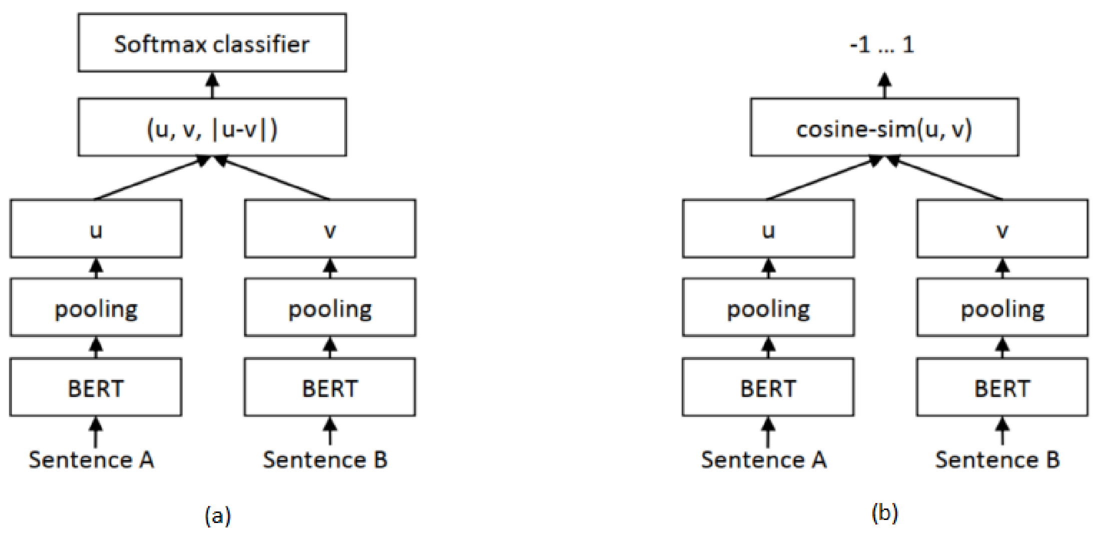

Figure 2.7: SBERT siamese architectures (REIMERS; GUREVYCH, 2019). Architecture (a) is used for classification, while Architecture (b) is leveraged for cosine similarity between sentences.

ding quality for other tasks, such as classification. In addition, the results in Thuma et al. (2023) enforce the quality of the generated sentence embeddings for classification tasks.

An important point is that BERT and SBERT rely on fine-tuning for updates, which generally demands a considerable amount of data. Therefore, using BERT and SBERT in text streaming scenarios can lead to problems such as an increase in the impact of concept drift on the performance of the dependent stream mining task due to the outdatedness of the language model.

2.6 Distributional Hypothesis

Another relevant definition behind one of the methods proposed in this thesis is the distributional hypothesis. According to Harris (1954), the distributional hypothesis states that "words that occur in the same contexts tend to have similar meanings". Therefore, this hypothesis considers that words that commonly appear together in the same contexts are somehow similar.

Consider the sentence "A sweet friendship refreshes the soul", displayed in Fig. 2.8. In addition, the legend suggests that *friendship* is the only unknown word. According to the distributional hypothesis, the surrounding words could be analyzed in order to understand or represent the unknown word. In the figure, a window of size $w = 1$ considers one word before and after the unknown word. Thus, for the given example, the representation of *friendship* could be obtained by regarding the representations from *sweet* and *refreshes*.

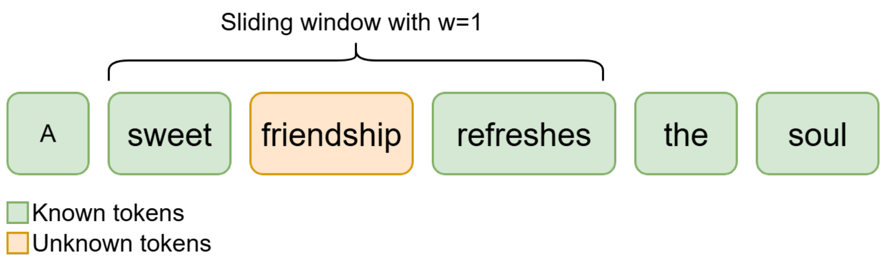

Figure 2.8: Example of use of the distributional hypothesis

Distributional Hypothesis is considered in this work due to its capability of providing information about co-occurring words/tokens. It means that, in case of usage of a given word in a distinct context than previously known, following the Distributional Hypothesis, the surrounding words would be different from the previously observed surrounding words.

The concept of the distributional hypothesis is generally applied to batch methods. Although the distributional hypothesis is not a recent concept, it is leveraged in a number of recent methods. For instance, the training step of Word2Vec (MIKOLOV et al., 2013b) either attempts to predict the surrounding words from a given word, i.e., skip-gram, or attempts to predict a given word from the surrounding words, i.e., continuous bag-of-words (CBOW). Fig. 2.9 graphically represents CBOW and Skip-gram.

Recently, Bravo-Marquez, Khanchandani & Pfahringer (2022a) presented a method that works in streaming settings, generating fixed-length vector representations on the fly: Incremental Word-Vectors. The authors generated vector representations by producing a bi-dimensional word-context matrix, in which context (column) is the surrounding words of a particular word (row). The cell corresponding to a word and a specific context contains the result of an incremental version of the positive pointwise mutual information (PPMI). Eq. 2.2 presents the calculation of PPMI:

$$\mathrm{PPMI}(w, c) = \max\left(0, \log_2\left(\frac{\mathrm{count}(w, c) \times D}{\mathrm{count}(w) \times \mathrm{count}(c)}\right)\right), \tag{2.2}$$

in which w corresponds to a particular word, c to a particular context, and D

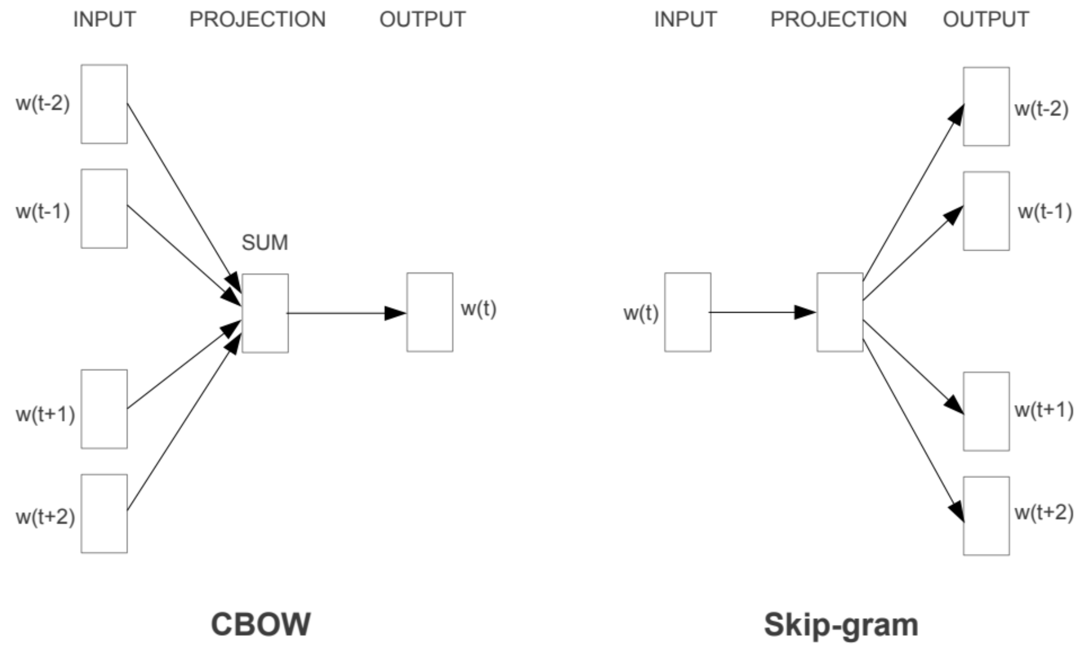

Figure 2.9: Word2Vec architectures (MIKOLOV et al., 2013a).

is the total number of tokens presented in the corpus. A negative PMI value suggests w and c co-occur less than by chance. The authors also mention that using PMI is not reliable unless calculated using a very large corpora. Therefore, keeping PMI positive (using PPMI) corrects the estimates provided by PMI. Furthermore, in their paper, the authors proposed a method for generating word-level sentiment drift, and therefore, they evaluated the representation power of the Incremental Word-Vectors.

Therefore, depending on the strategy and implementation, resorting to the Distributional Hypothesis enables the detection of changes in surrounding words over time and, in case of the appearance of new words, the representation of these words could be modeled from the surrounding words.

2.7 Concluding Remarks

This chapter covered important concepts and fundamentals, i.e., data and text streams, concept drift in text streams, concept drift detectors, bidirectional encoder representation from transformers (BERT), SentenceBERT, and distributional hypothesis.

As mentioned, if not updated, BERT and SBERT may suffer from concept drift/semantic shift over time. Generally, the updates of these language models happen through fine-tuning. Considering the scenario of textual data stream, which is fast and potentially infinite, stopping the predictions to fine-tune the language model can be disadvantageous, creating a bottleneck. Furthermore, the fine-tuning process generally takes long and demands a reasonable amount of data. Next chapter covers how related works handled concept drift in textual stream scenarios between 2018 and August 2024.

3 Related Works

A systematic literature review on concept drift adaptation in text stream scenarios between 2018 and August 2024 was conducted during this thesis. In this chapter, information such as the text drift categories, text drift detection methods, model update schemes, stream mining task addressed, and text representation methods were obtained from the aforementioned review, which is the second bullet in Section 1.5. This chapter includes the systematic review protocol, results, related works that leverage BERT in textual stream settings, and AdaNEN (GHAHRAMANIAN et al., 2024), a state-of-the-art method and competitor in this thesis.

3.1 A Review on Concept Drift Adaptation in Textual Stream Scenarios

The aforementioned review utilized the guideline proposed by Kitchenham & Charters (2007), which contains three steps: (a) planning the review, (b) conducting the review, and (c) reporting the review. The review collected papers from five sources of studies: IEEEXplore[1], Science Direct[2], ACM Digital Library[3],

[1]https://ieeexplore.ieee.org/
[2]https://www.sciencedirect.com/
[3]https://dl.acm.org/

Springer Link[4], and Scopus[5]. Four research questions guided this process and these are enlisted in Table 3.1. The primary question, $RQ1$, takes precedence, and the remaining questions are derived from $RQ1$.

Table 3.1: Research questions used in the systematic literature review.

ID	Research Questions
RQ1	"How to handle concept drift using ML approaches having as source text streams?"
RQ2	"Which type of application is addressed?"
RQ3	"Which type of token/word/sentence representation is used in the study?"
RQ4	"Which datasets were used to evaluate the proposed approach(es)?"

Since $RQ1$ was the main research question, it provided the terms for the research query. In addition, synonyms for the terms were used to broaden the query. Table 3.2 depicts keywords and synonyms used in the review. $RQ2$ focused on the applications the papers addressed when handling concept drift in textual streams. This question is crucial because it may help illustrate scenarios, the potential, and an increased interest in specific problems. Furthermore, $RQ2$ aimed at uncovering which ML methods were employed and how the models were updated, e.g., incrementally or regularly retrained. $RQ3$ intended to uncover the most common approaches to representing texts or smaller parts, such as tokens, words, and sentences. Finally, $RQ4$ pursued insights into the existence of consolidated datasets for the field, and their aspects, such as the level of labeling in the dataset, the data mining task employed, whether the dataset contains real-world data or it is synthesized, metrics used in those data mining tasks, and whether drifts are labeled in the dataset.

Table 3.2: Keywords and respective synonyms.

Keyword	Synonyms
concept drift	semantic shift, representation shift, semantic change
machine learning	-
text streams	textual streams, social network streams, Twitter streams, diachronic, text streaming
detection	-

The query presented below was developed using Table 3.2. *Semantic shift* and *representation drift* are closely related to *concept drift*, especially in the textual con-

[4]https://link.springer.com/
[5]https://www.scopus.com/

text. Semantic shift (or semantic change), according to Bloomberg (1933), refers to "innovations which change the lexical meaning rather than the grammatical function of a form". However, according to Fu et al. (2022), the representation shift in NLP relates to changes in the vector representation. The terminology *Social network streams* was included because they are a notable source of text streams produced directly by humans nowadays. The terminology *Twitter streams* was also used because Twitter, currently X[6], is a microblog (one of the most popular) and generated around 500 million tweets (posts) per day in 2022[7]. Furthermore, the term *diachronic* was included. When serving as an adjective for a dataset, *diachronic* refers to a dataset that contains data produced over time. The term *machine learning* was withdrawn because concept drift is mostly addressed by or in processes that use ML techniques, and preliminary tests showed that several unrelated works were retrieved. Therefore, the query used in the search was: `("concept drift" OR "semantic shift" OR "representation shift" OR "semantic change") AND ("text streams" OR "textual streams" OR "textual streaming" OR "social network streams" OR "twitter streams" OR "diachronic") AND ("detection")`. Each source has its parameters, but the full-text search was prioritized in all of them.

The inclusion and exclusion criteria used in the review are described below. The search was limited to papers published after 2018 because other previous secondary studies tackled similar problems (KUTUZOV et al., 2018; TAHMASEBIA; BORINA; JATOWT, 2021). However, a substantial difference between the review to the aforementioned works is that this review focused on papers that approach the problem of concept drift using text streams as a data source. Therefore, the inclusion and exclusion criteria considered are displayed in Table 3.3. It is also essential to note that this review protocol was last executed on November 3^{rd}, 2022.

Fig. 3.1 overviews the paper selection process. 870 papers were collected, considering the research query. The final calculated Cohen's Kappa coefficient reached 84.61%, which indicates a high agreement level between the researchers. In addition, the divergences were discussed after a thorough reading of the divergent papers, and a decision was reached on their inclusion or exclusion. After removing duplicates (n=178), non-article studies (n=5), non-primary studies

[6]<http://x.com>
[7]<https://www.dsayce.com/social-media/tweets-day/>

Table 3.3: Inclusion and exclusion criteria used in the review.

Ref	Inclusion criteria
IC1	The study is published in journals or conference proceedings
IC2	The study is published from 2018 (inclusive)
IC3	The study presents a method for handling concept drift
IC4	The study uses text streams as data source

Ref	Exclusion criteria
EC1	The study is not primary
EC2	The study is not written in English
EC3	The study is incomplete
EC4	The study is not an article
EC5	The study is duplicated
EC6	The study does not meet the inclusion criteria

(n=46), and unrelated studies (n=562+31=593), 48 articles were retained for a full reading and analysis.

3.1.1 Results and Insights

Considering the process depicted in Fig. 3.1, the reader's attention may be drawn by the high number of unrelated studies after screening the abstract. It occurred due to the keyword *diachronic*, which relates to something that evolves, especially concerning language. Most approaches that handle language evolution cannot work in streaming environments (about 60% of the initially selected papers). Therefore, those studies were removed from the paper selection.

Based on the information extracted from the selected papers using the research questions, the approaches for handling text drifts were categorized according to the following characteristics: text drift categories; drift detection types; (MU) model update; (TR) text representation; and (TRUS) text representation update scheme. The proposed taxonomy is depicted in Fig. 3.2.

Regarding **drift categories**, the papers were organized into: (i) *feature drift*, (ii) *real drift*, (iii) *semantic shift*, and (iv) *virtual drift*. *Feature drift* considers the changes in the importance of features, meaning that a subset of features may become necessary over time for an ML model, while other subsets may become obsolete (BARDDAL et al., 2017). The taxonomy subdivides *feature drift* into *changes in important words*, *changes in important dimensions*, and *vocabulary shift*. *Changes in important words* regards changes in the importance of words themselves or in dimensions that can be directly translated to words, such as in Bag-of-Words or TF-IDF representations. Conversely, changes in dimensions that cannot be translated to words, such as Word2Vec representations, are mentioned as *changes in important dimensions*. Finally, *vocabulary shift* (HOMBAIAH et al., 2021) can be considered one type of feature drift, since it ponders the changes

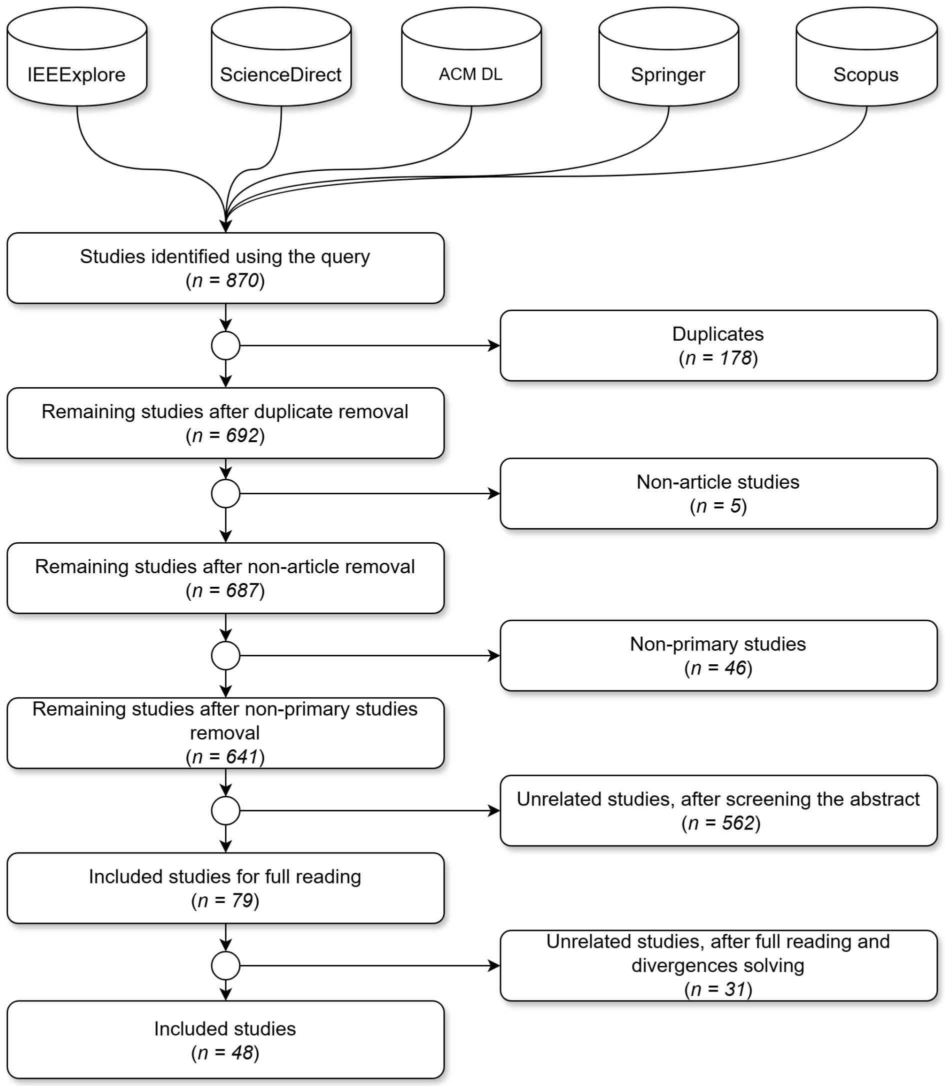
IEEExplore
ScienceDirect
ACM DL
Springer
Scopus
Studies identified using the query (*n = 870*)
Duplicates (*n = 178*)
Remaining studies after duplicate removal (*n = 692*)
Non-article studies (*n = 5*)
Remaining studies after non-article removal (*n = 687*)
Non-primary studies (*n = 46*)
Remaining studies after non-primary studies removal (*n = 641*)
Unrelated studies, after screening the abstract (*n = 562*)
Included studies for full reading (*n = 79*)
Unrelated studies, after full reading and divergences solving (*n = 31*)
Included studies (*n = 48*)

Figure 3.1: Process of papers selection of the systematic literature review. Each rounded-corner rectangle on the right side corresponds to an exclusion criterion. The numbers of remaining studies after each elimination are presented on the left side.

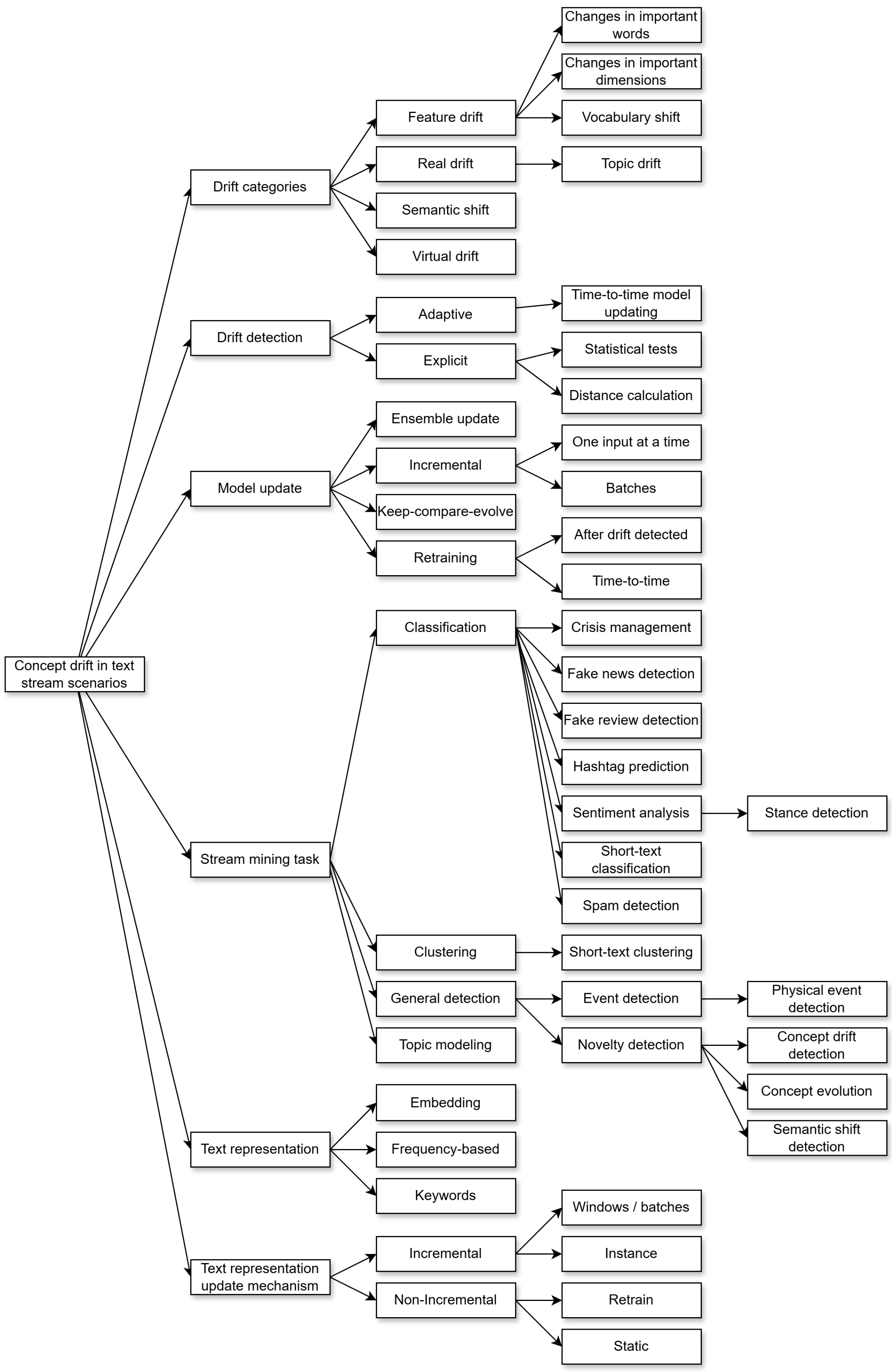

Figure 3.2: A taxonomy regarding concept drift in text stream scenarios.

of words in a vocabulary maintained by an approach as a type of text drift. Unlike the other subtypes of feature drift presented, vocabulary shift considers the changes, i.e., addition or removal of items, in the internal structure that stores the tokens. *Real drift* regards changes in $p(y|X)$ that can occur with or without changes in $p(X)$ (GAMA et al., 2014), where X regards the input features, while y corresponds to the class, and p is the probability. *Real drift* in a classification task refers to the change in the classes' boundaries, which may be accompanied by changes in the data distribution in X. The review considered *topic drifts* as an extension of *real drifts*. *Topic drifts* were encountered in applications regarding topic modeling, topic labeling, and short-text classification. Thus, a topic could drift by the change of either text labeled as a particular topic, i.e., $p(y|X)$, or by the change of a topic distribution in the stream, i.e., $p(X)$, or both simultaneously. *Semantic shift* considers changes in the meaning of tokens over time. Semantic shift is commonly handled in papers that study linguistic changes over years, decades, or even centuries. However, semantic changes can also occur in shorter periods, such as in weeks (STEWART et al., 2017). *Virtual drift* regards changes in data distribution, $p(X)$, without changing the boundaries between classes (GAMA et al., 2014).

Drift detection considers the types of drift detection, which can be *adaptive*, or *explicit*. *Adaptive*, also known as *blind adaptation* (GAMA et al., 2014), considers periodic updates in the machine learning model, while *explicit* regards the drift detection and posterior triggering of update mechanisms. Explicit detection can happen through statistical tests, such as Page-Hinkley (PAGE, 1954; SEBASTIÃO; FERNANDES, 2017) and distance calculation, such as Jensen-Shannon divergence (NIELSEN, 2019).

Regarding **model update**, it considers the manner the ML models are updated to overcome concept drift. *Ensemble update* included papers that propose mechanisms to create, update, and combine models as an ensemble. The update of ensembles generally includes removing outdated models and learning new base learners when necessary. *Incremental* regards models able to learn from new data without being retrained from scratch. The proposed taxonomy considers that incremental updates can happen one-by-one, i.e., *one input at a time*, or in *batches*, i.e., a small data collection. *Keep-compare-evolve* includes approaches that leverage other versions of a particular model to evaluate the occurrence of concept drift. That is, older versions are kept to serve as a reference to new models, and the approach can decide when to generate a new model and replace the older model. Hombaiah et al. (2021) implemented this particular model update scheme. Finally, *retraining* considers the complete retraining of models. The taxonomy includes, for this subdivision, the complete retraining *after a drift detection*, or periodically, i.e., *time-to-time*.

The taxonomy in Fig. 3.2 also considers the **stream mining task** approached in the papers. Each stream mining task, with the exception of topic modeling, is subdivided into applications. The stream mining tasks present in the papers were *classification*, *clustering*, *general detection*, and *topic modeling*. *Classification* is a task in which an algorithm generates a model, which its aim is to predict, with arbitrary accuracy, a label y based on an input X (BIFET et al., 2018). Applications that appeared in the studied papers and corresponded to classification were *crisis management*, *fake review detection*, *hashtag prediction*, *sentiment analysis* and *stance detection*, *short-text classification*, and *spam detection*. *Clustering* regards finding intrinsic patterns in the data, so that a model can infer data clusters in an unsupervised manner (BIFET et al., 2018). *Short-text clustering* was the only application found in the papers that fit in *clustering* task. *General detection* included *event detection* and *novelty detection*. *Event detection* considered papers that aimed to detect events, e.g., physical landslide detection (SUPREM; PU, 2019). Furthermore, some papers had *novelty detection* as the main application, which included *concept drift detection*, *concept evolution*, and *semantic shift detection*. The only work present in *concept drift detection* (MELLO et al., 2018), unlike all other works, had as the only task the concept drift detection without involving classic stream mining tasks. *Concept evolution* regards the appearance of unseen classes in the stream. *Semantic shift detection* considers the detection of change in meaning of words over time. Finally, *topic modeling* includes statistical tools to process textual data and identify the most relevant terms related to each theme (BLEI, 2012).

Another aspect evaluated in the papers was the **text representation** methods employed. The methods were subdivided into *embedding*, *frequency-based*, and *words*. *Embedding* regards representations that are often generated by methods such as BERT and Word2vec. These representations and their dimensions are not directly interpretable. *Frequency-based* representations are generally based on counting. Bag-of-Words and TF-IDF (term frequency - inverse document frequency) are examples of this category. Unlike representations from embeddings, their dimensions can be translated back to words. *Words* considers the words themselves. The most frequent category was frequency-based (7), followed by embeddings (7), and words (1). Please note that some methods repeated across the papers, such as Bag-of-words (a frequency-based method), are present in 10 papers.

At last, the **text representation update mechanism** aspect is also analyzed. The mechanisms were organized in *incremental*, and *non-incremental*. *Incremental* regards the methods in which the representations are updated, and it can happen in *windows* / *batches* or by *instance*. The *non-incremental* methods include representations that are generated from scratch, i.e., *retrain*, or are *static*, being the same during the learning process. Generally, methods fit this category when a particular approach leverages pre-trained language models. Only two methods are truly incremental (BRAVO-MARQUEZ; KHANCHANDANI; PFAHRINGER, 2022a; YANG et al., 2021).

Considering the dynamic nature of text streams and the fact that language is ever-evolving, it is crucial to develop incremental approaches for text stream-related scenarios. Generating text representations from scratch, or even fine-tuning models for this type of scenarios can be computationally intensive since streams are generally fast and potentially infinite (BIFET et al., 2018; GAMA et al., 2014), and those processes can create a bottleneck in the stream learning process.

3.2 On BERT Models in Textual Stream Scenarios

Several works use pre-trained BERT models and their variants as a part of an application. For example, Dusart, Pinel-Sauvagnat & Hubert (2021) used BERT for Twitter stream incremental summarization. The authors leveraged BERT pre-trained representations and word frequency to predict tweets' salience score, a metric to decide whether a tweet should be maintained in summary. Zhou et al. (2022) evaluated several pre-trained language models, such as Global Vectors (PENNINGTON; SOCHER; MANNING, 2014), ELMo (PETERS et al., 2017), BERT (DEVLIN et al., 2018), RoBERTa (LIU et al., 2019), and DistilBERT (SANH et al., 2019). The authors applied the pre-trained models in a classification task to identify the existence of asking for help, location, or information about the victim in tweets related to rescue situations. BERT models, together with different approaches, i.e., Linear classifier, Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN), obtained the best values for F1-Score.

In Thuma et al. (2023), SentenceBERT (SBERT) is used as a sentence representation method for a text stream classification task. The authors compared SBERT to Word2Vec (MIKOLOV et al., 2013b), Hashing Tricks (ATTENBERG et al., 2009), and Incremental Word-Vectors (BRAVO-MARQUEZ; KHANCHANDANI; PFAHRINGER, 2022a). SBERT obtained the best values in terms of accuracy, statistically equivalent to Hashing Tricks regarding running time.

Undoubtedly, leveraging pre-trained models saves time since training these models from scratch generally takes a long time, as well as demands a considerable amount of data. In addition, as a result of the pre-training process, these models can generally provide good-quality representations. However, none of the aforementioned works concern the language model update. Considering works regarding *semantic shift*, it is well-known that language is ever-evolving and, eventually, the language vector representations will be obsolete, leading to concept drift and semantic shift.

Generally, language models are updated through a fine-tuning process. However, BERT models demand a reasonable amount of data to generate stable representations, although other works resort to different strategies to use smaller amounts of data (ZHANG et al., 2020). Zhang et al. (2020) mention that the instability in the BERT's fine-tuning process has been known since BERT's release. In addition, the authors complement mentioning strategies for overcoming this instability, including a new regularization method (LEE; CHO; KANG, 2019) and a new early-stopping method (DODGE et al., 2020). An alternative strategy, with a different aim, includes extending the BERT model for new domains (TAI et al., 2020). The authors added the extension module to each layer in the original BERT, including a new vocabulary learned in the fine-tuning step. In addition, they developed a mechanism for weighting the embedding, considering the existence of words in the original and extended vocabularies. The authors evaluate the approach using named entity recognition and relation extraction. The running times draw attention: the authors use 64 and 128 hours, which is prohibitive for stream scenarios.

Intending to work using streaming scenarios, Hombaiah et al. (2021) evaluated strategies for the language model update based on dynamic content. The authors performed analyses using tweets from 2013 to 2019, and vocabulary shift analysis to assert the BERT models deteriorate over time. In addition, the authors proposed mechanisms for overcoming the model degradation: a method for dynamic model update based on the difference between models, and sampling methods to select more effective data to use in the BERT incremental training. According to Hombaiah et al. (2021), one of the sampling methods can be used to trigger incremental training. Although the authors performed extensive tests, they do not provide the run times, which is essential for streaming scenarios.

Unlike the aforementioned approaches, this thesis proposes methods for BERT language model update over time in textual streaming scenarios. This would allow for the development of ever-learning systems that resort to texts, mitigating the effects of concept drift and semantic shift. In addition, as a side-effect, the vocabulary would eventually be domain-specific. As the primary requirement, it should not create a bottleneck in the learning process.

3.3 AdaNEN

AdaNEN (GHAHRAMANIAN et al., 2024) is presented in this section because it corresponds to the state-of-the-art for this scenario addressed in this thesis. AdaNEN stands for Adaptive Neural Ensemble Network.

AdaNEN is a specific type of neural network that concatenates the different layers of the neural network instead of using only the output of the last layer. Besides, the output layer consists of an ensemble with isolated layers, whose outputs are combined in an actual output. Each isolated layer in the output layer is trained with different learning rates. This network can be trained in an online fashion, allowing it to work properly in text streaming settings. Fig. 3.3 shows the structure of AdaNEN in a setting of text data stream.

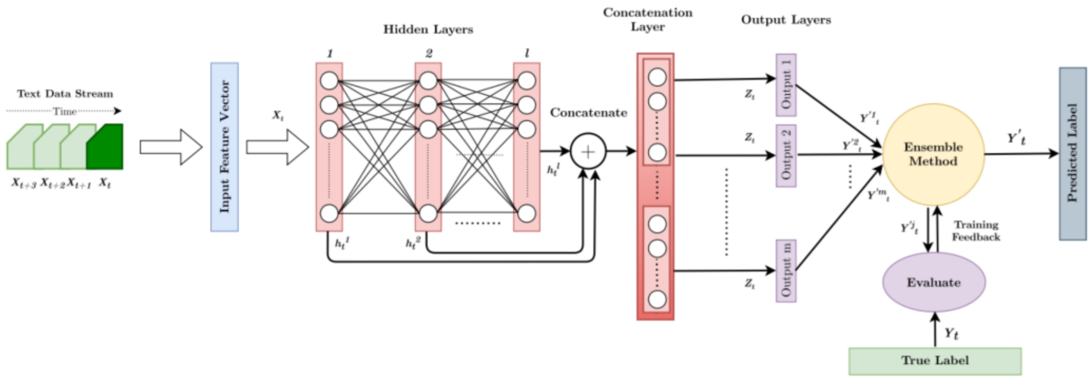

Figure 3.3: Structure of AdaNEN (GHAHRAMANIAN et al., 2024).

In the paper, the authors evaluated their approach using eight real text streams subject to concept drift (including abrupt, recurrent, gradual, and incremental). AdaNEN was compared to twelve other approaches, grouped in ensemble and neural methods. AdaNEN performed the best in all the experiments in terms of prequential accuracy. However, although it obtained the best accuracy results, it ranked 3.5 on average in elapsed times for textual datasets.

In this thesis, AdaNEN is leveraged in a different scenario, i.e., multiclass. Furthermore, AdaNEN is evaluated having as input the a vectorized text provided by SBERT, or SBERT combined with the Distributional-Hypothesis-based method, a method proposed in this thesis (Chapter 4.1).

3.4 Concluding Remarks

This chapter overviews the concept drift adaptation in textual stream scenarios. Several works were thoroughly evaluated regarding aspects such as the drift category, the type of drift detection, the model update, the approached stream mining task, the text representation, and the text representation update method.

These characteristics provide a complete view of the methods. In text stream scenarios, the manner in which the method generates and updates the text representations directly impacts the dependent machine learning model's performance over time. In addition, the update of the machine learning model may depend on the type of drift detection and which category of drift is handled. Of the 48 papers analyzed, only two approached the use of BERT-based models in textual streams.

Furthermore, other related works are discussed. These works include BERT-based models in textual stream scenarios. Some of them may include evolving methods for the machine learning model, and for the language model in different ways. This thesis proposes a method for SBERT models' update over time using the distributional hypothesis for the generation of new words' representation. The next chapter details the proposed methods.

4 Proposed Methods

This thesis addresses a SBERT model update process to be suitable for working in text stream scenarios subject to concept drift. Although SBERT models provide good-quality representations, text streaming environments can be dynamic, and the representations can become outdated over time. Traditional fine-tuning, depending on the amount of data, can be slow and take a long time to update the model, making the process unfeasible in streaming environments. Therefore, learning new words and updating representations is crucial for the never-ending functioning of a language model in text streams.

Based on the hypotheses described in Section 1.3, the Distributional Hypothesis is leveraged aiming at generating new words' representations by combining the representation of the surrounding words over time. The objective of this process is to overcome potential concept drifts in the text stream by having updated words' representations. The second hypothesis seeks to mitigate the effects of concept drift by maintaining a pool of two language models. The update of these models occurs in alternation, providing softer updates in the embeddings, compared to a single model fine-tuning.

Two methods are proposed in this thesis: (a) Distributional-Hypothesis-based representation update, and (b) Bi-SBERT adaptation approach. The former could be used in a fully incremental scenario and in the time-to-time update scenario, while the latter can be used only in the time-to-time scenario. These scenarios are detailed alongside the proposed methods.

4.1 Distributional-Hypothesis-based representation update

First, this section overviews the distributional-hypothesis-based approach, which is later detailed in terms of internal structures, weighting text vectorization method, the generation of new tokens method, and finally, the concluding remarks. This approach works in the setting depicted in Fig. 4.1, which graphically shows the interaction among the components. However, this approach is also suitable for the scenario presented in Section 4.2, which is not fully incremental.

The functioning of the Distributional-hypothesis-based method is detailed in Algorithm 1. Please note that the internal structures and the equations present in the aforementioned algorithm are described below, in Sections 4.1.1 and 4.1.4. For the proposed method, the sampling method utilized was the WordPieceRatio (please see Section 5.5).

Algorithm 1 Distributional-hypothesis-based update method

```
Require: $T \leftarrow$ Textual data stream
Require: $w$                                    ▷ Window size to consider surrounding words
Require: classifier                                               ▷ Incremental classifier
  ext_vocab ← {}                                                   ▷ External vocabulary
  SBERT ← pre-trained('paraphrase-miniLM-v6')
  while $T \neq \emptyset$ do                                 ▷ while the text stream is not empty
    text ← $T_i$                                    ▷ $i$ is the index of the text in the stream
    text ← clean(text)                          ▷ lowercase and remove special characters
    vectorized_text ← final_features(text)                                         ▷ Eq. 4.5
    $\hat{y}$ ← classifier.predict(vectorized_text)
                                                                   ▷ $y$ becomes available
    for each token ∈ text do
      if token ∉ SBERT.vocabulary then
        surr_tokens ← capture $w$ surrounding words of token
        new_repr ← repr(surr_tokens + token)                                      ▷ Eq. 4.6
        ext_vocab[token].update(new_repr)
      end if
    end for
  end while
```

For the sake of calculating the computational time complexity of the above algorithm, three aspects are considered: (a) SBERT, considering $O(L \cdot n^2 \cdot d)$

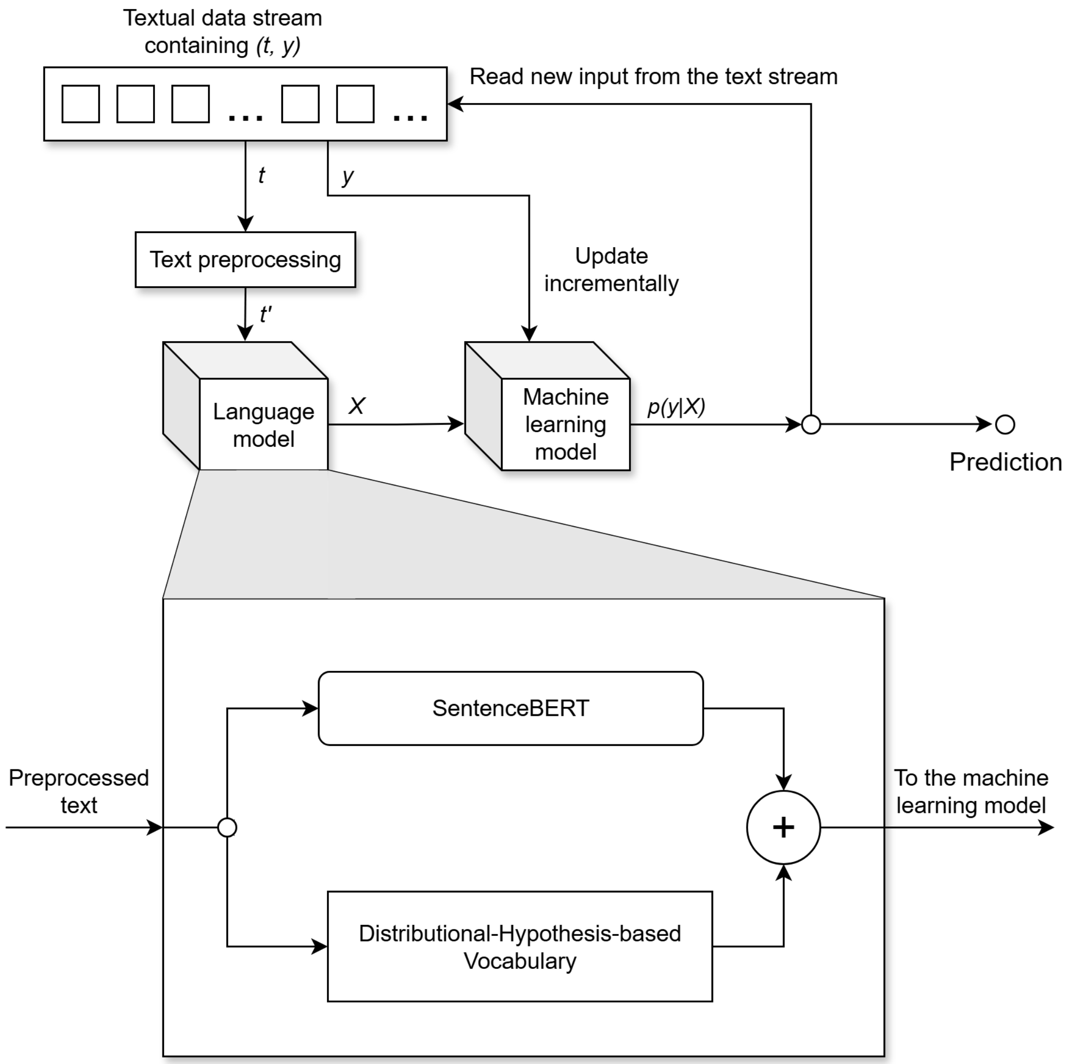
Textual data stream containing (t, y)
Read new input from the text stream
t
y
Text preprocessing
t'
Update incrementally
Language model
X
Machine learning model
p(y|X)
Prediction
SentenceBERT
Preprocessed text
+
To the machine learning model
Distributional-Hypothesis-based Vocabulary

Figure 4.1: Setting for the proposed distributed-hypothesis-based method. In this setting, a distributional-hypothesis-based vocabulary stores and updates the representation of new words over time. The text vectorized by Sentence-BERT is combined with the updated representations from the Distributional-Hypothesis-based vocabulary to provide a vector representation.

as its approximated complexity, where n is the number of tokens in the input sentence, d the dimension of the resulting embedding, and L the number of layers; (b) the iteration over the input text to combine the embeddings as $O(t)$, where t is the number of words, and (c) the stream itself as $O(m)$, where m is the stream length. It is well-known that the computational time complexity of the self-attention mechanism is $O(n^2d)$ (FARINA et al., 2024). Therefore, the computational time complexity of the algorithm is estimated to be $O(L{\cdot}n^2{\cdot}d{\cdot}t{\cdot}m)$, where m is the stream length.

4.1.1 Internal Structures

The proposed method considers two additional structures:

- A `vocabulary`, i.e., external vocabulary, which contains the new `tokens` discovered from the textual data stream;
- Each word in the vocabulary has a `representation storage`, which accumulates a given number of new representations of a particular word over time.

The **distributional-hypothesis-based vocabulary** stores the new tokens discovered during the text stream learning process. At first, it is virtually unlimited. However, it is intended to limit and maintain this structure to meet the requirements of the text stream. More details are given in Section 4.1.2.

Regarding the **representation storage**, for each word, a number of representations calculated over time using Eq. 4.6 can be stored. For example, consider a given word w, inexistent in the SBERT Vocabulary. This word appears over time at arbitrary moments i. Therefore, in the representation storage, for the word w, there will be a set of embeddings e generated at the moments of appearance, i.e., representation$(w) = \{e_0, e_1, \cdots, e_n\}$. Thus, at the moment that a particular word is used again, all the collected representations are weighted. Similarly to the vocabulary, the number of representations to be stored for a word is virtually infinite, but a choice for a reasonable number is desired.

4.1.2 Distributional-Hypothesis-based Vocabulary

The distributional-hypothesis-based vocabulary (DHV) is an external vocabulary, different from the language model's vocabulary. DHV stores the tokens absent in the language model's vocabulary, together with their vector representation, calculated as shown in Section 4.1.4.

Considering the streaming environment, it is a good practice to keep the structure concise while it is functioning. For this purpose, the Space-saving Algorithm (SSA) (MISRA; GRIES, 1982; MAY et al., 2017) was employed. More specifically, SSA is used to control the number of tokens in the DHV, by analyzing the tokens' frequency. SSA is a streaming algorithm designed to maintain a compact representation of a set of elements while allowing for efficient frequency estimation of these items. According to May et al. (2017), SSA is particularly useful in contexts where the data stream is too large to fit into memory, and retaining a fixed-size summary of this stream is necessary.

On the other hand, considering the multiple representations collected over time for each token and stored in the DHV, a regular queue structure, i.e., first-in-first-out, is used.

4.1.3 Weighted Text Vectorization

Text Vectorization refers to the transformation of unstructured data into structured data (THUMA et al., 2023). SBERT can perform feature extraction from texts, generating a fixed-length numeric vector. This is important for dependent machine learning methods since most cannot handle dynamic-length numeric vectors.

Subsequently, instead of exclusively using the vector representation generated by SBERT, its representation is combined with the external vocabulary representations proportionally. Eqs. 4.1, 4.2, 4.3, 4.4, and 4.5 are used to compute the weighted features extracted.

First, Eq. 4.1 computes the weight of the known tokens by SBERT. Here, a *token* means the same as *word*, i.e., a sequence of characters. The weight is calculated as follows:

$$\texttt{weight}_{\texttt{SBERT}} = \frac{|\texttt{tokens}_{\texttt{SBERT}}|}{|\texttt{tokens}|}, \tag{4.1}$$

where $\texttt{tokens}_{\texttt{SBERT}}$ are the words contained in SBERT's vocabulary, $\texttt{tokens}$ are all words of a given textual input, and $|\cdot|$ is the cardinality operator. This calculation enforces a proportional weight for the representation provided by SBERT. Since the result can be at most 1, the $\texttt{weight}_{\texttt{external}}$ (Eq. 4.2), which is the weight for the vector representations from the external vocabulary, is measured as follows:

$$\texttt{weight}_{\texttt{external}} = 1 - \texttt{weight}_{\texttt{SBERT}} \tag{4.2}$$

where $\texttt{weight}_{\texttt{SBERT}}$ is calculated in Eq. 4.1. The text, encoded by SBERT, is multiplied by the result of Eq. 4.1 in Eq. 4.3.

$$\text{sbert\_features}(\texttt{text}) = \text{encode}(\texttt{text}) \times \texttt{weight}_{\texttt{SBERT}} \tag{4.3}$$

Eq. 4.4 combines the results from Eq. 4.2 and the vector representations of each token contained in the external vocabulary. Eq. 4.4 is computed as follows:

$$\text{external\_features}(\texttt{tokens}) = \sum_{j=0}^{n} \frac{\text{repr}(\texttt{tokens}_j) \times \texttt{weight}_{\texttt{external}}}{n} \tag{4.4}$$

where n is the number of tokens of the input text contained in the external vocabulary, and repr($\texttt{tokens}_j$) is the vector representation of $\texttt{tokens}_j$. Finally, the results from Eqs. 4.3 and 4.4 are combined to generate the final representation, as depicted in Eq. 4.5.

$$\text{final\_features}(\texttt{text}, \texttt{tokens}) = \text{sbert\_features}(\texttt{text}) + \text{external\_features}(\texttt{tokens}) \tag{4.5}$$

The final_features calculated for a token in Eq. 4.5 is the final representation for an input text with words known by the external vocabulary. If no word of the input text is present in the external vocabulary, the final representation is the one provided by SBERT.

4.1.4 New Token Representation Generation

To generate new token representations over time, the method leverages the representations generated by the pre-trained SBERT model. Considering a window size w, Eq. 4.6 depicts how a representation is generated:

$$\text{repr}(\texttt{tokens}_i) = \sum_{k=i-w}^{i+w} \text{encode}(\texttt{tokens}_k, \ldots, \texttt{tokens}_i, \ldots, \texttt{tokens}_{i+w}), \tag{4.6}$$

where $\texttt{tokens}$ is the list of sequential tokens of an input text, k is the first index to be considered in the window, i is the index of the token of interest, and $\texttt{tokens}_i$ is the token of interest. Two clear constraints are: k cannot be lower than 0, and $i + w$ cannot be greater than the length of $\texttt{tokens}$. Please note that w is a hyperparameter.

Regarding w, the experiments were carried out using the value 5. The value of 5 was defined experimentally. Longer windows directly impact the elapsed time. Besides, Hombaiah et al. (2021) mentioned that the values chosen for window sizes in this context range between 3 and 17. Jurafsky & Martin (2008) also pointed out that "shorter windows are likely to obtain syntactic information, and longer windows are more useful for representing meaning".

Each token can accumulate several representations over time. As well as the vocabulary size, it is intended to limit the number of token representations stored over time. This thesis proposes three weighting schemes: (a) **average**, (b) **damped**, and (c) **vanishing**. The **average** method (Eq. 4.7) equally weighs the representations calculated over time. The **damped** scheme (Eq. 4.8) provides more weight to more recent calculated representations. On the other hand, **vanishing** (Eq. 4.9) is similar to the damped in the sense of older representations having lower weight than more recent representations. However, the damped weighting provides increased values mostly for the most recent values, while the vanishing scheme provide increasing values since the oldest representations.

If one opts for averaging the collected representations, represented by Eq. 4.7.

$$\text{final\_repr}(\texttt{token}) = \sum_{j=n}^{0} \left(\frac{\text{repr}(\texttt{token})_j}{n} \right), \tag{4.7}$$

where n is the number of collected representations for a particular token. On the other hand, the **damped** aggregation is represented by Eq. 4.8, as follows:

$$\text{final\_repr}(\texttt{token}) = \sum_{j=n}^{0} \left(\text{repr}(\texttt{token})_j \times ln(0.99 \times j) \right), \tag{4.8}$$

where n is the number of representations of a token. Visually, the behavior of this equation is presented in Fig. 4.2, in which the more to the left, the more recent, and then, the bigger the weight of a word's representation. The x-axis represents the index of a token representation collected over time, where 0 is the most recent representation, and 9 is the oldest.

$$\text{final\_repr}(\texttt{token}, \texttt{weight}) = \sum_{j=n}^{0} \left(\text{repr}(\texttt{token})_j \times \texttt{weight}^{\left(\frac{1}{j+1}\right)} \right), \tag{4.9}$$

where n is the number of representations of a particular token, and repr is the list of representations of a particular token. The factor $\left(\frac{1}{j+1}\right)$ represents the importance depending on the representation index, which favors the most recent representation. This scheme's behavior is visually shown in Fig. 4.3. In this figure, the vanishing scheme uses $\texttt{weight} = 0.10$. The bigger the weight parameter, the smaller the range of weights between the oldest and the most

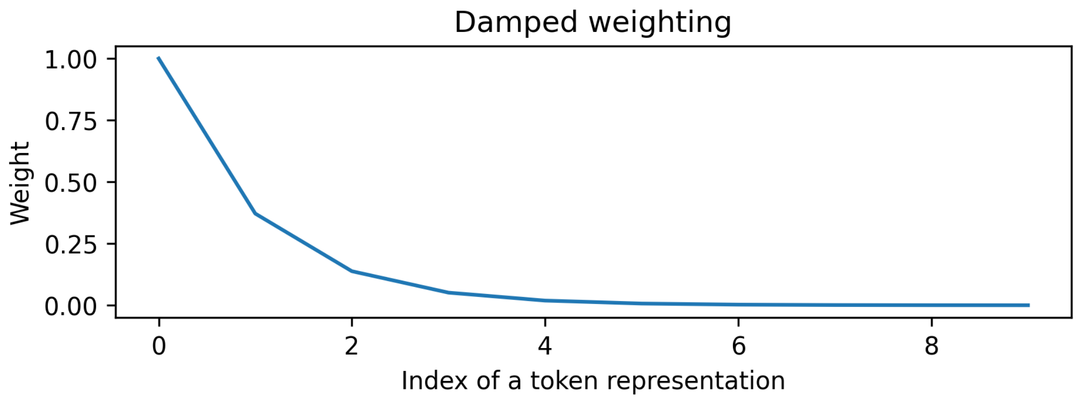

Figure 4.2: Damped weighting scheme of token representations.

recent representations. As in Fig. 4.2, the x-axis represent the index of a token representation collected over time, where the most recent representation (with bigger weights) is on the left, and the more to the right, the older.

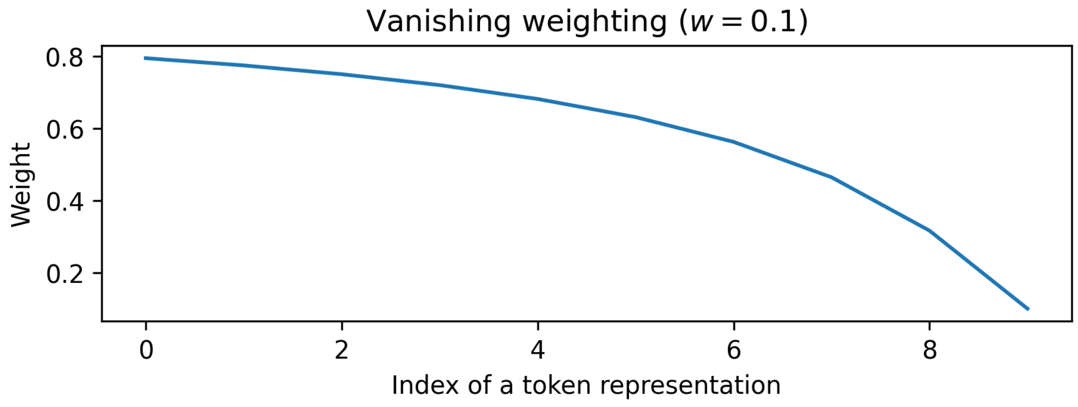

Figure 4.3: Vanishing weighting scheme of token representations.

Although these three weighting schemes are presented, new functions for the same purpose can be developed in future works.

Table 4.1 summarizes the parameters of the **distributed-hypothesis-based method**.

4.2 Bi-SBERT adaptation approach

Bi-SBERT adaptation approach is a method that relies on two SBERTs that share the responsibility of generating the vector representations for the incremental classifier. The rationale behind this approach is that using two methods

Table 4.1: Parameters available for the **distributional-hypothesis-based method**.

Symbol/Name	Description	Default value
w	Controls the number of surrounding words to be used to generate new representations	5
weighting	Defines how the representations collected will be aggregated	*damped* ($w = 0.1$)
vocabulary size	Controls the number of words stored in the vocabulary	1000
repr. limit	Controls the number of representations stored per word	100

can better reflect smooth changes in the text stream, because one of the models keeps older representations. Since the representations provided by the older model is combined with those of the updated model, any representation provided by the most recent model is tempered by the older model. Using Bi-SBERT requires leveraging a concept drift detector to fine-tune one of the models, i.e., the oldest one. This method is applied as follows:

1. A new text from the text data stream is received;
2. The text is processed and cleansed;
3. If only one SBERT is available, i.e., no drift was previously detected, the SBERT model generates the vector representation. On the other hand, if two SBERTs are available, the input text is applied to them, and the vector representations are averaged to generate only one vector representation;
4. The vector representation is applied to the machine learning model, which outputs a prediction;
5. The output prediction and the ground-truth label are input to the concept drift detector;
6. If a concept drift is deemed to have occurred (Section 4.2.1), the fine-tuning process starts (step 7, Section 4.2.2). Otherwise, the text stream continues to be processed (jump to step 1);
7. Texts are sampled from the buffer;
8. The oldest SBERT is fine-tuned using the items sampled from the buffer (Section 4.2.2). After the fine-tuning process, the oldest SBERT is removed, and the recently fine-tuned model is appended to the SBERTs pool. Later, the text stream continues to be processed (jump to step 1).

Fig. 4.4 visually describes the scenario and the proposed Bi-SBERT method.

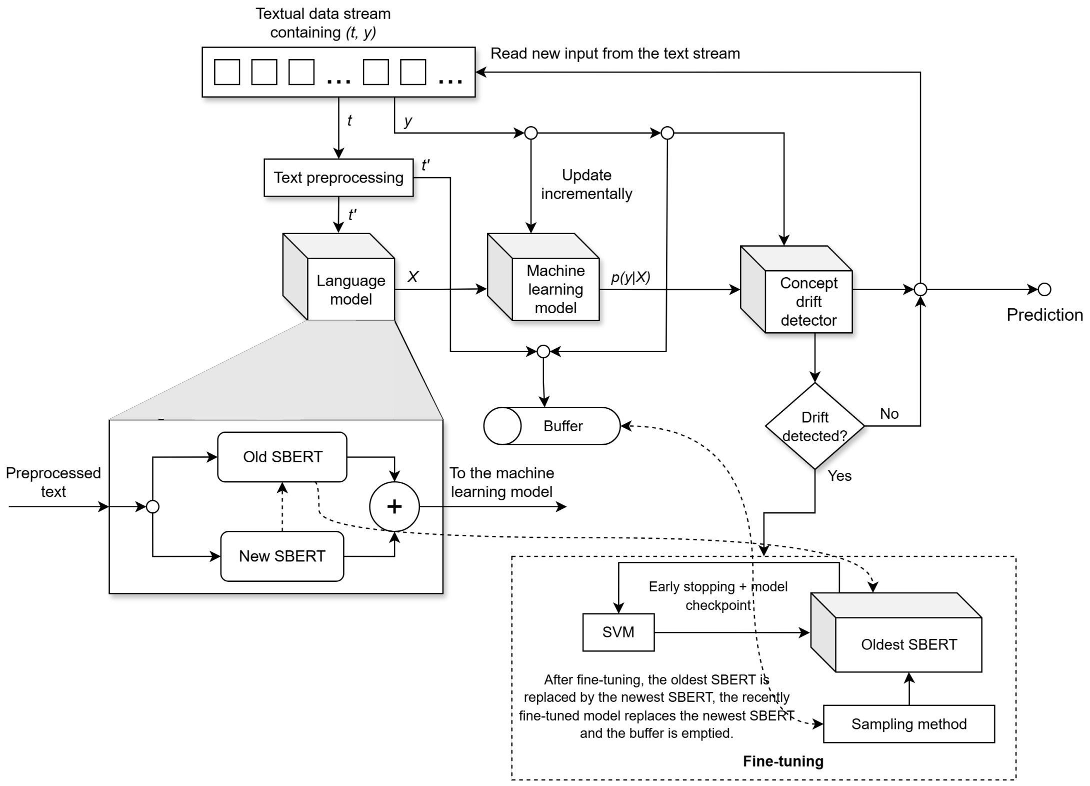
Textual data stream containing (t, y)
Read new input from the text stream
t
y
Text preprocessing
t'
t'
Update incrementally
Language model
X
Machine learning model
p(y|X)
Concept drift detector
Prediction
Buffer
Drift detected?
No
Yes
Preprocessed text
Old SBERT
New SBERT
+
To the machine learning model
Early stopping + model checkpoint
SVM
Oldest SBERT
Sampling method
After fine-tuning, the oldest SBERT is replaced by the newest SBERT, the recently fine-tuned model replaces the newest SBERT and the buffer is emptied.
Fine-tuning

Figure 4.4: Setting for the proposed Bi-SBERT method. In this setting, preprocessed texts are stored in a buffer for posterior use in fine-tuning. The method consists of combining representations generated by two (Old and New) SBERT models. The fine-tuning procedure is always applied to the *Old* model, using the texts stored in the buffer.

4.2.1 Drift Detection Process

During the development of this thesis, the tests suggested a – previously unexpected – behavior considering the drift detector. In the case of using hits and errors, i.e., 0s and 1s, as input for drift detectors, the drift detectors could not detect expected drifts in multiclass scenarios. Therefore, to overcome this limitation, **a pool of concept drift detectors** was created, where each concept drift detector monitors each class individually. Thus, for the multiclass cases, instead of using only one concept drift detector, a pool of them was leveraged, i.e., one detector per unique label. For the binary cases, the traditional use, i.e., only one drift detector, was employed.

Besides, this thesis leveraged ADWIN (BIFET; GAVALDA, 2007), a concept drift detector that detects the increase of both errors and hits. However, for this thesis, only the increase in errors is of interest. Therefore, a control mechanism was needed to prevent the detection of increase in hits. In addition, a drift is deemed to have occurred if the current Macro F1 (considering the text stream processing) was lower than the Macro F1 collected in the last occurred drift. The latter was developed to prevent drift from being detected if Macro F1 is in an increasing trend. Thus, three conditions must be met simultaneously for a drift to be deemed to have occurred: (a) the concept drift detector detects a drift, (b) considering updating ADWIN with 0 in case of hits and 1 for errors, ADWIN's current estimation must be bigger than its estimation before the last update (thus preventing detections regarding the increase in hits), and (c) the current Macro F1 must be lower than the Macro F1 collected in the last drift detected (by the drift detector only). Considering the multiclass scenario, a scheme with one concept drift detector per class is leveraged. This strategy prevents a concept drift detection to be impacted primarily by the majority class(es). Only the drift detector regarding the current item class is evaluated and updated over time. This separates drift behavior by class.

4.2.2 Fine-tuning Process

Three components are essential for the proposed fine-tuning process: (a) buffer(s), (b) sampling method, and (c) a linear support vector machine classifier (SVM). The buffer stores items collected throughout the text stream processing. For multiclass settings, a set of buffers is kept, one per label. The sampling method samples items from the buffer. Using one buffer per label allows class-based sampling methods. The sampled items are then split into train (90%) and test (10%) sets.

During the fine-tuning process, an SVM was employed to guide the language model fine-tuning. The choice of SVM is due to its ability to approximate linear separation quickly. In addition, the tests pointed out the lack of a clear correlation between loss values and Macro F1-scores obtained. Thus, instead of using the loss values, the Macro F1-scores obtained from the test set were used. In each epoch, an SVM is trained from scratch using the representations generated with the newly fine-tuning method. The Macro F1-scores obtained from the evaluation of the test set are used to define whether the language model should be fine-tuned for more iterations. In other words, the fine-tuning process leverages the early stopping (YAO; ROSASCO; CAPONNETTO, 2007) and model checkpoint procedures.

Algorithm 2 details the fine-tuning procedure. Alg. 2 requires a buffer with texts and their respective labels, a maximum patience, which represents the maximum number of epochs without improving the Macro F1-scores regarding the test set, and, since this method resorts to two SBERTs, both are passed to the procedure, under the names of *New* and *Old*. If only one SBERT model is available, after the fine-tuning procedure, it becomes the oldest, and the fine-tuned SBERT model becomes the newest. A holdout split is performed, generating the train and test sets. Between lines 1 and 3, the current patience and the `top_f1_score` are set as 0, and the language model to be updated is set as the `old_SBERT`. In line 4, the train and test sets are defined. In lines 5 and 6, the train and test sets are encoded (vectorized) by the language model. From line 7 to 19, until the maximum patience is reached, a new SVM is trained from scratch with the encoded items from the SBERT model. In line 10, the Macro F1-score is calculated, and if it is higher than the top Macro F1-score calculated so far (line 11), the fine-tuned SBERT model is stored as the *top language model* (line 12). The patience variable is incremented if the Macro F1-score is not higher than the top Macro F1-score (line 13). After, in line 17, the SBERT model is fine-tuned for one more epoch. In lines 18 and 19, the train and test sets are again vectorized, but with the recently fine-tuned language model. After the patience reaches the maximum patience, *old* SBERT becomes the name of reference for the new SBERT, and the name *new* SBERT points to the recently fine-tuned SBERT model, as it can be seen in lines 21 and 22.

Table 4.2 summarizes the parameters of the **distributed-hypothesis-based method**.

Table 4.2: Parameters available for the **Bi-SBERT method**.

Symbol/Name	Description	Default value
w	Controls the number of surrounding words to be used to generate new representations	5
weighting	Defines how the representations collected will be aggregated	*damped* (w = 0.1)
vocabulary size	Controls the number of words stored in the vocabulary	1000
repr. limit	Controls the number of representations stored per word	100
sampling method	Samples from the buffer based on characteristics of texts	WordPieceToken ratio
max patience	Maximum epochs for the fine-tuning	20

Algorithm 2 Fine-tuning procedure

```
Require: buffer                    ▷ Texts in the buffer also have their labels
Require: max_patience    ▷ Epochs without improving Macro F1 of the test set
Require: new_SBERT                              ▷ The newest language model
Require: old_SBERT                               ▷ The oldest language model
 1: patience ← 0
 2: top_f1_score ← 0
 3: language_model ← old_SBERT
 4: train_set, test_set ← train_test_split(buffer)
 5: train_set_encoded ← language_model.encode(train_set)
 6: test_set_encoded ← language_model.encode(test_set)
 7: while patience < max_patience do
 8:     svm ← LinearSupportVectorMachine()
 9:     svm.train(train_set_encoded)
10:     f1_test ← f1_scores(svm.predict(test_set_encoded))
11:     if f1_test > top_f1_score then
12:         top_language_model ← language_model
13:         top_f1_score ← f1_test
14:     else
15:         patience ← patience + 1
16:     end if
17:     language_model.fine_tune(train_set)
18:     train_set_encoded ← language_model.encode(train_set)
19:     test_set_encoded ← language_model.encode(test_set)
20: end while
21: old_SBERT ← new_SBERT                       ▷ Replaces the SBERT models
22: new_SBERT ← top_language_model
```

4.3 Concluding Remarks

This section presents the core methods of this thesis. The proposal mostly adheres to the concepts of text stream processing.

The proposed methods leverage pre-trained model vocabulary, which generally provides good quality vector representations. Learning new words through fine-tuning can demand a large amount of data and time, which is unsuitable for text stream settings. Therefore, the first proposed method leverages the distributional hypothesis to generate new representations, thus extending the vocabulary while improving the results regarding metrics.

Thus, an important structure addition is the external vocabulary. The new representations are generated from the pre-trained BERT vocabulary representations. It is possible to generate new representations from scratch in this setting, but mixing two vector representation generation processes would potentially create bias. Therefore, the SBERT's vocabulary is used as the original vocabulary.

Regarding the Bi-SBERT approach, the rationale is that two SBERTs could yield a single representation. Instead of using only one model, which under fine-tuning would incorporate recent changes providing a possible "abrupt" change in the embedding distribution, leveraging two models would encompass information from older and the more recent models at once, providing a representation that considers older and newer patterns.

Next chapter presents the experimental protocol to evaluate the aforementioned proposed methods.

5 Experimental Protocol

This chapter describes the experimental protocol adopted to assess the proposed methods. First, the software and hardware used in the experiments are detailed in Section 5.1. Next, the datasets are presented in Section 5.2. Then, details on the drift simulation process are described in Section 5.3. Section 5.4 provides information on concept drift detectors. Section 5.5 describes the sampling methods used in this thesis. Section 5.6 presents the metrics. Finally, Section 5.8 summarizes the setting defined for the experiments.

5.1 Software and Hardware

To obtain the results, this thesis leveraged the following tools: Python 3.9.12[1] for script development, Pandas[2] for data manipulation, Numpy[3] for matrix operations, Matplotlib[4] and Seaborn[5] for data visualization, SentenceTransformers[6] as implementation of SBERT, and RiverML[7] for text streams simulation, evaluation metrics, and implementation of concept drift detectors.

In addition, the experiments were executed on a machine with the following

[1]https://www.python.org/
[2]https://pandas.pydata.org/
[3]https://numpy.org/
[4]https://matplotlib.org/
[5]https://seaborn.pydata.org/
[6]https://www.sbert.net/
[7]https://riverml.xyz/

configuration: 24-core 13th Gen Intel(R) Core(TM) i9-13900K, with 32 CPUs, 128 GB of RAM, and 2 × GPUs NVIDIA GeForce RTX 4090.

5.2 Datasets

In this thesis, the experiments were run using six datasets: (a) Airbnb, (b) Amazon Polarity, (c) Amazon Reviews, (c) Yelp, (d) Yelp Full, and (e) Yelp Polarity. The downstream task related to this dataset is *text stream classification*. Table 5.1 shows some characteristics of each dataset.

Table 5.1: Categorization of the datasets used in the experiments.

Dataset	**Category**	**# of classes**	**Class proportion**
Airbnb	Multiclass	3	Very imbalanced ≈(2.5/2.5/95)%
Amazon Polarity	Binary	2	Balanced
Amazon Reviews	Multiclass	5	Balanced
Yelp	Multiclass	5	Imbalanced ≈(15.3/7.8/9.9/20.8/46.2)%
Yelp Full	Multiclass	5	Balanced
Yelp Polarity	Binary	2	Balanced

Below, a few metrics regarding the aforementioned datasets are presented. These numeric metrics can help understand behaviour and elapsed times regarding the models. Table 5.2 shows information about the number of characters, the number of tokens, the number of wordpieces, WordpieceToken ratio (WPT, explained in Section 5.5), number of unknown tokens, and mean ratio of unknown tokens per instance regarding the datasets in the scenario **Without drift**.

Table 5.2: Characteristics of each dataset in terms of number of characters, number of tokens, number of wordpieces, WordpieceToken ratio, and number of unknown tokens. The values are calculated per row, and then, averaged.

Dataset	**Chars.**	**Tokens**	**Wordpieces**	**WPT ratio**	**Unk. tokens**
Airbnb	256.69 ± 262.92	51.41 ± 53.59	53.28 ± 55.66	3.46% ± 0.09	1.31 ± 2.00
Amazon Polarity	415.39 ± 238.28	85.74 ± 48.59	90.45 ± 51.51	5.47% ± 0.07	3.25 ± 3.48
Amazon Reviews	422.73 ± 240.72	57.24 ± 49.07	91.99 ± 52.02	5.40% ± 0.06	3.29 ± 3.54
Yelp	575.62 ± 531.70	119.45 ± 110.68	126.17 ± 116.74	5.92% ± 0.09	4.83 ± 5.75
Yelp Full	743.27 ± 671.02	156.91 ± 141.54	168.31 ± 152.52	7.36% ± 0.10	7.89 ± 10.52
Yelp Polarity	736.52 ± 675.81	155.38 ± 142.58	153.27 ± 166.45	7.28% ± 0.11	7.68 ± 10.29

Resorting to Table 5.2, it is noticeable that the datasets are different considering the measured aspects. Airbnb contains shorter texts in general, leading also to fewer unknown tokens.

5.2.1 Airbnb

Airbnb dataset is a dataset based on data available in Inside Airbnb[8], filtered by language and classified in a zero-shot fashion using Llama3 (TOUVRON et al., 2023), through Ollama[9]. The complete treatment process is similar to that described in Garcia et al. (2024), and is described below.

The selected dataset contains data regarding New York Airbnb rents. This dataset could provide a large amount of data as New York is a frequent destination in the USA, being the most visited city in 2022[10]. Since this city receives worldwide tourists, the first step was to identify the language, filtering out the non-English posts. To this purpose, a pre-trained model for language identification, i.e., lid.176.ftz (JOULIN et al., 2016a; JOULIN et al., 2016b) was used.

As previously mentioned, Llama3 was used in zero-shot fashion for Airbnb review classification. The capability of Large Language Models (LLMs) such as GPT3 and Llama2 as zero-shot text classifiers is well known (WANG; PANG; LIN, 2023). Zero-shot is a prompt strategy for LLM in which the LLM is required to perform a task without providing any knowledge or examples (WANG; PANG; LIN, 2023). After this process, the dataset is ready to be used in the classification task. Totaling 879,919 records, it is a highly imbalanced dataset due to its domain, which tends to stop offering properties or rooms with bad reviews. Labels with value 0 tag a negative review; value 1 tags a neutral review, and value 2 tags a positive review.

5.2.2 Amazon Polarity

Amazon Polarity[11] is a dataset that contains products reviews collected from Amazon, spanning 18 years, according to its curators (ZHANG; ZHAO; LECUN, 2015). The label in the dataset corresponds to positive or negative. Those values were obtained according to the review's score (stars): considering values from 1 to 5, 1 and 2 became negative, 3 and 4 became positive, and reviews with 3 stars were discarded.

[8]<https://insideairbnb.com/>

[9]<https://ollama.com/>

[10]Available at: <https://www.cntraveler.com/story/most-visited-american-cities>. Accessed on May 27th, 2025.

[11]<https://huggingface.co/datasets/fancyzhx/amazon_polarity>

5.2.3 Amazon Reviews

Amazon Reviews[12] is a dataset formed by reviews scraped from Amazon. Amazon is a multinational technology company. Reviews were collected from Amazon's e-commerce. Our subset contains 3 million rows equally distributed across classes. This is a 5-class dataset, alluding to the number of stars, i.e., from 1 to 5.

5.2.4 Yelp dataset

Yelp dataset[13] corresponds to reviews on more than 500,000 businesses in eleven metropolitan areas. In addition, those reviews range from 1 to 5, which is the score (or stars) a user gives to a particular business. Therefore, it corresponds to a multiclass text stream classification. This dataset contains the following attributes: text, date, and stars.

5.2.5 Yelp Full

Yelp Full is a dataset corresponding to reviews and originally used in Zhang, Zhao & LeCun (2015). This dataset is similar to the Yelp dataset, but contains different texts. In addition, the Yelp Full dataset is balanced.

5.2.6 Yelp Polarity

Yelp Polarity dataset is the Yelp Full dataset after passing through a process to binarize it. Texts regarding 1 and 2 stars are deemed to be *negative*, 4 and 5 stars are considered *positive*, and 3 stars are ignored.

5.3 Textual Concept Drift Simulation

Although the presented datasets contain a variable number of instances, from each original dataset, five subsets were generated with 200,000 random instances each, respecting the temporal order, to ease the evaluation. Subsequently, these

[12]<https://www.kaggle.com/datasets/danielihenacho/amazon-reviews-dataset>
[13]<https://www.yelp.com/dataset>

subsets are copied, and the drift simulation methods proposed in Section 5.3 are applied.

This thesis concerns the real concept drift in textual data streams, i.e., changes in $p(y|X)$ over time. Even though there is a consensus that concept drift is generally present in textual streams (STEWART et al., 2017; HEUSINGER; RAAB; SCHLEIF, 2020a), it has been verified in the systematic literature review that most datasets do not have drifts labeled. In addition, the changes may not happen at a velocity sufficient for experimentation. Furthermore, for evaluation reasons, it is necessary to have drifts labeled, and synthesizing them becomes a prominent solution.

Thus, in order to have labeled drifts, three textual concept drift generation settings are proposed: (a) class swap, (b) class shift, (c) time-slice removal, and (d) adjective swap (BRAVO-MARQUEZ; KHANCHANDANI; PFAHRINGER, 2022a). These methods were applied to evaluate the ability of machine learning models to recover from text drift in text streams (GARCIA et al., 2024), i.e., paper #6 in Chapter 1.5. Although performing the aforementioned changes may not reflect real-world changes, the changes in the latent space can be enough for both language and classifier model updates.

5.3.1 Class Swap

In this setting, the classes are inverted only once. The intention is to simulate a change similar to an abrupt concept drift. Concerning the Yelp dataset, which is a 5-class dataset, the classes are inverted according to Table 5.3. Regarding binary datasets, the first class is replaced by the second one and vice-versa. These changes are designed to happen during the stream, and in our setting, they occur at $t = 50,000$. Furthermore, the Class Swap drift simulation aims to simulate *abrupt changes*, that is, it provides more drastic changes compared to the Class Shift drift simulation (Section 5.3.2).

Table 5.3: Class swap in 5-class dataset.

Original class	**Inverted class**
1	5
2	4
3	3
4	2
5	1

It is important to note that the middle class in multiclass setting, i.e., class 3 in the 5-class datasets, is not inverted given the odd number of classes involved. Fig. 5.1 visually shows the class swap method.

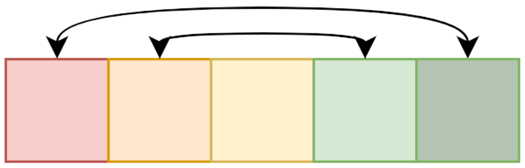

Figure 5.1: Representation of the class swap method. Each square represents a class and the arrows point to the swapped classes.

5.3.2 Class Shift

The Class Shift occurs three times, i.e., after 50,000 instances, during the text stream processing in this setting. The goal of the Class Shift procedure is to simulate incremental changes. Table 5.4 depicts the class rotation for the Yelp dataset.

Table 5.4: Class shift in a 5-class dataset.

Original class	**Rotated class**
1	2
2	3
3	4
4	5
5	1

Although one of the changes is discontinuous, i.e., from 5 to 1, the other changes are smoother, and this change from 5 to 1 can be assumed as a side effect. Apart from the 5 to 1 replacement, the other changes are smoother because the classes tend to be similar, e.g., from 1 to 2, from 2 to 3, and so on. In binary datasets, the effect is essentially the same as the class swap procedure. However, unlike the Class Swap procedure, the class shift happens three times during text stream processing. Fig. 5.2 visually describes this procedure.

5.3.3 Time-slices Removal

The textual concept drift generation process in this setting differs substantially from the aforementioned ones. In case of datasets that contain a timestamp

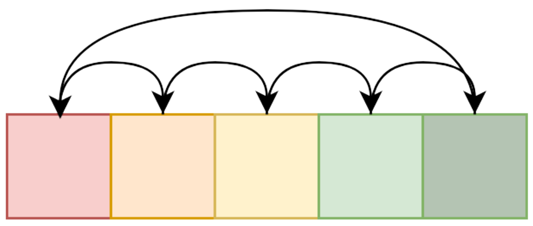

Figure 5.2: Representation of the class shift method. Each square represents a class, and the arrows point to the shifted classes.

attribute, and the complete dataset contains reviews comprising several years, three years are randomly selected to be left out of the sampling. Therefore, the expectation is to generate textual concept drifts in the gaps left by the years removed, since it is well that the characteristics of written texts tend to change over time (HAMILTON; LESKOVEC; JURAFSKY, 2016b). Fig. 5.3 graphically represents this type of concept drift.

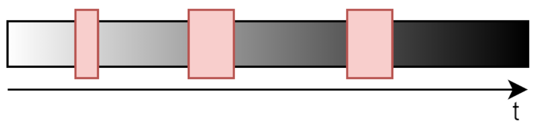

Figure 5.3: Representation of the time-slice removal method. Each red square represents a year excluded from the stream (black rectangle). The x-axis represents time.

5.3.4 Adjective Swap

The Adjective Swap method was first proposed in Bravo-Marquez, Khanchandani & Pfahringer (2022a). In that paper, the authors developed the adjective swap process to evaluate the ability of their method to recognize sentiment changes in specific words (tokens). Therefore, the authors replaced some adjectives with their antonyms. Although the authors focused on tokens, our aim by using the adjective swap method is to reverse the sense of a sentence. Please, note that, for a given input text, the adjectives are swapped, but the corresponding class remains the same. Fig. 5.4 represents the adjective swap method.

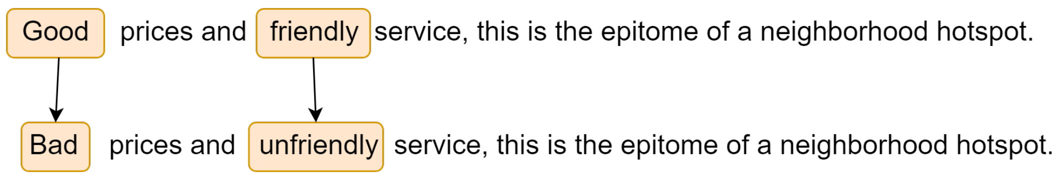
Good prices and friendly service, this is the epitome of a neighborhood hotspot.
Bad prices and unfriendly service, this is the epitome of a neighborhood hotspot.

Figure 5.4: Representation of the adjective swap method. The arrows point to the swapped adjectives.

5.4 Concept Drift Detectors

In the experiments, ADWIN (BIFET; GAVALDA, 2007) was used as the concept drift detector. The detector is utilized to trigger the SBERT's adaptation in cases where fine-tuning is used and whenever Bi-SBERT is used. In addition, ADWIN warns both increases and decreases in the mean of the observed variable, therefore, using the classifier's errors as input for the detector, the detector warns when the error increases (desirable) and when the error decreases (unnecessary in our case).

As pointed out in Chapter 4, for the Bi-SBERT method, ADWIN was used together with mechanisms and conditions to prevent undesired detections. The conditions are: (a) the current Macro F1-Score must be lower than the Macro F1-Score collected in the last ADWIN-only drift detection, and (b) the current ADWIN's estimation must be bigger than before the same estimation prior to the last ADWIN update. More details on this process are available in Chapter 4. In our work, ADWIN's default parameters were used, including the significance value, i.e., delta (δ) = 0.002.

ADWIN was chosen due to its consistency. In Garcia, Britto Jr & Barddal (2023), one of the works developed as a step for this thesis, i.e., publication #3 in Section 1.5, four concept drift detection methods were tested. ADWIN was more precise, i.e., it was the detector that detected drifts the closest to when the actual drifts happened, and triggered fewer false positives than HDDM (in its average and weighted versions) (FRIAS-BLANCO et al., 2014) and EDDM (BAENA-GARCIA et al., 2006).

5.5 Sampling Methods

During the algorithm execution, a buffer is filled with instances from the stream for the posterior model update. However, since the environment is a text stream, the initial idea was to select only a few instances from the buffer. This thesis leverages the WordpieceToken ratio sampling method (GARCIA et al., 2025; IMAI et al., 2024), developed during this thesis and evaluated in publications #5 and #7 in Section 1.5. The WordpieceToken ratio method showed to be helpful for the fine-tuning procedure, which provided better-quality representation for the classifier in 6 out of 8 scenarios in Garcia et al. (2025).

Consider a text population P, i.e., the instances stored in the buffer, $P = \{t_1, t_2, ..., t_n\}$, where n is the count of instances in the buffer, and t_i is a text instance of index i from the buffer. The weight w_i can be calculated considering different aspects, such as the length of an instance t_i, the confidence of a classifier on that instance, and so on. Next, the weights must be normalized to represent their probability of being sampled, i.e., $w'_i = \frac{w_i}{\sum_{j=1}^{n} w_j}$. Therefore, the sampling method's probability of sampling the j-th text in the buffer is w'_{ij}. Alg. 3 generalizes the weighted sampling. The main difference regards how the weights associated with the instances are calculated.

Algorithm 3 Algorithm of weighted sampling

```
Require: $n_s$                                   ▷ number of instances to be sampled
Require: buffer
Ensure: buffer ≠ ∅
  $w \leftarrow$ weights associated with instances $t$ in the buffer
  $\texttt{sum}_{\texttt{weights}} \leftarrow \sum_{j=1}^{n} w_j$
  for each $t_i \in$ buffer do
      $t_i.w' \leftarrow \frac{t_i.w}{\texttt{sum}_{\texttt{weights}}}$
  end for
  Sample $n_s$ instances using the weight of each buffer instance, i.e., $t_i.w'$, as
  probability
```

Wordpiece (WU et al., 2016) is a subword representation present in BERT. For example, the word `encompassing` could be tokenized using wordpieces as $[en, \#\#compass, \#\#ing]$, where the characters ## indicate the existence of previous wordpieces. Technically, wordpieces aids in the representation of unknown and out-of-vocabulary words. However, as expected, the generated representations for unknown words may not be as accurate as for known words. Therefore, this sampling method follows Eq. 5.1 to calculate the weights of the instances in the buffer:

$$\text{WordPieceTokenRatio}(\texttt{sentence}) = \frac{\text{count}(\text{wordpieces}(\texttt{sentence}))}{\text{count}(\text{words}(\texttt{sentence}))}, \tag{5.1}$$

where count(.) returns the length of the argument, wordpieces(.) returns a wordpiece list of a given sentence, and words(.) returns a token list (list of words) from a given text input. The rationale behind this calculation is: the higher the ratio, the more words are unknown to the vocabulary. Therefore, a high ratio indicates that the sentence is challenging for the SBERT language model since some words/tokens were "assembled" from wordpieces, and those words do not have their own representation.

5.6 Metrics

The selected metrics for this thesis proposal are: (a) Macro F1-Score and (b) run time.

Macro F1-Score is an interesting metric for imbalanced classification due to its ability to weight classes equally, preventing a misinterpretation of a classifier's performance based only on accuracy. F1-Score is calculated according to the Eq. 5.2.

$$\text{F1} = \frac{2}{\frac{1}{\texttt{precision}} + \frac{1}{\texttt{recall}}}, \tag{5.2}$$

where $\texttt{precision} = \frac{TP}{TP+FP}$, and $\texttt{recall} = \frac{TP}{TP+FN}$. This metric can achieve values between 0 and 1, in which the closer to 1, the better. In addition, F1-Score is the harmonic mean between precision and recall; therefore, for F1-Score to be high, both precision and recall need to be high simultaneously.

Macro-F1 Score is the calculated average between the F1-Scores regarding the classes in a given multiclass classification problem. Therefore, mathematically, Macro F1-Score can be described as:

$$\text{Macro-F1} = \frac{\sum_{i=1}^{c} \text{F1}(i)}{n}, \tag{5.3}$$

where $F1(i)$ corresponds to the F1-Score calculated for a given class i.

Since the chosen environment is a text stream setting, the time spent to evaluate the stream is relevant. Therefore, all the run times, in seconds, are stored for comparison. In addition, the wall time is considered, i.e., the elapsed time.

Regarding the Macro F1-Score, the evaluation is performed prequentially. This means that the proposed method receives an input text X_i, predicts a class $\hat{y}$, and then, the label y becomes available, and the model can learn from it. That is, testing and training steps take place in an interleaved fashion. In addition, the Macro F1-Scores are calculated incrementally, without resorting to windows.

5.7 Compared Methods

This thesis compares twelve methods. Considering the two methods proposed in this thesis, they were combined with different machine learning methods. Therefore, the combinations compared in this thesis are displayed in Table 5.5.

Above, the methods proposed in this thesis are in bold. The SBERT pretrained model used in all settings is `sentence-transformers/paraphrase-MiniLM-L6-v2`. This model generates 384-dimension embeddings. In addition, the Neural Network used in the combinations above has the architecture displayed in Fig. 5.5. In this figure, the blue rectangle represents the vector representation, coming from SBERT model, Bi-SBERT model, with or without the distributed-hypothesis-based method. The last layer has its size regarding the number of classes, i.e., $|C|$, according to the dataset used in the experiment.

For the above architecture, a dropout value of 20% was fixed, and RMSProp (HINTON, 2012) was used to optimize the parameters, which used similar values as Ghahramanian et al. (2024), i.e., learning rates of $1e-2$, $1e-3$, and $1e-4$, respectively, considering the architecture shown in Fig. 5.5. Besides, other parameters used were $\alpha = 0.99$, $\epsilon = 1e-08$, and momentum= 0, and ReLU as the activation function.

5.8 Settings

The approaches listed in Section 5.7 were evaluated in this thesis under

Table 5.5: Methods compared in this thesis, together with their descriptions. Bold values correspond to the methods proposed in this thesis.

#	Method	Description
1	AdaNEN	AdaNEN (GHAHRAMANIAN et al., 2024) is a state-of-the-art neural method designed for text stream classification (See Chapter 3.3).
2	AdaNEN + **Distr.-Hyp.-based method**	AdaNEN with SBERT representations incrementally updated with the **Distr.-Hyp.-based method** (See Chapter 4.1).
3	**Bi-SBERT** + NN	**Bi-SBERT** (see Chapter 4.2) coupled with a generic, incrementally-updated neural network (NN) (Fig. 5.5, Chapter 5.7).
4	**Bi-SBERT** + NN + **Distr.-Hyp.-based method** (fully incr.)	**Bi-SBERT** with incremental updates in representations.
5	**Bi-SBERT** + NN + **Distr.-Hyp.-based method** (fine-tuning)	**Bi-SBERT** with updates only when Bi-SBERT fine-tuning is triggered and finished.
6	SBERT + ISVM	SBERT with an Incremental Linear SVM as classifier.
7	SBERT + ISVM + **Distr.-Hyp.-based method**	SBERT representations incrementally updated with the **Distr.-Hyp.-based method** coupled to an Incremental Linear SVM as classifier.
8	SBERT + NN	SBERT coupled to a generic neural network (Chapter 5.7).
9	SBERT + NN + **Distr.-Hyp.-based method**	SBERT with an NN incrementally updated.
10	SBERT (Fine-tune) + NN	SBERT with fine-tuning with an incrementally updated NN.
11	SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fully incr.)	SBERT with fine-tuning and an incrementally updated NN, and incremental representation updates performed by **Distr.-Hyp.-based method**.
12	SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning)	SBERT with fine-tuning and an incrementally updated NN and representation updates performed by **Distr.-Hyp.-based method** only whenever the SBERT fine-tuning is finished.

the same experimental protocol. For each dataset presented in Section 5.2, we generated five stratified samples, i.e., subsets, with size 200,000 to simulate text streams. This setting is hereafter named **Without drift**. In addition, based on generated subsets, we simulated different types of drift according to the methods presented in Section 5.3. Table 5.6 lists the scenarios according to the subsets used in this thesis. Therefore, each method presented in Section 5.7 is executed 130 times, according to the aforementioned table.

Please note that only the Airbnb and Yelp datasets contain the Time-slice Removal type of drift applied in their subsets. This happened because this type of drift considers the timestamp, which is not available in the remaining datasets.

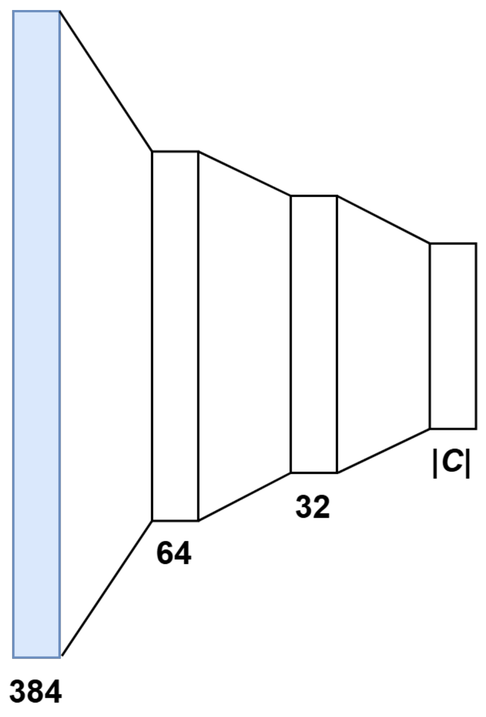

Figure 5.5: Neural network architecture used in the experiments.

5.9 Statistical Evaluation

This thesis compares the aforementioned methods regarding Macro F1-scores and time. The statistical evaluation focused on the Macro F1-scores. In this case, although the importance of elapsed times is acknowledged, the most important metric is the Macro F1-Scores. Besides, using box plots, it is easier to note the differences among elapsed times.

For the sake of statistical evaluation of the effectiveness of the distributional-hypothesis-based method proposed in this thesis, a given method was compared against the same method with the Distributional-Hypothesis-based method. For instance, considering the aforementioned list, one comparison performed was `SBERT + AdaNEN` vs. `SBERT + AdaNEN +` **`Distributional-Hypothesis-based method`**. Besides, the method **without** the Distributional-Hypothesis-based method is referred to as the *vanilla* version of an approach. Table 5.7 lists the approaches statistically compared in this thesis.

In this thesis, the Wilcoxon signed-rank test (WOOLSON, 2005) was employed. This test was chosen because of its ability to consider the ranking of two paired samples. This characteristic is important because some differences might

Table 5.6: Scenarios regarding datasets and subsets used in this thesis.

#	Dataset	Drift simulation	Subsets
1	Airbnb	Without drift	5
2		Class swap	5
3		Class shift	5
4		Time-slice removal	5
5		Adjective swap	5
6	Amazon Polarity	Without drift	5
7		Class swap	5
8		Class shift	5
9		Adjective swap	5
10	Amazon Reviews	Without drift	5
11		Class swap	5
12		Class shift	5
13		Adjective swap	5
14	Yelp	Without drift	5
15		Class swap	5
16		Class shift	5
17		Time-slice removal	5
18		Adjective swap	5
19	Yelp Full	Without drift	5
20		Class swap	5
21		Class shift	5
22		Adjective swap	5
23	Yelp Polarity	Without drift	5
24		Class swap	5
25		Class shift	5
26		Adjective swap	5
Total	6 datasets	-	130 subsets

be insufficient to be statistically significant in other statistical tests. Besides, the comparison directly addresses two groups: the results of a method *with* the Distributional-Hypothesis-based method, and *without* the method. Since it is not possible to guarantee data normality, the Wilcoxon signed-rank test is a suitable choice because it is non-parametric.

As explained in Section 5.8, each approach was executed 5 times per scenario. Although it would be possible to apply the Wilcoxon signed-rank test per scenario, the effect of randomness would be more significant in only 5 runs than in settings with more paired samples. Thus, the Wilcoxon signed-rank test was performed considering two main aspects: **datasets** and **drift scenarios**. The

Table 5.7: Statistical comparisons performed in this thesis.

#	Method 1	Method 2
1	AdaNEN	AdaNEN + **Distr.-Hyp.-based method**
2	**Bi-SBERT** + NN	**Bi-SBERT** + NN + **Distr.-Hyp.-based method** (fully incr.)
3	**Bi-SBERT** + NN	**Bi-SBERT** + NN + **Distr.-Hyp.-based method** (fine-tuning)
4	**Bi-SBERT** + NN + **Distr.-Hyp.-based method** (fully incr.)	**Bi-SBERT** + NN + **Distr.-Hyp.-based method** (fine-tuning)
5	SBERT + ISVM	SBERT + ISVM + **Distr.-Hyp.-based method**
6	SBERT + NN	SBERT + NN + **Distr.-Hyp.-based method**
7	SBERT (Fine-tune) + NN	SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fully incr.)
8	SBERT (Fine-tune) + NN	SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning)
9	SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fully incr.)	SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning)

rationale behind these choices is to have more paired data and evaluate if any dataset or drift scenario benefits from the methods proposed in this thesis.

To sum up, our null hypothesis h_0 states that there is no significant difference between Macro F1-scores from Method 1 and Method 2 (see Table 5.7). On the other hand, our alternative hypothesis h_a expresses that there is a significant difference between Macro F1-scores from Method 1 and Method 2. For the statistical tests, the significance value was defined as $\alpha = 5\%$.

6 Results

This chapter presents the results obtained using the proposed methods. First, an analysis organized by dataset and drift simulation types is performed. The assessed methods are also compared statistically, following the protocol described in Section 5.9. Afterward, an ablation study is performed, and finally, a discussion and concluding remarks are provided.

6.1 Per dataset

This section presents the results organized by dataset and drift simulation types, according to Table 5.6.

6.1.1 Airbnb

This subsection describes the results obtained for the Airbnb dataset, which is a 3-class dataset.

Without drift

Fig. 6.1 shows the Macro F1-Scores for the scenario **without drift** and the Airbnb dataset. The highest Macro F1-Scores reached around 70%, using the **Bi-SBERT** approaches (with and without the **Distributed-Hypothesis-based method** (with updated on fine-tuning)). In addition, the **Distributed-Hypothesis-based method** seems to slightly improve the results also for Ada-

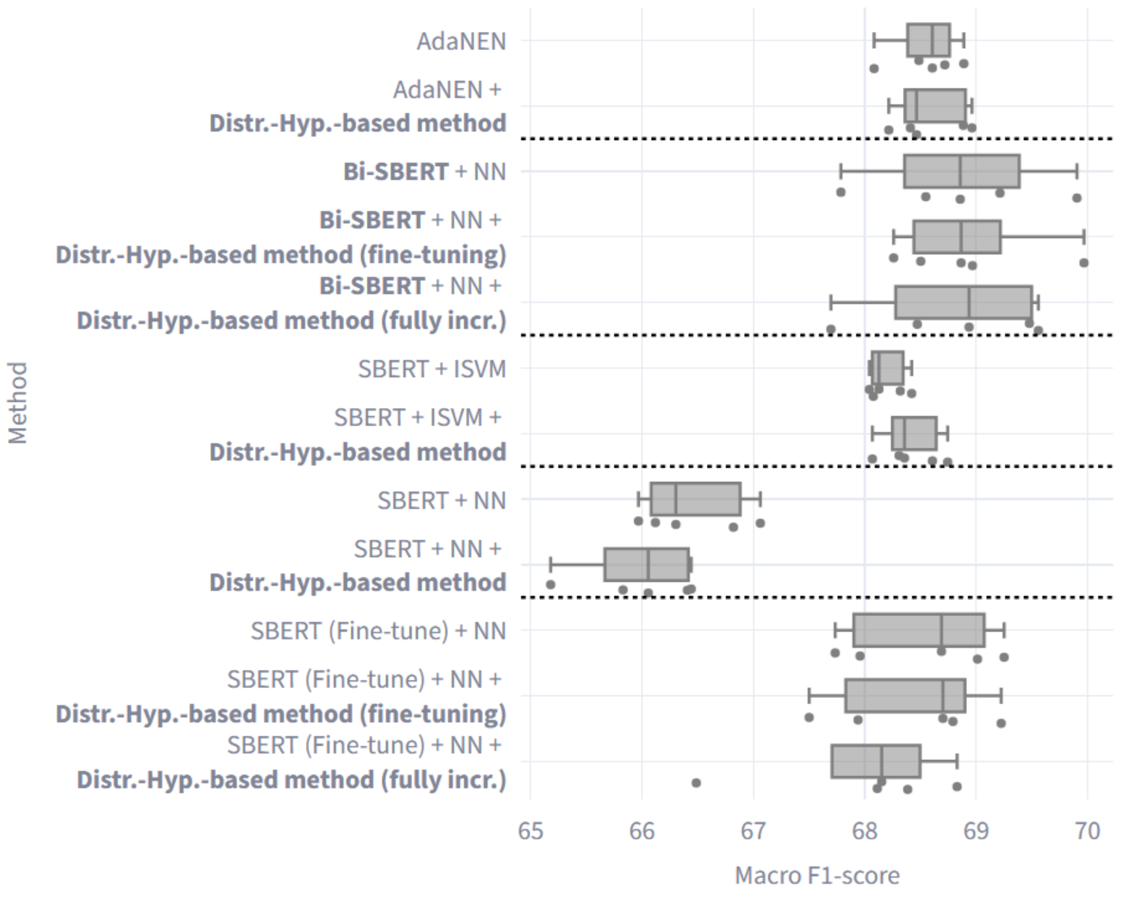

Figure 6.1: Results regarding Macro F1-Scores for the Airbnb dataset, in the scenario without drifts.

NEN and ISVM, compared to their vanilla version. The elapsed times are displayed in Fig. 6.2. Using the fully incremental approach leads to longer elapsed times, and the improvements in Macro F1-Scores come at a temporal cost.

In Appendix B, Table B.1 shows the results in terms of Macro F1 and elapsed time. The cells in green highlight the best values obtained regarding some given methods and the respective version with the Distributed-Hypothesis-based method. It is noteworthy that the highest mean Macro F1 was obtained by the **Bi-SBERT** + NN + **Distributed-Hypothesis-based method** (fine-tuning), which reached 68.91 ± 0.66. On the other hand, this method reached this Macro F1 at the expense of being around, on average, 800 seconds slower than the method without the **Distributed-Hypothesis-based method**, i.e., **Bi-SBERT** + NN. The fastest approach was the SBERT + ISVM, running in 622.85 seconds on average. In this setting, using the Distributed-Hypothesis-based method led to better values in three out of five methods.

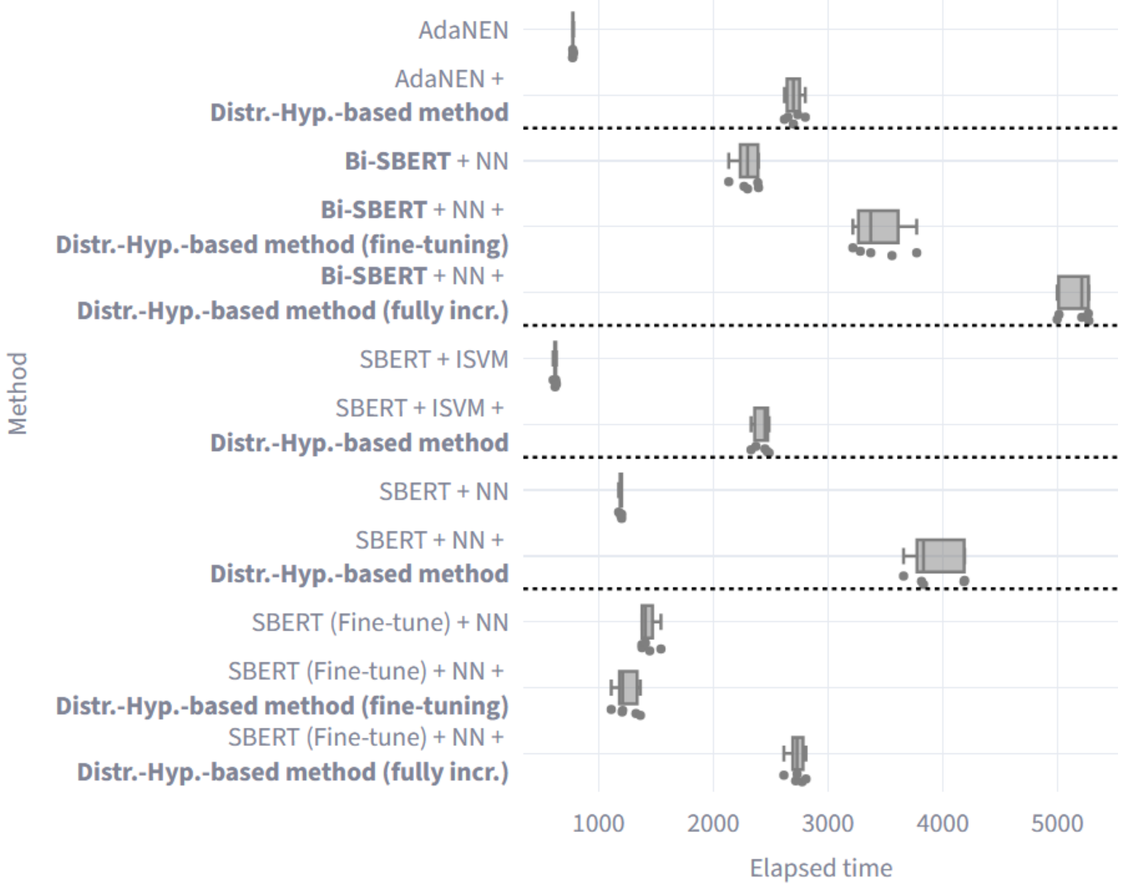

Figure 6.2: Elapsed times for the Airbnb dataset, in the scenario without drifts.

Class swap

Regarding the Airbnb dataset under the class swap drift, Fig. 6.3 shows the box plots with the Macro F1-scores collected during the experiments. AdaNEN reached the best average Macro F1-scores, i.e., 84.9406 ± 0.2226. Although AdaNEN + **Distr.-Hyp.-based method** could not reach the best average, it reached a better minimum and maximum Macro F1-values, compared to AdaNEN. Using the **Distr.-Hyp.-based method** led to better Macro F1-scores for **Bi-SBERT** + NN, ISVM, and SBERT (Fine-tune) + NN.

Table B.2 numerically displays the obtained results regarding Macro F1-scores and elapsed time. In general, the application of the **Distributional-Hypothesis-based method** tends to result in slight improvements in Macro F1-score across most approaches, although at the cost of an increase in wall time. For instance, in the SBERT + ISVM approach, the inclusion of the method led to an increase in Macro F1-score from 84.54 to 84.64, but the elapsed time

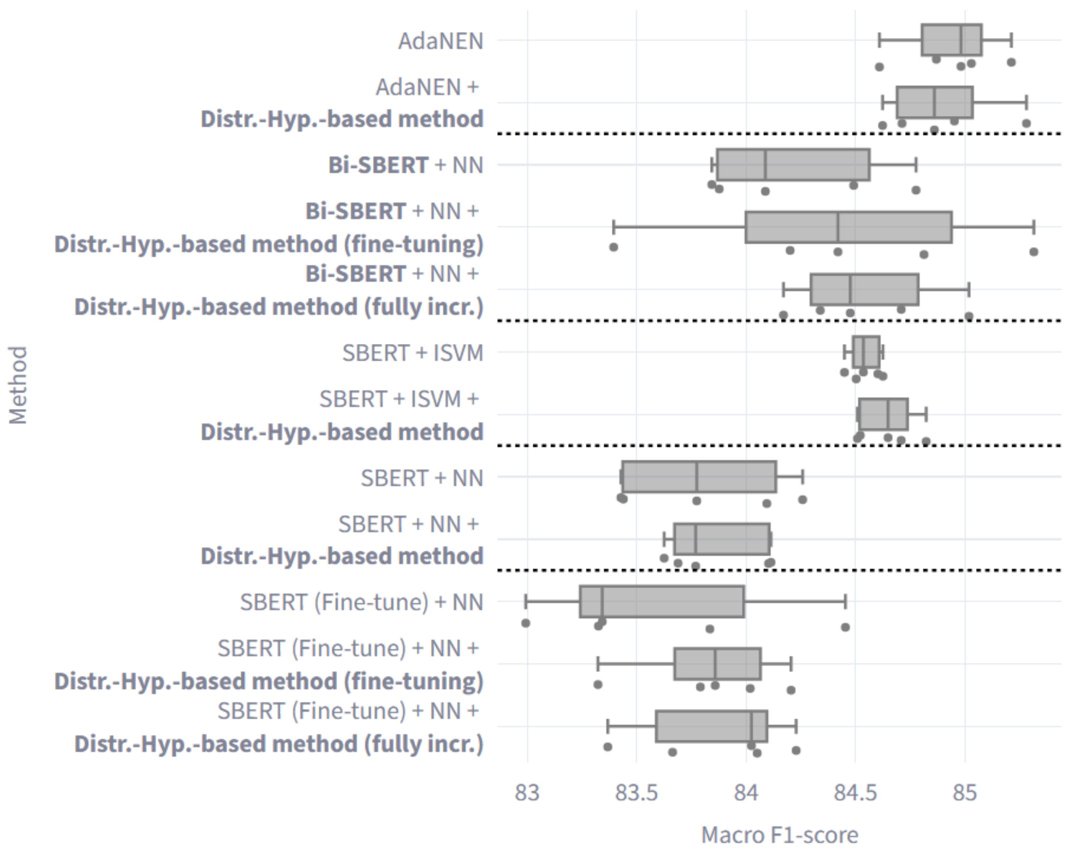

Figure 6.3: Results for the Airbnb dataset, in the scenario of Class Swap drift.

increased nearly fourfold, from 624.22 seconds to 2,408.65 seconds. A similar pattern is observed in the SBERT + NN setting, where the F1-score increased from 83.80 to 83.86, accompanied by a tripling of computational time. An additional observation regards the elapsed time for SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning), which, although with the addition of the **Distr.-Hyp.-based method**, was the fastest in the SBERT (Fine-tune) + NN group. It demands further analysis, but one hypothesis is that the **Distr.-Hyp.-based method** improved the representations in such a way that the number of concept drift detections over time was reduced, considering that fine-tuning is the most costly operation in this setting. The results show that, 5 out of 5 groups had the best Macro F1-score when resorting to **Distr.-Hyp.-based method**.

Class shift

In Appendix B, Fig. 6.5 shows the results in terms of Macro F1-scores for the tested approaches in the scenario with the Airbnb dataset, with the Class Shift

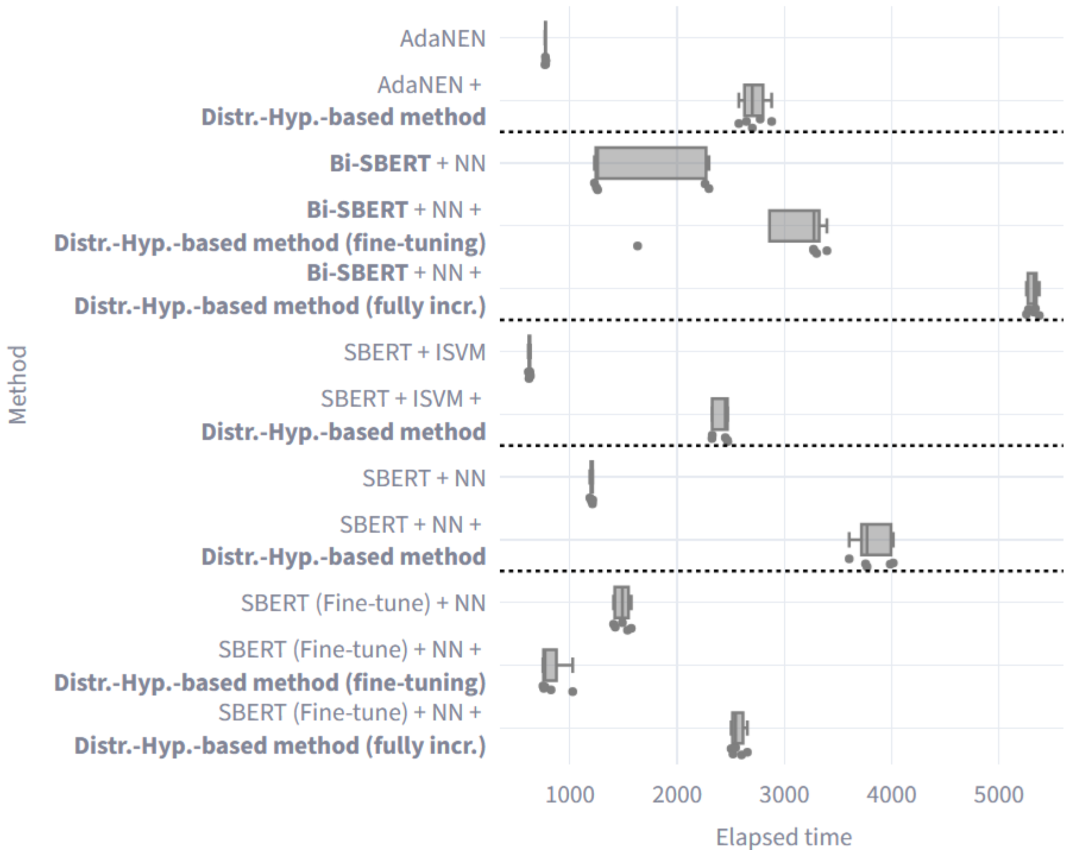

Figure 6.4: Elapsed times collected for the Airbnb dataset, in the scenario of Class Swap drift.

drifts. It can be noted that the Macro F1-scores for all the methods reached, on average, between 74% and 77%. An interesting aspect is that SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning) had one Macro F1-score below 72%. It demands further analysis to draw a conclusion about it, but one hypothesis could be an excessive number of fine-tune processes triggered by the concept drift detector. In addition, the Bi-SBERT approaches, especially the **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning), provided the highest Macro F1-scores observed in this setting.**

In Appendix B, Table B.3 lists the results in this scenario with respect to Macro F1-scores and elapsed time measured for the compared approaches. The values in green are the best in each group, separated by the dashed line. The overall performance trends suggest that the **Distributional-Hypothesis-based method** provided improvements in Macro F1-scores for most approaches, but with some increase in elapsed times. For example, in the SBERT + ISVM setting, the method enhanced performance from 75.60 to 75.65, but elapsed time increased from

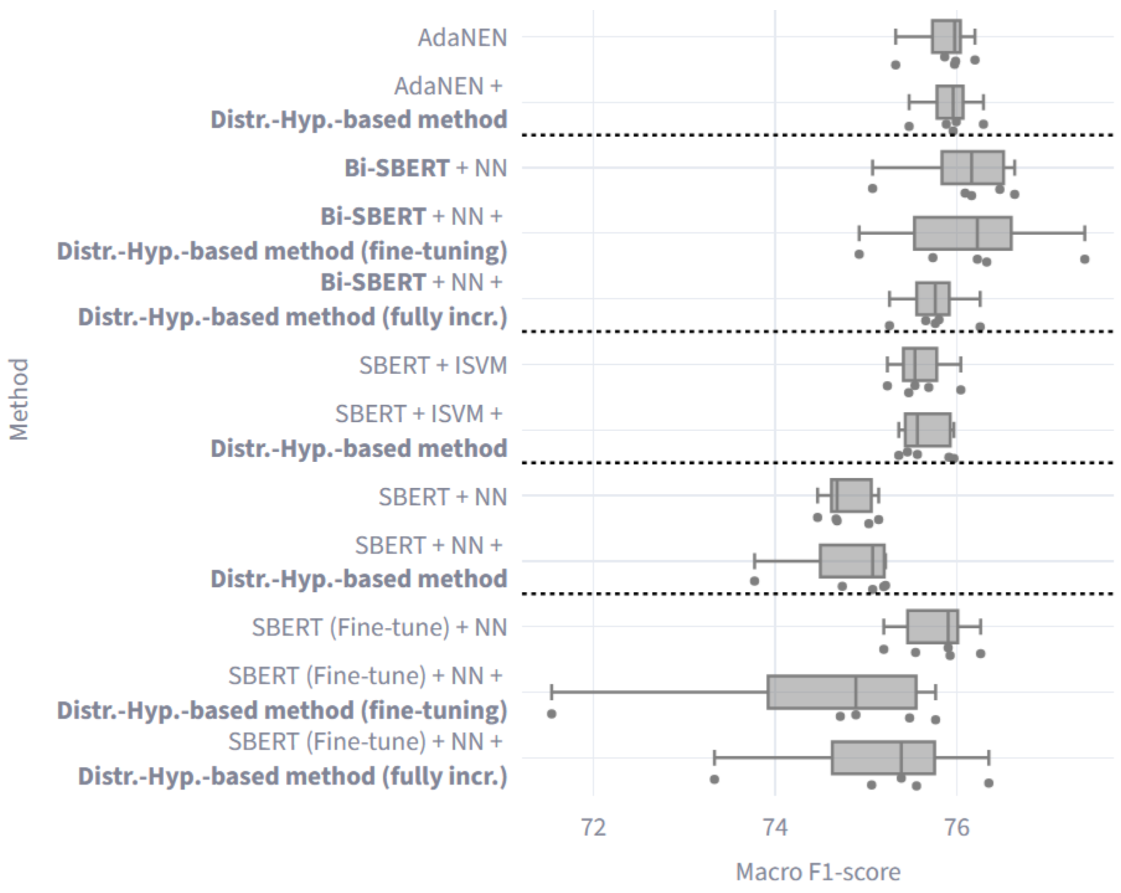

Figure 6.5: Results for the Airbnb dataset, in the scenario of Class Shift.

617.10 seconds to 2,396.87 seconds. A similar trend is observed in SBERT + NN, where the Macro F1-score remains almost unchanged at 74.80, although the elapsed time increased from 1,214.85 seconds to 3,866.35 seconds. The Macro F1-score for SBERT + NN + **Distr.-Hyp.-based method** is marked with a star because it was slightly better than SBERT + NN, which was rounded up. In this setting, the same pattern is again observed, as in the settings without drift and Class Swap: SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)** was the fastest in its group. However, it obtained the lowest Macro F1-score among the competitors in its group, being around 0.5 p.p. lower than the best in the group. As previously stated, this approach had the lowest Macro F1-score in one of its executions (overall in this setting), and it is possible that, with slight changes, it could perform better than observed. Methods with the **Distr.-Hyp.-based method** showed better performance in four out of five methods.

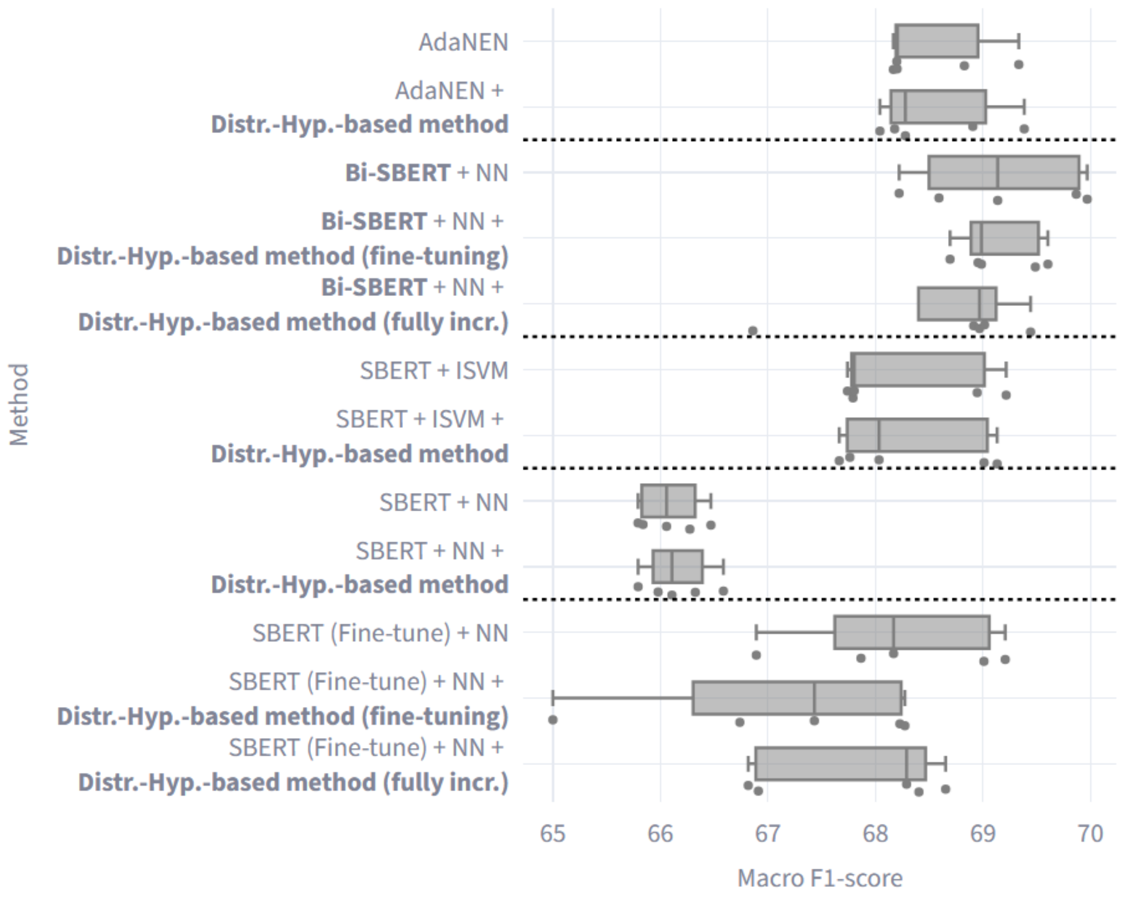

Figure 6.6: Results for the Airbnb dataset, in the scenario of Time-slice Removal.

Time-slice Removal

This subsection describes the results obtained for the Time-slice Removal type of drift. This setting is only available for the Airbnb and Yelp datasets, since they are the only ones that contain timestamp attributes. Recalling the Time-slice Removal drift, the instances from randomly selected years are removed from the subsets, and this justifies the need for timestamp columns.

Fig. 6.6 visually displays the box plots of the Macro F1-scores obtained by the approaches. **Bi-SBERT** + NN reaches the highest values, around 70%. On the other hand, SBERT + NN had the lowest average, while SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)** had the worst single execution performance, around 65%.

Table B.4 shows the values obtained in terms of Macro F1-scores and elapsed time. The results indicate that the **Distributional-Hypothesis-based method** contributed to slight, consistent improvements in classification performance across multiple approaches. In the SBERT + ISVM setting, the Macro F1-score increased from 68.30 to 68.32, while in the SBERT + NN setup, it rose from 66.09 to 66.16. Although these gains may appear small, they suggest that the method effectively refines decision boundaries. These improvements indicate that integrating the **Distributional-Hypothesis-based method** can enhance adaptability in drift scenarios, particularly in models that already demonstrate strong baseline performance. Similarly to previous scenarios, SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)** could run faster than SBERT (Fine-tune) + NN, that is, faster than the method *without* the proposed method. Using the **Distr.-Hyp.-based method** led to improvements in 3 out of 5 groups of approaches.

Adjective swap

This subsection describes the results for the Airbnb dataset in the scenario of adjective swap. Fig. 6.7 shows the box plot obtained by the compared approaches. All the results lied between 60% and 64%. **Bi-SBERT** + NN + **Distr.-Hyp.-based method** reached the highest value obtained in one execution, above 63.5%. SBERT + NN obtained, on average, the worst results.

Table B.5 shows the results for the Adjective Swap, regarding Macro F1-scores and elapsed times. The results demonstrate that the **Distributional-Hypothesis-based method** enhanced classification performance across different approaches. In SBERT + ISVM, the F1-score improved from 68.30 to 68.32, while in SBERT + NN, it increased from 66.09 to 66.16, indicating that the method contributes to better decision boundaries even in the presence of adjective swaps. These results suggest that integrating this method improves model adaptability, making it more resilient to textual variations that could otherwise impact classification accuracy. Although it increases the elapsed time, using the **Distr.-Hyp.-based method** led to improvements in 3 out of 5 groups of approaches.

6.1.2 Amazon Polarity

This section provides the results for the Amazon Polarity dataset. As seen in Section 5.2.2, this is a binary dataset.

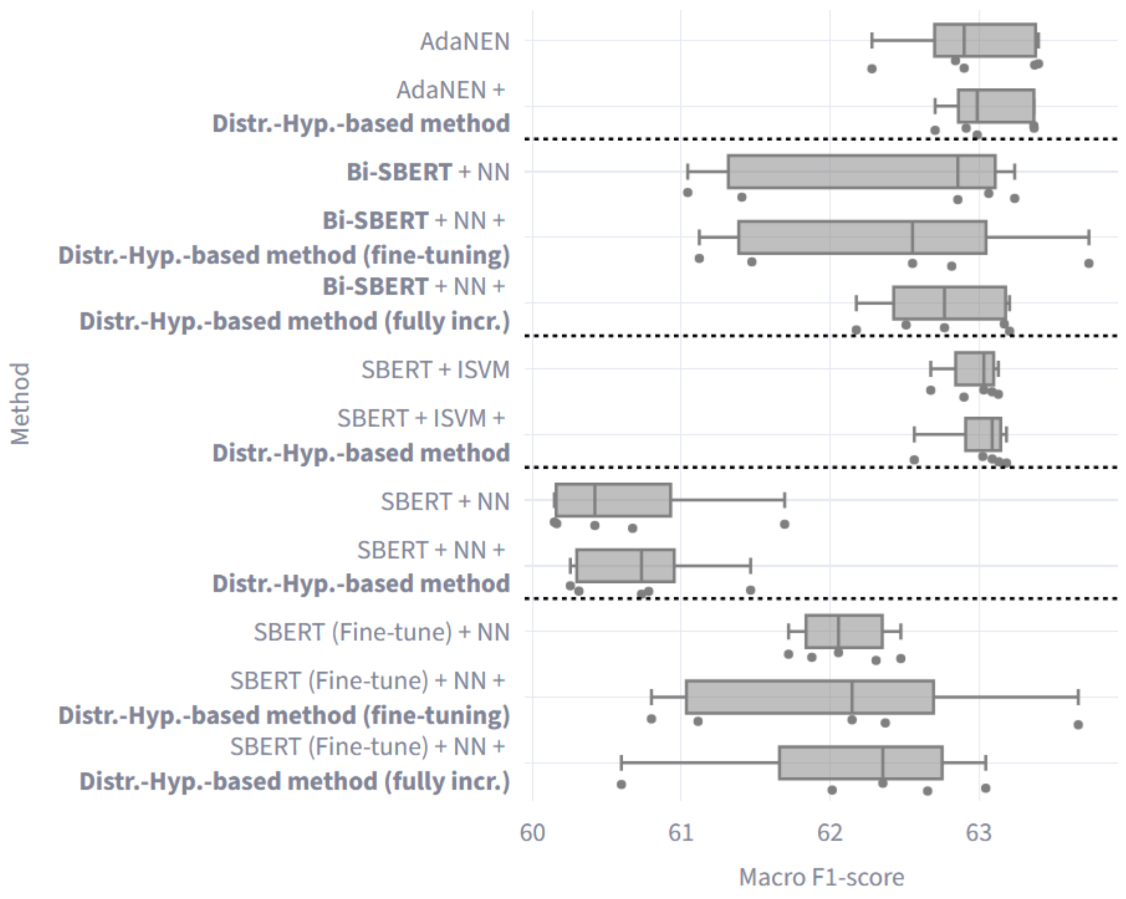

Figure 6.7: Results for the Airbnb dataset, in the scenario of Adjective Swap.

Without drift

Fig. 6.8 shows the Macro F1-Scores for the scenario **without drift** and the Amazon Polarity dataset. Since it is a binary dataset, it is expected for the methods to reach higher Macro F1-Scores compared to the Airbnb dataset. The highest Macro F1-Scores reached around 82.7%, using the SBERT + NN with the **Distributed-Hypothesis-based method**. In addition, the **Distributed-Hypothesis-based method** slightly improves the results also for the other methods, compared to their vanilla version, except for **Bi-SBERT** + NN.

In Appendix B, Table B.6 shows the results obtained in terms of Macro F1-Scores and Elapsed times collected in the experiments. Regarding Macro F1-Scores, clearly the integration of the **Distributional-Hypothesis-based** method contributes positively to the overall classification performance in this scenario without drift. It is important to recall that, although the scenario is "without drift", it may have natural drifts. The combination of SBERT + NN with the

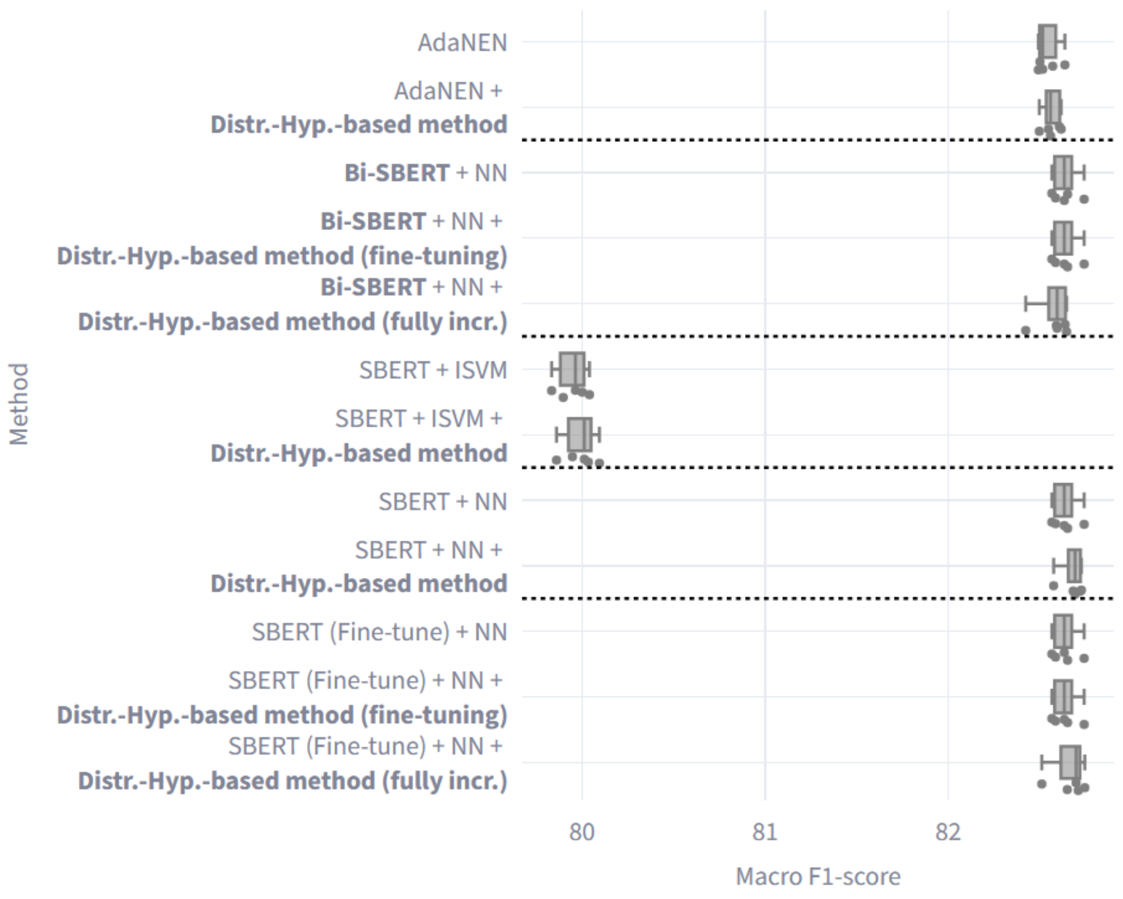

Figure 6.8: Results for the Amazon Polarity dataset, in the scenario without drifts.

method leads to an improvement from 82.64 to 82.68, the highest Macro F1-score among all models. Likewise, SBERT (Fine-tune) + NN benefits from the fully incremental variant, achieving 82.66, slightly above its original score. These results indicate that the method enhances model expressiveness and helps maintain or improve performance, without introducing instability. On the other hand, an increase in the elapsed time when using the **Distributional-Hypothesis-based** method is witnessed. Specifically with **Bi-SBERT** + NN, the exact Macro F1-Scores with and without the **Distributional-Hypothesis-based (fine-tuning)** method were obtained. It implies that no drift was detected during the experiment, leading to the very same results. Regarding the difference in the elapsed times in the aforementioned comparison, it may have occurred due to variations in the use of the computational system.

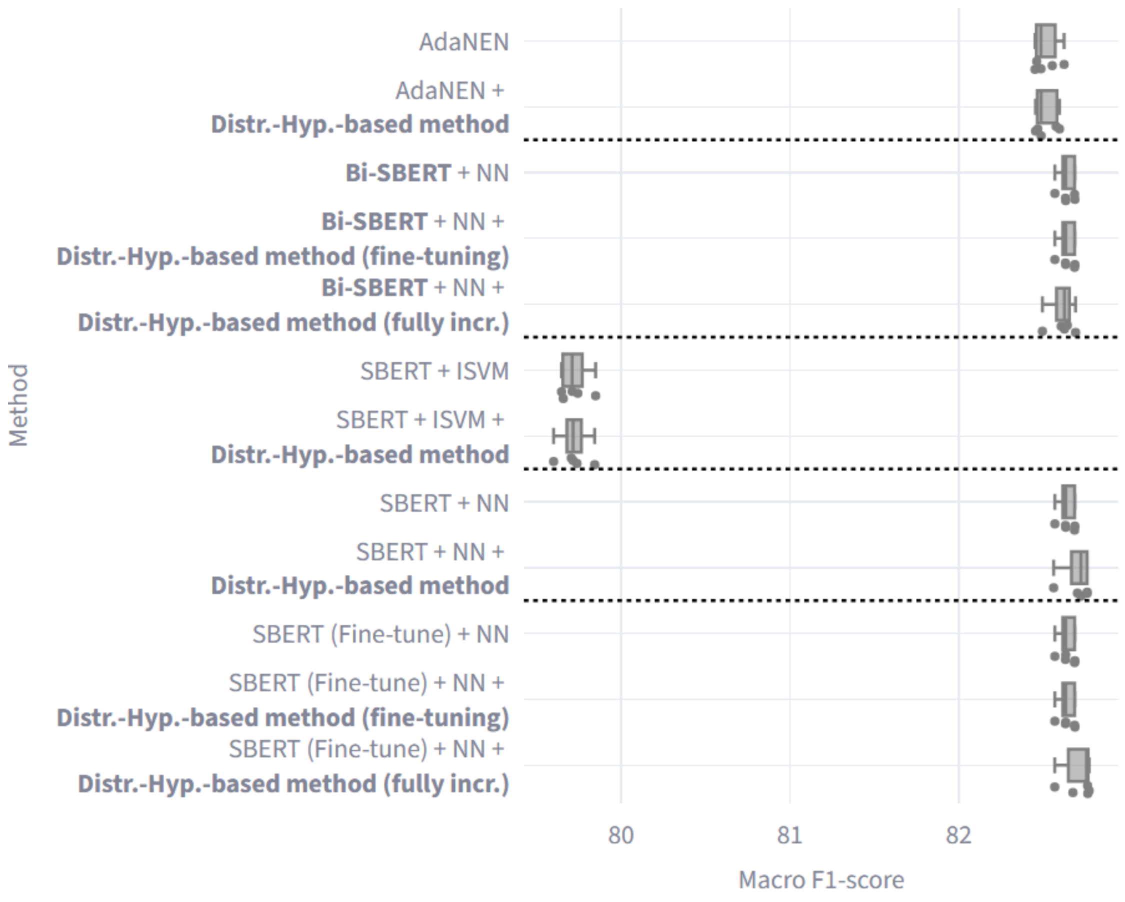

Figure 6.9: Results for the Amazon Polarity dataset, in the scenario of Class Swap.

Class Swap

Fig. 6.9 shows the results regarding Macro F1-Scores for the scenario of Class Swap using the Amazon Polarity dataset. The results obtained were similar to the scenario without drift, and again, the combination SBERT + ISVM had the worst Macro F1-Scores. On the other hand, the other approaches obtained Macro F1-Scores around 82.50%. Table B.7 precisely shows that using the **Distributed-Hypothesis-based method** led to a tie with the methods without it in 3 out of 5 opportunities, and better results in 2 out of 5. The ties suggest a very reduced number (or even none) drift detections.

Fig. 6.10 shows the results in terms of elapsed times. It is noticeable that using the Distributed-Hypothesis-based method leads to longer elapsed times, similarly to the scenario without drift. This pattern changes for the SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**, being faster than

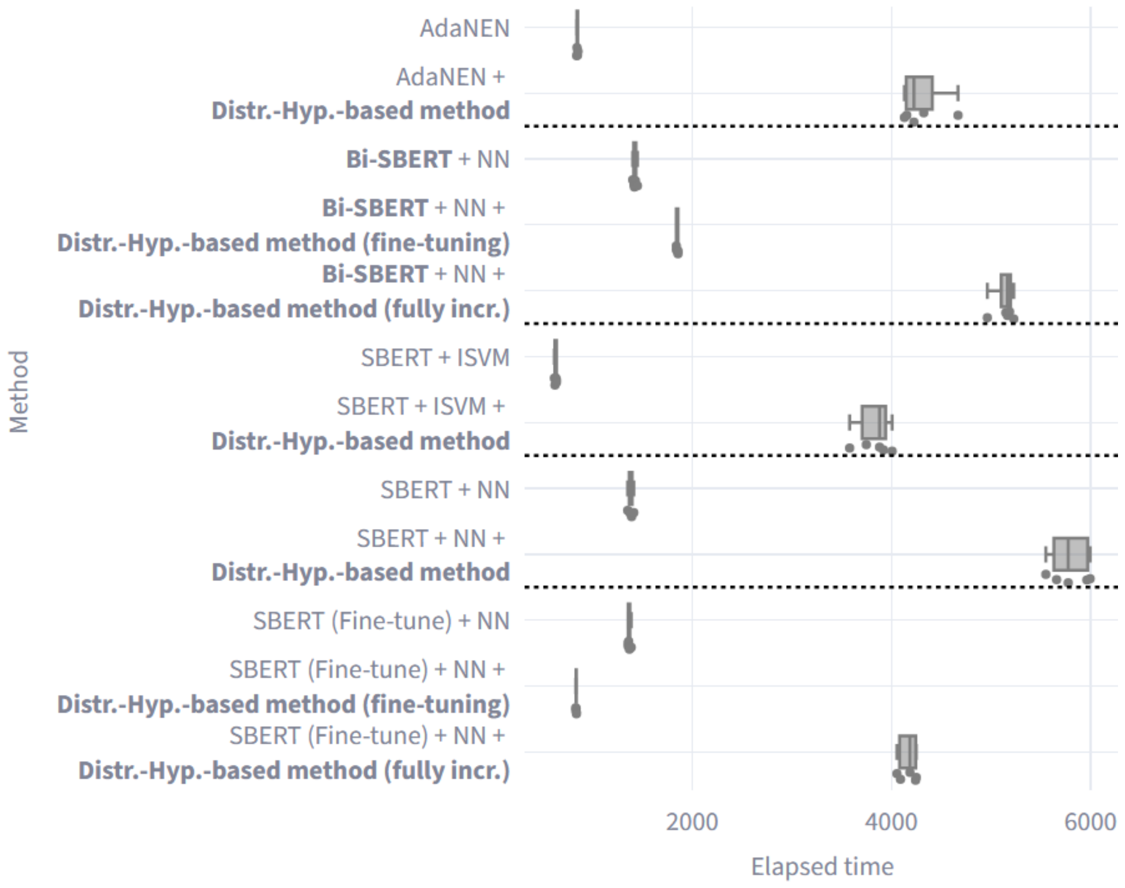

Figure 6.10: Results regarding elapsed times for the Amazon Polarity dataset, in the scenario of class swap.

its version without the **Distr.-Hyp.-based method (fine-tuning)**.

Class Shift

This subsection describes the results for the Class Shift scenario. Fig. 6.11 shows the results regarding Macro F1-Scores. It is possible to note that there were no significant changes compared to the without-drift and Class Swap scenarios. Using ISVM, similarly to its performance in the previously presented scenarios, led to the worst results among the methods. The other methods floated slightly above 82.5% of Macro F1-Scores. In Appendix B, Table B.8 shows the average Macro F1-Scores and elapsed times collected during the experiments. Regarding the Macro F1-Scores, leveraging the **Distributed-Hypothesis-based method** led to an improvement in 4 of 5 groups, while at least keeping the results of the vanilla version, e.g., **Bi-SBERT** + NN. It may have happened because of the lack of detection of drifts.

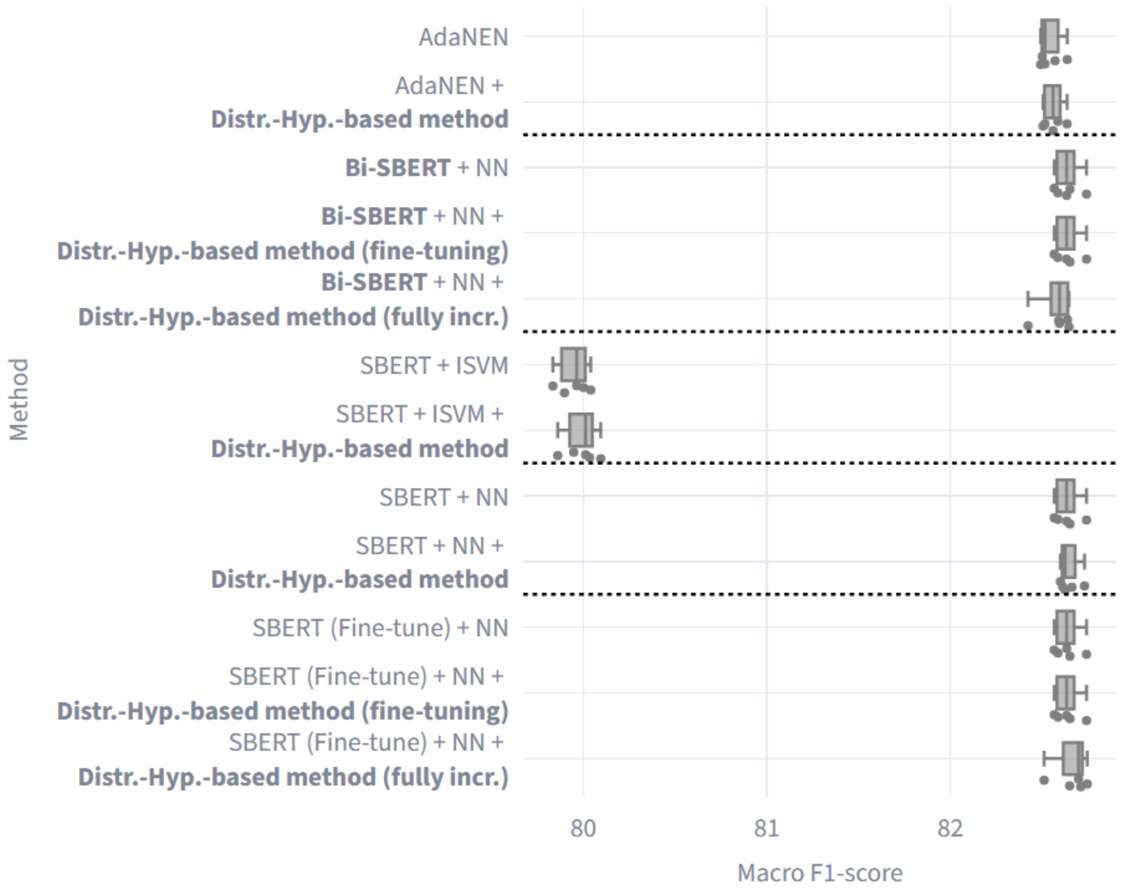

Figure 6.11: Results for the Amazon Polarity dataset, in the scenario of Class Shift.

Regarding the elapsed times, available in Fig. 6.12. Patterns similar to those in the Class Swap scenario were observed. In other words, leveraging the **Distributed-Hypothesis-based method** increased the elapsed times, with the only exception for the SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tuning)**.

Adjective swap

Finally, the scenario of Adjective Swap for the Amazon Polarity dataset is analyzed. Fig. 6.13 displays the results collected in terms of Macro F1-Scores. Different from the previously presented results for this dataset, in this scenario, the collected values vary significantly, making it possible to detect groups visually. In Appendix B, Table B.9 shows the precise results collected. Leveraging the **Distributed-Hypothesis-based method** provided better results in 4 out of 5 groups. SBERT (Fine-tune) + NN was the only method able to

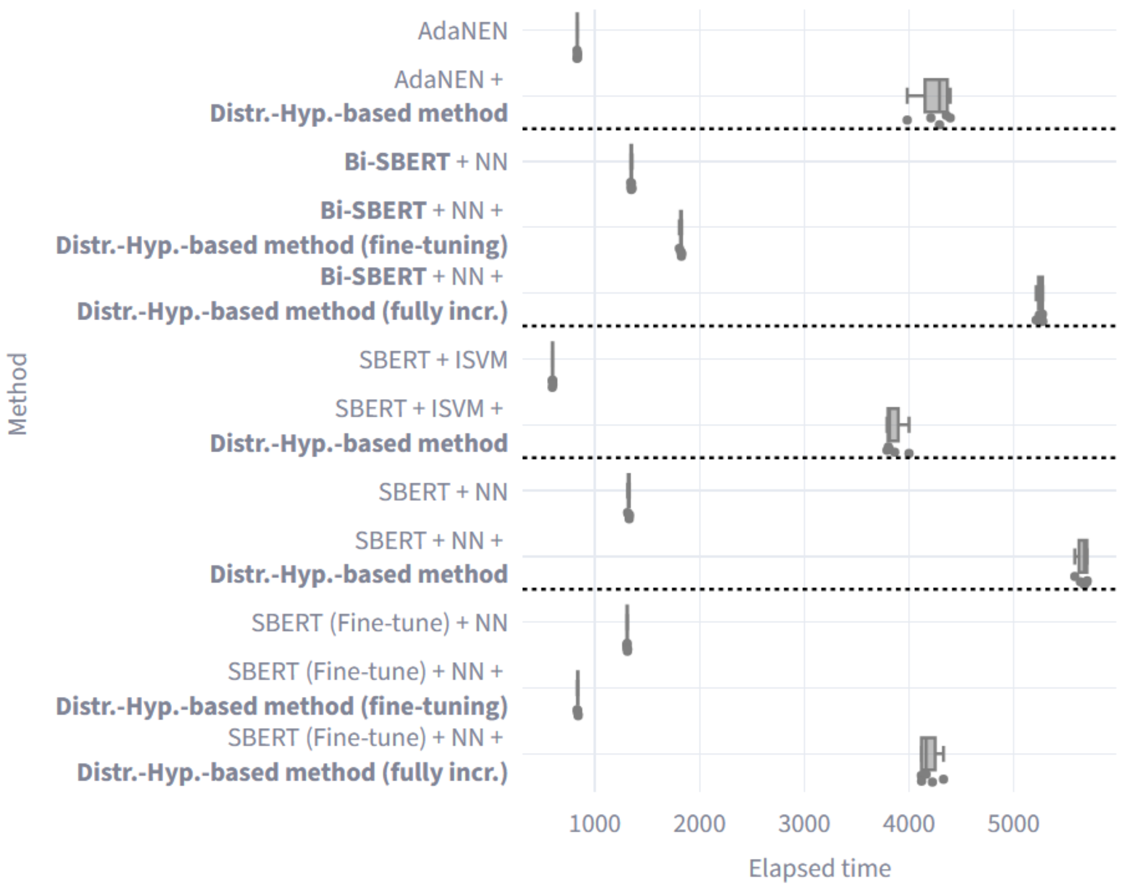

Figure 6.12: Results regarding elapsed times for the Amazon Polarity dataset, in the scenario of Class Shift.

obtain the best results within a group without resorting to the **Distributed-Hypothesis-based method**. A hypothesis for this was that the fine-tuning of SBERT played an important role in updating SBERT's internal representations, while the **Distributed-Hypothesis-based method** may have slowed this process when working together with the fine-tuned SBERT, leading to worse-quality representations.

Regarding the elapsed times, Fig. 6.14 shows the values collected. AdaNEN and SBERT + ISVM were the fastest methods, while Bi-SBERT + NN + **Distributed-Hypothesis-based method (fully incr.)** was the slowest. Similarly to the observations in the previous scenarios for this dataset, **Distributed-Hypothesis-based method** led to longer elapsed times, with the same exception of the SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tune)**.

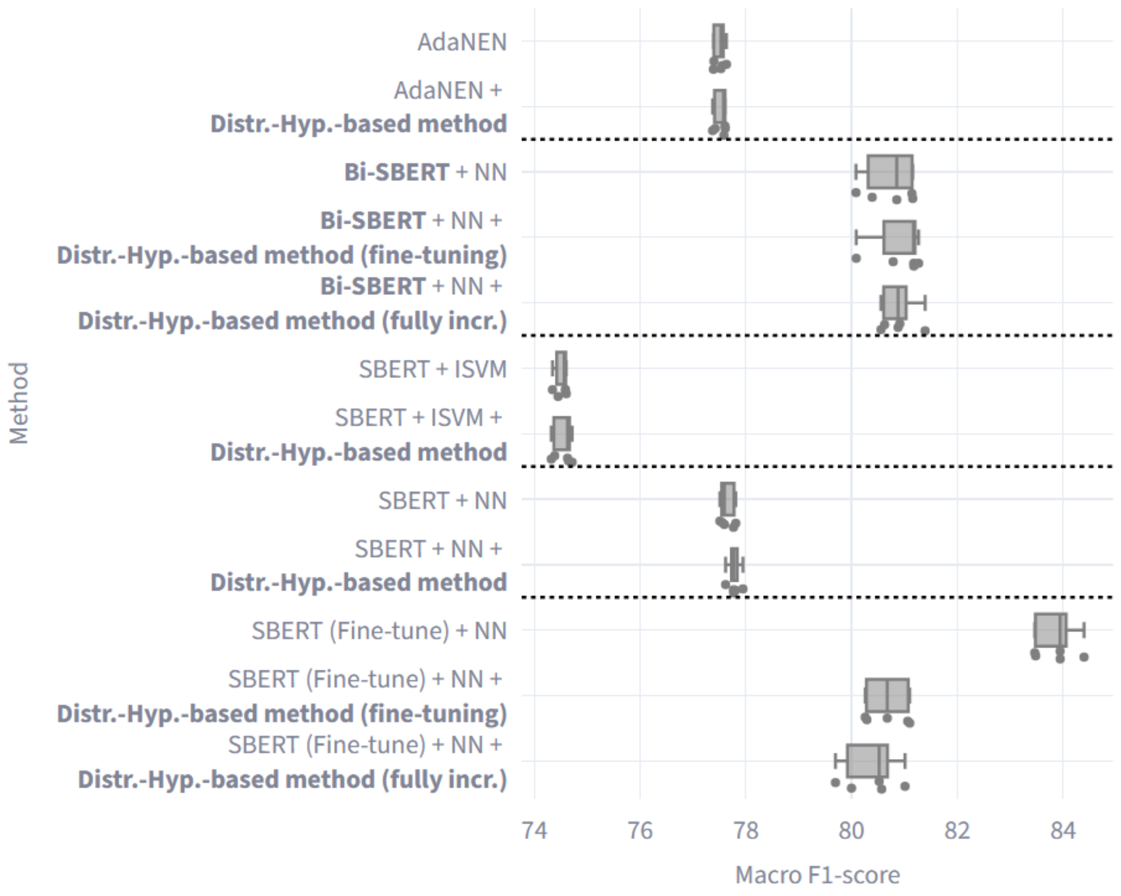

Figure 6.13: Results for the Amazon Polarity dataset, in the scenario of Adjective Swap.

6.1.3 Amazon Reviews

This subsection describes the results obtained using the Amazon Reviews dataset, which has five possible classes, ranging from one to five stars.

Without drift

For the scenario without drift, Fig. 6.15 shows the results regarding Macro F1-Scores. It is possible to verify that using SBERT + ISVM led to a lower Macro F1-Score, similarly to the scenarios regarding the Amazon Polarity dataset. Precisely, SBERT + ISVM (with and without the **Distr.-Hyp.-based method**) reached around 36.7% of Macro F1-Score, on average. Resorting to Table B.10, **Bi-SBERT** reached the best values among the methods, i.e., 43.68% on average. On the other hand, leveraging the **Distr.-Hyp.-based method** could not improve the results in four out of five comparisons, being the SBERT (Fine-tune) + NN +

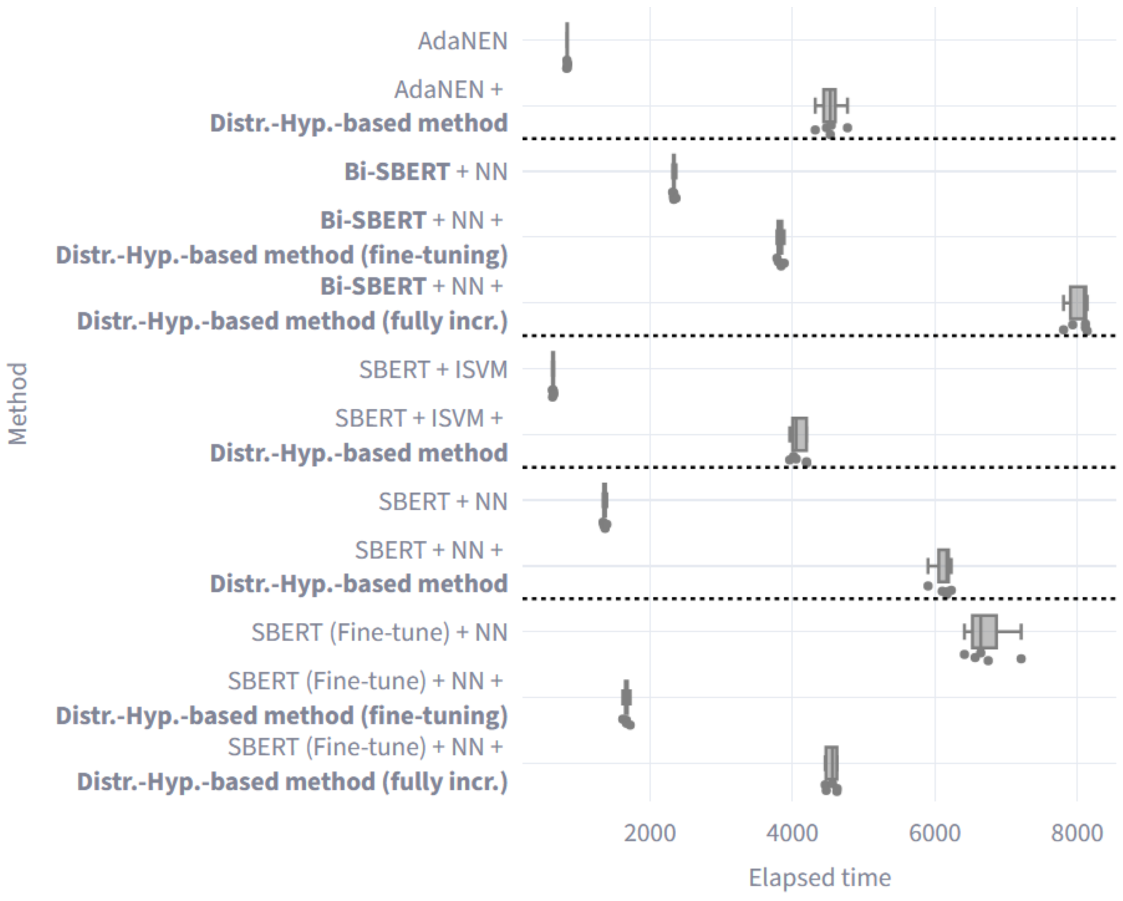

Figure 6.14: Results regarding elapsed times for the Amazon Polarity dataset, in the scenario of Adjective Swap.

Distr.-Hyp.-based method (fully incr.) equivalent to SBERT (Fine-tune) + NN, but reaching a bigger maximum Macro F1-Score.

Fig. 6.16 displays the elapsed times for each method. Similar to other scenarios, using **Distr.-Hyp.-based method** generally leads to longer elapsed times. Bi-SBERT + NN with **Distr.-Hyp.-based method (fully incr.)** was the slowest, while SBERT + ISVM was the fastest.

Class swap

Fig. 6.17 shows the results in terms of Macro F1-score. **Bi-SBERT** + NN reached the highest median of Macro F1-Scores. On the other hand, leveraging ISVM, similarly to the previously presented scenario, led to the lowest values of Macro F1-Scores. The other approaches, in general, lied between 41% and 44% of Macro F1. Differently, the results of SBERT (Fine-tune) + NN + **Distr.-**

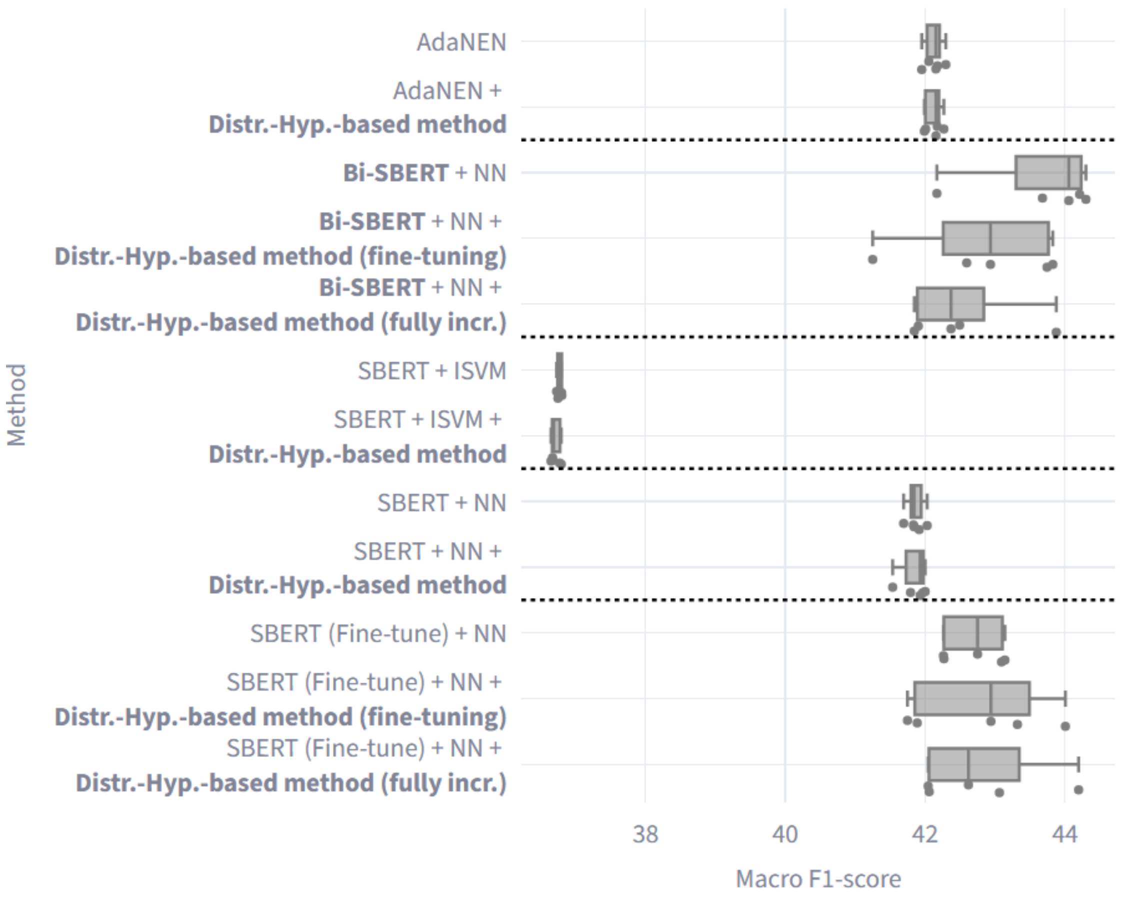

Figure 6.15: Results for the Amazon Reviews dataset, in the scenario without drift.

Hyp.-based method (fully incr.) may attract attention. The results obtained for this approach range from 37% to 44% of Macro F1, having a standard deviation essentially higher than the other approaches, as shown in Table B.11 available in Appendix B. Further analysis is required to understand this behavior. In this scenario, two out of five approaches obtained better results when leveraging the **Distr.-Hyp.-based method**.

Fig. 6.18 shows essentially the same patterns as the ones previously presented. ISVM was the fastest to run, while Bi-SBERT was the slowest, precisely when using the **Distr.-Hyp.-based method (fully incr.)**. Overall, using **Distr.-Hyp.-based method** leads to higher elapsed times.

Class shift

Regarding the Class Shift scenario, Fig. 6.19 shows the results regarding the Macro F1-Scores measured. Most methods reached a Macro F1 between 42

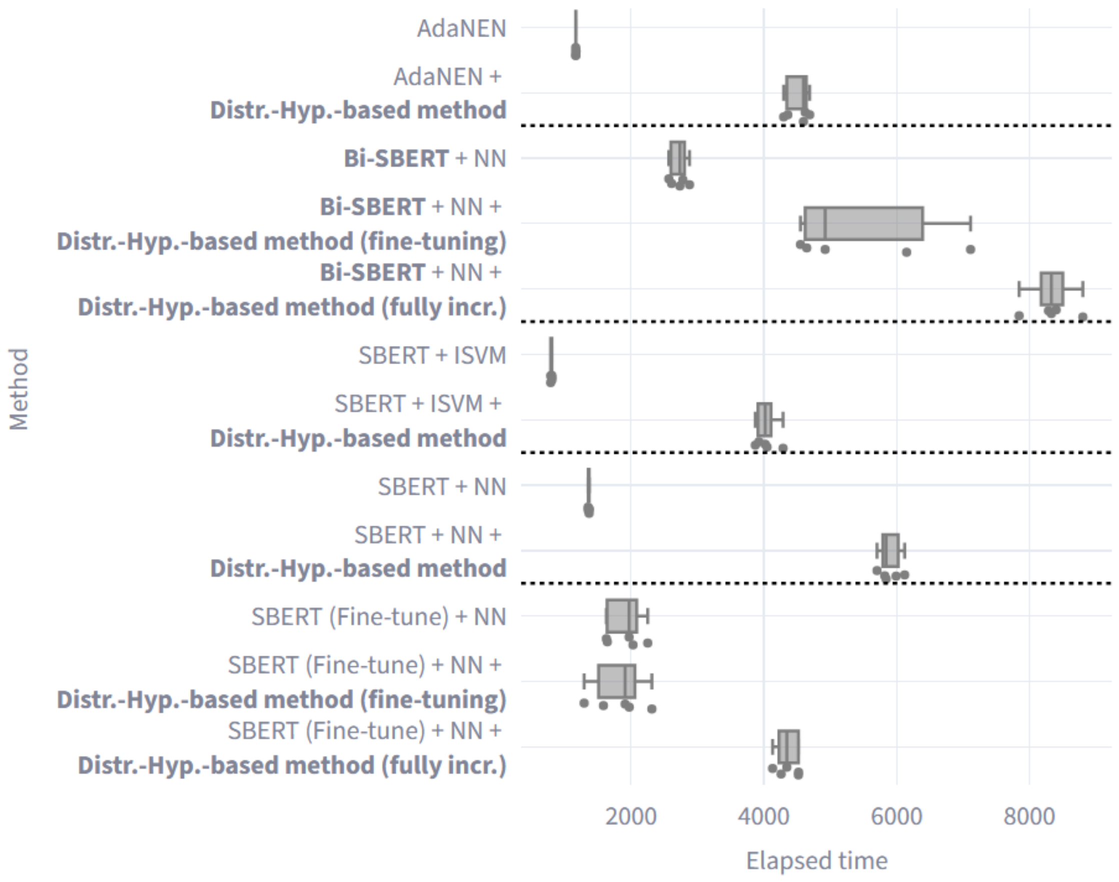

Figure 6.16: Results regarding elapsed times for the Amazon Reviews dataset, in the scenario without drift.

and 44% on average, while ISVM obtained scores below 37%. In this scenario, SBERT (Fine-tune) + NN reached the best Macro F1-Score, slightly above 44% in a single run. Resorting to Table B.12, in Appendix B, it is verified that using the proposed **Distr.-Hyp.-based method** led to better results in only one of the five groups.

Fig. 6.20 shows the elapsed times. The obtained results are similar to the ones shown before. AdaNEN and ISVM were the fastest ones, while Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)** was the slowest. Generally, using the **Distr.-Hyp.-based method** leads to longer elapsed times, and the only exception for this statement was the SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tune)**, which ran faster than its vanilla version.

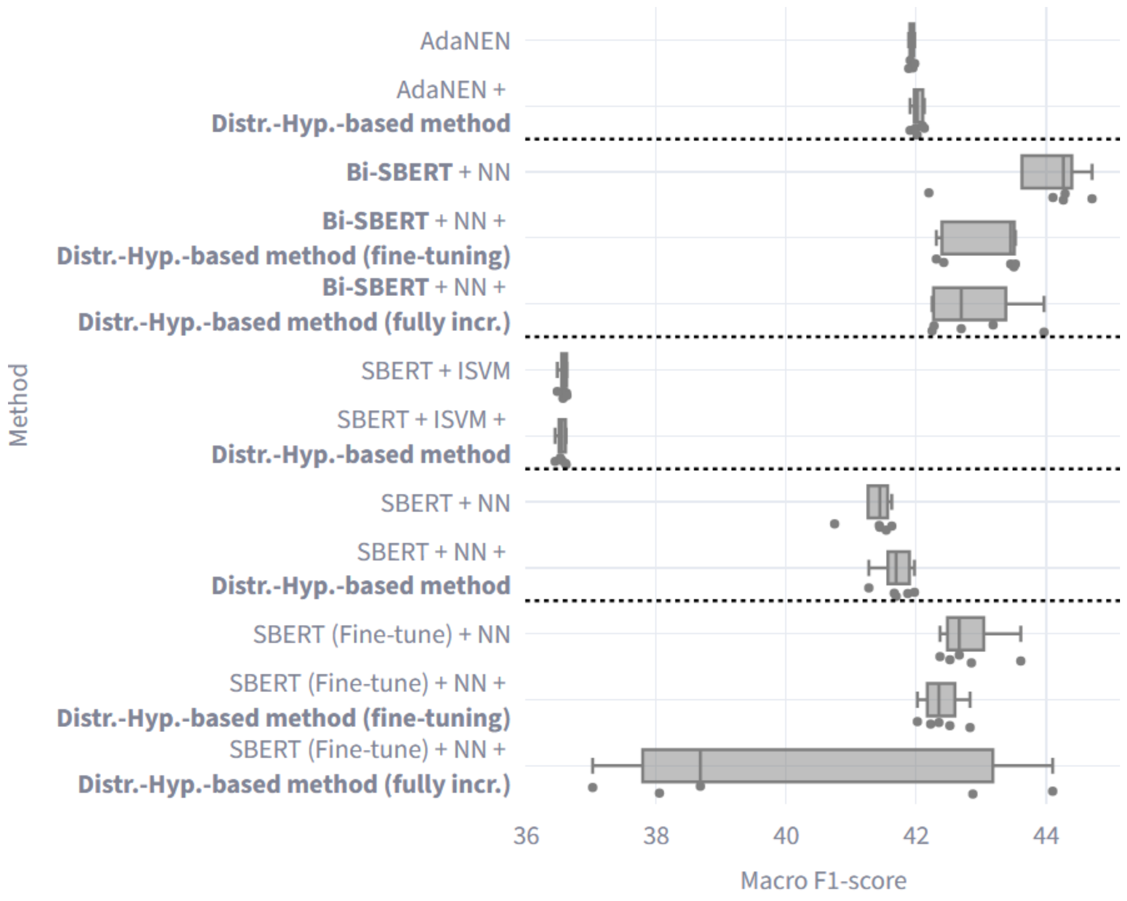

Figure 6.17: Results for the Amazon Reviews dataset, in the scenario of Class Swap.

Adjective Swap

Fig. 6.21 shows the results for the Adjective Swap scenario. Interestingly, AdaNEN obtained the best values in a steady manner. Other methods reached Macro F1-Scores below the ones AdaNEN obtained, and again, ISVM obtained the worst values. Table B.13 precisely shows the values obtained. Using the proposed **Distr.-Hyp.-based method** helped provide better values than the vanilla version in three out of five groups.

Fig. 6.22 shows the values for the elapsed times. Again, the elapsed times were very similar: **Bi-SBERT** being the slowest, and ISVM and AdaNEN the fastest. Interestingly, AdaNEN obtained the best mean values, while being close to the fastest method, e.g., ISVM. Precisely, AdaNEN ran in 782.31 ± 3.44 seconds, on average, and ISVM, was executed in 881.85 ± 14.14 seconds on average. As noticed, ISVM, although fast, cannot reach the same levels of Macro F1-Score as

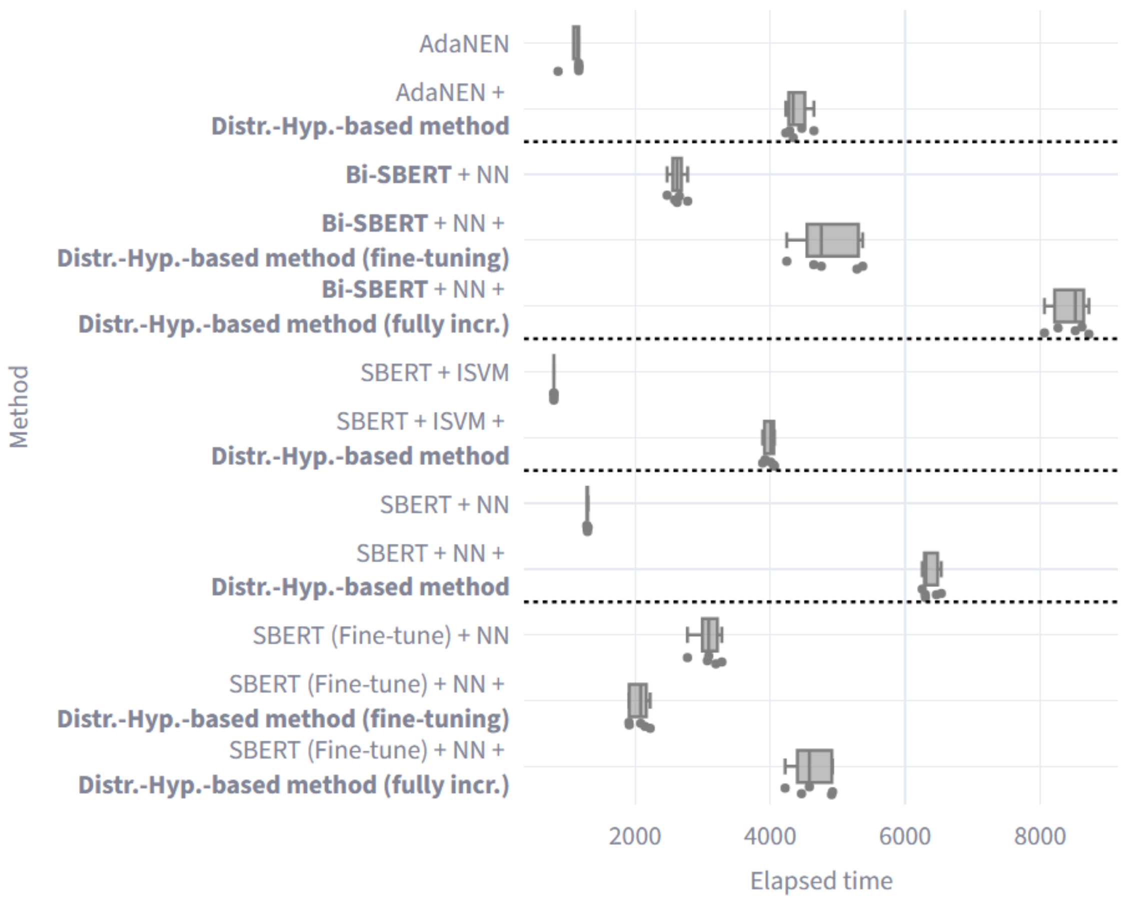

Figure 6.18: Results regarding elapsed times for the Amazon Reviews dataset, in the scenario of Class Swap.

AdaNEN.

6.1.4 Yelp

This subsection shows the results regarding the Yelp dataset. This dataset consists of five classes, and is one of the two datasets presented in this thesis that contains a timestamp column. This allowed the experiments with the Time-slice Removal, a proposed drift scenario.

Without drift

Fig. 6.23 shows the Macro F1-Scores for the Yelp dataset, in the scenario without drifts. In this setting, SBERT (Fine-tune) + NN reached the highest Macro F1-Scores, e.g., 50.04 ± 0.74. AdaNEN, with and without the **Distr.-Hyp.-based method**, reached the same results, 49.71 ± 0.05. **Bi-SBERT** + NN

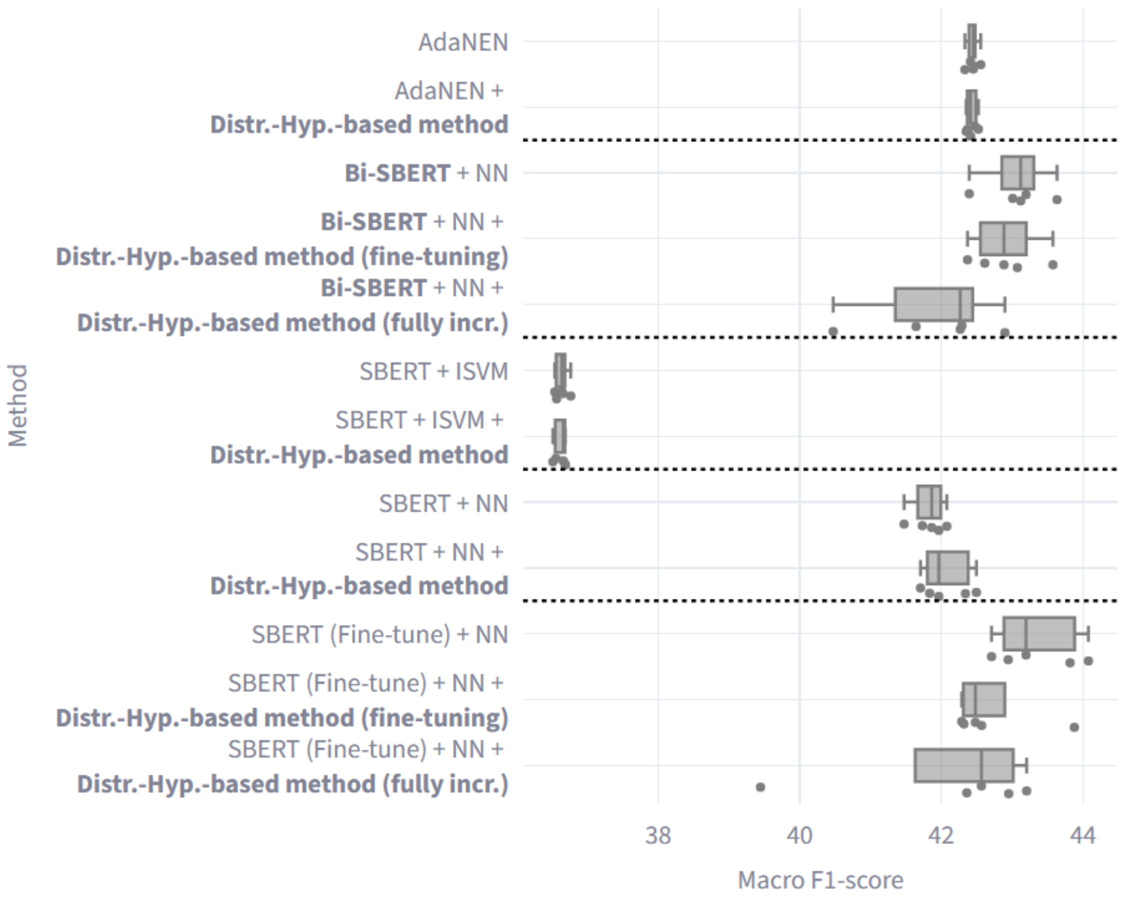

Figure 6.19: Results for the Amazon Reviews dataset, in the scenario of Class Shift.

reached 49.83 ± 0.23. Leveraging **Distr.-Hyp.-based method** led to better results compared to the vanilla version in only one out of five groups, having a draw in one case, i.e., AdaNEN. Complete results can be verified in terms of Macro F1-scores and elapsed times in Table B.14 in Appendix B.

Fig. 6.24 displays the elapsed times collected in the experiments. The results show that AdaNEN and SBERT + ISVM led to fast runs. The slowest was the **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)**. The elapsed times in other scenarios showed similar patterns to those observed in this scenario. Since the **Distr.-Hyp.-based method** does not use matrix operations and must iterate over the input, it is expected that the **Distr.-Hyp.-based method**, especially in its fully incremental version, could be slow.

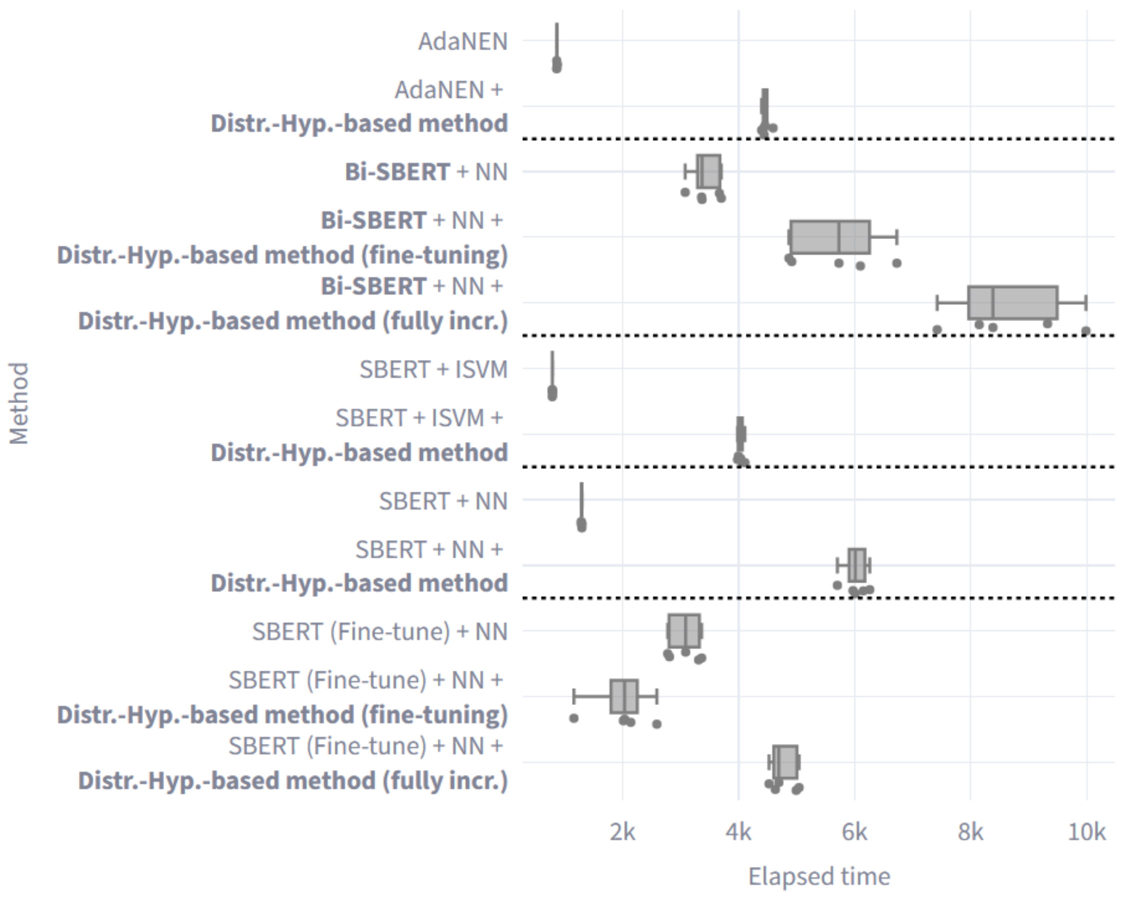

Figure 6.20: Results regarding elapsed times for the Amazon Reviews dataset, in the scenario of Class Shift.

Class Swap

This subsection shows the results for Class Swap drift in the Yelp dataset. The measured Macro F1-Scores are displayed in Fig. 6.25. **Bi-BERT** + NN obtained the best results for Macro F1-Scores. SBERT + ISVM obtained the lowest Macro F1-Scores. Using **Distr.-Hyp.-based method** led to better Macro F1-Scores compared to the vanilla version in three out of five groups, as can be checked in Table B.15. SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)** has a high range, suggesting that some runs could have been impacted by an excessive number of fine-tuning processes.

Fig. 6.26 shows the elapsed times, which followed the patterns previously observed: **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)** was the slowest, while SBERT + ISVM was the fastest. AdaNEN and SBERT + NN also performed fast, below 2,000 seconds for the complete experiment. Again,

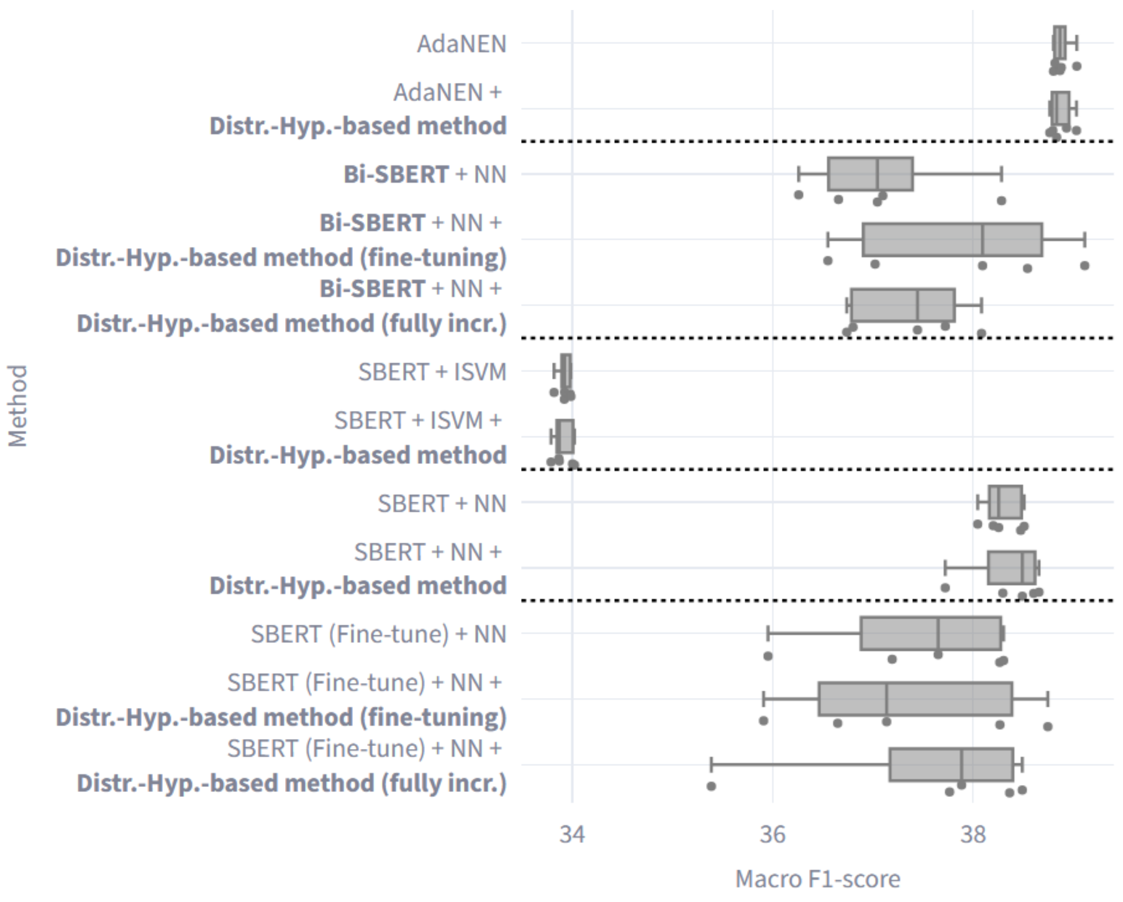

Figure 6.21: Results for the Amazon Reviews dataset, in the scenario of Adjective Swap.

SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)** executed faster than SBERT (Fine-tune) + NN.

Class shift

This subsection regards the results in the Yelp dataset under Class Shift. In this scenario, AdaNEN + **Distr.-Hyp.-based method** obtained the best results in terms of Macro F1-Score, e.g., 62.17 ± 0.09, according to Table B.16 in Appendix B. SBERT + ISVM obtained the worst results, although leveraging **Distr.-Hyp.-based method** in this case led to better results. In addition, using the **Distr.-Hyp.-based method** led to better results in four of five groups, with the SBERT (Fine-tune) + NN being the only exception.

Fig. 6.28 displays the elapsed times. SBERT + ISVM was the fastest, followed by AdaNEN and SBERT + NN. In this setting, a considerable range of values among the executions was noticed. For example, **Bi-SBERT** + NN + **Distr.-**

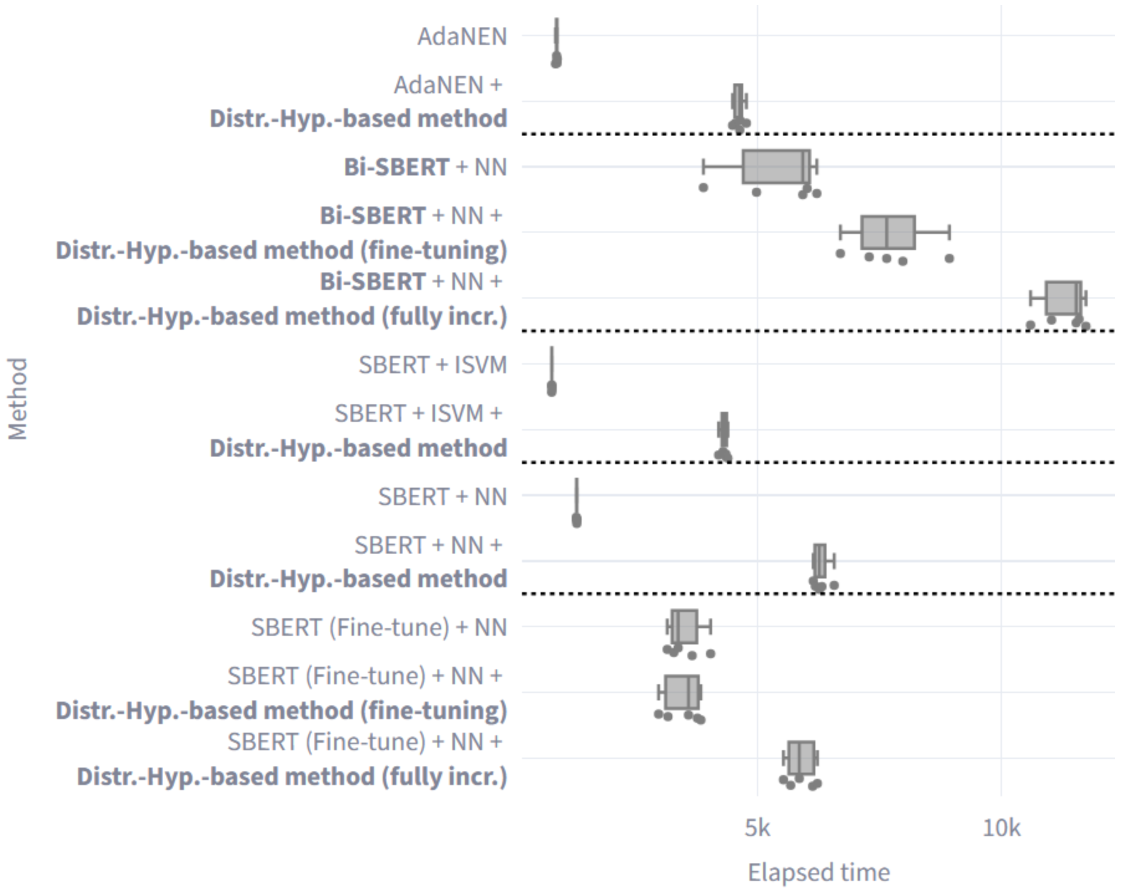

Figure 6.22: Results regarding elapsed times for the Amazon Reviews dataset, in the scenario of Adjective Swap.

Hyp.-based method (fully incr.) varied from below 8,000 and above 10,000 seconds. As previously observed, using **Distr.-Hyp.-based method** leads to slower executions, although, especially in this setting, it generally provided improvements in the Macro F1-Score.

Time-slice Removal

Fig. 6.29 shows the results for the scenario of Time-slice Removal. Resorting to Table B.17, it is verified that using the **Distr.-Hyp.-based method** did not provide any improvement in Macro F1-Scores, compared to the vanilla version. For AdaNEN, leveraging the **Distr.-Hyp.-based method** led to a draw. SBERT + ISVM obtained the worst results among the compared methods, similarly to other scenarios.

Fig. 6.30 shows the elapsed times for this setting. Similarly to the setting

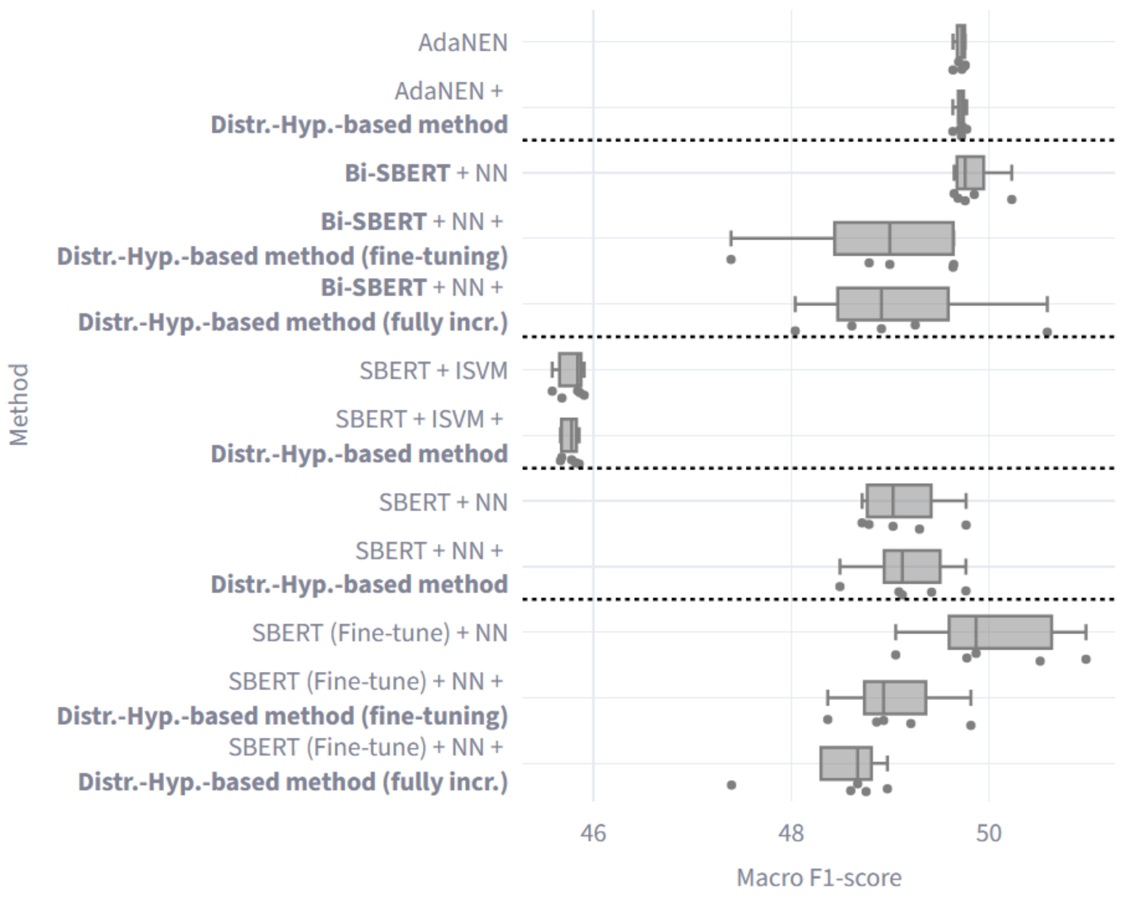

Figure 6.23: Results for the Yelp dataset, in the scenario without drift.

of Class Shift, a considerable range of elapsed times collected in the **Bi-SBERT**-based family, especially, was noticed.

Adjective swap

Fig. 6.31 shows the results regarding Macro F1-Scores for the Adjective Swap scenario in the Yelp dataset. The best isolated runs were obtained by **Bi-SBERT** + NN, specifically with the **Distr.-Hyp.-based method** in its fully incremental version. The worst values, similarly to the previous scenarios, were obtained by SBERT + ISVM. Resorting to Table B.18, it is noticed that leveraging the proposed **Distr.-Hyp.-based method** led to better results in two of the five groups. Besides, the **Bi-SBERT** group obtained the best average results, although within the group, the **Bi-SBERT** + NN, without the **Distr.-Hyp.-based method**, obtained the best results, i.e., 45.83 ± 0.81.

Fig. 6.32 shows the elapsed run times for this setting. SBERT + ISVM were the

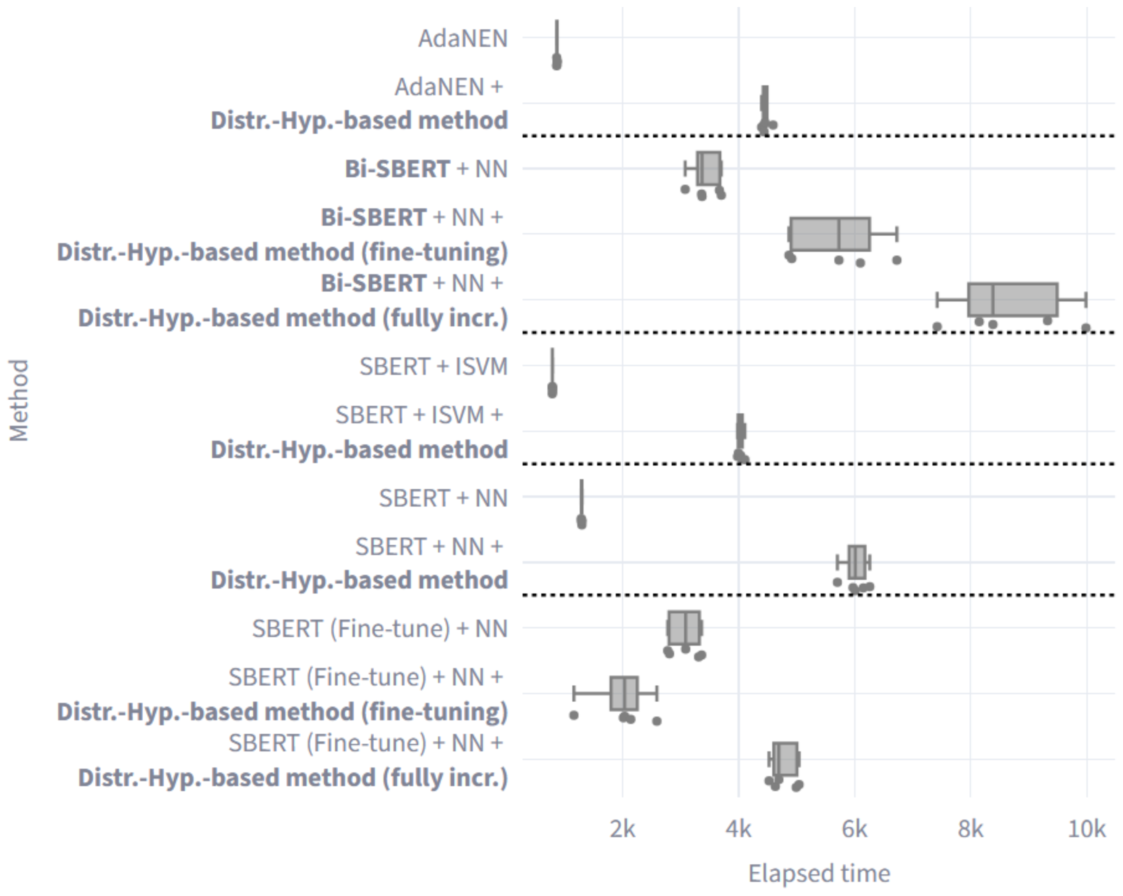

Figure 6.24: Results regarding elapsed times for the Yelp dataset, in the scenario without drift.

fastest to execute, running in 1057.12 ± 2.80 seconds. AdaNEN, which obtained competitive results in Macro F1-Scores, ran in 1115.48 ± 4.59 seconds, similar to SBERT + ISVM. On the other hand, the **Distr.-Hyp.-based method** generally leads to longer elapsed times, and the exception is the SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method**, which ran faster than SBERT (Fine-tune) + NN. The precise values mentioned can be verified in Table B.18 in Appendix B.

6.1.5 Yelp Full

This subsection provides the results obtained for the Yelp Full dataset, a 5-class dataset. The scenarios evaluated were "without drift", Class Swap, Class Shift, and Adjective Swap.

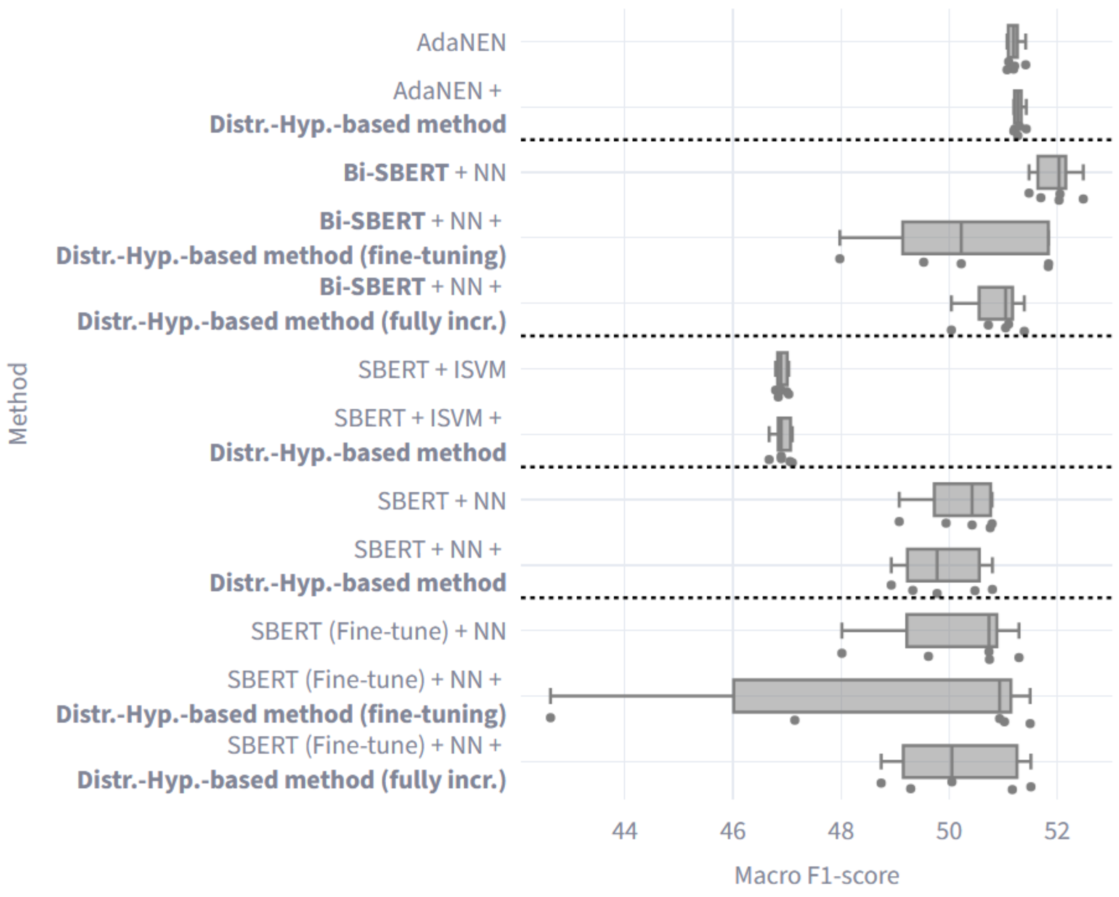

Figure 6.25: Results for the Yelp dataset, in the scenario of Class Swap.

Without drift

Fig. 6.33 shows the results in terms of Macro F1-Scores for the Yelp Full dataset in the scenario without drift. With the exception of SBERT + ISVM group, all methods obtained Macro F1-Scores between 50% and slightly above 52%. The best isolated run belonged to **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)**. In Appendix B, Table B.19 shows the precise results, and it is noticeable that leveraging the **Distr.-Hyp.-based method** led to better results in three out of five groups. **Bi-SBERT** + NN and SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** reached the best overall Macro F1-Scores, i.e., 51.61 ± 0.29 and 51.73, respectively.

Fig. 6.34 shows the elapsed times for this setting. The observations are similar to those performed previously. Interestingly, the SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**, differently from previous observations, obtained a longer elapsed time compared to SBERT (Fine-tune) +

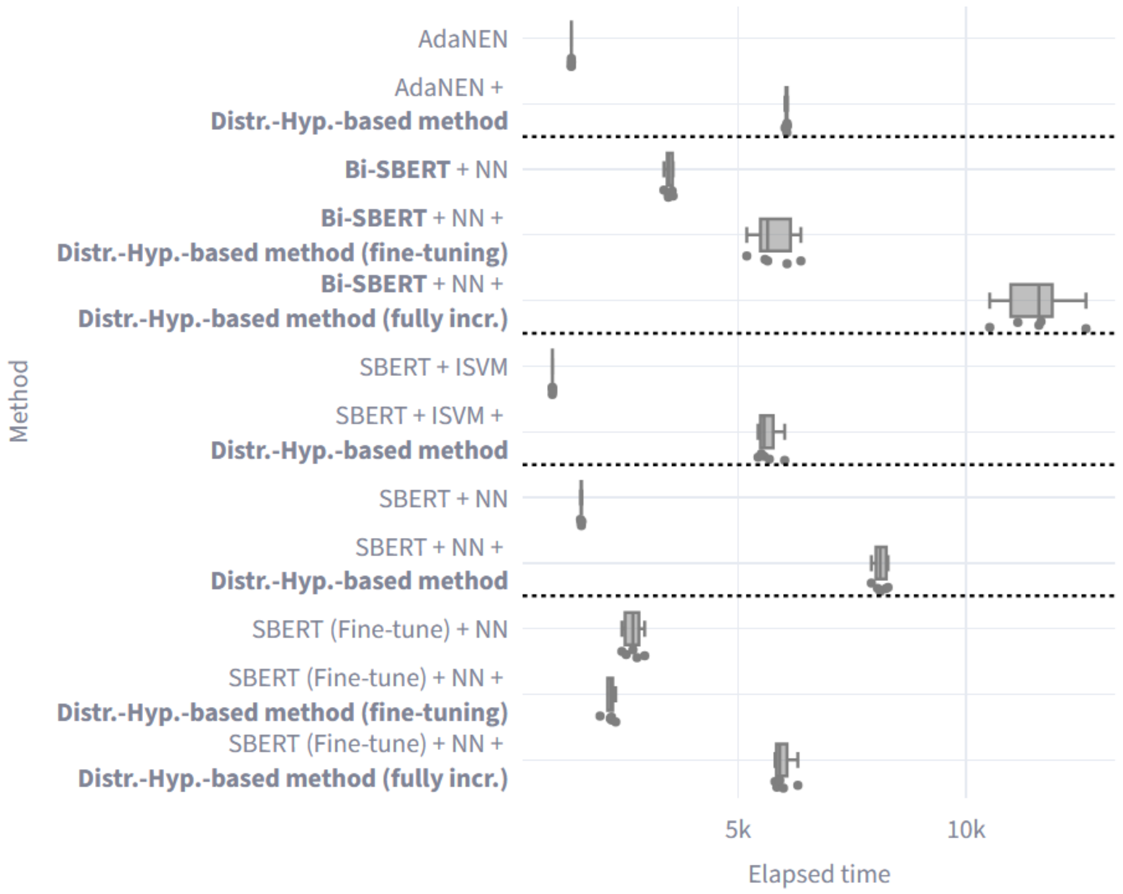

Figure 6.26: Results regarding elapsed times for the Yelp dataset, in the scenario of Class Swap.

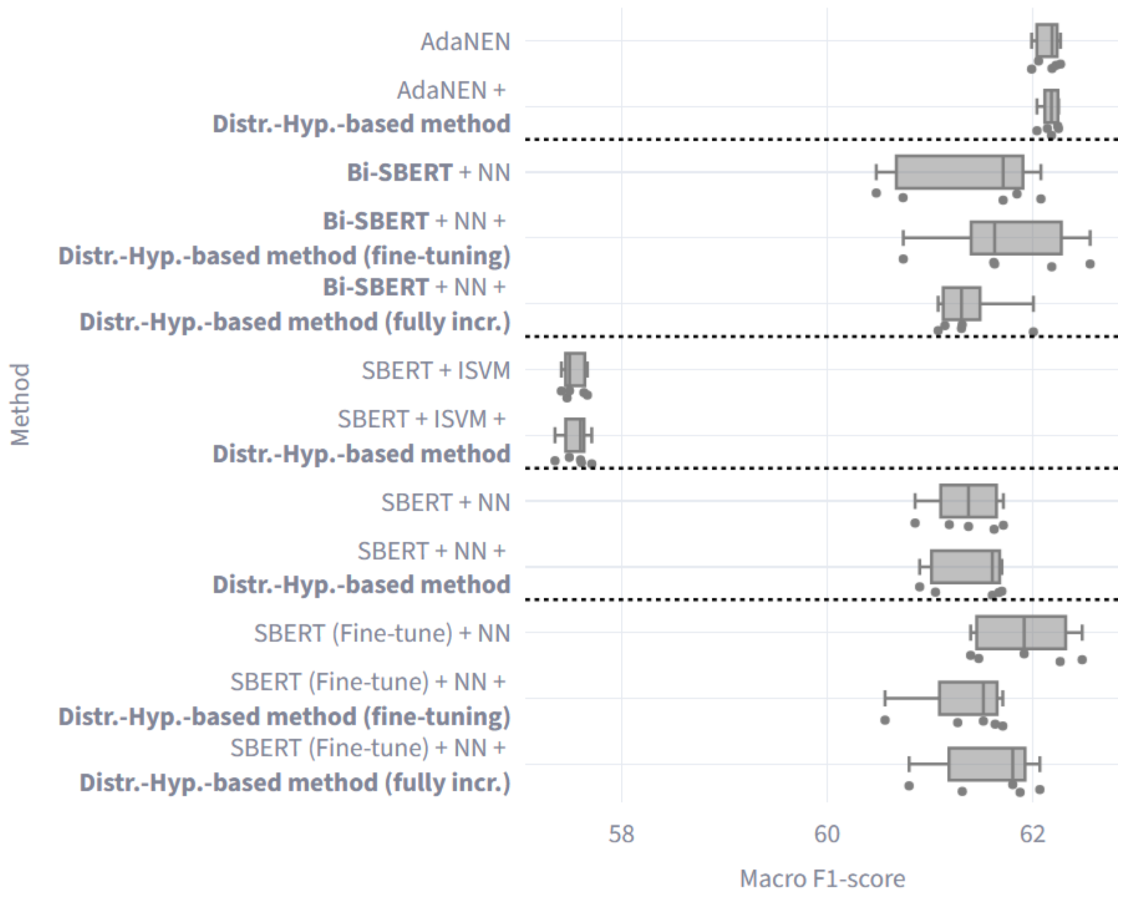

Figure 6.27: Results for the Yelp dataset, in the scenario of Class Shift.

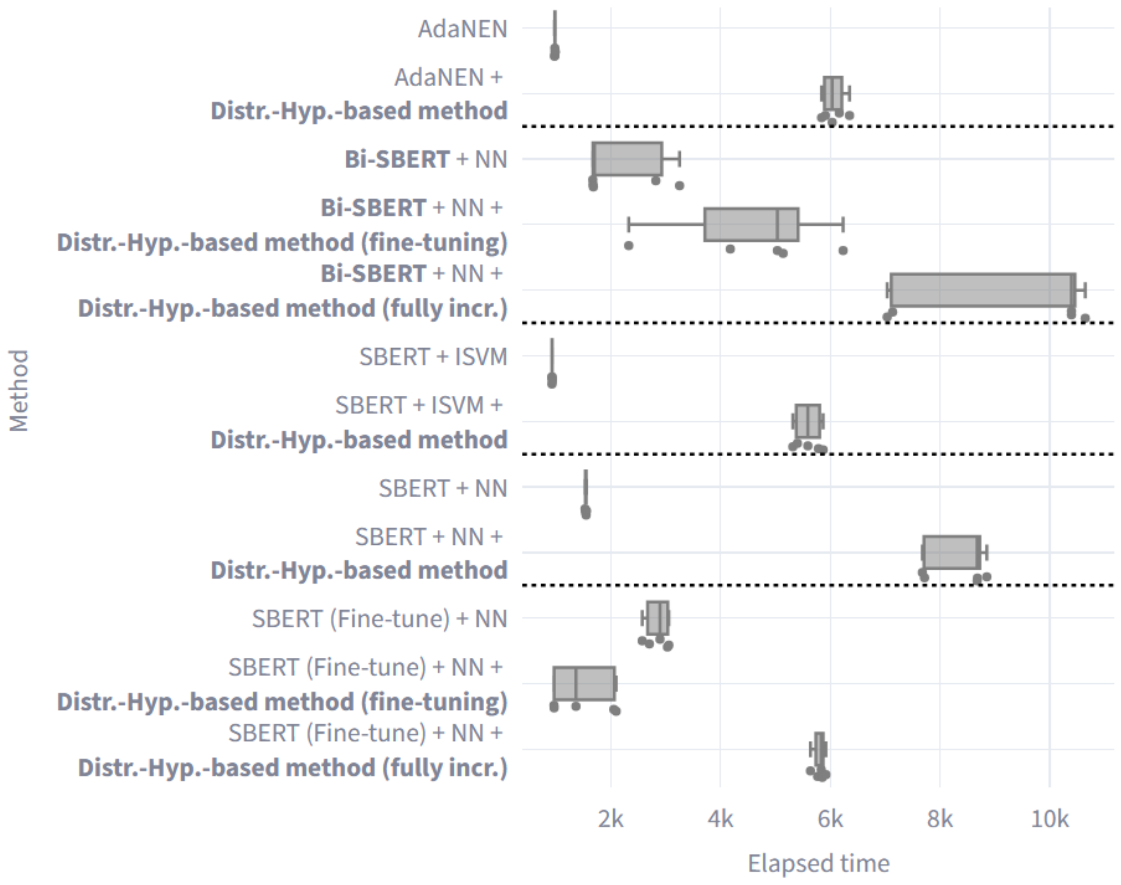

Figure 6.28: Results regarding elapsed times for the Yelp dataset, in the scenario of Class Shift.

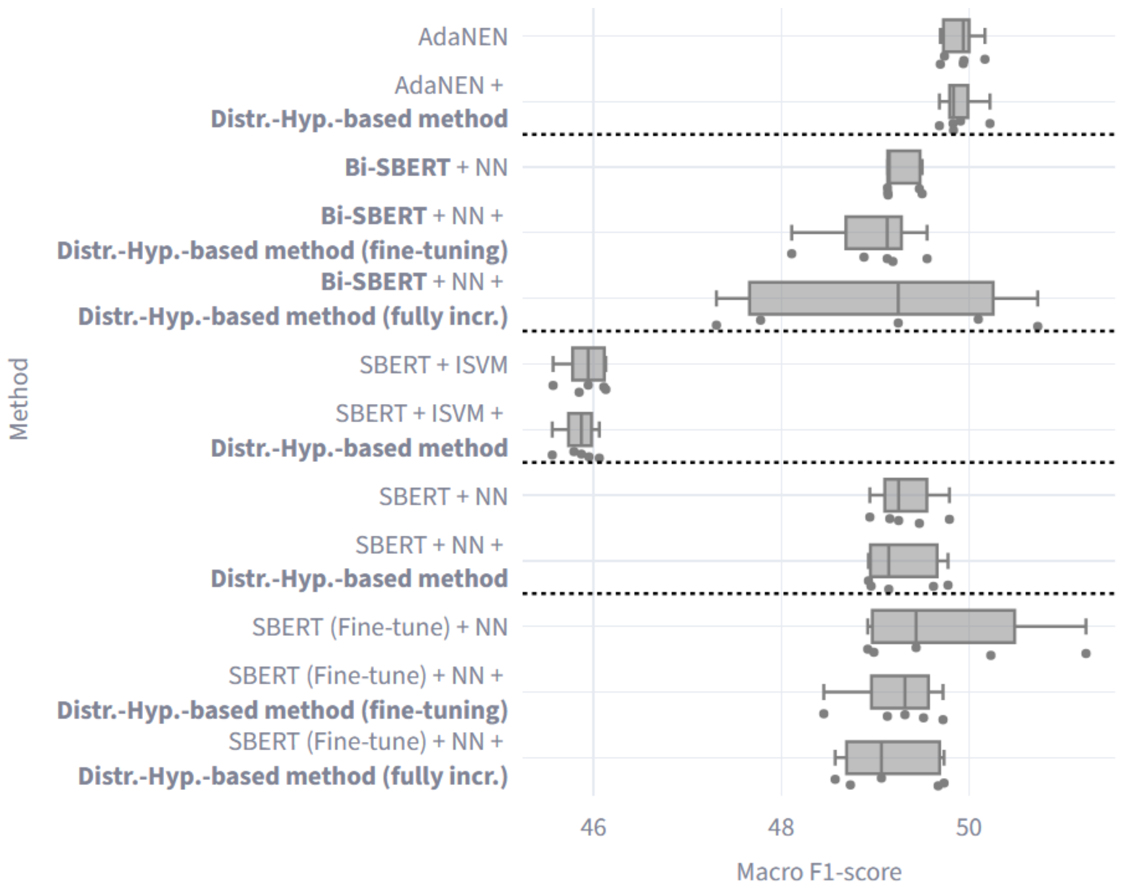

Figure 6.29: Results for the Yelp dataset, in the scenario of Time-slice Removal.

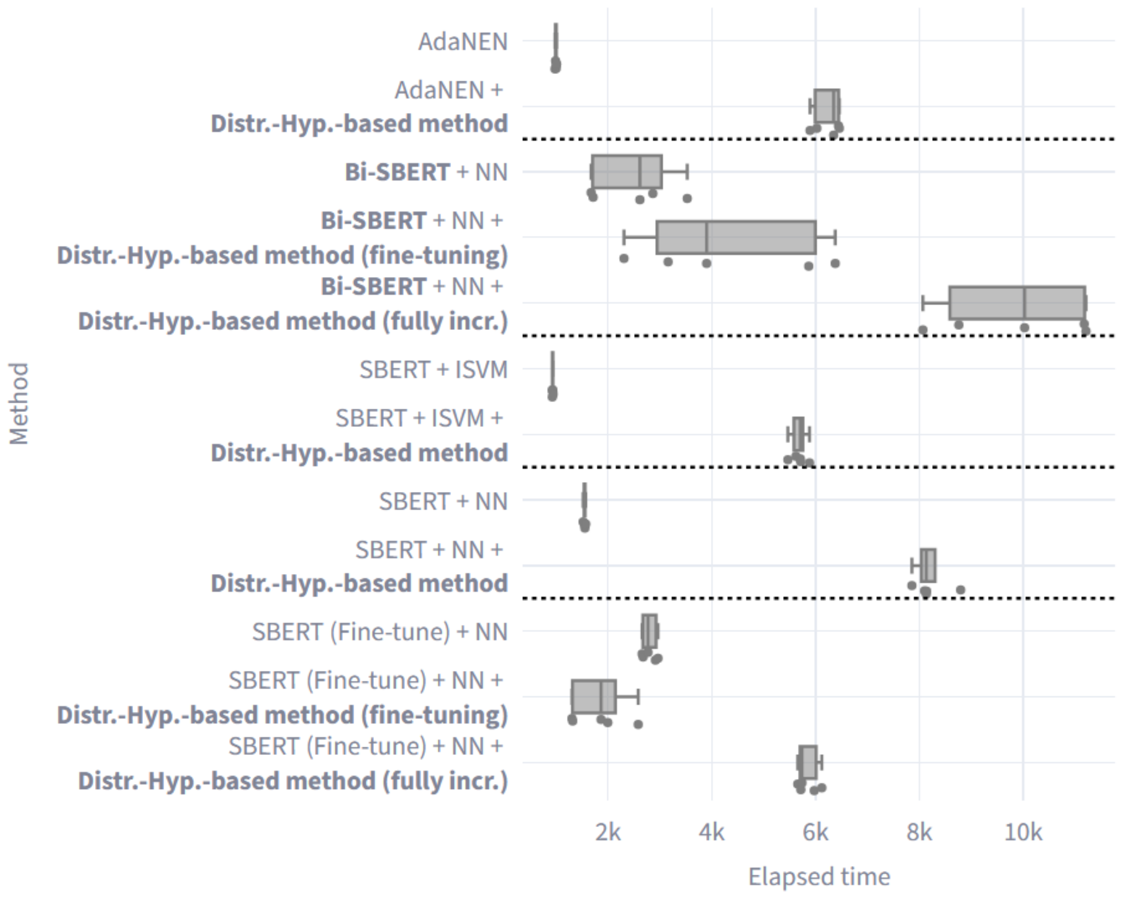

Figure 6.30: Results regarding elapsed times for the Yelp dataset, in the scenario of Time-slice Removal.

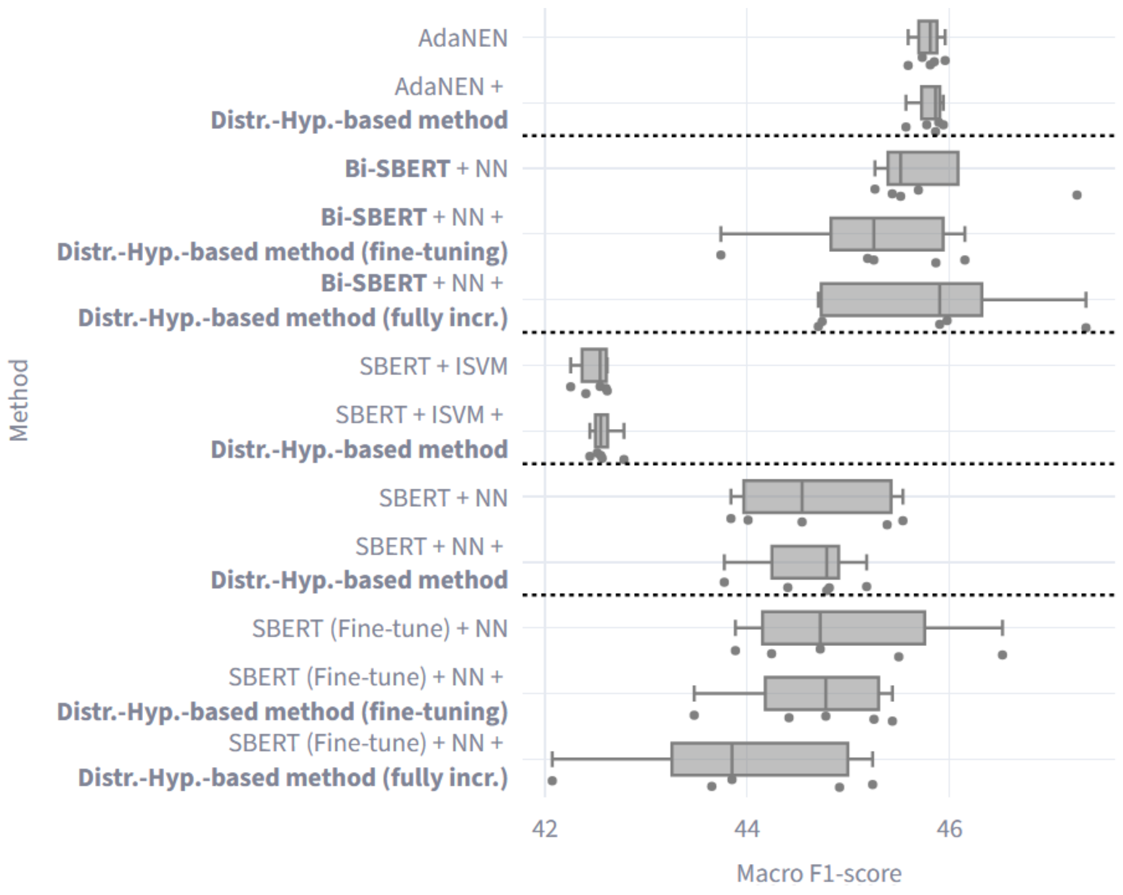

Figure 6.31: Results for the Yelp dataset, in the scenario of Adjective Swap.

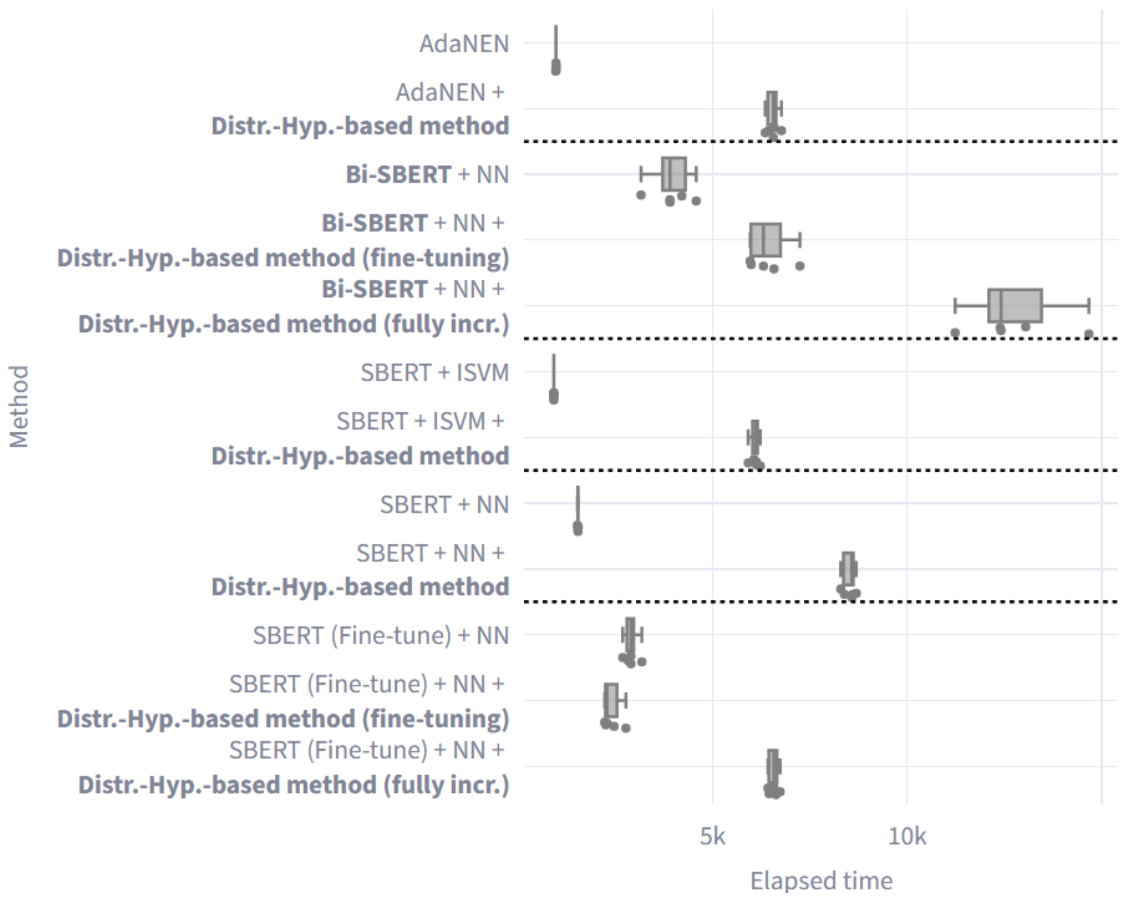

Figure 6.32: Results regarding elapsed times for the Yelp dataset, in the scenario of Adjective Swap.

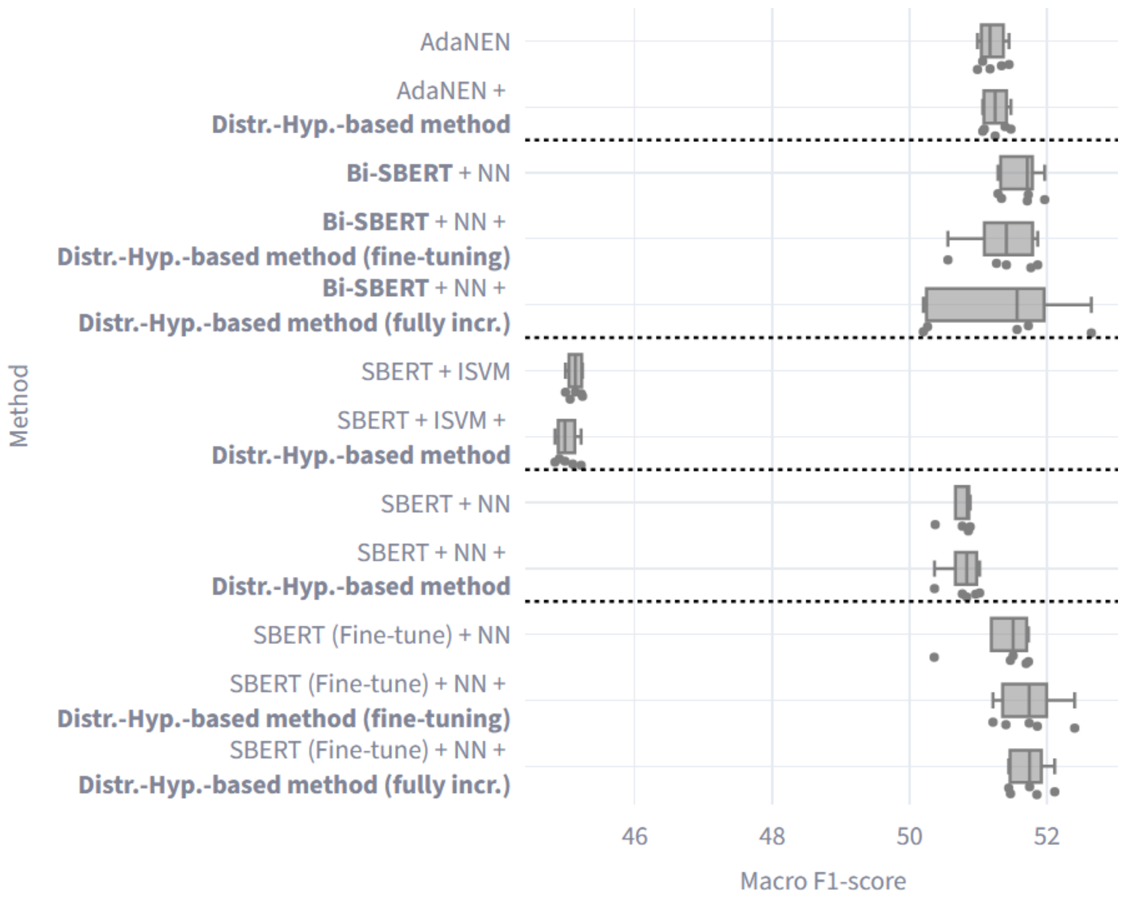

Figure 6.33: Results for the Yelp Full dataset, in the scenario without drift.

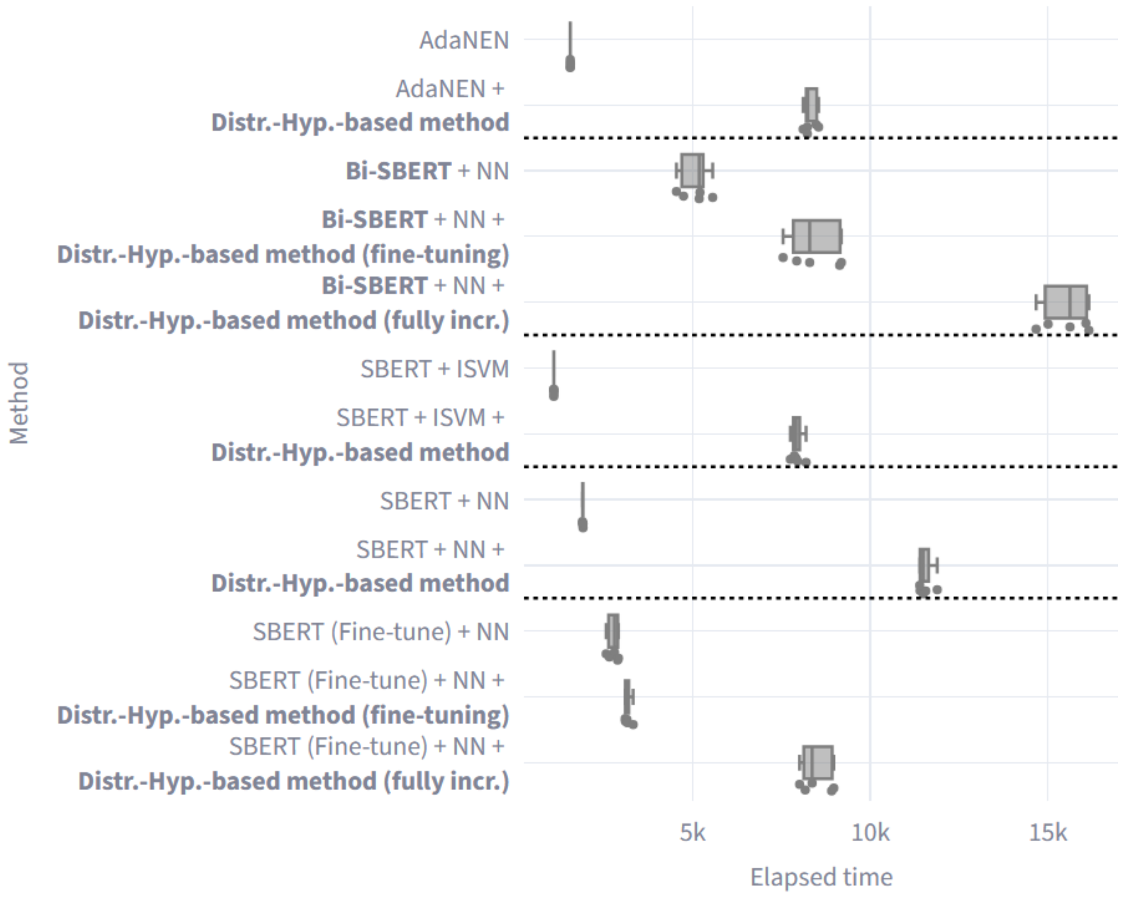

Figure 6.34: Results regarding elapsed times for the Yelp Full dataset, in the scenario without drift.

NN. **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)** obtained the longest elapsed time, while SBERT + ISVM was the fastest to run. In addition, AdaNEN took a competitive time to run.

Class Swap

Fig. 6.35 shows the results for Class Swap scenario in the Yelp Full dataset. Similarly to the runs without drift, most executions lied between 50 and 52% of Macro F1-Score. SBERT + ISVM reached the worst values, below 45% on average. In this setting, leveraging the **Distr.-Hyp.-based method** led to better results compared to the vanilla version in three of five groups. Verifying the Table B.20, in Appendix B, **Bi-SBERT** + NN reached the best overall Macro F1-Scores, i.e., 51.15 % ± 0.16.

Fig. 6.36 displays the results considering the elapsed times. An increase in

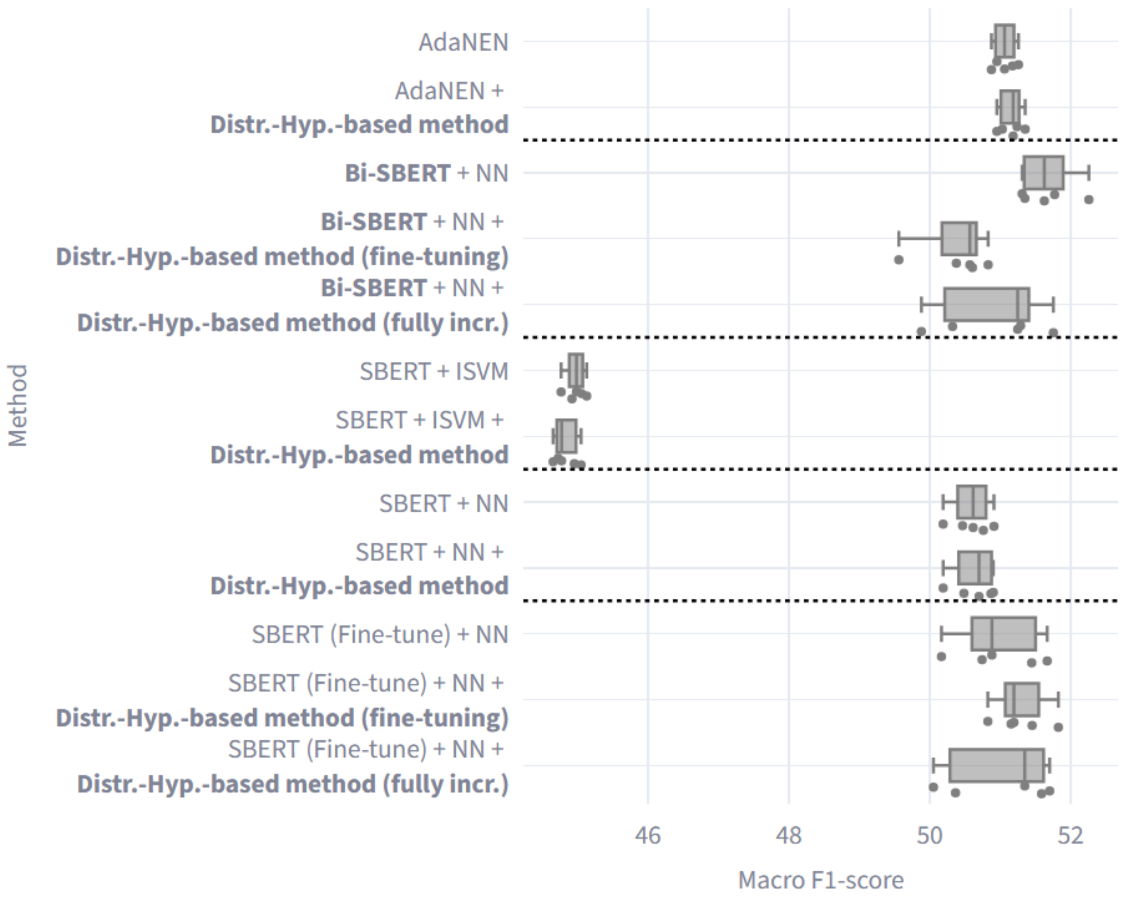

Figure 6.35: Results for the Yelp Full dataset, in the scenario of Class Swap.

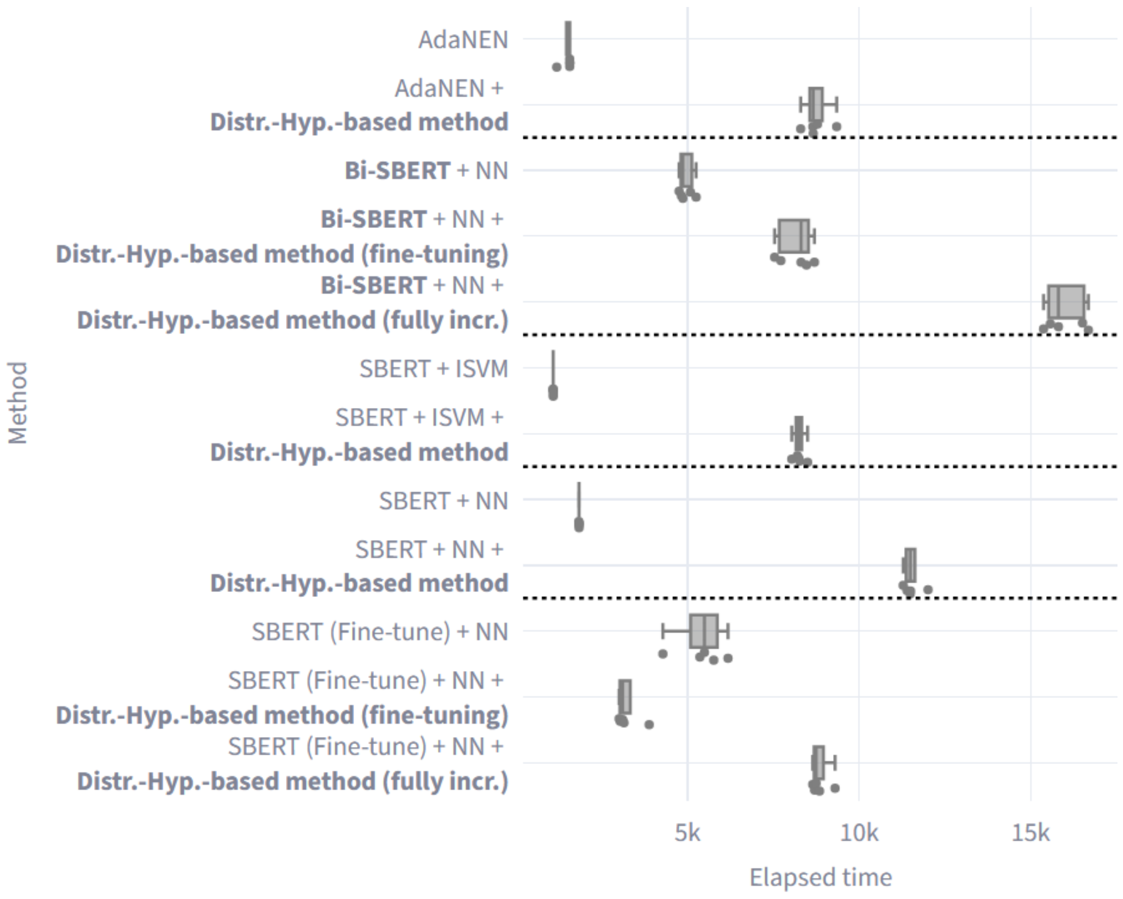

Figure 6.36: Results regarding elapsed times for the Yelp Full dataset, in the scenario of Class Swap.

run time was noted whenever the **Distr.-Hyp.-based method** is used, especially the fully incremental version. In addition, AdaNEN and SBERT + ISVM are considerably fast to run, while the Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)** took the longest to run.

Class Shift

Fig. 6.37 displays the Macro F1-Scores collected in the setting of Class Shift. Similarly to previous scenarios, most Macro F1-Scores lied next to or within a range between 50 and 52%. SBERT+ ISVM obtained the lowest results in terms of Macro F1-Scores. In this scenario, which is of fast changes, AdaNEN reached the best Macro F1-Scores. Precisely, resorting Appendix B, Table B.16, the"vanilla" AdaNEN reached 51.06% ± 0.15, and AdaNEN coupled with the **Distr.-Hyp.-based method** led to a slight improvement in the performance metric. Overall, leveraging **Distr.-Hyp.-based method** enhanced performance in four of five

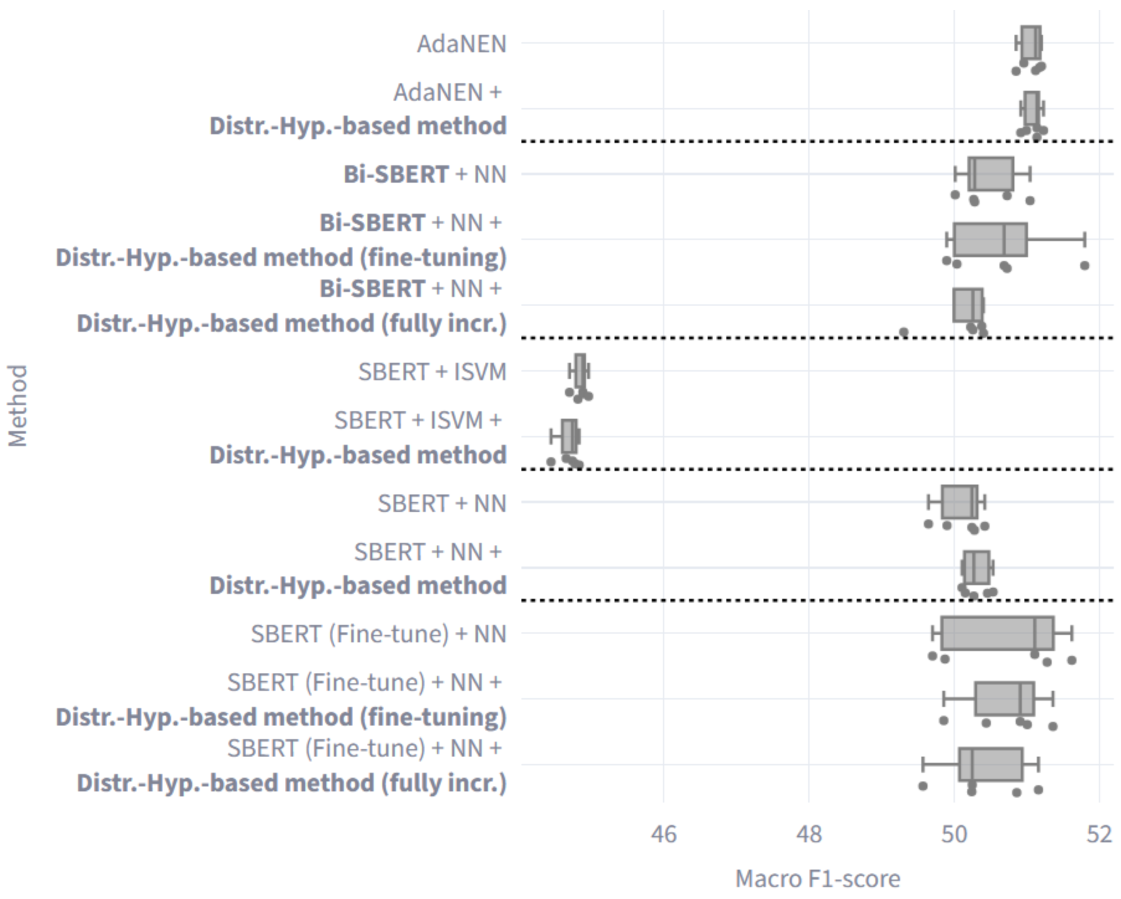

Figure 6.37: Results for the Yelp Full dataset, in the scenario of Class Shift.

results, obtaining at least a draw.

Fig. 6.38 shows the elapsed times measured in the experiments. The results show similar patterns to those previously observed. **Bi-SBERT** in general takes longer to run than SBERT + NN, SBERT + ISVM, and AdaNEN. Leveraging **Distr.-Hyp.-based method** generally led to longer elapsed times.

Adjective Swap

Fig. 6.39 shows the Macro F1-Scores obtained for the setting of Adjective Swap in the Yelp Full dataset. All methods had difficulties in keeping the same level of performance, as most of the methods obtained Macro F1-Scores between 46 and 48%, below the scores observed in previous scenarios. ISVM performed poorly compared to other methods, similarly to the observations of other methods. Resorting to Table B.22, SBERT + ISVM obtained 41.95 ± 0.05, and leveraging the **Distr.-Hyp.-based method** led to an even worse result.

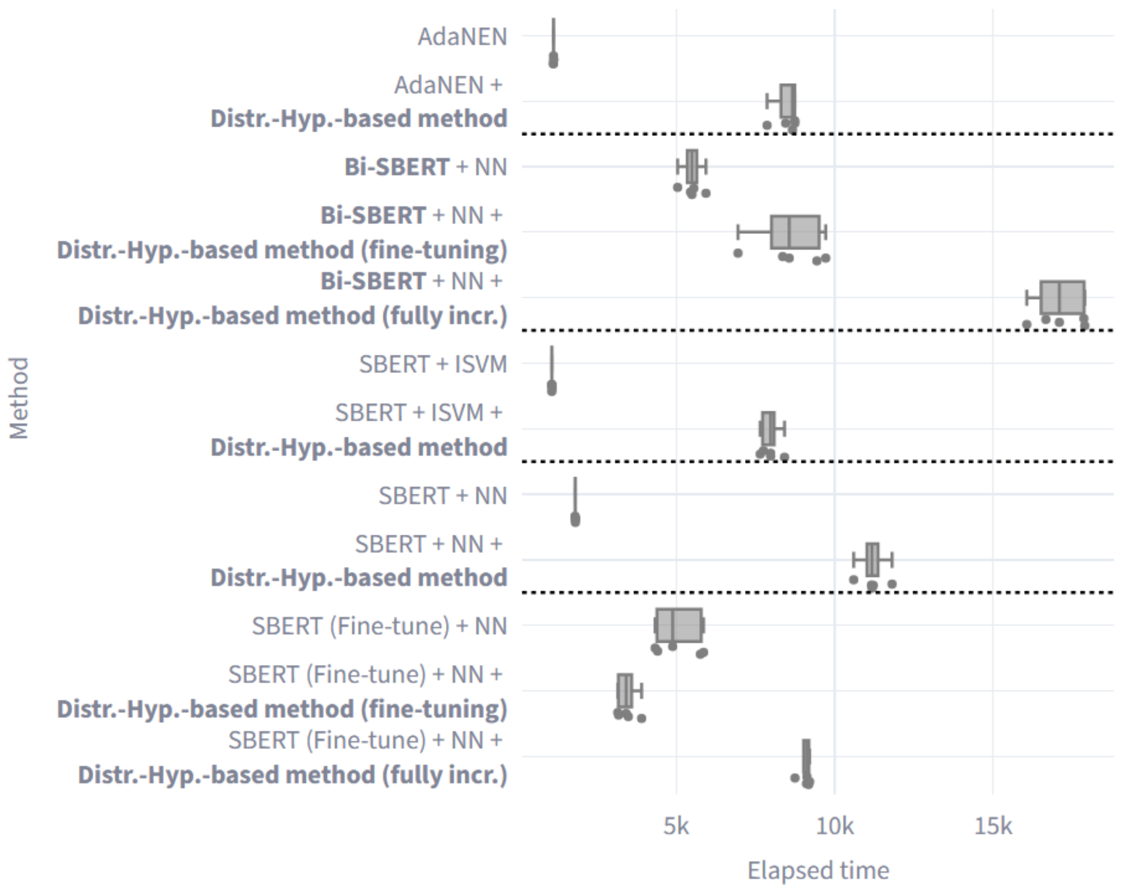

Figure 6.38: Results regarding elapsed times for the Yelp Full dataset, in the scenario of Class Shift.

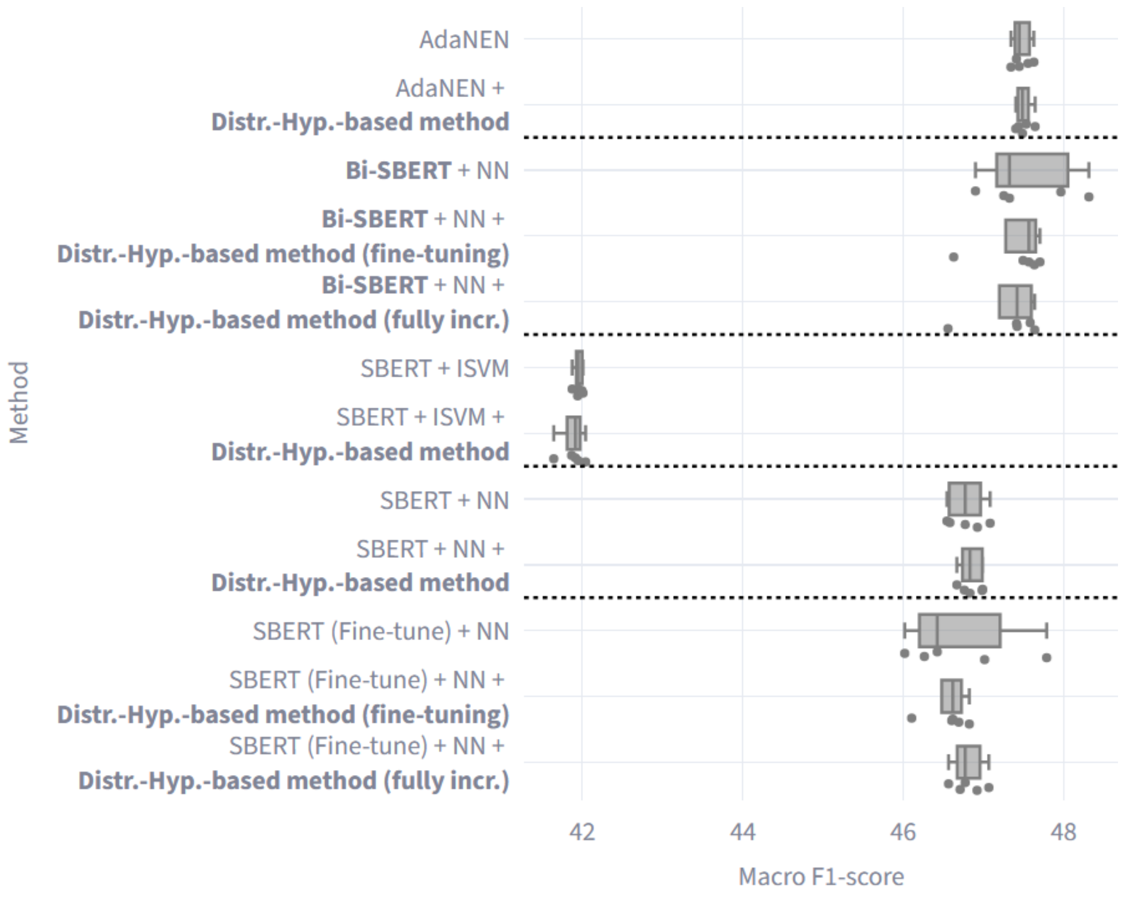

Figure 6.39: Results for the Yelp Full dataset, in the scenario of Adjective Swap.

Besides, **Bi-SBERT** + NN obtained the best Macro F1-Scores, reaching 47.55% ± 0.57. Leveraging the **Distr.-Hyp.-based method** led to better results in three of five groups.

Fig. 6.40 shows similar patterns to those previously observed. In this case, however, **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)** took even longer to run, e.g., longer than 20,000 seconds in an isolated run. In Appendix B, Table B.22 shows that the best elapsed times were obtained by the vanilla methods, with the exception of SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**, which performed faster than its vanilla version.

6.1.6 Yelp Polarity

This subsection presents the results for the Yelp Polarity dataset, which is a binary dataset. Therefore, it is expected to obtain higher levels of Macro F1-Scores.

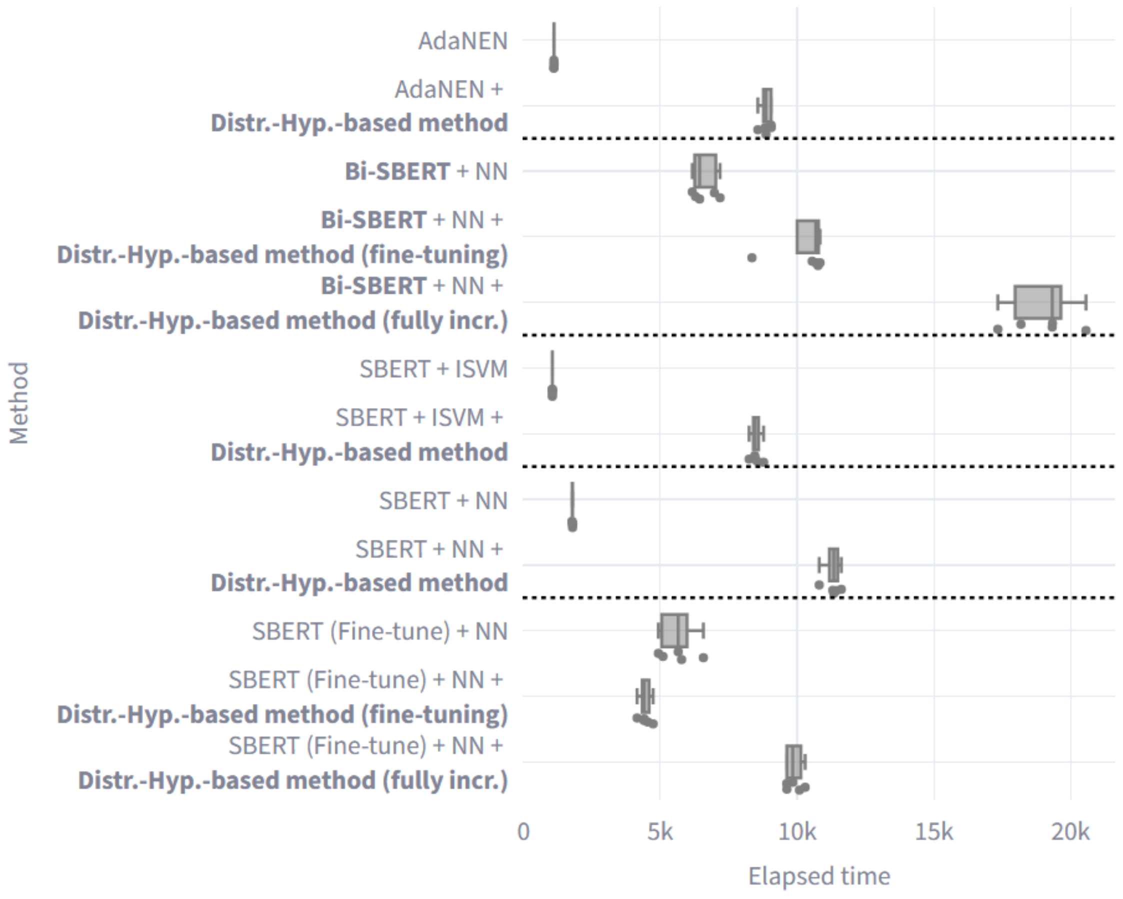

Figure 6.40: Results regarding elapsed times for the Yelp Full dataset, in the scenario of Adjective Swap.

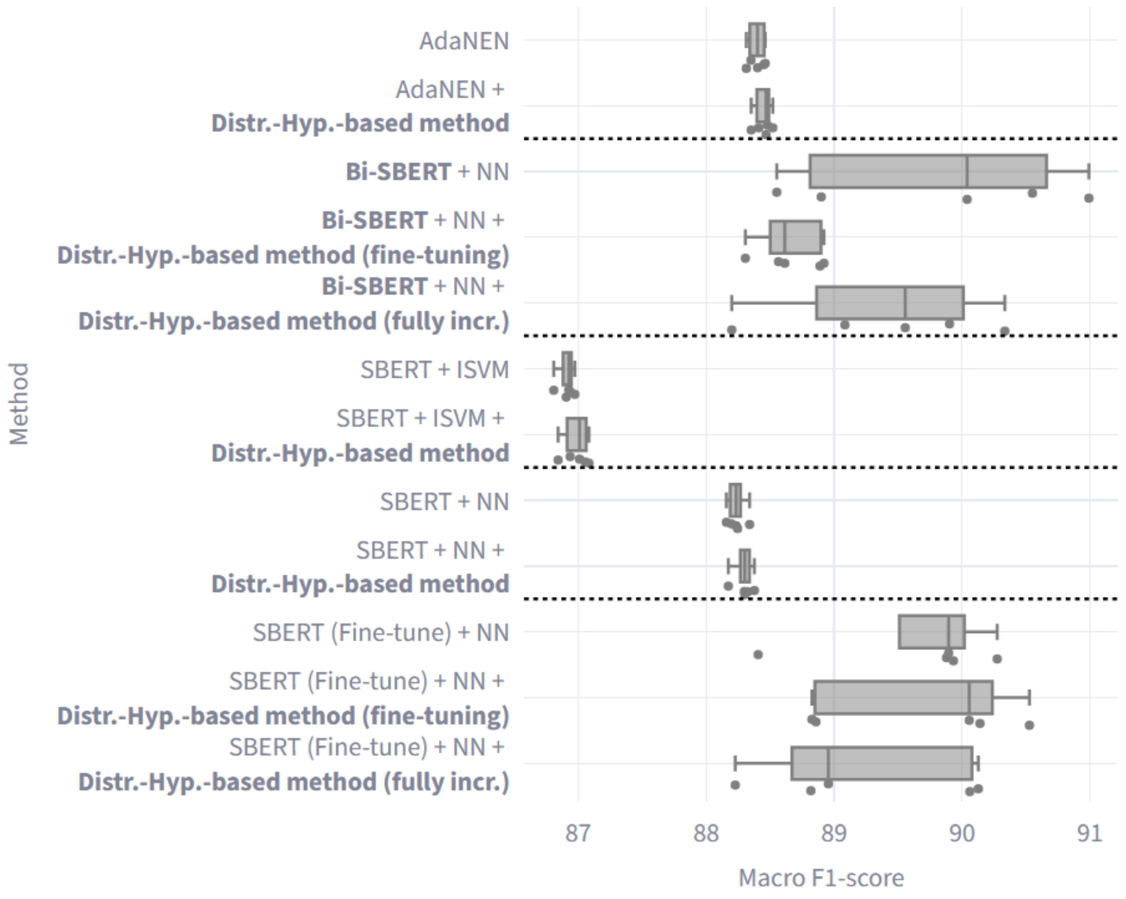

Figure 6.41: Results for the Yelp Polarity dataset, in the scenario without drift.

Without drift

Fig. 6.41 shows the results regarding Macro F1-Scores. The groups of AdaNEN, SBERT + ISVM, and SBERT + NN were steady, providing little variation across the runs. On the other hand, **Bi-SBERT** + NN provided results ranging from below 89% to 91%. SBERT (Fine-tune) + NN also provided a considerable range of results. SBERT + ISVM reached the lowest Macro F1-Scores, while **Bi-SBERT** + NN reached the best overall Macro F1-Scores, i.e., 89.81 ± 1.05. Leveraging the **Distr.-Hyp.-based method** provided better results than the vanilla version in four of five groups. The only exception regards **Bi-SBERT** + NN. The precise values provided in this paragraph can be obtained in Appendix B, Table B.23.

Fig. 6.42 shows the results with respect to the elapsed times. SBERT + ISVM performed the fastest, as observed in other scenarios. **Bi-SBERT** + NN + **Distr.-Hyp.-based method** was the slowest to run, but considerably below the

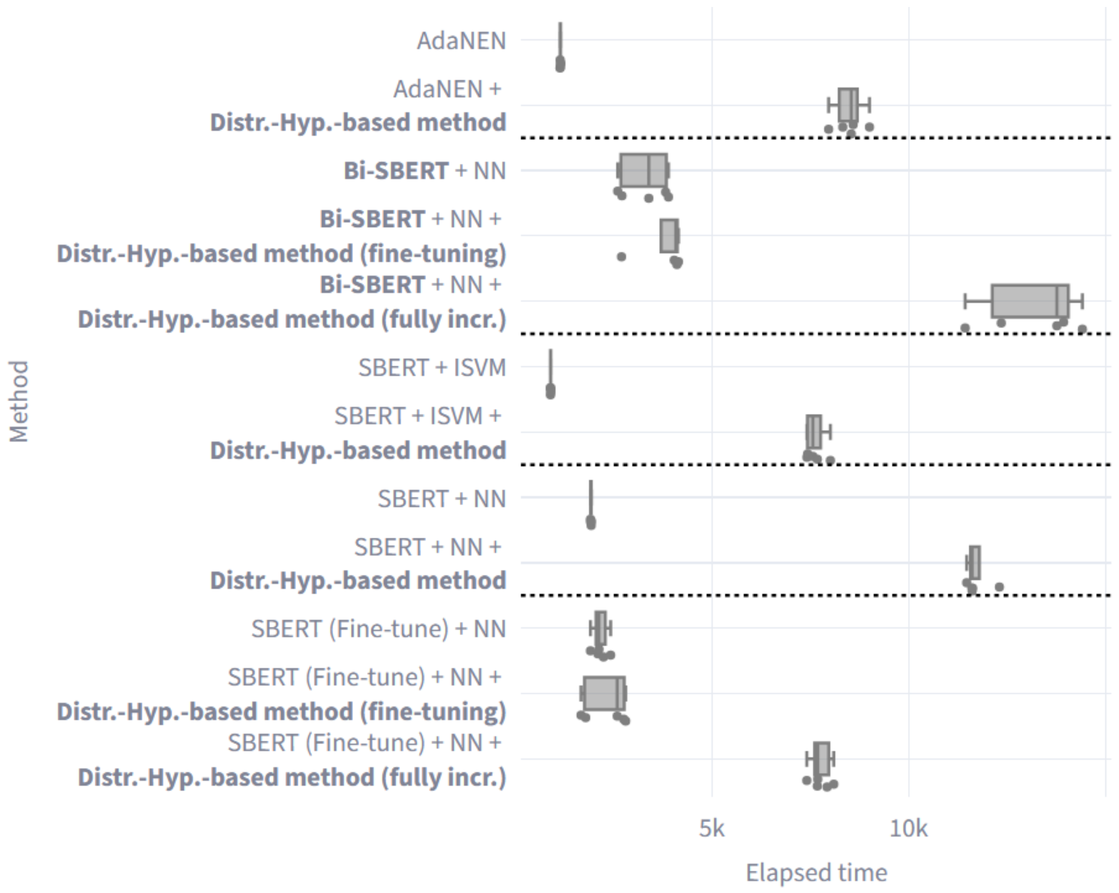

Figure 6.42: Results regarding elapsed times for the Yelp Polarity dataset, in the scenario without drift.

elapsed time observed in most scenarios with the Yelp Full dataset, below 14,000 seconds.

Class Swap

Fig. 6.43 shows the values of Macro F1-Score obtained during the experiments. Similarly to the scenario without drift, the groups of AdaNEN, SBERT + ISVM, and SBERT + NN performed steadily compared to the other methods. The groups of **Bi-SBERT** + NN and SBERT (Fine-tune) + NN had a considerable range of values, sometimes differing by more than 2 p.p. between the worst and the best result. This may be due to the fine-tuning process. Resorting to Appendix B, Table B.24, leveraging the proposed **Distr.-Hyp.-based method** led to better results in four of five groups, with SBERT (Fine-tune) + NN being the only exception. Overall, the highest average of Macro F1-Score was obtained by SBERT (Fine-tune) + NN, i.e., 89.47% ± 0.93.

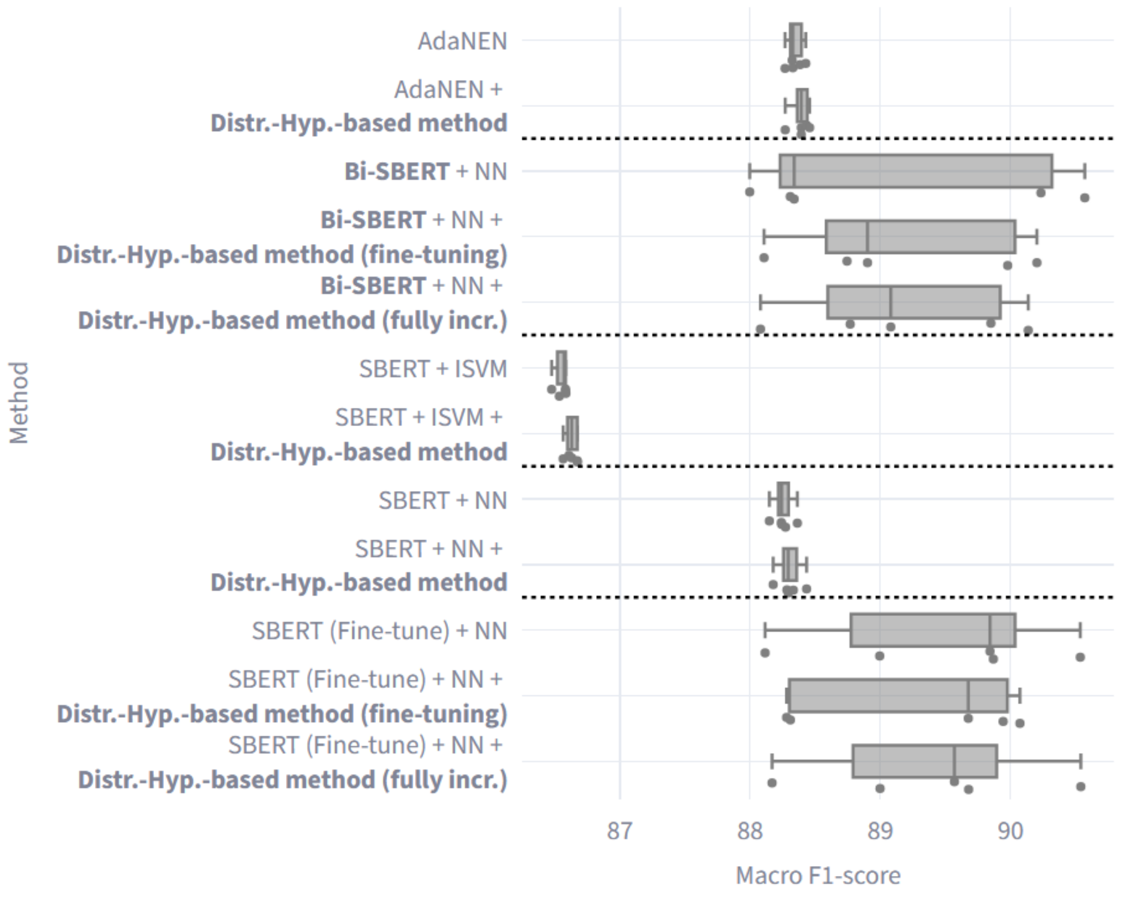

Figure 6.43: Results for the Yelp Polarity dataset, in the scenario of Class Swap.

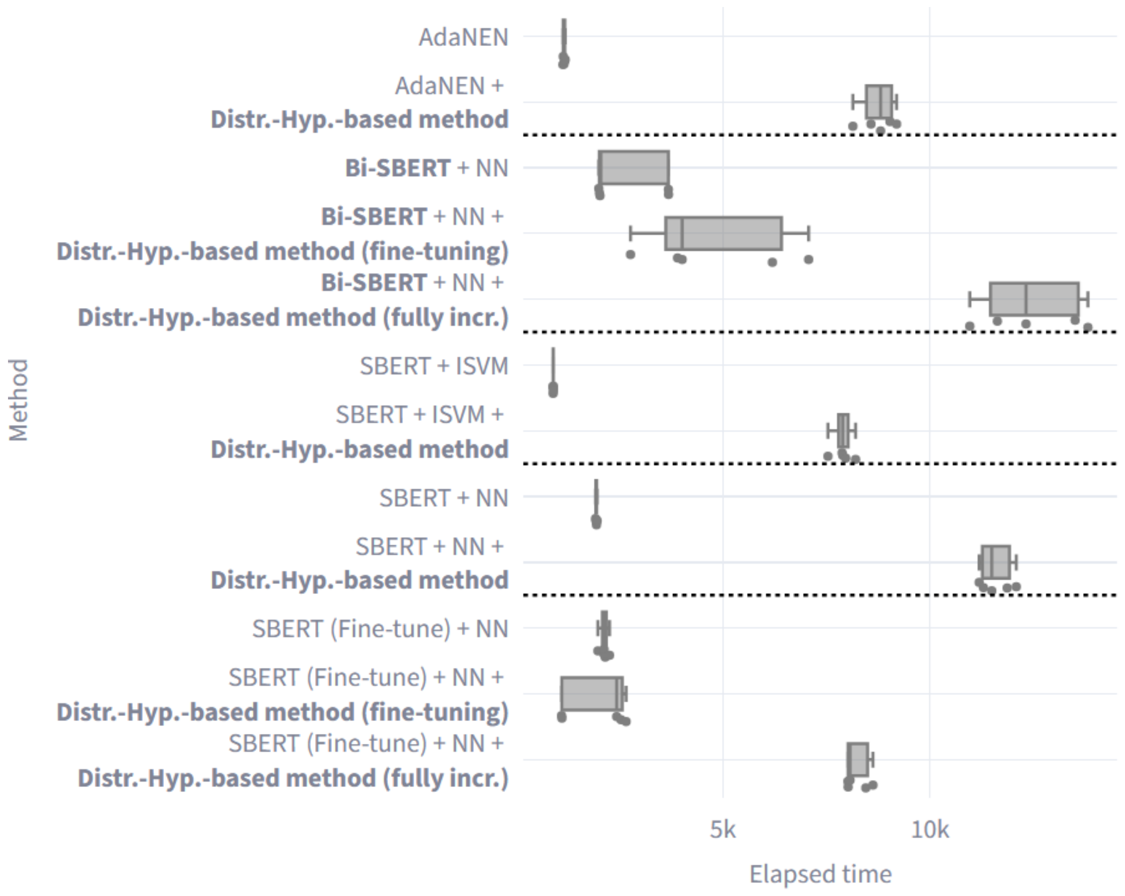

Figure 6.44: Results regarding elapsed times for the Yelp Polarity dataset, in the scenario of Class Swap.

Fig. 6.44 shows the elapsed times measured in the experiments. The methods that use fine-tuning have a higher variation in elapsed time. SBERT + ISVM, SBERT + NN, and AdaNEN performed steadily regarding elapsed time. **Bi-SBERT** + NN + **Distr.-Hyp.-based method** took the longest to run, due to its incremental characteristic. In this setting, SBERT (Fine-tune) + NN ran slightly faster than SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)** on average.

Class Shift

Fig. 6.45 displays the results obtained for the Class Shift scenario in the Yelp Polarity dataset. Additionally, methods with any kind of fine-tuning tended to have a high variation of Macro F1-Scores. The best result was obtained by **Bi-SBERT** + NN + **Distr.-Hyp.-based method**. More precisely, as shown in Appendix B, Table B.25, it reached 89.32% ± 1.03 of Macro F1-Score. Taking

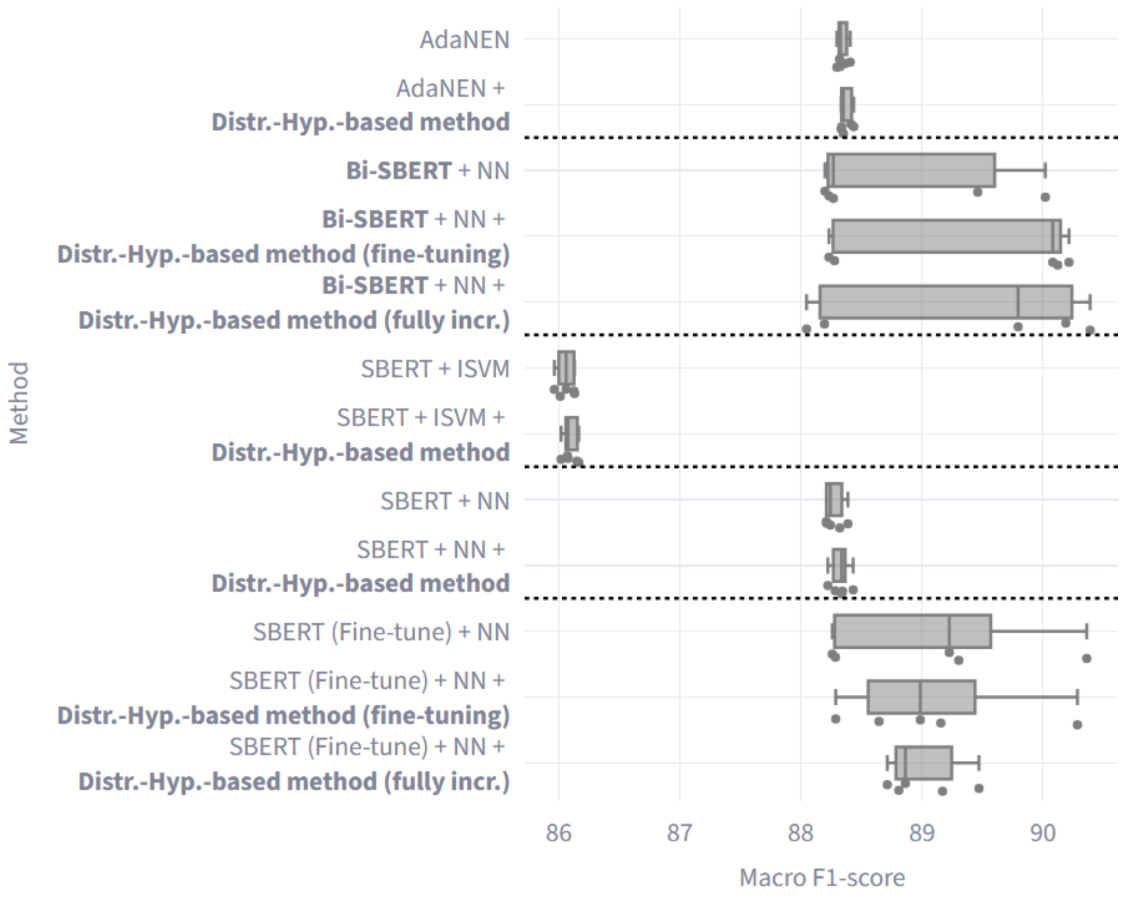

Figure 6.45: Results for the Yelp Polarity dataset, in the scenario of Class Shift.

advantage of the proposed **Distr.-Hyp.-based method**, four out of five methods had their results improved. Only SBERT (Fine-tune) + NN that performed slightly better than the methods with the **Distr.-Hyp.-based method**.

Fig. 6.46 shows the measured elapsed times in this setting. The results are similar to those previously observed: being subject to the triggering of the fine-tuning process leads to a wider range of elapsed time. In addition, AdaNEN and SBERT + ISVM ran faster than other methods. Leveraging the **Distr.-Hyp.-based method** in its fully incremental version leads to longer elapsed times.

Adjective swap

Fig. 6.47 shows the results regarding Macro F1-Scores. Again, the effects of being subject to the fine-tuning process triggering were noted. It provided a wider range of Macro F1-Scores. AdaNEN, SBERT + ISVM, and SBERT + NN groups were steady, having a smaller standard deviation. On the other

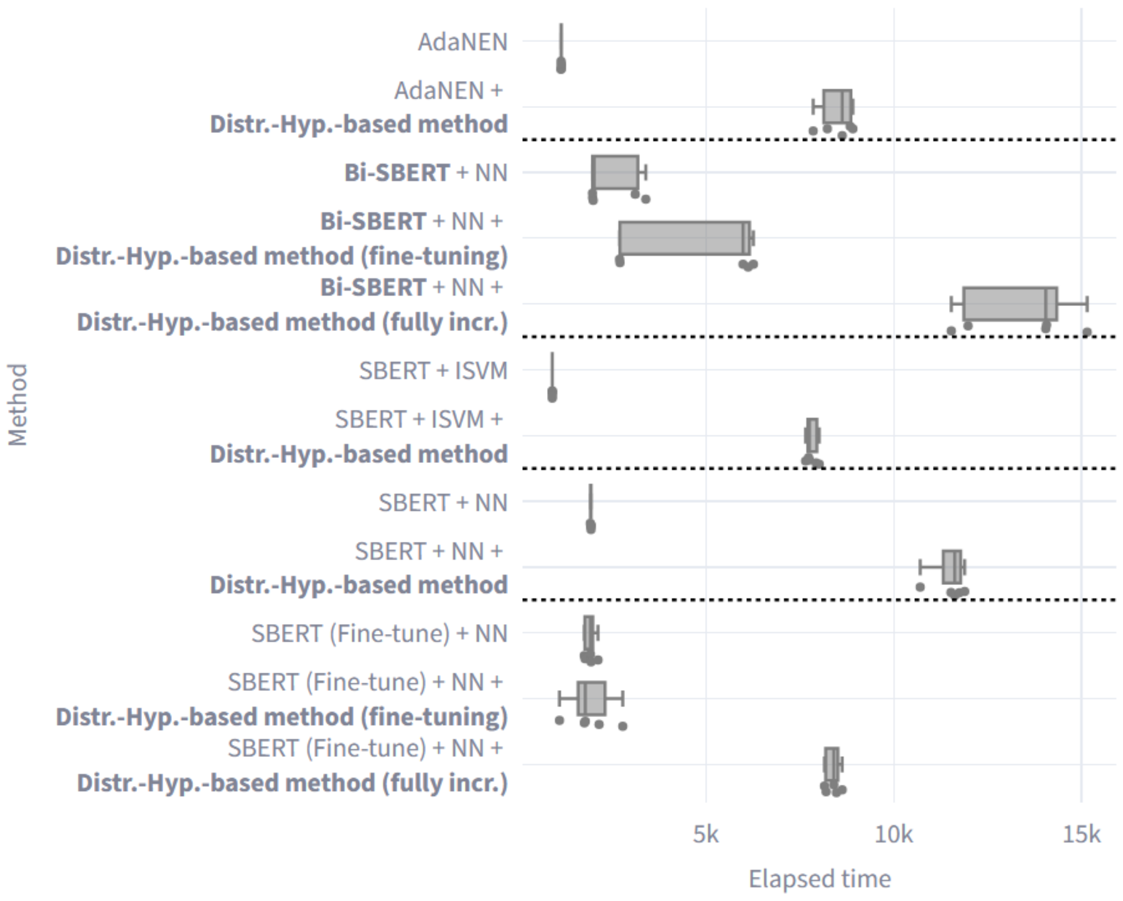

Figure 6.46: Results regarding elapsed times for the Yelp Polarity dataset, in the scenario of Class Shift.

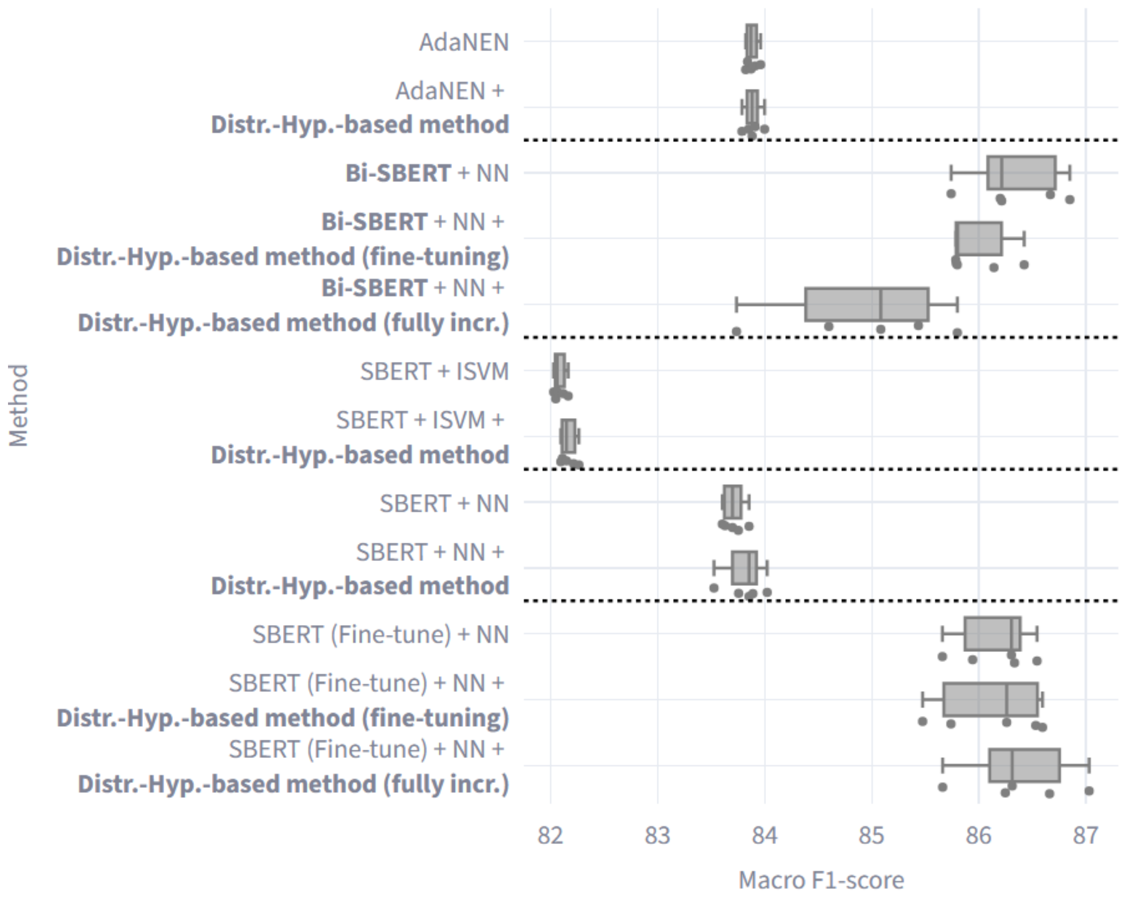

Figure 6.47: Results for the Yelp Polarity dataset, in the scenario of Adjective Swap.

hand, in this setting, AdaNEN could not reach a mean Macro F1-Score close to the best Macro F1-Score, keeping a level below 84%. Resorting to Appendix B, Table B.24, AdaNEN reached 83.88% ± 0.06, slight below AdaNEN + **Distr.-Hyp.-based method**, and the best results were obtained by SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**. Leveraging the proposed **Distr.-Hyp.-based method**, four of five methods could improve their results. The only exception was the **Bi-SBERT** + NN, which in its vanilla version could provide competitive performance.

Regarding the elapsed times, Fig. 6.48 shows the measured values. AdaNEN, SBERT + ISVM, and SBERT + NN were the fastest to run. The proposed **Distr.-Hyp.-based method** in its fully incremental version can lead to longer elapsed times, due to its iterative nature.

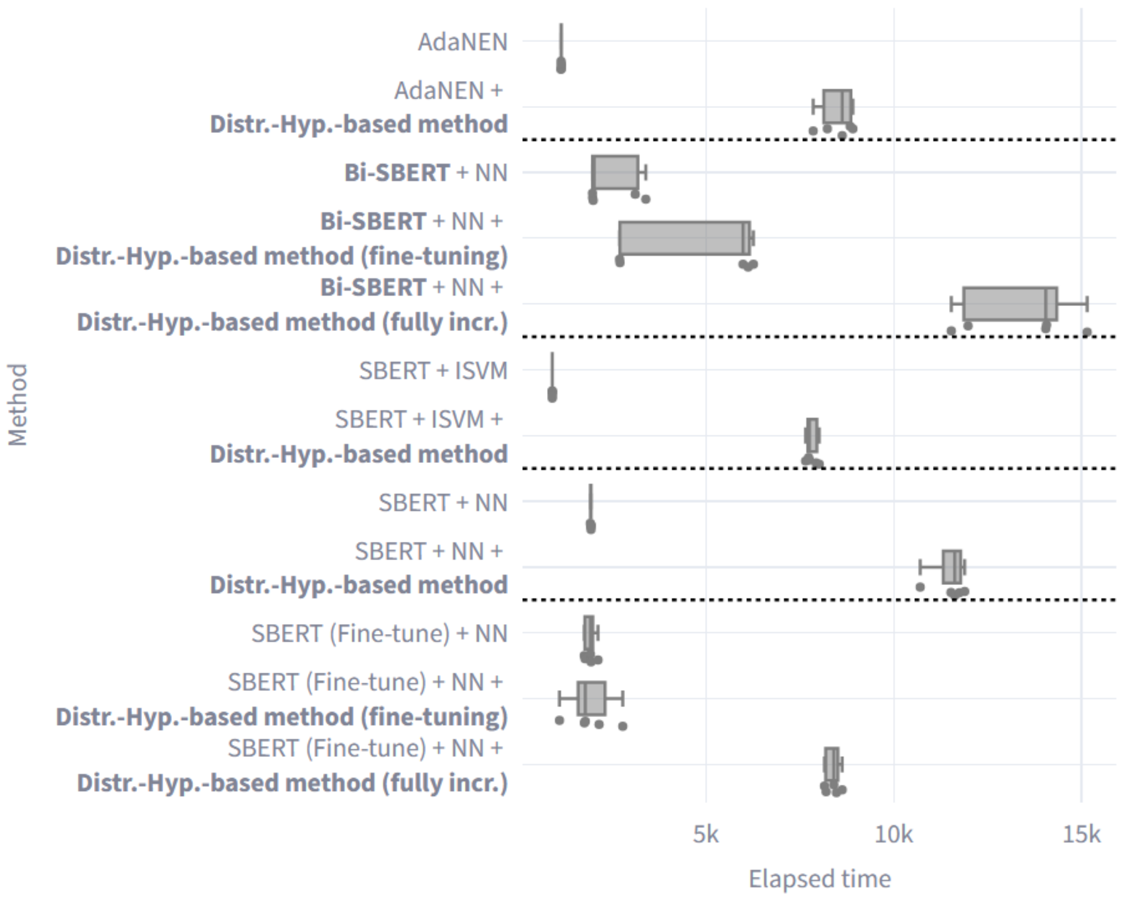

Figure 6.48: Results regarding elapsed times for the Yelp Polarity dataset, in the scenario of Adjective Swap.

6.1.7 Concluding Remarks

This section presented the results obtained from the experiments. The Macro F1-Scores and the measured elapsed times were reported. In addition, the results are tabulated in Appendix B.

Considering the presented results, leveraging the **Distr.-Hyp.-based method** frequently leads to improved results. When coupled with other models, the method can help update text representations, providing more accurate results. More specifically, it provided improvements or a draw with the vanilla version in 86 out of 130 opportunities, representing 66,15% of the chances. Table 6.1 shows the number of times that leveraging the **Distr.-Hyp.-based method** led to a win, tie, and loss compared to the model without the **Distr.-Hyp.-based method**.

Table 6.1: Improvements or draw provided by the addition of the Distributed-Hypothesis-based method.

Dataset	Wins	Ties	Losses
Airbnb	19	0	6
Amazon Polarity	14	5	1
Amazon Reviews	7	0	13
Yelp	10	2	14
Yelp Full	11	2	7
Yelp Polarity	15	1	4

Analyzing Table 6.1, Amazon Reviews and Yelp were the most difficult datasets, while in Airbnb and Yelp Polarity datasets, the proposed **Distr.-Hyp.-based method** led to wins most of the times. In Amazon Polarity, leveraging the proposed method resulted in a loss only once.

Regarding the **Bi-SBERT**, it obtained the best values among all methods in 11 out of 26 opportunities, corresponding to around 42%. It demonstrates that leveraging multiple language models can mitigate the effects of the concept drift phenomenon. However, the improvements, considering the **Bi-SBERT** and the methods that leverage the **Distributed-Hypothesis-based method**, came at a cost. There are increments in the elapsed times. For example, Table 6.2 shows the impact in terms of elapsed time of using the **Distributed-Hypothesis-based method** coupled to AdaNEN. Each number represents a multiple, having using vanilla AdaNEN corresponding to 1. Therefore, in the scenario without drift with the Airbnb dataset, AdaNEN + **Distributed-Hypothesis-based method**

Table 6.2: Comparison between AdaNEN and AdaNEN + **Distributed-Hypothesis-based method**. The number represents the impact of leveraging the **Distributed-Hypothesis-based method** in terms of elapsed times. In green, the smallest value, and in red, the highest multiple measured.

	Airbnb	**Amazon Polarity**	**Amazon Reviews**	**Yelp**	**Yelp Full**	**Yelp Polarity**
Without drift	3.48	5.08	3.86	5.23	5.35	7.37
Class Swap	3.51	5.08	3.98	4.58	5.89	7.52
Class Shift	3.48	5.09	5.14	6.18	7.49	7.41
Time Slice Removal	3.58	-	-	6.25	-	-
Adjective Swap	4.02	5.40	5.23	6.73	7.96	7.87

took 3.48 times more than AdaNEN alone. The tables regarding the other approaches are available in Appendix C.

6.2 Statistical Evaluation

In this section, the statistical evaluation was performed on the results, following the procedures presented in Section 5.9. Please, recall that five executions per setting are not generally enough for statistical evaluation. Therefore, the Wilcoxon signed-rank test was elected to perform the statistical evaluation using the executions across datasets and across scenarios, pairing methods without and with the **Distributed-Hypothesis-based method**.

6.2.1 AdaNEN vs. AdaNEN + **Distributed-Hypothesis-based method**

Table 6.3 shows the statistical comparison between AdaNEN and AdaNEN + **Distributed-Hypothesis-based method** across the datasets. It is noticeable that in all datasets, using the **Distributed-Hypothesis-based method** led to better results compared to the vanilla AdaNEN. Specifically, for Amazon Polarity, Yelp Full, and Yelp Polarity, **Distributed-Hypothesis-based method** significantly improved the results. The values highlighted in green indicate statistically significant differences.

Table 6.4 displays the results across the drift scenarios. Regardless of the datasets, leveraging the **Distributed-Hypothesis-based method** improved AdaNEN results in all scenarios, but the differences are statistically significant for

Table 6.3: Statistical comparison using the Wilcoxon signed-rank test between AdaNEN and AdaNEN + **Distributed-Hypothesis-based method** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.691519	AdaNEN + **Distributed-Hypothesis-based method**
Amazon Polarity	0.039989	AdaNEN + **Distributed-Hypothesis-based method**
Amazon Review	0.570597	AdaNEN + **Distributed-Hypothesis-based method**
Yelp	0.055070	AdaNEN + **Distributed-Hypothesis-based method**
Yelp Full	0.000105	AdaNEN + **Distributed-Hypothesis-based method**
Yelp Polarity	0.000322	AdaNEN + **Distributed-Hypothesis-based method**

the scenarios *Without drift*, Class Swap, and Class Shift.

Table 6.4: Statistical comparison using the Wilcoxon signed-rank test between AdaNEN and AdaNEN + **Distributed-Hypothesis-based method** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.502761	AdaNEN + **Distributed-Hypothesis-based method**
Class Swap	0.002020	AdaNEN + **Distributed-Hypothesis-based method**
Class Shift	0.017454	AdaNEN + **Distributed-Hypothesis-based method**
Time Slice Removal	0.845703	AdaNEN + **Distributed-Hypothesis-based method**
Adjective Swap	0.084065	AdaNEN + **Distributed-Hypothesis-based method**

6.2.2 Bi-SBERT + NN vs. Bi-SBERT + NN + Distributed-Hypothesis-based method (fine-tuning)

Table 6.5 shows the results of the comparison between **Bi-SBERT** + NN and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fine-tuning)**. In this case, there is no statistically relevant differences between both approaches.

Table 6.5: Statistical comparison using the Wilcoxon signed-rank test between **Bi-SBERT** + NN and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fine-tuning)** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.750993	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Amazon Polarity	0.345231	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Amazon Review	0.368277	**Bi-SBERT** + NN
Yelp	0.055070	**Bi-SBERT** + NN
Yelp Full	0.063723	**Bi-SBERT** + NN
Yelp Polarity	0.570597	**Bi-SBERT** + NN

Table 6.6 displays the results per drift scenario. **Bi-SBERT** + NN obtained statistically better results in the scenarios without drift and Class Swap. According to the results, leveraging the **Distributed-Hypothesis-based method** whenever a fine-tuning process is triggered does not improve the results coupled with the proposed **Bi-SBERT**.

Table 6.6: Statistical comparison using the Wilcoxon signed-rank test between **Bi-SBERT** + NN and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fine-tuning)** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.010181	**Bi-SBERT** + NN
Class Swap	0.017253	**Bi-SBERT** + NN
Class Shift	0.275832	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Time Slice Removal	0.556641	**Bi-SBERT** + NN
Adjective Swap	0.919297	**Bi-SBERT** + NN

6.2.3 Bi-SBERT + NN vs. Bi-SBERT + NN + Distributed-Hypothesis-based method (fully incr.)

Table 6.7 shows the comparison between the **Bi-SBERT** + NN and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fully incr.)**. **Bi-SBERT** + NN obtained statistically better results in three datasets, i.e., Amazon Review, Yelp, and Yelp Full. Leveraging the **Distributed-Hypothesis-based method (fully incremental)** led to a better average only in Amazon Polarity, but it was not statistically significant.

Table 6.7: Statistical comparison using the Wilcoxon signed-rank test between **Bi-SBERT** + NN and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fully incremental)** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.873988	**Bi-SBERT** + NN
Amazon Polarity	0.230513	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)**
Amazon Review	0.009436	**Bi-SBERT** + NN
Yelp	0.036682	**Bi-SBERT** + NN
Yelp Full	0.044054	**Bi-SBERT** + NN
Yelp Polarity	0.230513	**Bi-SBERT** + NN

Table 6.8 shows the results across scenarios. In all scenarios, **Bi-SBERT** + NN obtained the best results, being statistically superior in the scenarios without

drift and Class Swap.

Table 6.8: Statistical comparison using the Wilcoxon signed-rank test between **Bi-SBERT** + NN and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fully incremental)** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.006195	**Bi-SBERT** + NN
Class Swap	0.012048	**Bi-SBERT** + NN
Class Shift	0.084065	**Bi-SBERT** + NN
Time Slice Removal	0.322266	**Bi-SBERT** + NN
Adjective Swap	0.626346	**Bi-SBERT** + NN

6.2.4 Bi-SBERT + NN + Distributed-Hypothesis-based method (fine-tuning) vs. Bi-SBERT + NN + Distributed-Hypothesis-based method (fully incr.)

Table 6.9 shows the comparison between the **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fine-tuning)** and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fully incr.)**. Both versions are statistically equivalent, although **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fine-tuning)** obtained higher mean Macro F1-Scores.

Fig. 6.10 considers the results across the scenarios. Although regarding the datasets there was no statistically significant differences, in the Class Shift scenario, **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)** is statistically better than **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)**. Besides,

Table 6.9: Statistical comparison using the Wilcoxon signed-rank test between **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fine-tuning)** and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fully incremental)** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.894860	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Amazon Polarity	0.176853	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Amazon Review	0.097307	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Yelp	0.560171	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)**
Yelp Full	0.812355	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Yelp Polarity	0.545876	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**

Table 6.10: Statistical comparison using the Wilcoxon signed-rank test between **Bi-SBERT** + NN and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fully incremental)** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.839393	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)**
Class Swap	0.597808	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fully incr.)**
Class Shift	0.014538	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Time Slice Removal	0.431641	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**
Adjective Swap	0.685047	**Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)**

in 3 of 5 scenarios, **Bi-SBERT** + NN + **Distr.-Hyp.-based method (fine-tuning)** obtained higher averages.

6.2.5 SBERT + ISVM vs. SBERT + ISVM + **Distributed-Hypothesis-based method**

Table 6.11 shows the results for the statistical comparison between SBERT + ISVM and SBERT + ISVM + **Distributed-Hypothesis-based method**. In green, the statistically significant differences. In yellow, the only dataset in which SBERT + ISVM performed statistically better than SBERT + ISVM + **Distributed-Hypothesis-based method**. Besides, SBERT + ISVM + **Distributed-Hypothesis-based method** performed statistically better in 3 of 6 datasets. In the Yelp dataset, SBERT + ISVM + **Distributed-Hypothesis-based method** performed better, but in our experiments, it is not possible to state that it was statistically better.

Table 6.11: Statistical comparison using the Wilcoxon signed-rank test between SBERT + ISVM and SBERT + ISVM + **Distributed-Hypothesis-based method** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.017312	SBERT + ISVM + **Distributed-Hypothesis-based method**
Amazon Polarity	0.026642	SBERT + ISVM + **Distributed-Hypothesis-based method**
Amazon Review	0.089695	SBERT + ISVM
Yelp	0.936803	SBERT + ISVM + **Distributed-Hypothesis-based method**
Yelp Full	0.000027	SBERT + ISVM
Yelp Polarity	0.000002	SBERT + ISVM + **Distributed-Hypothesis-based method**

Regarding the scenarios, Table 6.12 shows the results of the statistical comparison. Across the scenarios, it was not possible to state any statistical difference; therefore, the methods were considered equivalent. However, the SBERT + ISVM + **Distributed-Hypothesis-based method** obtained better averages in 3 of 5 scenarios. This absence of statistical difference in the scenarios (with the existence of statistical difference in the datasets) may have happened due to the compensation that the results in some specific drift scenarios provided in terms of Macro F1-Score.

Table 6.12: Statistical comparison using the Wilcoxon signed-rank test between SBERT + ISVM and SBERT + ISVM + **Distributed-Hypothesis-based method** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.502761	SBERT + ISVM + **Distributed-Hypothesis-based method**
Class Swap	0.839393	SBERT + ISVM + **Distributed-Hypothesis-based method**
Class Shift	0.871208	SBERT + ISVM
Time Slice Removal	0.322266	SBERT + ISVM
Adjective Swap	0.236652	SBERT + ISVM + **Distributed-Hypothesis-based method**

6.2.6 SBERT + NN vs. SBERT + NN + Distributed-Hypothesis-based method

Table 6.13 shows the statistical comparison between SBERT + NN in the versions with and without the proposed **Distributed-Hypothesis-based method**. In green, the statistically significant differences. Therefore, across datasets, the **Distributed-Hypothesis-based method** significantly improved results in the Amazon Polarity and Yelp Polarity datasets. One hypothesis for this is the fact that both datasets are binary. Apart from that, it is noticeable that leveraging the **Distributed-Hypothesis-based method** led to improved results in 4 of 6 datasets.

Table 6.13: Statistical comparison using the Wilcoxon signed-rank test between SBERT + NN and SBERT + NN + **Distributed-Hypothesis-based method** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.811930	SBERT + NN
Amazon Polarity	0.029575	SBERT + NN + **Distributed-Hypothesis-based method**
Amazon Review	0.069580	SBERT + NN + **Distributed-Hypothesis-based method**
Yelp	0.832509	SBERT + NN
Yelp Full	0.082550	SBERT + NN + **Distributed-Hypothesis-based method**
Yelp Polarity	0.000322	SBERT + NN + **Distributed-Hypothesis-based method**

Considering the scenarios, Table 6.14 displays the results regarding the statistical comparison. In the Class Swap and Class Shift scenarios, using the

Distributed-Hypothesis-based method led to statistically significant, better results. Besides, only in scenario *Without drift*, SBERT + NN, without the **Distributed-Hypothesis-based method**, obtained better results.

Table 6.14: Statistical comparison using the Wilcoxon signed-rank test between SBERT + NN and SBERT + NN + **Distributed-Hypothesis-based method** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.871208	SBERT + NN
Class Swap	0.024786	SBERT + NN + **Distributed-Hypothesis-based method**
Class Shift	0.005776	SBERT + NN + **Distributed-Hypothesis-based method**
Time Slice Removal	0.275391	SBERT + NN + **Distributed-Hypothesis-based method**
Adjective Swap	0.328470	SBERT + NN + **Distributed-Hypothesis-based method**

6.2.7 SBERT (Fine-tune) + NN vs. SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tuning)**

The results regarding the comparison between SBERT (Fine-tune) + NN and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tuning)**. The green values highlight the statistically significant differences. The only statistically difference was obtained in the Amazon Polarity dataset, in which the best approach was the SBERT (Fine-tune) + NN. Leveraging the **Distributed-Hypothesis-based method (fine-tuning)** led to a higher average of Macro F1-Scores in the Amazon Review dataset, but the difference is not statistically relevant.

Table 6.15: Statistical comparison using the Wilcoxon signed-rank test between SBERT (Fine-tune) + NN and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tuning)** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.080196	SBERT (Fine-tune) + NN
Amazon Polarity	0.043114	SBERT (Fine-tune) + NN
Amazon Review	0.164957	SBERT (Fine-tune) + NN
Yelp	0.010511	SBERT (Fine-tune) + NN
Yelp Full	0.498009	SBERT (Fine-tune) + NN + **Distributional-Hypothesis-based method (fine-tuning)**
Yelp Polarity	0.898317	SBERT (Fine-tune) + NN

Table 6.16 shows the results across the evaluated drift scenarios. There are statistically significant differences in the Class Shift and Time-slice Removal sce-

narios. In all scenarios, the winner was the version *without* the **Distributional-Hypothesis-based method (fine-tuning)**.

Table 6.16: Statistical comparison using the Wilcoxon signed-rank test between SBERT (Fine-tune) + NN and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tune)** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.275832	SBERT (Fine-tune) + NN
Class Swap	0.696425	SBERT (Fine-tune) + NN
Class Shift	0.018555	SBERT (Fine-tune) + NN
Time Slice Removal	0.037109	SBERT (Fine-tune) + NN
Adjective Swap	0.076721	SBERT (Fine-tune) + NN

6.2.8 SBERT (Fine-tune) + NN vs. SBERT (Fine-tune) + NN + Distributed-Hypothesis-based method (fully incr.)

Table 6.17 shows the statistical comparisons to evaluate the improvements of using the **Distributed-Hypothesis-based method (fully incr.)** with the SBERT subject to the fine-tuning process. The only statistically significant difference occurred when using the Yelp dataset.

Table 6.17: Statistical comparison using the Wilcoxon signed-rank test between SBERT (Fine-tune) + NN and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fully incr.)** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.299612	SBERT (Fine-tune) + NN
Amazon Polarity	0.756166	SBERT (Fine-tune) + NN
Amazon Review	0.089695	SBERT (Fine-tune) + NN
Yelp	0.015973	SBERT (Fine-tune) + NN
Yelp Full	0.647655	SBERT (Fine-tune) + NN + **Distr.-Hypothesis-based method (fully incremental)**
Yelp Polarity	0.927279	SBERT (Fine-tune) + NN

Table 6.18 shows the statistical results obtained across the scenarios. It is noticeable that SBERT (Fine-tune) + NN reached the highest mean Macro F1-Scores, but only in the Class Shift scenario the difference is statistically significant.

Table 6.18: Statistical comparison using the Wilcoxon signed-rank test between SBERT (Fine-tune) + NN and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fully incr.)** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.063556	SBERT (Fine-tune) + NN
Class Swap	0.776569	SBERT (Fine-tune) + NN
Class Shift	0.040490	SBERT (Fine-tune) + NN
Time Slice Removal	0.130859	SBERT (Fine-tune) + NN
Adjective Swap	0.198076	SBERT (Fine-tune) + NN

6.2.9 SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tune)** vs. SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fully incr.)**

Finally, to statistically assess the difference between using the proposed **Distributed-Hypothesis-based method** in both incremental and fine-tune versions. Table 6.19 shows the results, and it is possible to see that leveraging the *fine-tune* variation of the **Distributed-Hypothesis-based method** led to better Macro F1-scores in 4 out of 6 times, although both methods were statistically equivalent.

Table 6.19: Statistical comparison using the Wilcoxon signed-rank test between SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tune)** and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fully incr.)** across the datasets. Green values highlight statistically different results.

Dataset	p-value	Winner
Airbnb	0.615043	SBERT (Fine-tune) + NN + **D.-H.-based (incr.)**
Amazon Polarity	0.261099	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**
Amazon Review	0.277355	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**
Yelp	0.366585	SBERT (Fine-tune) + NN + **D.-H.-based (incr.)**
Yelp Full	0.570597	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**
Yelp Polarity	0.869488	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**

Table 6.20 displays the results considering the scenarios. Similarly to the results verified across the datasets, no statistical difference was verified across the scenarios. However, the proposed **Distributed-Hypothesis-based method**, in its version executed after a fine-tuning is triggered, obtained the highest Macro F1-Scores on average in 4 out of 5 scenarios.

Table 6.20: Statistical comparison using the Wilcoxon signed-rank test between SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tune)** and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fully incr.)** across the proposed scenarios. Green values highlight statistically different results.

Scenario	p-value	Winner
Without drift	0.151887	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**
Class Swap	0.983834	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**
Class Shift	0.761065	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**
Time Slice Removal	0.193359	SBERT (Fine-tune) + NN + **D.-H.-based (incr.)**
Adjective Swap	0.745655	SBERT (Fine-tune) + NN + **D.-H.-based (fine-tuning)**

6.3 Ablation Study

To perform the ablation study, we selected the Yelp dataset. This choice was due to the fact that the models could obtain statistically different results in terms of Macro F1-Scores only in 2 out of 8 comparisons. The values for vocabulary size, window size, aggregation method, and the limit of representations to be stored were varied. On the other hand, AdaNEN was used as a machine learning approach to evaluate the proposed **Distributed-Hypothesis-based method**. The parameters' possible values are defined as: (a) window size: [3, 5, and 10]; (b) aggregation method: [average, damp, and vanishing]; (c) representation limit: [10, 50, and 100]; and (d) vocabulary size: [1000, 2500, and 5000].

Fig. 6.49 shows the plot between Macro F1-Scores and elapsed times. It is noticeable that window size = 10 leads to more concentrated elapsed times, although its median elapsed time is not the shortest: window size = 3 had the smallest median. In addition, window size = 3 had the highest median Macro F1-Score.

Fig. 6.50 shows a scatter plot similar to Fig. 6.49. However, Fig. 6.50 colors the scatter based on the aggregation method. It can be seen that using the *damp* aggregation leads to longer elapsed times, while providing slight improvements in Macro F1-Score.

Since the aggregation method *average* provided a fast elapsed time and a reasonable Macro F1-Score, the window size was fixed as 10 and the aggregation method as *average* to evaluate different parameters. Therefore, the vocabulary size was the first to be changed . Fig. 6.51 shows that using vocabulary size = 2500 leads to longer elapsed times, although it obtained the highest mean Macro F1-Score.

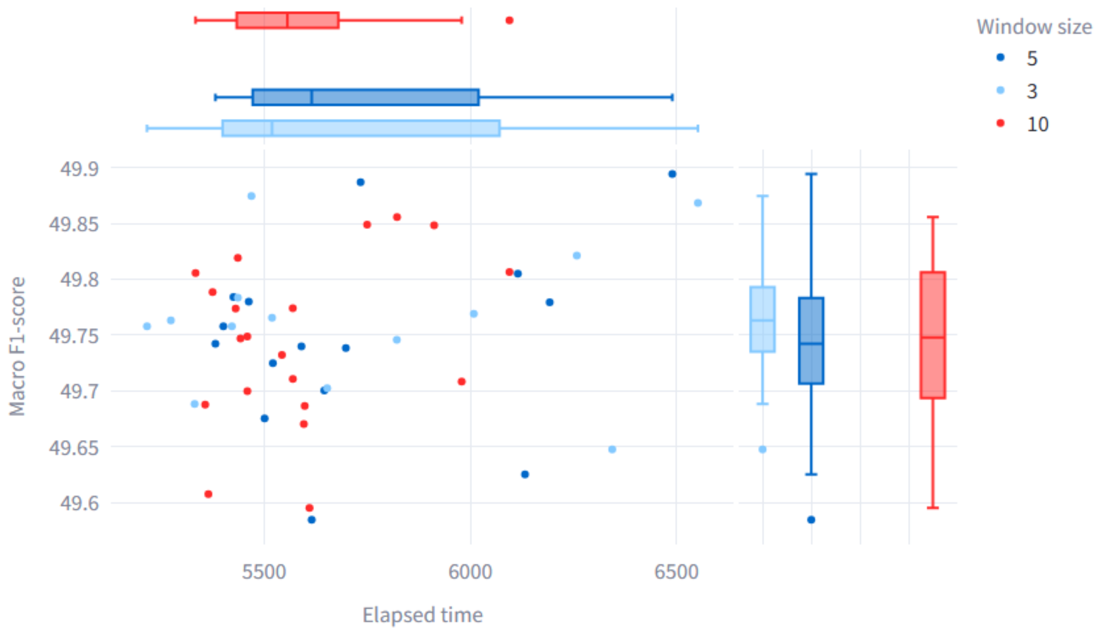

Figure 6.49: Information crossing between Macro F1-Scores and Elapsed times. The scatter is colored according to the window size.

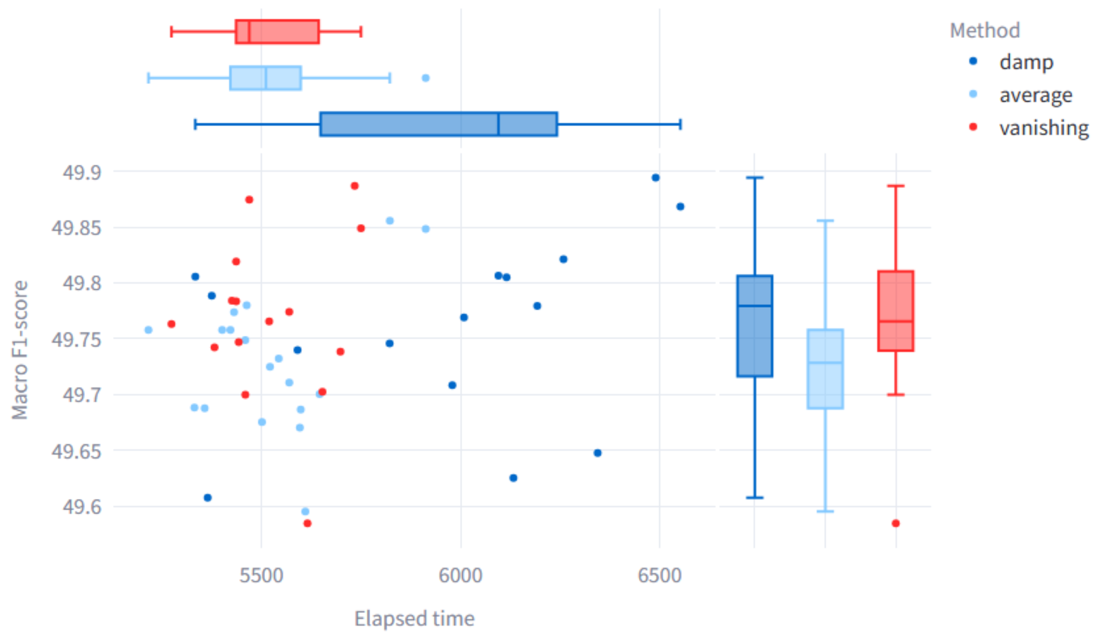

Figure 6.50: Information crossing between Macro F1-Scores and Elapsed times. The scatter is colored according to the aggregation method.

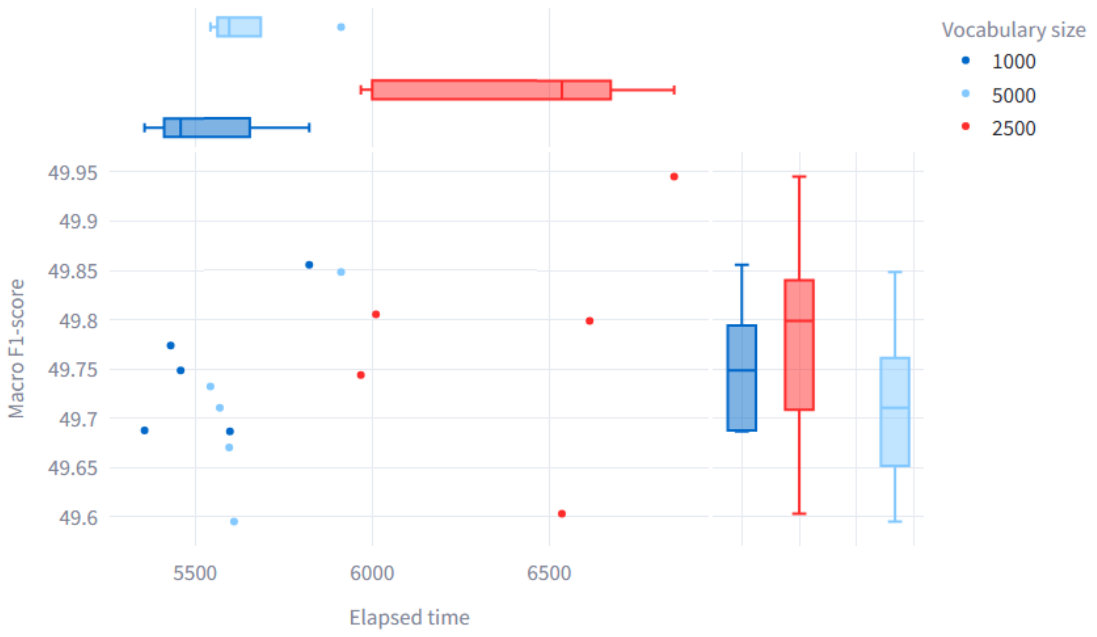

Figure 6.51: Information crossing between Macro F1-Scores and Elapsed times. The scatter is colored according to the vocabulary size.

Finally, considering the usage of vocabulary size = 1000 led to shorter elapsed times than using vocabulary size = 2500 and a similar mean Macro F1-Score. Therefore, the vocabulary size was set to 1000 and varied the limit number of representations per token. Fig. 6.52 graphically shows the variation of the representation limit.

Analyzing Fig. 6.52, the average values obtained are similar in terms of Macro F1-Score and elapsed times. However, using the representation limit = 5 leads to the highest mean of Macro F1-Score, and the shortest mean elapsed time. Therefore, it is possible to conclude that the representation limit impacts the elapsed time and Macro F1-Scores very little. However, having such a small representation limit, i.e., 5, as the best value may signify that keeping a bigger number of representations may lead to lower-quality embeddings.

6.4 A closer look at Bi-SBERT's results

Since Hypothesis 2 regarded the use of a pool of two SBERTs, an analysis of the Bi-SBERT's results is performed. The ranks are calculated over the 26 scenarios and are displayed in Fig. 6.53. Note that, on average, **Bi-SBERT** + NN was the approach that reached rank 1 the most frequently, i.e., 8 out of 26

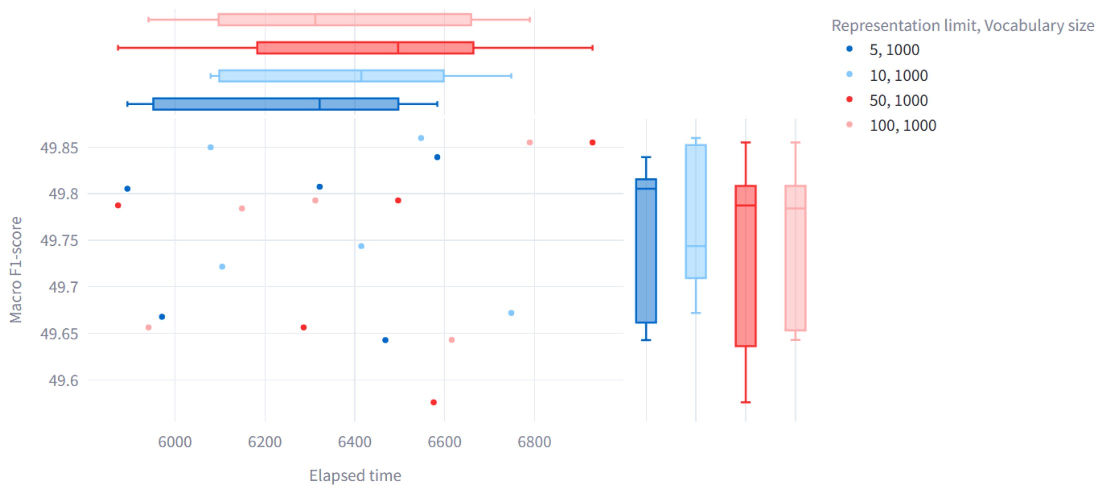

Figure 6.52: Information crossing between Macro F1-Scores and Elapsed times. Variations of the representation limit are compared, maintaining the vocabulary size = 1000.

scenarios. This consists of a win by a large margin, since the second that ranked 1 most frequently was SBERT(Fine-tune) + NN and SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**.

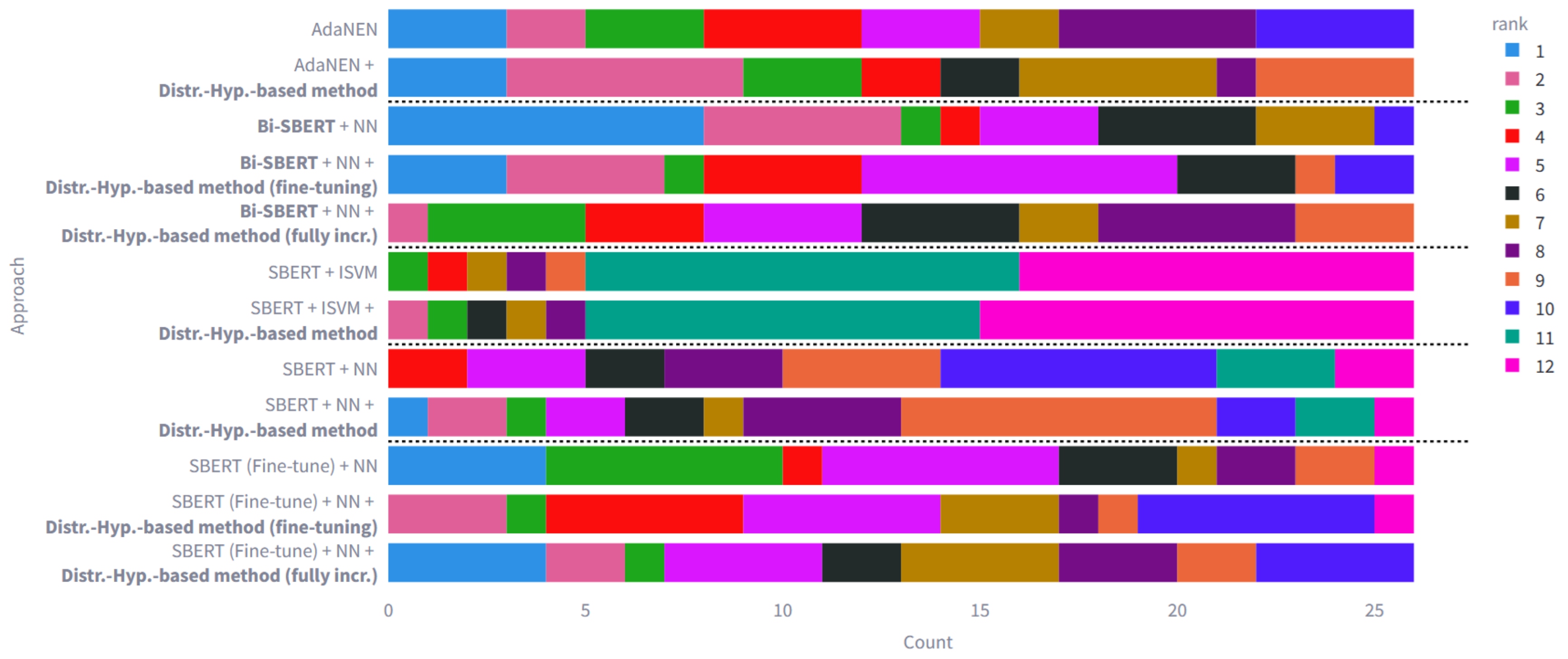

Figure 6.53: Ranks of the approaches evaluated in this thesis across all the 26 scenarios presented in Chapter 5.8.

In addition, calculating the average rank of the methods across the experiments leads to important observations. Table 6.21 shows the calculated average ranks per approach. **Bi-SBERT** + NN obtained the best rank, and the same method coupled with the **Distr.-Hyp.-based method (fine-tuning)** reached the second best. Other observations are reinforced, such as the AdaNEN + **Distr.-Hyp.-based method** with a higher rank than its vanilla version. The same happens for SBERT + NN and SBERT + ISVM. Apart from **Bi-SBERT**, only SBERT (Fine-tune) + NN obtained a higher rank than its version with the **Distr.-Hyp.-based method**.

Table 6.21: Average rank per approach. The lower the rank, the better.

Approach	Average rank
Bi-SBERT + NN	3.653846
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	4.500000
AdaNEN + **Distr.-Hyp.-based method**	4.730769
SBERT (Fine-tune) + NN	4.884615
AdaNEN	5.423077
Bi-SBERT + NN + Distr.-Hyp.-based method (fully incr.)	5.807692
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	5.884615
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	6.307692
SBERT + NN + **Distr.-Hyp.-based method**	7.500000
SBERT + NN	8.538462
SBERT + ISVM + **Distr.-Hyp.-based method**	10.307692
SBERT + ISVM	10.461538

6.5 Concluding Remarks

Considering the comparison between a *vanilla* approach and the approach coupled with the **Distr.-Hyp.-based method**, there are 42 comparisons. Of those 42 comparisons, leveraging the **Distr.-Hyp.-based method** led to a higher mean Macro F1-Score in 19 opportunities. Resorting to statistically significant differences, leveraging the proposed **Distr.-Hyp.-based method** resulted in 8 wins, 33 equivalences and only one lower statistically significant result.

Fig. 6.54 shows the behavior and impact of using the **Distributed-Hypothesis-based method** according to the dataset. The dashed line is 3.5, half the number of comparisons. Using the **Distributed-Hypothesis-based method** led to better results in the Amazon Polarity and Yelp Full datasets, while Yelp and Amazon Reviews had the smallest impact, resulting in poor results.

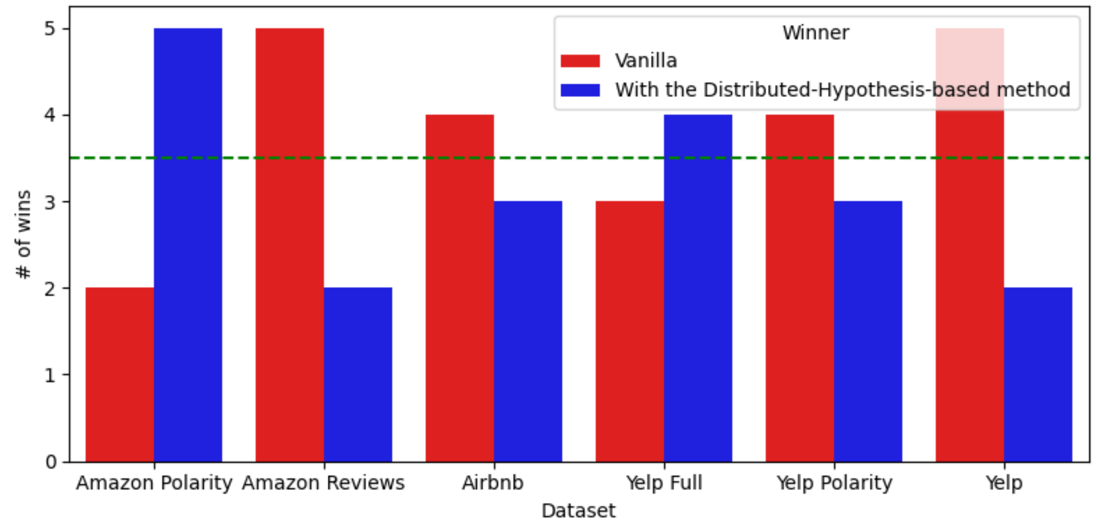

Figure 6.54: Number of wins and losses comparing averages of Macro F1-scores between methods with and without the Distributed-Hypothesis-based method, regarding the statistical evaluation only. The dashed line corresponds to the average number of runs.

Recalling Table 5.2, an investigation on the characteristics of the datasets that might have impacted the performance of the **Distributed-Hypothesis-based method** was conducted. A correlation study indicated that most characteristics obtained from the datasets have little Pearson correlation. However, among the calculated values, the highest correlations regard the average Wordpiece-Token ratio (32.51%), and Unknown ratio (29.88%). Although these are weak positive correlations, they indicate that methods together with the Distributed-Hypothesis-based method can lead to better results when applied to datasets with higher rates of unknown tokens. Please note that *unknown* tokens are tokens not existent in the language model's vocabulary. Despite these observations, further investigation is demanded to draw a more robust conclusion.

Regarding the **Bi-SBERT** approaches, it was noticed through the evaluations that its capacity of reaching good performance in terms of Macro F1-Scores. **Bi-SBERT** + NN reached the best rank in 8 out of 26 scenarios. Besides, the **Bi-SBERT** + NN + **Distr.-Hyp.-based method** reached rank #1 in 3 out of 26 scenarios, demonstrating its ability to smooth the effects of changes in textual data streams subject to concept drift. It can be observed that **Bi-SBERT** obtained a higher average rank compared to AdaNEN, a state-of-the-art method.

Considering the ablation study, it was possible to verify that **Distr.-Hyp.-**

based method performs similar across the settings, with the biggest changes occurring in terms of elapsed times. Still, the best combination of parameters is: window size = 10, vocabulary size = 1000, and representation limit = 5.

7

Conclusion, Limitations, and Future Works

This thesis approached the problem of concept drift adaptation in textual data streams. For the development of this thesis, several aspects of textual data streams were studied, from the basic characteristics to the methods that are applied to textual streams subject to concept drift, through a systematic review of the literature.

Some parts of the study resulted in seven papers thus far:

1. The performance and quality of the embeddings generated by representation methods that could be leveraged in text streams were evaluated (THUMA et al., 2023);

2. As previously mentioned, a systematic review on concept drift adaptation in text stream mining was developed, providing a better understanding of the scenario, the sort of applications, textual drift detection methods, textual drift types, and so on (GARCIA et al., 2025);

3. Scenarios subject to fast drifts were studied, and sentiment drifts could be verified within time intervals shorter than a few days, expanding to a few minutes and hours (GARCIA; Britto Jr; BARDDAL, 2023);

4. A limited graph to evaluate streams and detect groups periodically to analyze hashtags co-occurrence was developed, finding out that a given hashtag, i.e., #mybodymychoice, has been undergoing a drift at least since 2018, not only related to COVID vaccination, but also to drug liberation (GARCIA; BRITTO Jr; BARDDAL, 2024);

5. A textual sampling method was proposed and evaluated combined with loss functions for SentenceBERT models, showing the sampling method's efficiency (GARCIA et al., 2025);

6. In addition, most (3 out of 4) of the textual drift scenarios were proposed in this thesis, to enable experimentation and evaluation of the methods proposed in this thesis (GARCIA et al., 2024);
7. Finally, the impact of updating language models through fine-tuning in pre-defined periods was evaluated, compared to maintaining a static model, and to pre-trained language models fine-tuned using only texts from the last period analyzed, showing that fine-tuning periodically can be more beneficial than using a "new" pre-trained language model fine-tuned with only recent data (IMAI et al., 2024).

During the development of the systematic review of the literature, one of the gaps detected was the lack of representation update methods. To bridge this gap, in this thesis, a method directed to its use in textual data streams was proposed. In addition, a method to overcome concept drift in textual data streams by joining representations provided by a pair of SentenceBERT models, updated alternately after each drift detection, was proposed.

The Research Question of this thesis was defined in Chapter 1.1 as: "Does the distributional hypothesis help reach concept drift adaptation for BERT-based language models in textual data stream classification scenarios?". Two hypotheses were defined in Section 1.3: **(Hypothesis 1)** A distributional-hypothesis-based method overcomes the effects of concept drift by allowing higher quality representations, and **(Hypothesis 2)** Combining a pool of two language models to generate vector representations mitigates the effects of concept drift by preventing abrupt model updates (using fine-tuning). Regarding **Hypothesis 1**, the method proposed in this thesis obtained good results in the adaptation in scenarios of textual concept drift, providing better representations, especially in situations with a higher rate of unknown words (from the language model's point of view). However, as can be seen in Appendix C, there is room for improvement in the elapsed times.

Considering **Hypothesis 2**, i.e., "Combining a pool of two language models to generate vector representations mitigates the effects of concept drift by preventing abrupt model updates (using fine-tuning)", the **Bi-SBERT** + NN obtained the best Macro F1-Score in 8 out of 26 scenarios (Table 5.6) of evaluation, corresponding to 30.77% of the scenarios. Besides, the version with the **Distributional-Hypothesis-based method (fine-tuning)** reached the second-best average rank, and reached rank 1 in 3 out of 26 scenarios, which is around 11.54%. Together, both reached the best Macro F1-Scores in around 42% of the scenarios. It is important to note that **Bi-SBERT** + NN obtained a higher average rank than AdaNEN, a state-of-the-art method.

Recalling the specific objectives defined in Section 1.2, they are: (a) To perform a systematic literature review to understand the state-of-the-art on concept drift adaptation in textual data stream settings; (b) To execute experiments using pre-trained and online learning models to determine a baseline regarding textual data stream classification; (c) To develop methods based on distributional hypothesis to complement a pre-trained model, and based on the combination of representations calculated over time in continuous processing of textual data streams settings; and (d) To evaluate the proposed methods and compare them against different methods, including AdaNEN, a state-of-the-art method. The specific objective (a) was accomplished, and the resulting paper was published in 2025 (GARCIA et al., 2025). For objective (b), several experiments were performed, resulting in papers such as Thuma et al. (2023) and Garcia et al. (2025). At that moment, Incremental SVM was defined as the baseline. Regarding objectives (c) and (d), two methods were developed, described, and evaluated in this thesis: (i) the **distributional-hypothesis-based method**, and (ii) Bi-SBERT. **Bi-SBERT** obtained a higher rank than AdaNEN, and the distributional-hypothesis-based method could improve the results of most methods, including AdaNEN, which reached a better rank than *vanilla* AdaNEN. An important point about the proposed methods is that both are agnostic to the language model, which means that the methods can be used to extend/generate new embeddings from any language model that generates embeddings.

Although the methods provided improvements in a number of scenarios, there are a few limitations that are worth mentioning: (a) only one embedding model was used in the experiments; (b) only five executions were performed per setting; (c) some of the drift scenarios proposed and used in this thesis may not be realistic; and (d) for the **Bi-SBERT**, in its fine-tuning procedure, only SVM was tested to check performance improvements across the fine-tuning epochs. Besides, as stated in the systematic review and corroborated in Ghahramanian et al. (2024), neural approaches are typically overlooked in textual stream scenarios. Taking this into account, only two types of neural approaches, AdaNEN (GHAHRAMANIAN et al., 2024), and a generic neural network, described in Chapter 5.7, were evaluated. Therefore, it appears that there remains potential for improvement in this regard, as the generic neural network's architecture was not optimized through any systematic optimization procedure, such as grid search, for instance. Regarding the limitation (a), more recent models, such as ModernBERT (WARNER et al., 2025), could be evaluated in the proposed scenarios. In addition, the authors in Warner et al. (2025) mention that "encoder-only models remain widely used in a variety of non-generative downstream applications", mostly due to their ability to provide quality while being smaller than decoder-only models.

Considering the concept drift detection strategy, although the use of detectors other than ADWIN was documented, preliminary experiments with EDDM, DDM, and HDDM did not show improvements, with a high number of false positives, which slows the complete process due to the excessive triggering of fine-tuning processes in the methods that have it. In Garcia, Britto Jr & Barddal (2023), experiments with different drift detectors were assessed, and ADWIN performed the best. Taking the results into account, ADWIN was employed in the scenarios in this thesis.

As future works, it is intended to: (a) study and propose new, more realistic drift scenarios specific for texts; (b) evaluate the impacts of the proposed method in scenarios such as online clustering; (c) evaluate impacts on quality in embeddings provided by different language models, such as ModernBERT; (d) study and propose similar, but non-iterative methods, leveraging matrix multiplication in order to reduce the running time; and (e) study and propose neural architectures for text streaming scenarios.

References

ATTENBERG, J.; WEINBERGER, K.; DASGUPTA, A.; SMOLA, A.; ZINKEVICH, M. Collaborative Email-spam Filtering with the Hashing Trick. In: *Proceedings of the Sixth Conference on Email and Anti-spam*. [S.l.: s.n.], 2009.

BAENA-GARCIA, M.; CAMPO-ÁVILA, J. del; FIDALGO, R.; BIFET, A.; GAVALDA, R.; MORALES-BUENO, R. Early Drift Detection Method. In: CITESEER. *Fourth International Workshop on Knowledge Discovery from Data Streams*. [S.l.], 2006. v. 6, p. 77–86.

BARDDAL, J. P.; GOMES, H. M.; ENEMBRECK, F.; PFAHRINGER, B. A Survey on Feature Drift Adaptation: Definition, Benchmark, Challenges and Future Directions. *Journal of Systems and Software*, Elsevier, v. 127, p. 278–294, 2017.

BARROS, R. S.; CABRAL, D. R.; JR, P. M. G.; SANTOS, S. G. RDDM: Reactive Drift Detection Method. *Expert Systems with Applications*, Elsevier, v. 90, p. 344–355, 2017.

BARROS, R. S. M.; SANTOS, S. G. T. C. A Large-scale Comparison of Concept Drift Detectors. *Information Sciences*, Elsevier, v. 451, p. 348–370, 2018.

BEHNAMGHADER, P.; ADLAKHA, V.; MOSBACH, M.; BAHDANAU, D.; CHAPADOS, N.; REDDY, S. LLM2Vec: Large language models are secretly powerful text encoders. *Conference on Language Modeling (COLM)*, 2024.

BELOTTI, F.; BIANCHI, F.; PALMONARI, M. UNIMIB@ DIACR-Ita: Aligning Distributional Embeddings with a Compass for Semantic Change Detection in the Italian Language. *EVALITA Evaluation of NLP and Speech Tools for Italian-December 17th, 2020*, p. 451, 2020.

BIFET, A. Classifier Concept Drift Detection and the Illusion of Progress. In: SPRINGER. *Artificial Intelligence and Soft Computing: 16th International Conference, ICAISC 2017, Zakopane, Poland, June 11-15, 2017, Proceedings, Part II 16*. [S.l.], 2017. p. 715–725.

BIFET, A.; GAVALDA, R. Learning from Time-changing Data with Adaptive Windowing. In: SIAM. *Proceedings of the 2007 SIAM International Conference on Data Mining*. [S.l.], 2007. p. 443–448.

BIFET, A.; GAVALDA, R.; HOLMES, G.; PFAHRINGER, B. *Machine Learning for Data Streams: with Practical Examples in MOA*. [S.l.]: MIT Press, 2018.

BLEI, D. M. Probabilistic topic models. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 4, p. 77–84, 2012.

BLOOMBERG, L. *Language*. [S.l.]: George Allen & Unwin, 1933.

BOWMAN, S. R.; ANGELI, G.; POTTS, C.; MANNING, C. D. A Large Annotated Corpus for Learning Natural Language Inference. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. [S.l.: s.n.], 2015.

BRAVO-MARQUEZ, F.; KHANCHANDANI, A.; PFAHRINGER, B. Incremental word vectors for time-evolving sentiment lexicon induction. *Cognitive Computation*, v. 14, n. 1, p. 425–441, 2022.

BRAVO-MARQUEZ, F.; KHANCHANDANI, A.; PFAHRINGER, B. Incremental word vectors for time-evolving sentiment lexicon induction. *Cognitive Computation*, Springer, p. 1–17, 2022.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.

DODGE, J.; ILHARCO, G.; SCHWARTZ, R.; FARHADI, A.; HAJISHIRZI, H.; SMITH, N. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

DUSART, A.; PINEL-SAUVAGNAT, K.; HUBERT, G. TSSuBERT: Tweet Stream Summarization using BERT. *arXiv preprint arXiv:2106.08770*, 2021.

FARINA, M.; AHMAD, U.; TAHA, A.; YOUNES, H.; MESBAH, Y.; YU, X.; PEDRYCZ, W. Sparsity in Transformers: A Systematic Literature Review. *Neurocomputing*, Elsevier, v. 582, p. 127468, 2024.

FENG, X.; HE, X.; WANG, C.; WANG, C.; ZHANG, J. Towards a unified analysis of kernel-based methods under covariate shift. In: *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*. [S.l.: s.n.], 2024. v. 36, p. 73839–73851.

FRIAS-BLANCO, I.; CAMPO-ÁVILA, J. del; RAMOS-JIMENEZ, G.; MORALES-BUENO, R.; ORTIZ-DÍAZ, A.; CABALLERO-MOTA, Y. Online and Non-parametric Drift Detection Methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 27, n. 3, p. 810–823, 2014.

FU, C.-L.; CHEN, Z.-C.; LEE, Y.-R.; LEE, H.-y. AdapterBias: Parameter-efficient Token-dependent Representation Shift for Adapters in NLP Tasks. *arXiv preprint arXiv:2205.00305*, 2022.

GAMA, J.; FERNANDES, R.; ROCHA, R. Decision Trees for Mining Data Streams. *Intelligent Data Analysis*, IOS Press, v. 10, n. 1, p. 23–45, 2006.

GAMA, J.; ŽLIOBAITĖ, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A Survey on Concept Drift Adaptation. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 46, n. 4, p. 1–37, 2014.

GAO, Z.; FENG, A.; SONG, X.; WU, X. Target-dependent Sentiment Classification with BERT. *IEEE Access*, IEEE, v. 7, p. 154290–154299, 2019.

GARCIA, C.; LEITE, D.; ŠKRJANC, I. Incremental Missing-data Imputation for Evolving Fuzzy Granular Prediction. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 28, n. 10, p. 2348–2362, 2019.

GARCIA, C. M.; ABILIO, R. S.; KOERICH, A. L.; Britto Jr, A. d. S.; BARDDAL, J. P. Concept drift adaptation in text stream mining settings: A systematic review. *ACM Transactions on Intelligent Systems and Technology*, 2025.

GARCIA, C. M.; Britto Jr, A. d. S.; BARDDAL, J. P. Event-driven sentiment drift analysis in text streams: An application in a soccer match. In: IEEE. *2023 International Conference on Machine Learning and Applications (ICMLA)*. [S.l.], 2023. p. 1920–1927.

GARCIA, C. M.; BRITTO Jr, A. d. S.; BARDDAL, J. P. Temporal analysis of drifting hashtags in textual data streams: A graph-based application. *Expert Systems with Applications*, Elsevier, v. 257, p. 125007, 2024.

GARCIA, C. M.; KOERICH, A. L.; Britto Jr, A. d. S.; BARDDAL, J. P. Methods for generating drift in text streams. *arXiv preprint arXiv:2403.12328*, 2024.

GARCIA, C. M.; KOERICH, A. L.; Britto Jr, A. d. S.; BARDDAL, J. P. Improving sampling methods for fine-tuning sentencebert in text streams. In: SPRINGER. *International Conference on Pattern Recognition (ICPR)*. [S.l.], 2025. p. 445–459.

GHAHRAMANIAN, P.; BAKHSHI, S.; BONAB, H.; CAN, F. A novel neural ensemble architecture for on-the-fly classification of evolving text streams. *ACM Transactions on Knowledge Discovery from Data*, v. 18, n. 4, p. 1–24, 2024.

GOMES, H. M.; BIFET, A.; READ, J.; BARDDAL, J. P.; ENEMBRECK, F.; PFHARINGER, B.; HOLMES, G.; ABDESSALEM, T. Adaptive Random Forests for Evolving Data Stream Classification. *Machine Learning*, Springer, v. 106, p. 1469–1495, 2017.

HAMILTON, W. L.; LESKOVEC, J.; JURAFSKY, D. Cultural Shift or Linguistic Drift? Comparing Two Computational Measures of Semantic Change. In: NIH PUBLIC ACCESS. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. [S.l.], 2016. v. 2016, p. 2116.

HAMILTON, W. L.; LESKOVEC, J.; JURAFSKY, D. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. [S.l.: s.n.], 2016.

HARRIS, Z. S. Distributional Structure. *Word*, Taylor & Francis, v. 10, n. 2-3, p. 146–162, 1954.

HEUSINGER, M.; RAAB, C.; SCHLEIF, F.-M. Analyzing Dynamic Social Media Data via Random Projection - a New Challenge for Stream Classifiers. In: IEEE. *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. [S.l.], 2020. p. 1–8.

HEUSINGER, M.; RAAB, C.; SCHLEIF, F.-M. Analyzing dynamic social media data via random projection-a new challenge for stream classifiers. In: IEEE. *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. [S.l.], 2020. p. 1–8.

HINTON, G. *RMSProp: Divide the gradient by a running average of its recent magnitude*. 2012. 26–31 p. Disponível em: <https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>.

HOANG, M.; BIHORAC, O. A.; ROUCES, J. Aspect-based Sentiment Analysis using BERT. In: *Proceedings of the 22nd Nordic Conference on Computational Linguistics*. [S.l.: s.n.], 2019. p. 187–196.

HOMBAIAH, S. A.; CHEN, T.; ZHANG, M.; BENDERSKY, M.; NAJORK, M. Dynamic Language Models for Continuously Evolving Content. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2021. p. 2514–2524.

IMAI, B. Y. L.; GARCIA, C. M.; ROCHA, M. V.; KOERICH, A. L.; Britto Jr, A. d. S. de; BARDDAL, J. P. Is it fine to tune? evaluating sentencebert fine-tuning for brazilian portuguese text stream classification. In: IEEE. *IEEE International Conference on Big Data*. [S.l.], 2024.

JAMSHIDI, S.; MOHAMMADI, M.; BAGHERI, S.; NAJAFABADI, H. E.; REZVANIAN, A.; GHEISARI, M.; GHADERZADEH, M.; SHAHABI, A. S.; WU, Z. Effective text classification using bert, mtm lstm, and dt. *Data & Knowledge Engineering*, Elsevier, v. 151, p. 102306, 2024.

JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; MIKOLOV, T. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

JOULIN, A.; GRAVE, E.; BOJANOWSKI, P.; DOUZE, M.; JÉGOU, H.; MIKOLOV, T. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. [S.l.: s.n.], 2008.

KITCHENHAM, B.; CHARTERS, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering. *Keele University and Durham University Joint Report*, Citeseer, 2007.

KOJIMA, T.; GU, S. S.; REID, M.; MATSUO, Y.; IWASAWA, Y. Large language models are zero-shot reasoners. In: *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*. [S.l.: s.n.], 2022. v. 35, p. 22199–22213.

KUTUZOV, A.; ØVRELID, L.; SZYMANSKI, T.; VELLDAL, E. Diachronic Word Embeddings and Semantic Shifts: a Survey. *arXiv preprint arXiv:1806.03537*, 2018.

LEE, C.; CHO, K.; KANG, W. Mixout: Effective regularization to finetune large-scale pretrained language models. *arXiv preprint arXiv:1909.11299*, 2019.

LEE, J.; TANG, R.; LIN, J. What Would Elsa do? Freezing Layers during Transformer Fine-tuning. *arXiv preprint arXiv:1911.03090*, 2019.

LEITE, D.; BALLINI, R.; COSTA, P.; GOMIDE, F. Evolving Fuzzy Granular Modeling from Nonstationary Fuzzy Data Streams. *Evolving Systems*, Springer, v. 3, p. 65–79, 2012.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

MAY, C.; DUH, K.; DURME, B. V.; LALL, A. Streaming Word Embeddings with the Space-saving Algorithm. *arXiv preprint arXiv:1704.07463*, 2017.

MELLO, R. F. de; RIOS, R. A.; PAGLIOSA, P. A.; LOPES, C. S. Concept Drift Detection on Social Network Data using Cross-recurrence Quantification Analysis. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, AIP Publishing LLC, v. 28, n. 8, p. 085719, 2018.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; DEAN, J. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, v. 26, 2013.

MISRA, J.; GRIES, D. Finding repeated elements. *Science of Computer Programming*, Elsevier, v. 2, n. 2, p. 143–152, 1982.

NIELSEN, F. On the Jensen–Shannon Symmetrization of Distances Relying on Abstract Means. *Entropy*, MDPI, v. 21, n. 5, p. 485, 2019.

PAGE, E. S. Continuous Inspection Schemes. *Biometrika*, JSTOR, v. 41, n. 1/2, p. 100–115, 1954.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. GloVe: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543.

PETERS, M. E.; AMMAR, W.; BHAGAVATULA, C.; POWER, R. Semi-supervised Sequence Tagging with Bidirectional Language Models. *arXiv preprint arXiv:1705.00108*, 2017.

RABINOVICH, E.; VETZLER, M.; ACKERMAN, S.; ANABY-TAVOR, A. Reliable and interpretable drift detection in streams of short texts. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics: Industry Track (ACL 2023)*. [S.l.]: Association for Computational Linguistics, 2023. p. 438–446.

RAINA, V.; LIUSIE, A.; GALES, M. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. *arXiv preprint arXiv:2402.14016*, 2024.

REIMERS, N.; GUREVYCH, I. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019. Disponível em: <http://arxiv.org/abs/1908.10084>.

ROSS, G. J.; ADAMS, N. M.; TASOULIS, D. K.; HAND, D. J. Exponentially Weighted Moving Average Charts for Detecting Concept Drift. *Pattern Recognition Letters*, Elsevier, v. 33, n. 2, p. 191–198, 2012.

RYZHOVA, A.; RYZHOVA, D.; SOCHENKOV, I. Detection of Semantic Changes in Russian Nouns with Distributional Models and Grammatical Features. In: *Computational Linguistics and Intellectual Technologies: Papers from the Annual Conference Dialogue*. [S.l.: s.n.], 2021.

SÁ, J. M. C. de; SILVEIRA, M. D.; PRUSKI, C. Survey in Characterization of Semantic Change. *arXiv preprint arXiv:2402.19088*, 2024.

SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

SBERT.NET. *Cross-encoders*. 2022. Disponível em: <https://www.sbert.net/examples/applications/cross-encoder/README.html>.

SEBASTIÃO, R.; FERNANDES, J. M. Supporting the Page-Hinkley test with Empirical Mode Decomposition for Change Detection. In: SPRINGER. *Foundations of Intelligent Systems: 23rd International Symposium, ISMIS 2017, Warsaw, Poland, June 26-29, 2017, Proceedings 23*. [S.l.], 2017. p. 492–498.

STEWART, I.; ARENDT, D.; BELL, E.; VOLKOVA, S. Measuring, Predicting and Visualizing Short-term Change in Word Representation and Usage in VKontakte Social Network. In: *Eleventh International AAAI Conference on Web and Social Media*. [S.l.: s.n.], 2017.

SUBAKTI, A.; MURFI, H.; HARIADI, N. The Performance of BERT as Data Representation of Text Clustering. *Journal of Big Data*, SpringerOpen, v. 9, n. 1, p. 1–21, 2022.

SUPREM, A.; PU, C. ASSED: a Framework for Identifying Physical Events Through Adaptive Social Sensor Data Filtering. In: *Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems*. [S.l.: s.n.], 2019. p. 115–126.

TAHMASEBIA, N.; BORINA, L.; JATOWT, A. Survey of Computational Approaches to Lexical Semantic Change Detection. *Computational Approaches to Semantic Change*, Language Science Press, v. 6, p. 1, 2021.

TAI, W.; KUNG, H.; DONG, X. L.; COMITER, M.; KUO, C.-F. exBERT: Extending pre-trained models with domain-specific vocabulary under constrained training resources. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. [S.l.: s.n.], 2020. p. 1433–1439.

THUMA, B. S.; VARGAS, P. S. de; GARCIA, C.; Britto Jr, A. de S.; BARDDAL, J. P. Benchmarking Feature Extraction Techniques for Textual Data Stream Classification. In: *International Joint Conference on Neural Networks*. [S.l.: s.n.], 2023.

TOUVRON, H.; LAVRIL, T.; IZACARD, G.; MARTINET, X.; LACHAUX, M.-A.; LACROIX, T.; ROZIÈRE, B.; GOYAL, N.; HAMBRO, E.; AZHAR, F. et al. Llama: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*, 2023.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is All You Need. *Advances in Neural Information Processing Systems*, v. 30, 2017.

WANG, Y.; GONG, C.; JI, X.; YUAN, Q. Text classification for evaluating digital technology adoption maturity based on bert: An evidence of industrial ai from china. *Technological Forecasting and Social Change*, Elsevier, v. 211, p. 123903, 2025.

WANG, Z.; PANG, Y.; LIN, Y. Large language models are zero-shot text classifiers. *arXiv preprint arXiv:2312.01044*, 2023.

WARES, S.; ISAACS, J.; ELYAN, E. Data Stream Mining: Methods and Challenges for Handling Concept Drift. *SN Applied Sciences*, Springer, v. 1, p. 1–19, 2019.

WARNER, B.; CHAFFIN, A.; CLAVIÉ, B.; WELLER, O.; HALLSTRÖM, O.; TAGHADOUINI, S.; GALLAGHER, A.; BISWAS, R.; LADHAK, F.; AARSEN, T. et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In: *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [S.l.: s.n.], 2025. p. 2526–2547.

WILLIAMS, A.; NANGIA, N.; BOWMAN, S. R. A Broad-coverage Challenge Corpus for Sentence Understanding through Inference. *arXiv preprint arXiv:1704.05426*, 2017.

WOOLSON, R. F. Wilcoxon signed-rank test. *Encyclopedia of Biostatistics*, Wiley Online Library, v. 8, 2005.

WU, Y.; SCHUSTER, M.; CHEN, Z.; LE, Q. V.; NOROUZI, M.; MACHEREY, W.; KRIKUN, M.; CAO, Y.; GAO, Q.; MACHEREY, K. et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

YANG, S.; HUANG, G.; ZHOU, X.; MAK, V.; YEARWOOD, J. EWNStream +: Effective and Real-time Clustering of Short Text Streams using Evolutionary Word Relation Network. *International Journal of Information Technology & Decision Making*, World Scientific, v. 20, n. 01, p. 341–370, 2021.

YAO, Y.; ROSASCO, L.; CAPONNETTO, A. On early stopping in gradient descent learning. *Constructive Approximation*, Springer, v. 26, n. 2, p. 289–315, 2007.

ZHANG, T.; WU, F.; KATIYAR, A.; WEINBERGER, K. Q.; ARTZI, Y. Revisiting few-sample BERT fine-tuning. *arXiv preprint arXiv:2006.05987*, 2020.

ZHANG, X.; ZHAO, J.; LECUN, Y. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, v. 28, 2015.

ZHAO, E.; LIU, A.; ANANDKUMAR, A.; YUE, Y. Active learning under label shift. In: *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)*. [S.l.]: PMLR, 2021. (Proceedings of Machine Learning Research, v. 130), p. 3412–3420.

ZHOU, B.; ZOU, L.; MOSTAFAVI, A.; LIN, B.; YANG, M.; GHARAIBEH, N.; CAI, H.; ABEDIN, J.; MANDAL, D. VictimFinder: Harvesting rescue requests in disaster response from social media with BERT. *Computers, Environment and Urban Systems*, Elsevier, v. 95, p. 101824, 2022.

A

Byproduct datasets

During the development of this thesis, two datasets, of different domains, have been collected, treated, and leveraged: (a) a dataset regarding tweets during a soccer match; and (b) a dataset regarding co-occurrence of hashtags in the domain of the hashtag *#mybodymychoice*.

More specifically, in Garcia, Britto Jr & Barddal (2023), a number of tweets (posts from the X social network, former Twitter) was collected about a soccer match between Sport Club Internacional (Brazil) and Club Social y Deportivo Colo Colo (Chile) in the Sudamericana Cup 2022. The byproduct dataset contains labeled drifts, which is rare for textual datasets. Paper (GARCIA; Britto Jr; BARDDAL, 2023) aimed to detect sentiment drifts in a short window (the duration of a soccer match). The sentiments were calculated using a lexicon-based classifier, in which each word was indicated as positive, negative, or neutral. Besides, Incremental Word-Vectors (IWV) (BRAVO-MARQUEZ; KHANCHANDANI; PFAHRINGER, 2022b) was leveraged to monitor the most important words related to selected keywords, e.g., *@scinternacional* and *inter*. Using IWV allowed tracing back the drift-generator event. In this paper, ADWIN was the most precise concept drift detector, providing the fewest false alarms and the closest detections compared to the actual event: around 2m21s. This byproduct dataset contains 37,125 rows and is available at <https://github.com/cristianomg10/sentiment-drift-analysis-text-stream-football>.

For the paper (GARCIA; BRITTO Jr; BARDDAL, 2024), a dataset with 255,131 valid tweets was collected, regarding the hashtag #mybodymychoice, but removing the re-posts, also known as retweets. The dataset comprises posts between

February 2018 and December 2022. In this paper, the drifts regarded the use of the aforementioned hashtag in different contexts other than its original, i.e., women's rights and bodily autonomy. An incremental graph was proposed, and the analysis was performed annually, using the resulting graph. The analysis leveraged the Girvan-Newman method to find hashtag communities, and thus, identify the contexts of use of the #mybodymychoice hashtag. Although it was identified that 2021 was the year with the most significant drift, the community detection allowed discovering that the hashtag was used in different communities during the entire analyzed period. The dataset is available at: <https://github.com/cristianomg10/temporal-analysis-of-drifting-hashtags-in-textual-data-streams-a-graph-based-application/>.

B

Tables with Macro F1-scores and elapsed times

This section shows the tables with Macro F1-scores and elapsed times collected across the experiments performed in this thesis.

Table B.1: Results for the Airbnb dataset, in the scenario without drifts. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	68.56 ± 0.30	777.43 ± 3.54
AdaNEN + **Distr.-Hyp.-based method**	68.59 ± 0.32	2,702.26 ± 71.55
Bi-SBERT + NN	68.86 ± 0.79	2,297.82 ± 105.77
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fine-tuning)	68.91 ± 0.66	3,441.14 ± 225.93
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fully incr.)	68.83 ± 0.77	5,154.02 ± 137.08
SBERT + ISVM	68.20 ± 0.16	622.85 ± 10.85
SBERT + ISVM + **Distr.-Hyp.-based method**	68.42 ± 0.26	2,422.66 ± 67.91
SBERT + NN	66.45 ± 0.47	1,193.72 ± 10.81
SBERT + NN + **Distr.-Hyp.-based method**	65.98 ± 0.52	3,937.24 ± 239.54
SBERT (Fine-tune) + NN	68.53 ± 0.66	1,431.97 ± 68.88
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning)	68.43 ± 0.70	1,244.48 ± 102.15
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fully incr.)	67.99 ± 0.89	2,730.19 ± 72.94

Table B.2: Results for the Airbnb dataset, in the scenario with Class Swap drift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	84.94 ± 0.22	773.75 ± 3.19
AdaNEN + **Distr.-Hyp.-based method**	84.89 ± 0.25	2,715.78 ± 117.84
Bi-SBERT + NN	84.22 ± 0.41	1,658.97 ± 566.46
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fine-tuning)	84.43 ± 0.72	2,975.34 ± 752.78
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fully incr.)	84.54 ± 0.33	5,314.98 ± 48.79
SBERT + ISVM	84.54 ± 0.07	624.22 ± 7.61
SBERT + ISVM + **Distr.-Hyp.-based method**	84.64 ± 0.13	2,408.65 ± 75.45
SBERT + NN	83.80 ± 0.38	1,202.97 ± 10.79
SBERT + NN + **Distr.-Hyp.-based method**	83.86 ± 0.23	3,826.86 ± 172.16
SBERT (Fine-tune) + NN	83.59 ± 0.57	1,486.56 ± 71.25
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning)	83.84 ± 0.33	825.34 ± 116.63
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fully incr.)	83.87 ± 0.35	2,565.45 ± 62.07

Table B.3: Results for the Airbnb dataset, in the scenario with Class Shift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	75.87 ± 0.33	775.64 ± 2.65
AdaNEN + **Distr.-Hyp.-based method**	75.92 ± 0.29	2,697.12 ± 53.77
Bi-SBERT + NN	76.09 ± 0.61	2,228.08 ± 94.21
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fine-tuning)	76.13 ± 0.91	3,247.28 ± 594.17
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fully incr.)	75.75 ± 0.36	4,949.32 ± 460.49
SBERT + ISVM	75.60 ± 0.30	617.10 ± 6.55
SBERT + ISVM + **Distr.-Hyp.-based method**	75.65 ± 0.27	2,396.87 ± 102.22
SBERT + NN	74.80 ± 0.28	1,214.85 ± 20.95
SBERT + NN + **Distr.-Hyp.-based method**	74.80 ± 0.60*	3,866.35 ± 133.41
SBERT (Fine-tune) + NN	75.77 ± 0.41	1,498.30 ± 60.89
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning)	74.48 ± 1.70	996.06 ± 165.61
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fully incr.)	75.14 ± 1.12	2,730.61 ± 85.23

Table B.4: Results for the Airbnb dataset, in the scenario with Time-slice Removal. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	68.55 ± 0.52	772.46 ± 4.28
AdaNEN + **Distr.-Hyp.-based method**	68.56 ± 0.57	2,765.02 ± 116.29
Bi-SBERT + NN	69.16 ± 0.77	2,330.43 ± 94.86
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fine-tuning)	69.15 ± 0.39	3,533.20 ± 208.72
Bi-SBERT + NN + **Distr.-Hyp.-based method** (fully incr.)	68.64 ± 1.02	5,312.15 ± 217.68
SBERT + ISVM	68.30 ± 0.72	617.47 ± 7.52
SBERT + ISVM + **Distr.-Hyp.-based method**	68.32 ± 0.70	2,368.15 ± 95.74
SBERT + NN	66.09 ± 0.29	1,224.94 ± 5.98
SBERT + NN + **Distr.-Hyp.-based method**	66.16 ± 0.31	3,693.80 ± 114.41
SBERT (Fine-tune) + NN	68.23 ± 0.93	1,397.82 ± 76.49
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fine-tuning)	67.14 ± 1.35	1,056.16 ± 205.71
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method** (fully incr.)	67.82 ± 0.88	2,636.62 ± 95.33

Table B.5: Results for the Airbnb dataset, in the scenario with Adjective Swap. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	62.96 ± 0.46	771.97 ± 4.56
AdaNEN + **Distr.-Hyp.-based method**	63.07 ± 0.29	3103.59 ± 139.38
Bi-SBERT + NN	62.32 ± 1.02	2390.51 ± 125.66
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	62.34 ± 1.06	3630.71 ± 140.16
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	62.77 ± 0.44	6081.00 ± 120.61
SBERT + ISVM	62.97 ± 0.18	618.06 ± 5.74
SBERT + ISVM + **Distr.-Hyp.-based method**	63.00 ± 0.25	2773.52 ± 106.06
SBERT + NN	60.62 ± 0.64	1181.10 ± 5.80
SBERT + NN + **Distr.-Hyp.-based method**	60.71 ± 0.49	4175.33 ± 168.29
SBERT (Fine-tune) + NN	62.09 ± 0.31	1458.86 ± 77.26
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	62.02 ± 1.14	1421.83 ± 120.73
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	62.13 ± 0.94	3168.93 ± 92.63

Table B.6: Results for the Amazon Polarity dataset, in the scenario without drift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	82.54 ± 0.06	839.61 ± 4.29
AdaNEN + **Distr.-Hyp.-based method**	82.57 ± 0.05	4268.55 ± 136.84
Bi-SBERT + NN	82.64* ± 0.07	1462.22 ± 16.28
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	82.64* ± 0.07	1896.42 ± 6.72
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	82.58 ± 0.09	5085.96 ± 69.48
SBERT + ISVM	79.95 ± 0.08	625.84 ± 8.98
SBERT + ISVM + **Distr.-Hyp.-based method**	79.99 ± 0.09	3877.81 ± 128.56
SBERT + NN	82.64 ± 0.07	1366.40 ± 16.92
SBERT + NN + **Distr.-Hyp.-based method**	82.68 ± 0.06	5829.91 ± 88.40
SBERT (Fine-tune) + NN	82.64 ± 0.07	1384.68 ± 19.95
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	82.64 ± 0.07	841.73 ± 3.59
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	82.66 ± 0.09	4108.01 ± 121.42

Table B.7: Results for the Amazon Polarity dataset, in the scenario of Class Swap. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	82.52* ± 0.07	846.67 ± 4.88
AdaNEN + **Distr.-Hyp.-based method**	82.52* ± 0.07	4301.84 ± 219.16
Bi-SBERT + NN	82.64* ± 0.05	1423.71 ± 17.75
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	82.64* ± 0.05	1849.25 ± 8.48
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	82.61 ± 0.07	5139.36 ± 102.28
SBERT + ISVM	79.72* ± 0.08	628.37 ± 9.26
SBERT + ISVM + **Distr.-Hyp.-based method**	79.72 ± 0.09	3828.28 ± 166.58
SBERT + NN	82.64 ± 0.05	1382.34 ± 22.63
SBERT + NN + **Distr.-Hyp.-based method**	82.70 ± 0.08	5789.58 ± 191.53
SBERT (Fine-tune) + NN	82.64 ± 0.05	1366.87 ± 12.92
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	82.64 ± 0.05	834.68 ± 2.47
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	82.71 ± 0.09	4165.71 ± 87.96

Table B.8: Results for the Amazon Polarity dataset, in the scenario of Class Shift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	82.54 ± 0.06	833.90 ± 2.71
AdaNEN + **Distr.-Hyp.-based method**	82.56 ± 0.05	4247.48 ± 163.26
Bi-SBERT + NN	82.64* ± 0.07	1349.62 ± 5.17
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	82.64* ± 0.07	1823.07 ± 8.87
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	82.58 ± 0.09	5255.87 ± 26.33
SBERT + ISVM	79.95 ± 0.08	597.77 ± 2.54
SBERT + ISVM + **Distr.-Hyp.-based method**	79.99 ± 0.09	3854.50 ± 86.50
SBERT + NN	82.64 ± 0.07	1324.91 ± 7.19
SBERT + NN + **Distr.-Hyp.-based method**	82.65 ± 0.05	5659.46 ± 49.53
SBERT (Fine-tune) + NN	82.64 ± 0.07	1310.10 ± 3.75
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	82.64 ± 0.07	837.81 ± 3.64
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	82.66 ± 0.09	4191.62 ± 88.44

Table B.9: Results for the Amazon Polarity dataset, in the scenario of Adjective Swap. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	77.50 ± 0.11	838.32 ± 5.29
AdaNEN + **Distr.-Hyp.-based method**	77.52 ± 0.11	4529.97 ± 162.83
Bi-SBERT + NN	80.73 ± 0.47	2337.18 ± 17.70
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	80.90 ± 0.49	3827.78 ± 39.29
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	80.87 ± 0.33	8021.25 ± 145.22
SBERT + ISVM	74.52 ± 0.11	640.14 ± 9.51
SBERT + ISVM + **Distr.-Hyp.-based method**	74.54 ± 0.17	4086.04 ± 108.12
SBERT + NN	77.65 ± 0.13	1362.77 ± 20.28
SBERT + NN + **Distr.-Hyp.-based method**	77.78 ± 0.12	6117.04 ± 127.16
SBERT (Fine-tune) + NN	83.85 ± 0.39	6716.80 ± 302.08
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	80.68 ± 0.40	1669.62 ± 36.89
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	80.36 ± 0.52	4550.11 ± 79.77

Table B.10: Results for the Amazon Reviews dataset, in the scenario without drift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	42.13 ± 0.13	1168.91 ± 1.98
AdaNEN + **Distr.-Hyp.-based method**	42.12 ± 0.12	4513.40 ± 172.91
Bi-SBERT + NN	43.68 ± 0.88	2715.19 ± 126.94
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	42.87 ± 1.05	5476.53 ± 1115.96
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	42.50 ± 0.82	8332.70 ± 341.82
SBERT + ISVM	36.77 ± 0.03	800.50 ± 9.86
SBERT + ISVM + **Distr.-Hyp.-based method**	36.71 ± 0.07	4030.72 ± 161.10
SBERT + NN	41.86 ± 0.12	1361.59 ± 8.77
SBERT + NN + **Distr.-Hyp.-based method**	41.85 ± 0.19	5895.44 ± 161.82
SBERT (Fine-tune) + NN	42.70 ± 0.43	1904.48 ± 266.78
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	42.78 ± 0.96	1815.03 ± 389.81
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	42.80 ± 0.89	4356.12 ± 169.05

Table B.11: Results for the Amazon Reviews dataset, in the scenario of Class Swap. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	41.94 ± 0.04	1106.06 ± 137.33
AdaNEN + **Distr.-Hyp.-based method**	42.03 ± 0.09	4396.86 ± 165.95
Bi-SBERT + NN	43.91 ± 0.99	2624.66 ± 110.28
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	43.04 ± 0.62	4862.21 ± 467.51
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	42.87 ± 0.72	8436.88 ± 269.47
SBERT + ISVM	36.57 ± 0.06	798.46 ± 1.46
SBERT + ISVM + **Distr.-Hyp.-based method**	36.55 ± 0.07	3988.42 ± 76.94
SBERT + NN	41.36 ± 0.35	1294.30 ± 7.58
SBERT + NN + **Distr.-Hyp.-based method**	41.69 ± 0.27	6369.44 ± 122.22
SBERT (Fine-tune) + NN	42.80 ± 0.49	3084.48 ± 192.47
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	42.39 ± 0.31	2056.49 ± 140.43
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	40.14 ± 3.14	4621.19 ± 300.22

Table B.12: Results for the Amazon Reviews dataset, in the scenario of Class Shift. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	42.44 ± 0.08	869.31 ± 4.22
AdaNEN + **Distr.-Hyp.-based method**	42.43 ± 0.07	4464.08 ± 76.00
Bi-SBERT + NN	43.07 ± 0.45	3436.99 ± 255.83
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	42.90 ± 0.46	5667.82 ± 793.38
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	41.91 ± 0.92	8652.56 ± 1008.90
SBERT + ISVM	36.63 ± 0.09	791.44 ± 2.46
SBERT + ISVM + **Distr.-Hyp.-based method**	36.62 ± 0.08	4034.47 ± 49.84
SBERT + NN	41.82 ± 0.23	1295.08 ± 6.08
SBERT + NN + **Distr.-Hyp.-based method**	42.07 ± 0.34	6019.08 ± 210.64
SBERT (Fine-tune) + NN	43.35 ± 0.58	3069.99 ± 272.78
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	42.71 ± 0.67	1989.30 ± 518.02
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	42.11 ± 1.52	4777.28 ± 228.08

Table B.13: Results for the Amazon Reviews dataset, in the scenario of Adjective Swap. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	38.88 ± 0.09	881.85 ± 14.14
AdaNEN + **Distr.-Hyp.-based method**	38.87 ± 0.11	4615.85 ± 107.55
Bi-SBERT + NN	37.07 ± 0.76	5404.46 ± 971.70
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	37.86 ± 1.06	7707.50 ± 832.39
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	37.36 ± 0.58	11293.32 ± 472.63
SBERT + ISVM	33.92 ± 0.07	782.31 ± 3.44
SBERT + ISVM + **Distr.-Hyp.-based method**	33.91 ± 0.10	4317.91 ± 74.21
SBERT + NN	38.30 ± 0.19	1291.62 ± 5.44
SBERT + NN + **Distr.-Hyp.-based method**	38.35 ± 0.38	6295.62 ± 167.84
SBERT (Fine-tune) + NN	37.47 ± 0.97	3499.54 ± 353.84
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	37.34 ± 1.16	3464.20 ± 380.36
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	37.58 ± 1.26	5883.08 ± 293.62

Table B.14: Results for the Yelp dataset, in the scenario without drift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	49.71* ± 0.05	1272.44 ± 155.96
AdaNEN + **Distr.-Hyp.-based method**	49.71* ± 0.05	6653.41 ± 598.90
Bi-SBERT + NN	49.83 ± 0.23	3395.02 ± 103.76
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	48.89 ± 0.92	3887.80 ± 1431.45
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	49.08 ± 0.95	10162.13 ± 1288.08
SBERT + ISVM	45.78 ± 0.14	935.02 ± 3.92
SBERT + ISVM + **Distr.-Hyp.-based method**	45.76 ± 0.08	5515.56 ± 134.79
SBERT + NN	49.12 ± 0.43	1638.33 ± 19.36
SBERT + NN + **Distr.-Hyp.-based method**	49.18 ± 0.47	8239.97 ± 375.34
SBERT (Fine-tune) + NN	50.04 ± 0.74	1903.69 ± 253.29
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	49.04 ± 0.53	1494.41 ± 551.36
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	48.48 ± 0.62	5828.85 ± 188.71

Table B.15: Results for the Yelp dataset, in the scenario of Class Swap. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	51.20 ± 0.14	1340.92 ± 1.97
AdaNEN + **Distr.-Hyp.-based method**	51.28 ± 0.09	6062.20 ± 21.61
Bi-SBERT + NN	51.95 ± 0.38	3486.93 ± 80.22
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	50.28 ± 1.64	5775.83 ± 459.53
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	50.86 ± 0.51	11508.16 ± 775.86
SBERT + ISVM	46.91 ± 0.10	924.96 ± 3.14
SBERT + ISVM + **Distr.-Hyp.-based method**	46.92 ± 0.17	5644.68 ± 229.49
SBERT + NN	50.20 ± 0.71	1556.81 ± 11.52
SBERT + NN + **Distr.-Hyp.-based method**	49.86 ± 0.78	8126.15 ± 145.33
SBERT (Fine-tune) + NN	50.08 ± 1.31	2682.82 ± 196.91
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	48.65 ± 3.79	2184.53 ± 127.99
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	50.15 ± 1.19	5975.72 ± 199.62

Table B.16: Results for the Yelp dataset, in the scenario with Class Shift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	62.15 ± 0.12	981.15 ± 4.95
AdaNEN + **Distr.-Hyp.-based method**	62.17 ± 0.09	6060.15 ± 203.66
Bi-SBERT + NN	61.37 ± 0.71	2220.64 ± 760.71
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	61.75 ± 0.69	4583.00 ± 1457.20
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	61.37 ± 0.37	9121.69 ± 1861.09
SBERT + ISVM	57.54 ± 0.11	929.89 ± 3.26
SBERT + ISVM + **Distr.-Hyp.-based method**	57.55 ± 0.14	5592.65 ± 237.91
SBERT + NN	61.35 ± 0.35	1543.66 ± 10.28
SBERT + NN + **Distr.-Hyp.-based method**	61.39 ± 0.38	8321.18 ± 573.56
SBERT (Fine-tune) + NN	61.91 ± 0.48	2852.65 ± 210.30
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	61.34 ± 0.46	1489.87 ± 559.86
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	61.57 ± 0.51	5803.39 ± 106.30

Table B.17: Results for the Yelp dataset, in the scenario of Time-Slice Removal. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	49.90* ± 0.19	997.22 ± 12.47
AdaNEN + **Distr.-Hyp.-based method**	49.90* ± 0.20	6233.06 ± 257.77
Bi-SBERT + NN	49.28 ± 0.19	2479.15 ± 790.21
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	48.97 ± 0.54	4323.37 ± 1746.99
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	49.03 ± 1.47	9848.63 ± 1414.83
SBERT + ISVM	45.92 ± 0.23	933.92 ± 6.96
SBERT + ISVM + **Distr.-Hyp.-based method**	45.85 ± 0.19	5678.36 ± 151.51
SBERT + NN	49.32 ± 0.32	1549.25 ± 18.14
SBERT + NN + **Distr.-Hyp.-based method**	49.29 ± 0.39	8204.80 ± 350.28
SBERT (Fine-tune) + NN	49.77 ± 0.98	2796.77 ± 137.86
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	49.23 ± 0.49	1814.29 ± 531.49
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	49.16 ± 0.53	5840.97 ± 197.40

Table B.18: Results for the Yelp dataset, in the scenario with Adjective Swap. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	45.79 ± 0.14	971.73 ± 3.58
AdaNEN + **Distr.-Hyp.-based method**	45.81 ± 0.15	6539.95 ± 158.57
Bi-SBERT + NN	45.83 ± 0.81	3948.88 ± 522.03
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	45.24 ± 0.93	6414.81 ± 526.38
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	45.73 ± 1.09	12748.13 ± 1257.25
SBERT + ISVM	42.48 ± 0.15	917.11 ± 1.67
SBERT + ISVM + **Distr.-Hyp.-based method**	42.57 ± 0.13	6089.80 ± 114.67
SBERT + NN	44.66 ± 0.78	1536.57 ± 7.21
SBERT + NN + **Distr.-Hyp.-based method**	44.59 ± 0.53	8505.05 ± 162.80
SBERT (Fine-tune) + NN	44.97 ± 1.06	2899.25 ± 178.79
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	44.67 ± 0.78	2399.19 ± 227.59
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	43.94 ± 1.25	6560.20 ± 128.36

Table B.19: Results for the Yelp Full dataset, in the scenario without drift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	51.20 ± 0.19	1554.10 ± 2.24
AdaNEN + **Distr.-Hyp.-based method**	51.25 ± 0.18	8316.15 ± 184.43
Bi-SBERT + NN	51.61 ± 0.29	5047.18 ± 405.01
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	51.37 ± 0.52	8417.48 ± 728.61
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	51.28 ± 1.04	15502.45 ± 652.61
SBERT + ISVM	45.13 ± 0.11	1090.88 ± 2.29
SBERT + ISVM + **Distr.-Hyp.-based method**	45.01 ± 0.15	7932.90 ± 163.41
SBERT + NN	50.75 ± 0.21	1906.16 ± 7.70
SBERT + NN + **Distr.-Hyp.-based method**	50.79 ± 0.26	11544.95 ± 201.48
SBERT (Fine-tune) + NN	51.35 ± 0.57	2762.92 ± 149.15
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	51.73* ± 0.46	3167.46 ± 91.39
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	51.73* ± 0.28	8483.92 ± 435.89

Table B.20: Results for the Yelp Full dataset, in the scenario of Class Swap. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	51.06 ± 0.16	1485.47 ± 169.64
AdaNEN + **Distr.-Hyp.-based method**	51.15 ± 0.16	8746.98 ± 380.90
Bi-SBERT + NN	51.66 ± 0.38	4951.00 ± 206.04
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	50.39 ± 0.49	8143.77 ± 496.65
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	50.90 ± 0.77	15992.23 ± 577.74
SBERT + ISVM	44.97 ± 0.14	1077.86 ± 4.95
SBERT + ISVM + **Distr.-Hyp.-based method**	44.83 ± 0.17	8248.08 ± 165.71
SBERT + NN	50.59 ± 0.28	1831.46 ± 3.69
SBERT + NN + **Distr.-Hyp.-based method**	50.63 ± 0.30	11536.01 ± 278.43
SBERT (Fine-tune) + NN	50.98 ± 0.60	5411.32 ± 707.21
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	51.29 ± 0.37	3231.18 ± 365.69
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	51.01 ± 0.75	8847.19 ± 262.13

Table B.21: Results for the Yelp Full dataset, in the scenario of Class Shift. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	51.06 ± 0.15	1134.22 ± 5.20
AdaNEN + **Distr.-Hyp.-based method**	51.08 ± 0.13	8495.96 ± 369.71
Bi-SBERT + NN	50.46 ± 0.41	5500.19 ± 317.93
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	50.63 ± 0.75	8607.89 ± 1087.36
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	50.11 ± 0.46	17111.95 ± 786.45
SBERT + ISVM	44.85 ± 0.10	1080.73 ± 4.60
SBERT + ISVM + **Distr.-Hyp.-based method**	44.69 ± 0.15	7957.80 ± 295.68
SBERT + NN	50.10 ± 0.32	1819.79 ± 5.22
SBERT + NN + **Distr.-Hyp.-based method**	50.30 ± 0.19	11194.01 ± 428.51
SBERT (Fine-tune) + NN	50.71* ± 0.87	5055.82 ± 725.98
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	50.71* ± 0.58	3435.35 ± 303.24
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	50.41 ± 0.62	9075.65 ± 186.64

Table B.22: Results for the Yelp Full dataset, in the scenario of Adjective Swap. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	47.48 ± 0.11	1115.48 ± 4.59
AdaNEN + **Distr.-Hyp.-based method**	47.50 ± 0.09	8874.77 ± 199.91
Bi-SBERT + NN	47.55 ± 0.57	6612.38 ± 444.65
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	47.41 ± 0.44	10238.13 ± 1060.59
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	47.32 ± 0.44	18946.67 ± 1231.98
SBERT + ISVM	41.95 ± 0.05	1057.12 ± 2.80
SBERT + ISVM + **Distr.-Hyp.-based method**	41.88 ± 0.15	8493.86 ± 190.50
SBERT + NN	46.78 ± 0.23	1790.02 ± 10.75
SBERT + NN + **Distr.-Hyp.-based method**	46.85 ± 0.14	11302.34 ± 300.75
SBERT (Fine-tune) + NN	46.70 ± 0.71	5608.39 ± 649.35
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	46.57 ± 0.27	4451.21 ± 214.31
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	46.81 ± 0.19	9894.43 ± 294.13

Table B.23: Results for the Yelp Polarity dataset, in the scenario without drift. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	88.40 ± 0.06	1151.71 ± 7.45
AdaNEN + **Distr.-Hyp.-based method**	88.45 ± 0.07	8488.14 ± 380.09
Bi-SBERT + NN	89.81 ± 1.05	3290.30 ± 604.99
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	88.66 ± 0.25	3823.66 ± 626.01
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	89.42 ± 0.82	13183.87 ± 1241.61
SBERT + ISVM	86.91 ± 0.06	904.24 ± 4.24
SBERT + ISVM + **Distr.-Hyp.-based method**	86.99 ± 0.10	7623.40 ± 243.19
SBERT + NN	88.24 ± 0.07	1927.87 ± 10.08
SBERT + NN + **Distr.-Hyp.-based method**	88.30 ± 0.08	11727.32 ± 330.76
SBERT (Fine-tune) + NN	89.68* ± 0.73	2166.71 ± 189.03
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	89.68* ± 0.79	2328.35 ± 548.01
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	89.24 ± 0.83	7763.24 ± 261.49

Table B.24: Results for the Yelp Polarity dataset, in the scenario with the Class Swap drift. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	88.35 ± 0.06	1162.84 ± 22.75
AdaNEN + **Distr.-Hyp.-based method**	88.39 ± 0.07	8749.74 ± 413.81
Bi-SBERT + NN	89.09 ± 1.21	2684.74 ± 913.25
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	89.19 ± 0.88	4790.96 ± 1777.53
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	89.18 ± 0.83	12445.39 ± 1212.87
SBERT + ISVM	86.55 ± 0.05	902.93 ± 4.19
SBERT + ISVM + **Distr.-Hyp.-based method**	86.63 ± 0.05	7895.19 ± 238.99
SBERT + NN	88.25 ± 0.08	1942.24 ± 14.66
SBERT + NN + **Distr.-Hyp.-based method**	88.30 ± 0.09	11583.72 ± 380.54
SBERT (Fine-tune) + NN	89.47 ± 0.93	2131.44 ± 100.43
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	89.26 ± 0.89	1970.25 ± 791.28
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	89.39 ± 0.88	8236.75 ± 280.67

Table B.25: Results for the Yelp Polarity dataset, in the scenario of Class Shift. Values in green are the best in each group, split by the dashed line.

Approach	Macro F1	Elapsed time
AdaNEN	88.34 ± 0.04	1147.07 ± 5.67
AdaNEN + **Distr.-Hyp.-based method**	88.37 ± 0.05	8494.49 ± 444.15
Bi-SBERT + NN	88.84 ± 0.85	2495.35 ± 703.80
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	89.39 ± 1.03	4757.74 ± 1875.61
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	89.32 ± 1.12	13357.03 ± 1535.34
SBERT + ISVM	86.06 ± 0.07	909.05 ± 0.83
SBERT + ISVM + **Distr.-Hyp.-based method**	86.10 ± 0.06	7818.47 ± 150.50
SBERT + NN	88.28 ± 0.08	1933.86 ± 9.59
SBERT + NN + **Distr.-Hyp.-based method**	88.32 ± 0.08	11498.22 ± 463.72
SBERT (Fine-tune) + NN	89.09 ± 0.87	1906.78 ± 147.00
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	89.07 ± 0.76	1918.44 ± 615.81
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	89.01 ± 0.31	8372.99 ± 195.74

Table B.26: Results for the Yelp Polarity dataset, in the scenario with Adjective Swap drift. Values in green are the best in each group, split by the dashed line.

Approach	**Macro F1**	**Elapsed time**
AdaNEN	83.88 ± 0.06	1134.67 ± 5.81
AdaNEN + **Distr.-Hyp.-based method**	83.89 ± 0.08	8929.00 ± 237.99
Bi-SBERT + NN	86.33 ± 0.44	3586.92 ± 159.61
Bi-SBERT + NN + **Distr.-Hyp.-based method (fine-tuning)**	85.99 ± 0.29	6357.71 ± 407.28
Bi-SBERT + NN + **Distr.-Hyp.-based method (fully incr.)**	84.93 ± 0.80	14974.83 ± 199.89
SBERT + ISVM	82.09 ± 0.06	891.25 ± 3.60
SBERT + ISVM + **Distr.-Hyp.-based method**	82.17 ± 0.07	8048.90 ± 241.32
SBERT + NN	83.71 ± 0.10	1977.10 ± 25.82
SBERT + NN + **Distr.-Hyp.-based method**	83.81 ± 0.18	11922.21 ± 433.77
SBERT (Fine-tune) + NN	86.16 ± 0.35	2164.45 ± 101.34
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fine-tuning)**	86.12 ± 0.49	2717.41 ± 211.41
SBERT (Fine-tune) + NN + **Distr.-Hyp.-based method (fully incr.)**	86.38 ± 0.51	8766.22 ± 271.59

C
Tables with impact on the elapsed times

This section lists the tables that show the impact of using the Distributed-Hypothesis-based method in terms of elapsed times.

Table C.1: Comparison between **Bi-SBERT** and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (incremental)**. The number represents the impact of leveraging the **Distributed-Hypothesis-based method** in terms of elapsed times. In green, the smallest value, and in red, the highest multiple measured.

	Airbnb	**Amazon Polarity**	**Amazon Reviews**	**Yelp**	**Yelp Full**	**Yelp Polarity**
Without drift	2.24	3.48	3.07	2.99	3.07	4.01
Class Swap	3.20	3.61	3.21	3.30	3.23	4.64
Class Shift	2.22	3.89	2.52	4.11	3.11	5.35
Time Slice Removal	2.28	-	-	3.97	-	-
Adjective Swap	2.54	3.43	2.09	3.23	2.87	4.17

Table C.2: Comparison between **Bi-SBERT** and **Bi-SBERT** + NN + **Distributed-Hypothesis-based method (fine-tuning)**. The number represents the impact of leveraging the **Distributed-Hypothesis-based method** in terms of elapsed times. In green, the smallest value, and in red, the highest multiple measured.

	Airbnb	**Amazon Polarity**	**Amazon Reviews**	**Yelp**	**Yelp Full**	**Yelp Polarity**
Without drift	1.50	1.30	2.02	1.15	1.67	1.16
Class Swap	1.79	1.30	1.85	1.66	1.64	1.78
Class Shift	1.46	1.35	1.65	2.06	1.57	1.91
Time Slice Removal	1.52	-	-	1.74	-	-
Adjective Swap	1.52	1.64	1.43	1.62	1.55	1.77

Table C.3: Comparison between SBERT + ISVM and SBERT + ISVM + **Distributed-Hypothesis-based method (fine-tuning)**. The number represents the impact of leveraging the **Distributed-Hypothesis-based method** in terms of elapsed times. In green, the smallest value, and in red, the highest multiple measured.

	Airbnb	**Amazon Polarity**	**Amazon Reviews**	**Yelp**	**Yelp Full**	**Yelp Polarity**
Without drift	3.89	6.20	5.04	5.90	7.27	8.43
Class Swap	3.86	6.09	5.00	6.10	7.65	8.74
Class Shift	3.88	6.45	5.10	6.01	7.36	8.60
Time Slice Removal	3.84	-	-	6.08	-	-
Adjective Swap	4.49	6.38	5.52	6.64	8.03	9.03

Table C.4: Comparison between SBERT + ISVM and SBERT + ISVM + **Distributed-Hypothesis-based method (fine-tuning)**. The number represents the impact of leveraging the **Distributed-Hypothesis-based method** in terms of elapsed times. In green, the smallest value, and in red, the highest multiple measured.

	Airbnb	**Amazon Polarity**	**Amazon Reviews**	**Yelp**	**Yelp Full**	**Yelp Polarity**
Without drift	3.30	4.27	4.33	5.03	6.06	6.08
Class Swap	3.18	4.19	4.92	5.22	6.30	5.96
Class Shift	3.18	4.27	4.65	5.39	6.15	5.95
Time Slice Removal	3.02	-	-	5.30	-	-
Adjective Swap	3.54	4.49	4.87	5.54	6.31	6.03

Table C.5: Comparison between SBERT (Fine-tune) and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (incremental)**. The number represents the impact of leveraging the **Distributed-Hypothesis-based method** in terms of elapsed times. In green, the smallest value, and in red, the highest multiple measured.

	Airbnb	**Amazon Polarity**	**Amazon Reviews**	**Yelp**	**Yelp Full**	**Yelp Polarity**
Without drift	1.91	2.97	2.29	3.06	3.07	3.58
Class Swap	1.73	3.05	1.50	2.23	1.63	3.86
Class Shift	1.82	3.20	1.56	2.03	1.80	4.39
Time Slice Removal	1.89	-	-	2.09	-	-
Adjective Swap	2.17	0.68	1.68	2.26	1.76	4.05

Table C.6: Comparison between SBERT (Fine-tune) and SBERT (Fine-tune) + NN + **Distributed-Hypothesis-based method (fine-tuning)**. The number represents the impact of leveraging the **Distributed-Hypothesis-based method** in terms of elapsed times. In green, the smallest value, and in red, the highest multiple measured.

	Airbnb	**Amazon Polarity**	**Amazon Reviews**	**Yelp**	**Yelp Full**	**Yelp Polarity**
Without drift	0.87	0.61	0.95	0.79	1.15	1.07
Class Swap	0.56	0.61	0.67	0.81	0.60	0.92
Class Shift	0.66	0.64	0.65	0.52	0.68	1.01
Time Slice Removal	0.76	-	-	0.65	-	-
Adjective Swap	0.97	0.25	0.99	0.83	0.79	1.26