University of Milano-Bicocca

Department of Informatics, Systems and Communication
Master of Science in Computer Science

Bridging a GAP: Text Style Transfer from Journalistic to Conversational for enhanced social media dissemination of news

Supervisor
Gabriella Pasi

Dissertation by
Mattia Piazzalunga
851931

Academic Year 2023/2024

Colophon

This dissertation, entitled "Bridging a gap: text style transfer from journalistic to conversational for enhanced social media dissemination of news", was written by Mattia Piazzalunga as the final project for the Master’s degree in Computer Science at the Department of Informatics, Systems and Communication of the University of Milano-Bicocca.

The document was drafted using LaTeX software, and the cited sources have been meticulously documented according to the BibTeX citation style.

The thesis was completed on October 25, 2024 and supervised by Professor Gabriella Pasi.

Email: mattiapiazzalunga@outlook.com

Our future success hinges on
our ability to embrace AI
and use it for good
Brad Smith

Abstract

In a society where precise and reliable information is essential for public well-being, the Digital News Report 2023 from the Reuters Institute for the Study of Journalism [81] highlights the crisis facing the news industry. The main challenges stem from the inability of news agencies to effectively adapt to digital media, where the audience prefers personalized access to news, a key strength of social networks. Additionally, the language used on these platforms is predominantly conversational, in stark contrast to the traditional style of journalism. In this context, research cannot remain passive. This thesis aims to identify a technique that allows the automated dissemination of news on new platforms, reducing the costs associated with human intervention in the production and adaptation of news to a conversational register. To this end, the Natural Language Processing task known as Text Style Transfer (TST) is introduced and adapted to the case study. The new task developed in this dissertation, derived from TST, is called *Journalistic to Conversational Text Style Transfer* (*J2C-TST*) and aims to transform the pragmatics of news from a journalistic (standard) style to a conversational (social) one. The chapters of the thesis examine the state of the art and the main neural architectures referenced in the literature, providing a solid theoretical framework to understand the design process adopted throughout the thesis. To support the new task, two datasets are presented: an italian one (*J2C_new_IT*) originating from collaboration with ANSA.it, and an english one (*J2C_news_EN*). An analysis is also conducted on the latter to formally define the "conversational" style, which, while maintaining the same levels of subjectivity, formality, and sentiment, is characterized by greater brevity and a focus on the essential information for the reader. To support the J2C-TST task, four language models are introduced: *T5-v1_1-news-style-J2C-EN-v1* and its reduced version *T5-v1_1-news-style-J2C-small-EN-v1* for the english task, and *mT5-news-style-J2C-IT-v1* and *mT5-news-style-J2C-small-IT-v1* for the italian task. To assess the correct stylistic transformation towards the conversational target, two classifiers are also presented: *RoBERTa-base-news-style-CLS-journalistic-conversational-EN-v1* and *XLM-RoBERTa-base-news-style-CLS-journalistic-conversational-IT-v1*, capable of distinguishing between journalistic and conversational styles in both languages, with potentially extendable utility. Finally, to define accurate and resource-efficient J2C-TST models, alongside careful hyperparameter optimization and modifications to existing libraries, the validity of using state-of-the-art compression and data augmentation techniques for the task is explored. The dissertation concludes with an analysis of the results obtained and a reflection on future prospects.

Preface

When I was 10 years old and in the fourth grade, I began to develop a passion for technology and innovation. It was during this time that I first imagined that computer science could become my future field of study, even though I didn't yet fully understand all its nuances. It might have seemed like a premature intuition, especially considering that many young people remain undecided about their path until university; yet, today I can confidently say that intuition was spot on. In 2014, I started high school, and as my passion grew, so did my desire to delve deeper into every aspect of computer science. Back then, just like today, I wasn't satisfied with simply creating a functional program: I wanted to fully understand the technical details underlying it, with the goal of creating something not only innovative but also efficient and ethical. This desire led me to the decision to continue with my university studies. After graduating with honors in 2019, I decided to pursue a master's degree. Over the years, my interest shifted from software engineering to artificial intelligence and natural language processing. These fields fascinated me for their innovative potential and the yet unexplored research opportunities. Today, scientific research is one of my greatest passions. The work I present in this thesis is the result of years of study and months of experimentation. The project, born out of collaboration with ANSA, provided me with the extraordinary opportunity to tackle complex scientific problems and, in my small way, to contribute to advancing knowledge in the NLP field, defining a new downstream task that is both challenging and impactful for society.

I hope that the results of this work can be useful for further research and exploration, even during my PhD journey. This thesis thus marks the end of a magnificent academic path, but the beginning of a new adventure in the world of academia and research.

Acknowledgements

This section is presented in italian, the native language of the author.

Sono stati anni intensi, pieni di sfide e di emozioni che hanno lasciato un segno profondo. Ogni difficoltà superata, ogni notte passata sui libri, ogni traguardo raggiunto ha contribuito a rendere questo percorso indimenticabile. Cinque anni fa ho intrapreso un viaggio senza sapere esattamente dove mi avrebbe portato, ma con il cuore pieno di speranza e determinazione. Oggi, raggiungere questo obiettivo è molto più di una semplice conclusione: è un inizio, una chiave che apre la porta verso un nuovo mondo di scoperte e crescita, quello del dottorato. Molte persone sono state fondamentali in questo viaggio: colleghi, amici e la mia famiglia hanno contribuito a rendere tutto questo possibile. Mi hanno sostenuto nei momenti più difficili, quando le sfide sembravano insormontabili, e mi hanno guidato con la loro presenza costante e il loro affetto.

Alla mia relatrice, **Gabriella Pasi**, che con infinita pazienza e dedizione mi ha accompagnato in questi ultimi mesi, vorrei dedicare un ringraziamento speciale. La sua capacità di trasmettermi la passione per la ricerca ha reso questo percorso profondamente stimolante. Sono felice che sarà lei a guidarmi in questo nuovo capitolo del dottorato, certa che, saranno anni di grandi sfide e soddisfazioni.

Un ringraziamento speciale va ad **ANSA.it**, e in particolare a Marco Beltrame, Gian Andrea Casavecchia e Alessandro Barberi, per aver creduto nella mia proposta e per aver collaborato con me per costruire il dataset italiano di trasferimento stilistico delle notizie. Grazie alla loro disponibilità e professionalità, ho avuto l'opportunità di contribuire al sostegno all'indutria dell'informazione italiana. ANSA e il suo straordinario team sono la dimostrazione di come una realtà consolidata possa non solo abbracciare il mondo universitario e della ricerca, ma anche favorire la propria crescita sostenendo le idee e le ambizioni dei giovani studenti. Grazie per avermi dato fiducia e per aver reso possibile questa incredibile collaborazione.

Infine, vorrei ringraziare la **mia famiglia** e i **miei amici**, che sono stati un supporto fondamentale lungo tutto questo percorso. Nei momenti di difficoltà, sono stati il mio punto di riferimento, sempre pronti a sostenermi e a darmi la forza di andare avanti. La loro presenza costante e il loro incoraggiamento mi hanno dato la motivazione necessaria per affrontare le sfide e raggiungere questo traguardo. Non saprò mai ringraziarli abbastanza per tutto l'affetto e il sostegno che mi hanno dato.

Contents

Chapter 1

Introduction

1.1 Overview of the research context

Being informed about what is happening in the world, from local to international news, is fundamental for promoting freedom of thought, expanding personal knowledge, and well-being. Shared knowledge among citizens, fueled by news from certified sources, also makes them active in society and capable of fully exercising their civic rights and duties, an aspect at the core of a healthy democracy. Equally important is the role of information in ensuring transparency and accountability: public and private institutions must be subject to adequate oversight, and news plays an essential role in highlighting misconduct or abuses of power.

"The shaping of the mind or character", states the Oxford english Dictionary [24], an abstract definition of information that conveys the "power" it wields. Free access to information is recognized and protected by the most important international Bills of Rights: Article 19 of the "Universal Declaration of Human Rights" [80] as well as Article 11 of the "Charter of Fundamental Rights of the European Union" [113] emphasize the indispensability of receiving and imparting information and ideas through any media, without interference by public authority and regardless of frontiers.

In a society where information is a necessity, Newman et al. [81] in the Digital News Report 2023 from the Reuters Institute for the Study of Journalism provide clear evidence of the crisis affecting the news industry. The main reason for this decline in public interest is related to the difficulties that news agencies face in transitioning to a digital media environment, a change driven by generations that rely almost entirely on digital media for content access. To make matters worse, year after year, the number of direct accesses to news agency websites or apps decreases: society prefers a personalized and ubiquitous access to news, guaranteed by social networks, a trend promoted by people's growing dependence on these intermediaries. The industry cannot, and must not, remain indifferent to this revolution.

It is therefore necessary to acknowledge the growing dissatisfaction with the way news content is presented: publishers must convince the public that the news deserves their attention. The challenge is, more than ever, to restore relevance and trust by meeting the needs of users, providing real value to consumers, and ensuring the accuracy of information that only recognized news agency can certify.

In this context, research cannot remain indifferent.

1.2 The news industry landscape

Before delving into the specifics of the research objectives, it is important to fully understand the research context, systematically highlighting the crisis affecting the news industry. In this section, the access habits of people to information are analyzed, with particular reference to news. The goal is to study current trends: which communication channels are used, the importance attributed to quality information, and the challenges that news agencies must face. The most important statistical study cited in this section is that of Newman et al. [81], aimed at understanding how news is consumed in a selection of countries, analyzing, as far as possible, the habits of nationally representative quotas by age, gender, and region in each of the 46 markets examined, net of the biases present in them.

1.2.1 The fall of news engagement

A first trend to analyze concerns the importance, nowadays, attributed to the retrieval of news. According to Newman et al. [81] in the Digital News Report 2023, the proportion of consumers who tend to avoid it is close to historical highs, with 36% of the population in the 46 markets analyzed. The technical term is "selective avoidance", which has increased by 7 percentage points since 2017 1.1, although it has slightly decreased compared to 2022.

In particular, about half of the avoiders (53%), generally young people or those with a low level of education, do so in a general or periodic manner. A second group tends to avoid the news by adopting more specific actions, exercising control less frequently or refraining from consulting certain categories.

Unlike the COVID-19 pandemic, where there was a reversal of the trend, neither the cost of living crisis nor the war in Ukraine have led to a sustained increase in news consumption. In a large group of countries, survey data show a decline in weekly consumption and an overall lower interest. Despite the political and economic threats many people face, therefore, less than half (48%)

of the aggregated sample defines themselves as very or extremely interested in them, a percentage value down from 63% in 2017 1.1.

The decline in interest in news gathering is also highlighted by the volume of the "news" query on Google, which has seen a drop of a dramatic 30% over five years, 16% in the last year alone.

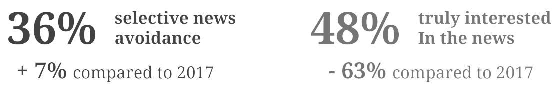


Figure 1.1: Interest in the news by the population.

Therefore, trying to re-engage people with information becomes fundamental. In particular, it is necessary to arouse interest in the younger population. One idea could be to use channels or tools that are ubiquitous in the daily life of the digital society.

1.2.2 Disinformation and trust in news

A longstanding concern among the population is the fear of misinformation and disinformation. According to the Digital News Report 2023 by Newman et al. [81], the shift in attention towards new communication channels does not seem to have alleviated this concern. Across markets, well over half of individuals (56%) state they are worried about identifying the difference between what is real and fake on the internet when it comes to news, an increase of 2% from 2022 1.2. Those who say they primarily use social media as a news source are much more concerned (64%) 1.2 compared to people who do not use it at all (50%) 1.2. This does not mean that the use of social media causes misinformation, but that the problems documented on these platforms and the greater exposure to a wider range of sources seem to have an impact on the trust in the information that the population has access to.

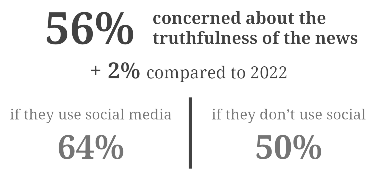


Figure 1.2: Concern about the truthfulness of the news.

The dissemination of quality information, promoted and verified by well-known press agencies, can therefore fit into this context. Of course, they cannot handle the burden of managing various communication channels "manually," but AI/ML techniques for the automatic distribution of certified content, starting from news written in the conventional form, could prove effective.

1.2.3 The digital transformation of Media

According to the Digital 2024 Global Overview Report [109], as of January 2024, the world population has reached 8,08 billion, of which 66,2% (5,34 billion) use the Internet 1.3: a well-known trend that has already revolutionized traditional media companies. However, whether they have completed their initial digital transformation, shifting from a focus on print or broadcasts to digitally focused brands, matters little. Agencies find themselves, in a digital society, facing continuous transformation. For example, today, new generations often avoid direct discovery of news, show little interest in conventional offerings, while embracing the more participatory, personal, and personalized options of social platforms. An important figure is, once again, provided by the Digital 2024 Global Overview Report [109]: 62,3% of the global population, equivalent to 5,04 billion individuals (5,6% more than the previous year) use social media, tools that base their operation precisely on the personalization of content. This accounts for 93,2% of internet users 1.3. In contrast, only 68% make use of online press content, down 7,5% from the previous year. And printed news? Only 62,1% of internet users between the ages of 16 and 64 make use of it, a drop of 10,1% in one year 1.3.

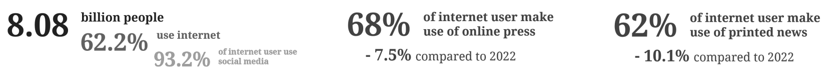


Figure 1.3: Inversion of trends and issues.

Returning to the problem of the decline in direct discovery of information, in all 46 markets of the Digital News Report 2023 by Newman et al. [81], only one-fifth of participants (22%) state that they start their information journeys through a website or an app, -10% compared to 2018 1.4. Although publishers in some small Northern European markets have managed to reverse this trend, younger groups everywhere show a preference for accessing news through secondary routes. In particular, at an aggregate level, a turning point has been reached in recent years, with a preference for social media (30% of respondents 1.4). Obviously, these are averages, which depend on the state of belonging and the age of the respondent, but a very strong figure: press agencies are losing audience on traditional media, especially among young people.

22% **start their journeys through a website or app**
- 10% compared to 2018

30% **prefer to access news through social media**
new trend

Figure 1.4: New points of access to news.

Regardless of population differences, social media is a resource: every day, according to the Digital 2024 Global Overview Report [109], people spend 6h 40m on the internet where, on average, 2h 23m on social media (-5,5% compared to the previous year) and only 1h and 41m is spent on online/printed press media, a dramatic drop of 22,2% in one year. Furthermore, 34,2% of respondents emphasize that access to information is the main reason for using social media: traditional press agencies cannot remain indifferent.

This is the media environment that the public embraces, the "new normal" in which journalists and media must find their space if they want to connect with the audience. This thesis focuses precisely on this, trying to automatically adapt, through Deep Learning techniques, the "old" journalistic language, still used and fundamental in "high-level" publishing, to a more "social" language, called conversational, used by new platforms.

Moreover, while mainstream journalists often lead conversations about news on Twitter and Facebook, they struggle to attract attention on new networks like Instagram, Snapchat, and TikTok, where often personalities, influencers, and ordinary people are more prominent, even when it comes to informative conversations. Therefore, it is necessary to act to reverse this trend, and having an automatic tool to support the dissemination of quality information across all the countless channels becomes essential to reach everyone and reduce costs.

1.3 Research goals

Once the research context and the issues affecting the information market are thoroughly understood, it is possible to delve into the specifics of the research goals of this thesis.

This dissertation aims to develop a methodology that facilitates the dissemination of news on social networks, the primary communication channels used by contemporary society. The dominant language on these platforms is characterized by a simple and conversational style, in contrast to the one used in traditional journalism. This more accessible communication approach, marked by the brevity of the texts, is one of the key elements of social media's success. The dissertation starts from this consideration: currently, a "costly" human intervention is required to adapt the style of news before it is published on social media, but Machine Learning models may offer a solution. If properly trained, these models can automatically transform journalistic texts into a more "social" style, suitable for communication on these platforms.

Currently, there is no specific task in the literature that addresses the described stylistic change in text. For this reason, starting from the reference task in Natural Language Processing (NLP), namely Text Style Transfer (TST), the intention is to formalize a new task aimed at converting prose, using Machine Learning (ML), from journalistic style (standard) to the conversational style typical of social media. Since this is an innovative task without existing resources, this dissertation aims to define datasets that can support this task for both the italian and english languages. Specifically, the goal is to define two corpora, one for each language. For the italian corpus, the ongoing collaboration with ANSA.it [1], Italy's leading news agency, plays a fundamental role. Not only does ANSA.it ensure the quality of the dataset, but its involvement also highlights the significance of the proposed task. Moreover, the corpora aim to be useful for other types of analyses beyond those presented in this thesis.

By alternating theoretical and practical aspects, this dissertation also seeks to outline a framework on the state of the art in the NLP field and the specific task under examination, providing a foundation for understanding the design pipeline used in this thesis. All of this aims to achieve a second objective: the creation of models to support the task in both languages. The idea is to develop, in particular, two models for each language: one large and highly accurate, and one smaller and more efficient. Throughout the thesis, special attention will be paid to the need for efficient solutions, and in this context, theoretical references can help to understand the societal impact of more sustainable models.

What is presented in this dissertation represents only a starting point, a first attempt to support the news market. Therefore, in the final part of the thesis, it is important to outline future goals that will improve the proposed task and related ones.

In summary, while news agencies struggle to adapt to the use of new social media platforms, Machine Learning, and particularly Deep Learning, can offer valuable support to the information industry. In this work, their application is focused on aligning the linguistic style of news with that of social networks, a task often overlooked by publishers due to the costly human intervention it requires. This dissertation is a first contribution to an industry that is one of the main drivers of societal growth.

[1] For the writing of this thesis, a collaboration is underway with ANSA.it (`https://www.ansa.it/`), a well-known italian news agency, which has helped highlight the importance of the task that is being formalized.

1.3.1 Open science and transparent research practices

This thesis is written in adherence to the principles of Open Science, a set of practices that promote transparency and the open sharing of the scientific process. As emphasized by the European Commission [2], Open Science facilitates sharing and collaboration, thereby accelerating the discovery process, improving research quality, and making science more impactful and central to human and social development. For this reason, not only are research results made freely accessible, but also the models, code, and procedures necessary to replicate experiments are provided, thus fostering greater transparency, scientific progress, and reducing duplication. This approach aligns with the philosophy of "Green AI", which aims for more sustainable development of artificial intelligence.

The code used in this study, the datasets presented, and the models generated are available at this link: `https://github.com/mattiapiazzalunga/MasterThesis`

1.4 Structure of the thesis

In this final introductory section, after outlining the research context and objectives in relation to the news market, and highlighting the importance of the task examined in this thesis for supporting the information market, it is appropriate to present the structure of this study:

1. As previously highlighted in Chapter 1, "Introduction", the research context was discussed, and current trends in the news market were analyzed. The focus was on the crisis affecting the information industry and the need for research support, thereby outlining the objectives of this dissertation
2. In Chapter 2, "Exploring Text Style Transfer", the main task of the thesis is introduced, which aims, as highlighted in the introduction, to convert text from a standard journalistic style to a conversational one. This goal, pursued through the use of machine learning models, aims to promote the dissemination of accurate news on new digital platforms without the need for direct human intervention. The new task, not previously addressed in the literature, is thus formalized, providing the theoretical foundations necessary to understand it.
3. In Chapter 3, "Text processing and representation", the transition from theory to implementation in the NLP process is described theoretically, with a focus on the initial stages of the NLP pipeline. The importance

[2] `https://research-and-innovation.ec.europa.eu/strategy/strategy-2020-2024/our-digital-future/open-science_en`

of noise removal to improve data quality and the preparation of words for machine and/or model understanding is explained, along with a description of the state of the art in the field and the current limitations.

4. In Chapter 4, "Deep learning approaches in NLP", the state of the art of NLP models and the downstream task under study is presented through a theoretical discussion. Although this task is innovative, there are related tasks that can serve as a starting point. Furthermore, the sections discuss deep learning models, with a particular focus on transformer architecture and foundational models, highlighting their advantages and limitations. The chapter concludes by presenting the issues related to the high resource consumption of today's 'large' models and proposing cutting-edge solutions to address them.

5. In Chapter 5, "Evaluating TST models: key metrics and methods", a theoretical discussion is provided on how to evaluate the effectiveness of the models presented in the dissertation. Three main areas of evaluation are highlighted, supported by four automatic metrics well-known in the literature and by a custom neural model introduced in the dedicated section. The functioning and role of all these elements in the evaluation process are, also, presented.

6. In Chapter 6, "Introducing J2C datasets", which focuses on implementation and result analysis, the two datasets generated to support the new style transfer task, in both italian and english, are presented. Additionally, a brief exploratory analysis on the english corpus is conducted to highlight their potential in other areas of study, which also helps to characterize the two styles involved in the transfer.

7. In Chapter 7, "Optimizing J2C models: from theory to practice", the core of this dissertation, six models are presented. Two of these are fine-tuned models (pretrained model size 'base'), one for the italian language and one for the english language, developed to support the news market through the task proposed in this dissertation. To pursue the goals of 'green AI' and create efficient, fast, and environmentally friendly models, 'small' counterparts of the base models have also been developed. In addition to the generation models, two style classifiers are introduced, whose purpose will be discussed in more detail in the following sections, as they are a key tool for assessing the quality of the produced outputs. Finally, to improve the models' results, state-of-the-art compression and data augmentation techniques are analyzed, whose validity is evaluated in relation to the considered task.

8. In the concluding chapter (8), "Final remarks: NLP and news", the dissertation wraps up by summarizing the results achieved and revisiting the datasets and models presented. At the same time, possible future di-

rections for research are outlined, based on the theoretical limitations identified and the implementations carried out in this thesis. The proposed areas for development range from optimizing the results of the downstream task analyzed to improving elements associated with NLP in a broader sense.

Together, these chapters construct an in-depth exploration of how cutting-edge machine learning models can address modern needs, supporting, through a specially designed downstream task, the quality and accessibility of information in the digital age.

Chapter 2

Exploring Text Style Transfer

After providing an introduction to this dissertation and presenting the current trends influencing the information industry, this chapter introduces the primary task of the thesis: Text Style Transfer (TST), a downstream task of Natural Language Processing (NLP). In this research, TST is used to convert text from a formal journalistic style to a social/conversational one. This approach, as previously highlighted, aims to support the dissemination of news through new digital communication channels by means of an automatic tool. Simultaneously, the theoretical areas of NLP, specifically NLG and NLU, are introduced to emphasize their connection with TST.

2.1 Natural language processing: A concise introduction

This thesis involves working with text, its understanding, and generation. Therefore, it is necessary to introduce what Natural Language Processing is.

From an early age, humans learn language by discovering patterns and models, in an attempt to compose sentences, questions, and commands. In 1999, Feldman [27] highlighted that if we could formally define these models and describe them to a computer, then we would be able to teach a machine how we understand and speak.

Natural Language Processing (NLP) is a discipline that emerged in the 1950s, combining artificial intelligence (AI) with linguistics to form an interdisciplinary field aimed at fulfilling the desire for human-machine interaction and simplifying computer use for non-experts. The primary goal of NLP is to enable machines to understand, interpret, generate, and respond to human language, or similar, in a meaningful and useful way. NLP is an extensive subject, rooted in the need for automatic translation between the 1940s and 1950s.

It is divided into two overlapping fields: Natural Language Understanding (NLU) and Natural Language Generation (NLG), according to Khurana et al. [52] (schematized in figure 2.1).

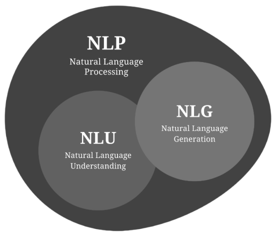

NLP
Natural Language Processing
NLG
Natural Language Generation
NLU
Natural Language Understanding

Figure 2.1: Branches of the NLP.

The task that will be studied and analyzed in this dissertation is a sub-area of NLP; therefore, it is essential to thoroughly understand the limits, potential, and fields of NLP because we will delve into technical/mathematical aspects.

2.2 From phonetics to pragmatics: NLU explained

The branch of NLP that deals with the understanding of natural language (NLU) faces a fundamental problem in the correct processing of prose: how to attribute meaning. As complex as it is to even explain how humans are capable of interpreting a text, it is even more difficult to define objective and unambiguous levels of depth of understanding. This is where linguistics comes into play. The contribution to formalization has been numerous, with various linguists succeeding one another, and research in this area of language is still active. There are different classifications in levels of understanding, and in this dissertation, we use the form, perhaps the most widespread, formalizing six in accordance with, among many, a paper by Feldman from 1999 [27]. They are reported here in ascending order of intensity of understanding:

1. Phonetics or Phonology. Phonetics concerns the concrete study of sounds. They are analyzed as physical entities, called phones, without considering their ability to distinguish the meanings of words. Phonology, on the other hand, is the study of the organization and function of sounds in conversation. It focuses on the abstract aspect of sounds, called phonemes, where a phoneme is a “sound” that leads to a difference in the semantics of the word.

   This level is not important for written text, but rather for understanding in spoken language.

2. Morphology. Morphology is concerned with the structure and meaning of words. Specifically, it analyzes how words change form through

the combination of morphemes, the smallest units of meaning in a language. Morphology, therefore, investigates the mechanisms by which simple meaning-bearing units are organized into more complex meanings.

3. Syntax. Syntax concerns the grammatical structure of sentences, that is, how words can be organized to form grammatically correct sentences. A language is composed of thousands of words: mathematically playing permutations and combinations leads to an almost infinite number of ways in which they can be combined in an attempt to form phrases. However, only some of the combinations make sense for a language. By analyzing syntactic aspects, it is possible to understand how words can be syntactically combined in a way that reflects the meaning to be conveyed. It is also important to highlight that a sentence conveys meaning and relationships between terms even if the "global" semantics of it are not known. Using, in a statement, a verb in the past rather than the present, for example, is clear temporal information. The output of the syntactic level, thus, underlines the structural dependency between words.

4. Semantics. Semantics concerns the formation of meanings, that is, how words and phrases convey information. It examines not only words for their dictionary meaning but also for the sense that comes from the context of the sentence. More formally, semantics focuses on the relationship between linguistic signs (the signifiers) and what they represent (the meaning), as well as the reciprocal relations between meanings.

   But how does this process occur? To understand the semantic content of a sentence, humans rely on their knowledge of language and the concepts present in it, but machines cannot rely on these techniques. As highlighted by Khurana et al. [52], semantic processing at the machine level determines the possible meanings of a sentence by processing its logical structure, recognizing the most frequent words, and understanding their interactions. For example, an algorithm might understand the context of a sentence if words related to it are used.

5. Discourse. Discourse pertains to the use of language in more extended forms of communication. It analyzes how sentences are organized within a broader text, such as a conversation, an article, or a document. Discourse involves the analysis of relationships between sentences, the structure of the text, and overall coherence. Through a broader study than that of semantics, this level can extract additional meanings. By interpreting the relationships between sentences, as Khurana et al. [52] emphasize, it is possible, for example, to resolve coreference by finding all expressions that refer to the same entity in a text.

6. Pragmatics. It is important to emphasize from the outset that the pragmatic level depends on a set of world knowledge that comes from outside the text content. Pragmatics concerns the use of language in real contexts, where the context includes both the linguistic and the situational one.
   When a sentence is not specific and when the meaning of a linguistic expression becomes uncertain due to contextual, intentional, and interpretive factors, pragmatic ambiguity may arise. As Walton pointed out in 1996 [117], it occurs when different people draw different interpretations of the text, depending on the context. While semantic analysis focuses on the literal meaning of words or phrases, pragmatic analysis focuses on the meaning that readers/listeners perceive based on factors not directly related to the structure of the words or phrases.

   In short, pragmatics involves the analysis of how meaning is influenced by context: it can be affected by the situation, the speaker's intention, and the listener's interpretation. This level, for example, attempts to capture the presence of sarcasm in a conversation.

A graphical feedback to what is stated is shown in figure 2.2.

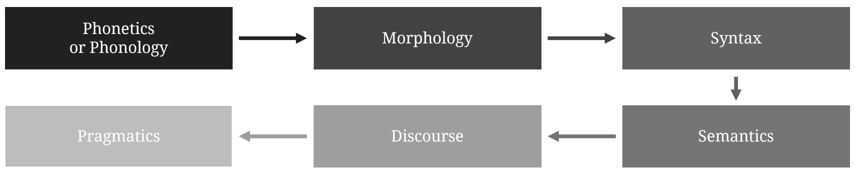


Figure 2.2: NLU levels.

The depth of understanding required for the task of this dissertation will be highlighted next.

2.3 The role of style in linguistics

Why, at this point, is it necessary to talk about style? Because the task discussed in this dissertation involves a transfer of text style, from a formal/semi-formal journalistic style to a conversational/informal digital style.

According to the Cambridge Dictionary [77], style is defined as a distinctive manner of expression, the way in which 'to say something.' It is an intrinsic characteristic of every utterance that, as highlighted by Jin et al. [47], can be seen as a protocol that regularizes communication.

An important insight for the definition of style is that of McDonald et al. from 1985 [74], which is the way semantics are expressed. According to this theory, style is a consequence of the decisions made during the transition from the level of conceptual representations to the linguistic level.

Already in 1987, Hovy [43] highlighted the importance, in text generation, of studying the objectives that underlie the production of the text itself, suggesting how prose can be expressed in different ways, varying its form, in an attempt to convey more information than that contained in the literal meaning of the words. Why? To produce a stronger effect on the recipient of the speech. These additional pieces of information express the interpersonal goals of the speaker towards the listener and, in general, their perception of the pragmatic aspects of the conversation. Already in 1987, therefore, the need to personalize the linguistic style of a generative model was highlighted, to avoid producing the same result for all listeners/readers, in all circumstances.

In order for generative programs to produce varied and information-rich texts, as Hovy points out, such programs must have some means to represent the relevant characteristics of the listener, the context of the conversation, and the interpersonal objectives. All pragmatic aspects.

2.3.1 A data-driven perspective on style

In the previous section, the concept of "style" was introduced and its linguistic aspects were defined. Formalizing a definition, Biber et al. [12] state that style is the set of non-functional linguistic characteristics of a text, that is, those not necessary to convey the main meaning; the "aesthetic" aspects of prose. From a practical point of view, stylistic classification based on the linguistic definition is prescriptive and constrained, as it outlines style based on theoretical constraints and rigid characteristics.

Working with natural language, however, is complex and attributing a style to a text, especially if short, can be a really difficult task. For this reason, as highlighted by Mou et al. [79], it is useful to relax the definition of style. A data-driven and empirical definition, more important in the context of this dissertation, states that given two corpora differentiated by their style, the invariance between them is the content, or the informative essence, while the variation is the style (figure 2.3).

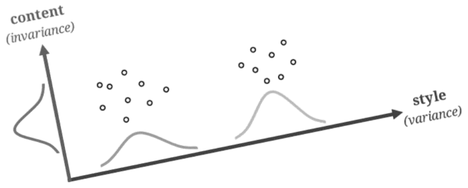


Figure 2.3: Style vs content in two corpora.

Why is this data-driven definition more important in the context of NLP and deep learning? As Jin et al. [47] emphasizes, this new definition extends the concept of style to all the general attributes of the text. Moreover, it allows the text pertaining to a style to be less bound by rules and rigid characteristics. Often, in fact, the creation of corpus occurs automatically by exploiting, for example, meta-information, which leads to texts not rigidly classified in a linguistic style: there can be ambiguities in affiliation. This does not mean that there are no "data-driven" attributes that correspond to the definition of style dictated by linguistics, but it ensures greater flexibility in working with style and text.

The data-driven definition, in any case, is suitable to be exploited in a study, as in this dissertation, where deep neural algorithms are used that will learn the concept of style by distinguishing it in the different text corpora.

2.4 From intent to expression: the NLG process

Once the concept of style and its placement within NLU are understood, to rigorously define the downstream task of NLP discussed in this dissertation, it is appropriate to introduce the sub-area of NLP dedicated to Natural Language Generation (NLG), a field that studies the process of producing coherent and meaningful text/speech (output) given an input that is not always linguistic.

To fully understand NLG, as highlighted by McDonald in 1986 [72], it is beneficial to start by adequately delineating the differences compared to NLU. The process of "Understanding", as previously emphasized, aims to deduce the content conveyed by the text and the likely intentions of the speaker who produced it, based on the words and, possibly, their intonation. In contrast, the generation process proceeds from intention and content towards expressive form. Additionally, while in NLU the ambiguity related to the different meanings that can be attributed to the text represents a challenge, in NLG, ambiguity is, theoretically, nonexistent initially. However, the linguistic choices made during text production can introduce ambiguity for the recipient. With a flow opposite to NLU, as McDonald pointed out [72], one might think that a generation process is organized like a comprehension process, but with the stages in reverse order. To some extent this is true, but to emphasize this order of linguistic representational levels would mean losing the special character of generation, which is primarily a process of planning: a progressive refinement.

In any case, as Gatt et al. point out [30], providing a precise definition of NLG is a challenging task: although the literature agrees on the output of this task, namely the text, the input could vary substantially. In a broader and more formal concept, McDonald in 2010 [73] defined this problem as "the source problem": currently, it is not rigorously known where a generation system should start to speak or produce text like humans: the source is a "mental state" within a speaker, who has "intentions" and who acts in a "situation", all terms of art with very slippery meanings. Assuming that the mental state has a formal representation, there are dozens, often conflicting among research groups.

Understanding that the "generator" can be interpreted as an individual who intends to communicate something [73], the problem of NLG, in its most common and theoretical form, is divided into six sub-tasks in order to transition from the system's input to textual output. This division is in accordance with Reiter et al. [95] and Gantt et al. [30]:

1. Content determination. The NLG system must first decide which information to include in the statement to be produced as output, that is, it must choose what to communicate. The selection should be made based on the target audience and communicative intentions [30], so much so that content determination approaches depend heavily on the application domain in most cases. The output of this activity consists of "messages" [95], which are formal representations defining entities, concepts, and relationships.
2. Text structuring/discourse planning. After deciding which "messages" to communicate, the system must define the order in which the information should be presented. A text, as Reiter et al. highlight [95], is not just a random collection of pieces of information: the structural order (e.g., beginning, middle, and end) can positively impact the reading and understanding of the prose. A possible output of Text Structuring is a tree structure that connects the "messages."
3. Sentence aggregation. In this step, the decision concerns which information to present in each sentence [30]. Not all the selected "messages" need to be expressed in separate statements; by combining multiple messages into a single phrase, the output can become more fluid and readable.
4. Lexicalization. Once the content of the phase is defined, the system can start converting it into natural language. After organizing the information, it is necessary to choose the appropriate linguistic resources (phrases and words) to communicate them, taking into account the complexity of being able to express a single "block" in many different ways. Lexicalization must consider the application domain and the associated reference vocabulary, as well as the stylistic constraints imposed on the NLG system. This important step, as Reiter highlights [95], also allows for subtle pragmatic modifications of the output.
5. Referring expression generation (REG). The purpose of this activity is to select words or phrases aimed at identifying domain entities, increasing the clarity of the discourse and avoiding unnecessary repetition of

information. Unlike lexicalization, REG is specifically defined by Reiter [95] as a discrimination task, whose objective is to identify the terms necessary to distinguish one object from another. The way the system refers to these entities depends, among other things, on whether they have already been mentioned [30] and if they need to be distinguished from other similar concepts. For this purpose, "referential forms" [30] are used, which are linguistic expressions useful for referring to entities, such as pronouns, proper names, or descriptions: a description based on properties can be useful for disambiguating multiple item with the same referential form.

6. Linguistic realisation. The linguistic realization represents the final step in the natural language generation process. In this phase, the selected words and phrases are combined to form well-structured sentences from a syntactic and morphological perspective. This process, heavily influenced by the target language, aims to make the text as smooth as possible by determining the word order and generating the correct morphological forms (e.g., verb conjugation and subject-verb agreement). Additionally, the linguistic realization phase might involve the addition of functional words in the text, such as auxiliary verbs and prepositions, and the insertion of correct punctuation.

A graphical feedback to what is stated is shown in figure 2.4.

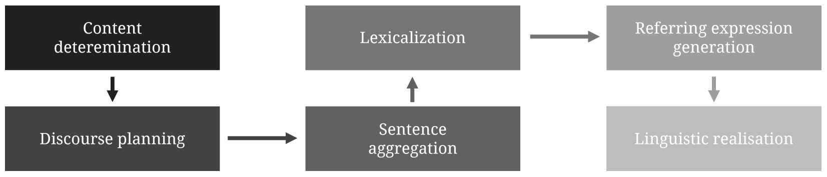


Figure 2.4: NLG activities.

This set of activities does not correspond to the concrete components useful for the implementation of a natural language generation (NLG) system, as highlighted by Mellish et al. [75], but represents the tasks most commonly identified as independent functions within NLG systems. The steps indicated are, therefore, to be considered highly theoretical, also because they are less rigidly applicable in today's deep neural architectures. However, they are useful for understanding the difficulty of the NLG task and for providing an idea of the possible internal reasoning of a "black box" neural network.

In any case, to conclude, central to NLG is, as highlighted by Reiter et al. in 1997 [95], generating information that is easily understandable by people, even non-experts. This is closely related to the concept of style which, working at a pragmatic level, allows for easier comprehension of the generated text.

2.5 Adapting linguistic style: the role of TST

As highlighted in the previous sections, language is situational [47], and the stylistic features of the text tend to vary depending on the "situations" [87]: each statement is placed in a specific time, place, and scenario. Machine Learning algorithms can generate language accurately based on context; however, the system's task must go beyond simple textual generation, it is necessary to model the language by parameterizing it with a style, in an attempt to make it user-centered.

The values of the style attributes (formal, informal, social, etc.), therefore, are to be chosen based on the context in which the communicative act takes place, and pragmatics is the branch of linguistics that studies this link even if, as highlighted in the previous section, style in NLP is to be understood as a broader and data-driven concept: an attribute of the text.

But, which branch of NLP deals with style transfer? Text style transfer TST is the NLP task that aims to perform a stylistic adaptation of prose, controlling the attributes of the text generated in the NLG phase. In particular, TST aims to vary the stylistic properties of the text, preserving its content (invariance in transfer).

Research in this field has regained attention in recent years thanks to the performance of deep neural models, but its origin is far back in time and lays the foundations in computer vision. Why is research so interested in TST? Because, as previously highlighted, it makes the processing of natural language, particularly its generation (NLG), personalized to the user.

Formalizing the TST task, according to Mou et al. [79].

INPUT

- $x_s = x_s^1 x_s^2 ... x_s^n$ the source sentence with the style attribute associated a_s;
- a_t the desired style attribute.

OUTPUT

- $x_t = x_t^1 x_t^2 ... x_t^n$ the target sentence, with style attribute associated a_t.

CONDITIONS

- a_t must be the desired style;
- $a_t \neq a_s$ for it to be a style transfer, otherwise the input could be copied to the output;
- x_s and x_t must share the content. If this condition were not guaranteed, the problem would degenerate, potentially resulting in obtaining the same x_t regardless of the input x_s provided, with an alteration, in any case, of the semantics/discourse.

In detail [47]. Defining x_s as the source sentence with the style attribute associated a_s and x_t as the target sentence with the style attribute associated a_t, the TST (Text Style Transfer) models $p(x_t|x_s; a_t)$, that is, it models the language x_t conditioned to the style a_t and the source sentence x_s, in an attempt to preserve its content. An example is shown in the figure 2.5.

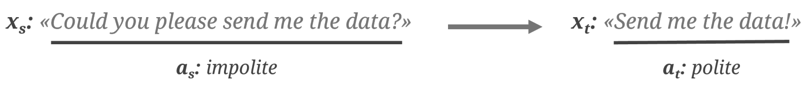

x_s: «Could you please send me the data?»
a_s: impolite
x_t: «Send me the data!»
a_t: polite

Figure 2.5: Example of TST.

2.5.1 J2C-TST: converting journalistic to conversational style

Once the relevant NLP downstream task for this dissertation has been identified, it is essential to delve into the ways in which it can be leveraged and adapted to the specific use case. As previously illustrated, the primary goal of this thesis is to support the struggling news market by reducing the time and cost associated with disseminating information on social media. Social platforms use simple and conversational language, which stands in stark contrast to the formal language typically used by traditional journalists for publishing in newspapers or on official websites.

The integration of social platforms by news agencies should facilitate a broader and more accurate dissemination of news, helping to reduce the spread of misinformation and, in general, supporting the information sector. The idea behind this work is to convert text content from journalistic language to conversational language, eliminating the need for human intervention in this translation. To this end, the Text Style Transfer task has been introduced, which will be used to transform between the two styles.

Unfortunately, in the literature, there is no downstream TST task like the one under consideration; therefore, it is useful to formally define it. Considering a parallel dataset consisting of text pairs <x_s, x_t> , where x_s is in the source style a_s = 'journalistic', and x_t is in the target style a_t = 'conversational', it is possible to reformulate the TST process for the specific case addressed in this thesis, adapting the formalization proposed by Mou et al. [79].

INPUT

- $x_s = x_s^1 x_s^2 ... x_s^n$ the source sentence with the style attribute associated a_s = "journalistic";
- a_t = "conversational" the desired style attribute.

OUTPUT

- $x_t = x_t^1 x_t^2 ... x_t^n$ the target sentence, with style attribute associated a_t.

CONDITIONS

- $a_t \neq a_s$ and therefore it is a stylistic transfer;
- x_s and x_t must share the content.

The task formally defined here can be called Journalistic2Conversational TST, or briefly, J2C-TST. The entire dissertation and associated research aim to provide tools to support this new task, starting with the definition of an ad-hoc datasets.

Chapter 3

Text processing and representation

In this chapter, we transition from theory to implementation, detailing the fundamental steps necessary for a machine to understand text. Additionally, the use of the techniques discussed within modern natural language processing architectures is examined, with particular reference to those employed in this dissertation.

Specifically, the focus will be on exploring the initial generic/main steps of the NLP pipeline (figure 3.1), a structured sequence of computational operations applied to a text, designed to transform and analyze textual data to extract meaningful information and support natural language processing applications. Given the completion of the data collection phase, we will delve into the details of harmful noise removal, the four stages of text processing (sentence segmentation, tokenization, normalization of word forms, and stop word removal), and finally, text representation.

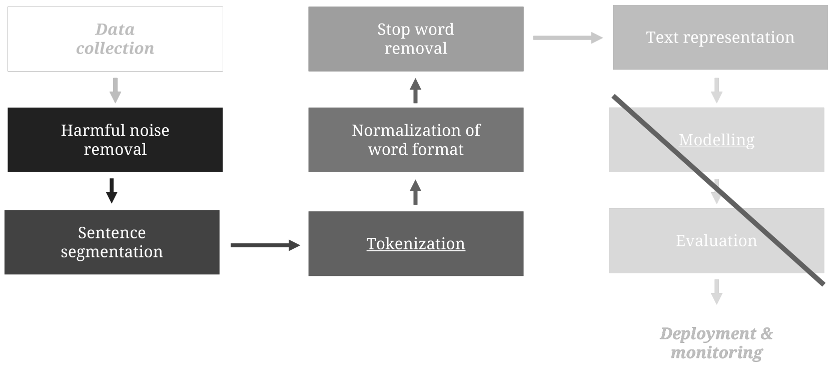


Figure 3.1: NLP pipeline, first steps.

3.1 Enhancing text quality: removing harmful noise

The "Harmful Noise Removal" phase is not often included in traditional NLP pipelines as a separate stage from "text processing." However, it might be beneficial to consider it as an autonomous step, since many text processing operations are optional and depend on the specific task, whereas the removal of harmful noise is always a good practice. As will be further emphasized in the following sections, some textual elements, although seemingly irrelevant, may prove useful in specific downstream tasks. However, what is not useful and therefore must be removed is anything external to the analysis that has the potential to obscure the true meaning of the text.

Why this "noise"? To create the corpus needed to train NLP models, textual data is often gathered from noisy sources. This noise can be a hindrance to algorithms, making it difficult to decipher and interpret the meaning of words within the text. Managing this issue in NLP has drawn significant attention, especially with the growing need to use data from social media platforms [2].

Defining exactly what constitutes "noise" can be complex. However, as emphasized by Al Sharou et al. [2], it is crucial to distinguish between:

- Harmful noise: content that must be removed or normalized as it negatively affects the intended meaning of the text and/or system performance.
- Useful noise: content that should be retained to enhance system performance. This type of noise includes desired elements that convey relevant meaning for the specific task and/or performance.

Even though much of the harmful noise is addressed during normalization and tokenization stages, this section discusses those irrelevant data that are often, if not always, to be removed and that are not normally handled in other stages. Here is a list of these elements:

- Whitespace Removal. This phase involves eliminating excessive or irregular spaces between words, such as more than a single space between words.
- Formatting Characters. This step involves the removal of HTML, XML, and markup tags from which the text of interest is extracted.
- Irrelevant Information. Removing information that is not part of the text to be analyzed, such as comments or annotations, is essential to keep the model focused on the specific task.
- Noise from Optical character recognition (OCR). Scanned documents can introduce noise due to errors in optical character recognition. Implementing advanced OCR techniques and post-processing steps is necessary to mitigate these errors and ensure accurate text extraction.

- Elimination of Meta-informative data. It is often necessary to remove metadata, document headers, or other non-relevant formatting information.
- Filtering of Repetitions and Duplicates. This step includes identifying and removing repetitive phrases and sections.
- Filtering of Advertising or Promotional Content. In many cases, it is useful to remove irrelevant promotional messages from the text.
- Boilerplate text. This step involves removing non-informative text that may appear on every page of a document (e.g., copyright notices, "next page" links).
- Removal of NaN Values. This step requires the elimination of missing or null values that can negatively impact subsequent analyses.

Once the primary harmful noise has been eliminated, it is possible to proceed with text processing.

3.2 Text processing: preparing words for NLP

Text processing is often the first step in any NLP application and consists of several levels of textual processing. The goal is to enable the machine to work with the text: after gathering the texts/documents of interest into a corpus and after removing the harmful noise, they must be made computable by transforming them in a formal way. It is necessary to access the words!

But what is a word? In linguistics, various ambiguities make it difficult to provide a single formal definition. What is considered a word can vary depending on the specific task. Punctuation, for example, plays an important role in delimiting boundaries and identifying certain aspects of meaning (such as questions, exclamations, etc.): in some downstream tasks, punctuation marks are therefore treated as words. A meaningful definition could be the one reported by Goldberg in his book [31]: "the smallest unit of meaning."

Text processing is one of the initial stages of the NLP pipeline, essential for preparing text and words before representing them in a machine-readable format. The primary goal of this macro-stage is to understand, as emphasized, what should be considered a "word" and to transform the text into a standardized format through the application of four main phases:

1. Sentence segmentation
2. Tokenization (word segmentation);
3. Normalization of word formats;
4. Stop words removal.

This chapter will describe a general approach to text processing for NLP, while simultaneously clarifying the steps useful for the downstream task studied in this dissertation.

3.2.1 Dividing text: behind sentence segmentation

Sentence segmentation represents a crucial step in text processing as it allows for dividing the text into meaningful units. Specifically, sentence tokenization focuses on dividing the text into individual sentences, an essential aspect for tasks that require analysis at this level of detail, such as automatic summarization or machine translation.

But how can a sentence be identified? To understand this aspect, it is useful to refer to the work of Jurafsky et al.[50]. The main cues for identifying sentence boundaries include punctuation marks such as periods, question marks, and exclamation points. However, while question marks and exclamation points are clear and unequivocal signals, periods present greater ambiguity. The latter, in fact, could be interpreted either as sentence boundary markers or as part of an abbreviation, a topic that will be explored in the section dedicated to word tokenization. Since many ambiguities at the word level also exist at the sentence level, as will be evident, it might be useful to address sentence and word tokenization together.

Another ambiguity related to periods arises in indirect speech [50]: in standard typographic practice, especially in North America, closing quotation marks are placed after the final punctuation of the sentence. As a result, the sentence boundary does not coincide with the period but with the quotation marks that follow the period. Additionally, it is important to consider that other punctuation marks could also divide what would otherwise be considered a single sentence, as in the case of colons.

This section does not delve into the details of the techniques and all the ambiguities for sentence segmentation, as they are closely related to the discussion that will be presented for word tokenization. In any case, it is important to highlight that once sentences are identified, it is crucial to separate them with a specific token that delineates them in the text.

3.2.2 Tokenization: dividing text for machine understanding

It is important to highlight that, in practice, words are not used as the internal unit of representation in many NLP applications. The input strings are tokenized into tokens, which can be words or other elements [71]. Tokens are, therefore, the basic units of processing.

The process of dividing the text into tokens is called tokenization, and it is performed by a tokenizer, which must make important decisions about what to consider as a token. Generally, for languages that use spaces between words, such as Western languages, this is a good splitting point (space-based tokenization), but it is not sufficient because there are various ambiguities in linguistics [71] [50]:

- Periods. Often, words are not surrounded by white spaces but are accompanied by punctuation. While removing punctuation from tokens may seem like the most intuitive solution, it can actually create problems. Most periods, in fact, indicate the end of a sentence, while others are part of abbreviations. How to handle periods, therefore, becomes a non-trivial aspect to consider and heavily depends on the downstream task at hand.
- Clitic. A clitic is a linguistic term that refers to a word or a part of a word that is grammatically independent but phonologically dependent on another word. In english, clitics often appear in the form of contractions. How should these apostrophes be treated?
- Hyphenation. Some compound words are joined by hyphens, especially when they are combined to form a single idea or noun. In other cases, the hyphen is an integral part of the word itself, as in "e-mail." Additionally, hyphens are used when a word is too long to fit at the end of a line in a text. Deciding whether to generate a single token or multiple tokens can be a challenging problem in this context, also because those listed are just some of the possibilities that lead the writer to use the hyphen.
- Compound expressions. Depending on the application, tokenization algorithms may need to treat compound expressions, such as "New York", or phone numbers as a single token. The same principle applies to phrasal verbs.
- And how should all the symbols commonly used online today, such as :), be treated? Not to mention the proper names of programming languages, aircraft, etc., that use symbols and punctuation within the name itself. It is often advisable to retain some of them and address them during the normalization phase, which will be discussed later.

The ambiguities mentioned are just some of the possible ones: generating the same semantic type expressed in different formats and languages presents a challenge. For languages that do not use spaces to delineate words, the situation becomes even more complex. In Chinese, for example, a single character can be considered a ’word’, while in languages like Japanese and Thai, the character is too small a unit, but the word is not adequate. This necessitates the use of word segmentation algorithms during the tokenization phase. The tokenizer, therefore, heavily depends on the language, the documents, and the context.

At this point, it is necessary to emphasize the current practice in tokenization for large language models: tokenization is not limited to words alone, but instead, bottom-up algorithms are used to define tokens [50]. Instead of considering tokens as individual words or characters, large text corpora are utilized to automatically determine which segments should be treated as tokens. This approach is particularly advantageous in handling unknown words, a significant challenge in natural language processing. NLP algorithms, in fact, learn the structure of the language from a training corpus and then apply this knowledge to a separate test corpus. However, the system can encounter difficulties with words that have never been seen before. To address this issue, modern tokenizers leverage sets of tokens that include segments smaller than words, known as subwords. Subwords can be either arbitrary substrings or meaningful units, such as morphemes. This way, any unrecognized word can be represented by a sequence of known subword units or, if necessary, as a sequence of individual letters. Most tokenization schemes belonging to this category consist of two main components [50]: a token learner and a token segmenter. The token learner processes a raw training corpus to generate a vocabulary, which is a set of tokens. The token segmenter, on the other hand, takes a raw sentence from the test corpus as input and divides it into the tokens present in the vocabulary. Two algorithms are widely used in this context: Byte-Pair Encoding (BPE), proposed by Sennrich et al. in 2016 [102], and the Unigram Language Model developed by Kudo in 2018 [55]. A particularly relevant library that implements both of these methods is SentencePiece, created by et al. [56]. As Jurafsky et al. suggest [50], the term SentencePiece is often used as a synonym for tokenization based on the Unigram Language Model.

To conclude and provide consistent definitions useful for understanding the results of this dissertation, it is important to differentiate token, type, and term as outlined by Manning et al. in 2008 [70]:

- A token is an instance of a sequence of characters in a particular document that are grouped together as a useful semantic unit for processing;
- A type is the class of all tokens containing the same character sequence;
- A term is a (perhaps normalized) type that is included in the vocabulary;

The set of index terms could be entirely distinct from the tokens, but in practice, in modern systems, they are closely tied to the tokens in the document [70]: they are usually derived from them through various normalization processes.

3.2.3 Standardizing text: key techniques in normalization

The normalization process aims to transform words or tokens into a standardized format. This process is essential because, in many cases, two character sequences may differ slightly from each other but still need to be treated as equivalent. The most common way to normalize is to implicitly create equivalence classes, typically named after one of the members of the set [70]. However, the normalization phase may also require the removal of some elements to emphasize the importance of others in the text.

It is important to emphasize from the outset that the utility of normalization depends on the specific task, the models used, the chosen tokenizer, and various other factors. Although the standardization process might lead to the loss of some orthographic information, it can prove to be extremely valuable in enhancing the consistency and accuracy of natural language processing in certain tasks.

At this point, it is appropriate to introduce some normalization techniques [50] [70] [107] (for some techniques debated in the literature, their applicability to various tasks is discussed):

- Capitalization. A common and simple example of word normalization is converting text to lowercase (case folding). While this technique can be useful in many tasks like Text Classification, it may also lead to the merging of words that should ideally remain distinct. For instance, many proper nouns are derived from common nouns and are only distinguishable by their capitalized initial letter. Therefore, in certain contexts, this uniformity can result in the loss of important information and potentially compromise the model's accuracy.
- Spell Correction. It's common for user-generated texts shared online to contain spelling errors. By correcting these spelling and grammatical mistakes, the unintended consequence of having the same term transcribed differently is minimized, ensuring greater consistency and clarity in the text.
- Accents/Diacriticals. Diacritical marks on characters in english have a somewhat marginal status, and one might, therefore, want to merge different words into one. However, in many other languages, diacritics are a regular part of the writing system and distinguish different sounds.
- HTML or XML Entities. Special character entities like &amp;, &lt;, and &gt; that appear in HTML or XML text should be decoded.
- Slang and Abbreviation. Abbreviations, informal writing styles, shorthand, and slang are commonly used in text. It is important to handle these informal terms by standardizing them or replacing them with their full, proper forms to accurately reflect their intended meaning.
- Removal of Numbers. The removal of numbers, for historical reasons, was often applied in text normalization processes. However, this approach can lead to a loss of meaning in the sentence, as numbers may

contain crucial information for correctly understanding the context or the user's intent.

- Standardization of Multiple Punctuation Marks. A commonly used normalization technique involves standardizing multiple punctuation marks into a single one. The repeated punctuation marks of note are the period, question mark, and exclamation point. However, Effrosynidis et al. in 2017 [26] highlighted that the repetition of these symbols represents important information: it indicates the presence of emotions in the text. Therefore, it is crucial to consider the context of the specific task before proceeding with normalization.
- Removal of Word Elongation. In texts, especially on social media platforms, words with repeated characters may appear (e.g., "coooooool"). The preprocessing step for the removal of elongation involves replacing these elongated words with their original form, to avoid, for example, a classifier interpreting the two forms as distinct words and undervaluing the elongated ones due to their lower frequency in the text. Once again, pay attention to the specific task at hand!
- Handling Negations. Handling negations is a crucial preprocessing technique in sentiment analysis, as omitting negation words can alter the tone of surrounding sentences, leading to classification errors. For some downstream tasks (such as TC), a commonly shared technique to manage them involves tagging the words following the negation, up to a punctuation mark, with a special indicator (e.g., adding the prefix "NEG_"). This way, the model can correctly recognize and interpret the negated context, thereby improving the accuracy of the analysis.
- Removal of Emojis and Emoticons. In the literature, a classic preprocessing step involves the removal of non-textual characters such as emojis and emoticons. However, in 2015, Wang et al. [118] demonstrated that certain emoticons in online social interactions are strong and reliable indicators of sentiment polarity in the text. In nearly half of the cases analyzed by the authors, emoticons were the only element expressing either positive or negative sentiment. A similar argument can be made for emojis. Hogenboom et al.[41] also emphasized the importance of these polarity signals: considering emoticons significantly improved overall sentiment classification accuracy at the paragraph level, increasing it from 22% to 94%. Pay close attention to the downstream task at hand...
- Lemmatization & Stemming. In linguistics, word forms refer to the different inflected versions of a word, such as "cat" and "cats." While these are distinct word forms, they share the same lemma, which is the base or dictionary form of a word. Lemmatization, the process of identifying the lemma for a given word form, is especially crucial in morpho-

logically rich languages, where words can have many inflected forms. In contrast, for most tasks in english, recognizing different word forms without reducing them to their lemma is often sufficient.

However, lemmatization algorithms can be complex. For this reason, a simpler and cruder method is sometimes used, which primarily involves cutting off the final suffixes of words. This naive version of morphological analysis is called stemming. An example of this is the Porter stemmer, a widely used stemming algorithm [91].

Lemmatization produces better results than stemming algorithms, but not significantly enough in most cases to outweigh the computational cost.

In general, lemmatization and stemming have long been common preprocessing steps for traditional machine learning models. However, since the advent of deep learning models, lemmatization is rarely used as a preprocessing step.

- Other Issues. There are many possible normalization techniques, and they depend on the specific task and dataset under consideration. For example, dates, times, and similar elements can appear in multiple formats, which could be standardized to ensure consistency.

These are just a few examples of normalization. It should be emphasized, once again, that the choice of normalization techniques strongly depends on the specific task and the language in question. Some languages, like Japanese, for instance, use multiple writing systems simultaneously (Chinese characters, hiragana, katakana, and Latin characters), making it challenging to develop a normalization system that can effectively handle all these variations[70].

3.2.4 Zipf's law and the concept of stop words in text processing

Sometimes, extremely common words that might seem to have little value for certain specific downstream NLP tasks are completely excluded from the vocabulary. These words are called stop words.

Zipf's law is an empirical law defined in 1935 by Zipf et al. [126] that describes the distribution of word frequencies in a natural text. It is based on the observation that a few words are extremely common, while most are very rare. In other words, the frequency of a word is inversely proportional to its rank in the list of most frequent words: the most common word occurs twice as often as the second most frequent word, three times as often as the next word, and so on, down to the least frequent word. Formally, denoting a word by w, its frequency in the text by f(w), and its rank by r(w), Zipf's law is defined by the following formula:

$$f(w) \propto \frac{1}{r(w)}$$

This observation leads to Luhn's analysis from 1957 [68]. Luhn suggests that the frequency with which a word appears in a document can be used as an indicator of its importance. According to Zipf's law, very frequent words are often common words, and thus not useful in determining the significance of a document's content. To improve textual analysis, Luhn proposes eliminating these common words from the text. This can be done by comparing the words in the text to a list of common words or by establishing an upper frequency cutoff above which words are considered too common to be meaningful.

Although this was not the only insight derived from Luhn's 1957 work, it helps formalize the concept of stop words. These can be categorized into two main types:

- Generic Stopwords. These are language-specific and ubiquitous words commonly found across various contexts.
- Domain-Specific Stopwords. These are words that are specific to a particular domain.

Using a list of stop words significantly reduces the vocabulary size without causing much harm, but not only that. Information retrieval (IR) and text classification systems, for instance, can experience a performance boost when stop words are eliminated. On the other hand, in Machine Translation, Language Modeling, and Text Summarization, common words cannot be removed because they help ensure fluency and the quality of the output.

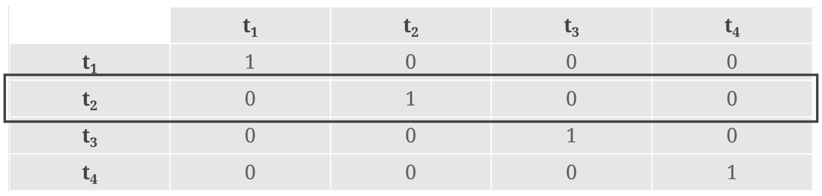
Over the years, the practice of removing stop words in certain tasks has progressively decreased, thanks to improvements in SOTA models and increased computational power. In Information Retrieval systems, for example, there has been a shift from the standard use of extensive stop word lists to an approach where such lists are increasingly less used, to the point of being no longer necessary [70].

3.3 The power of embeddings in representing word meaning

A text representation is necessary to make the text computable by an algorithm. The challenge is to understand how to represent a word: a model of word meaning is needed that allows us to draw inferences for tasks related to meaning [50].

Vector representations, which depict a word as an ordered list of numbers, as highlighted by Miutra et al. [78], are essential for enabling models to leverage language. Different vector representations offer various levels of generalization: some treat each term as a unique entity, while others learn to identify common attributes. It is, therefore, crucial to understand how they work and what types exist [78].

One of the earliest (and simplistic) vector representations is the local representation, or one-hot encoding, where each term in a fixed-size vocabulary V is represented by a binary vector $\mathbf{v} \in \{0,1\}^{|T|}$. In this type of representation, each position in the vector corresponds to one of the terms in the vocabulary, and only one position has the value 1 (corresponding to the term itself), while all other positions are 0. For example, the generic term t_i in this representation is represented by a vector that has the value 1 in the position corresponding to t_i and 0 in all other positions, as shown in the figure 3.2.



	t_1	t_2	t_3	t_4
t_1	1	0	0	0
t_2	0	1	0	0
t_3	0	0	1	0
t_4	0	0	0	1

Figure 3.2: An example of one-hot encoding.

However, this representation has some significant limitations. One of the main issues is that no information is shared between words, meaning there is no commonality between similar terms. In other words, the one-hot representation treats each word as a completely isolated entity, without recognizing any semantic similarity or relationship between different words. This is problematic because, as Harris pointed out in 1954 [35], there is no one-to-one correspondence between a word and its associated meaning; the context in which a word is used is crucial for determining its meaning: words that appear in similar contexts tend to have similar meanings. This connection between the similarity in the distribution of words and the similarity in their meaning is called the distributional hypothesis: Harris, in 1954, observed that synonymous words tend to appear in the same environments, with the amount of difference in meaning between two words roughly corresponding to the amount of difference in their environments [35].

In distributed representations each term is represented by a vector $\mathbf{v} \in \mathbb{R}^{|k|}$ (rows of a matrix), which can be either sparse or dense, with features (columns of a matrix) that are either manually constructed or derived by creating a latent space where the individual dimensions are not interpretable on their own. Leveraging Harris's hypothesis [35] and Osgood's 1957 [83] intuition of using a point in a multidimensional space to represent a word, in addition, a term can thus be represented as a point in a multidimensional semantic space (figure 3.3). This space is derived, after defining the features, from the distributions of neighboring words, utilizing the distributional hypothesis.

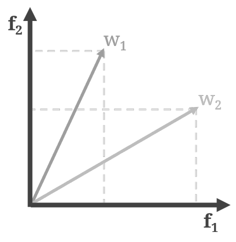


Figure 3.3: An example of multidimensional semantic space.

To summarize, a word is represented by a vector of features, where the weight associated with a feature in the vector is derived from the co-occurrence of that feature in the context of the word.

Going into detail, two macro-techniques have been introduced to identify the features of vector representations [78]: manual (observed features) and automatic (latent features).

- Observed features. Observed feature spaces can be categorized based on the choice of distributional features and the different weighting schemes applied to the raw counts [78], choices that require, in this case, human manual intervention.

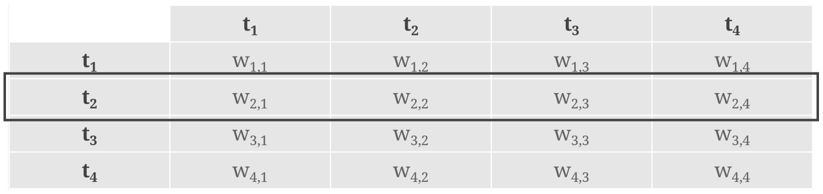
  The first vector models of meaning were based on a co-occurrence matrix, a technique used to represent the frequency with which words co-occur in a text corpus. A common application of this approach is the term-document matrix (figure 3.4), widely used in IR, from which row vectors representing individual terms can be derived: words with similar meanings tend to have similar vectors because they appear in similar documents.

	d_1	d_2	d_3	d_4
t_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$
t_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$
t_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$
t_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$

  Figure 3.4: An example of term-document matrix.

  An alternative to using the term-document matrix is the term-term matrix, also known as the word-word matrix or the term-context matrix

[50] (figure 3.5), where the columns are labeled not by documents, but by words from the vocabulary. The matrix therefore has a dimensionality of |V|×|V|, where |V| is the size of the vocabulary, and each cell records the number of times a row word (target) and a column word (context) co-occur within a specific context in the training corpus. This context might be defined as the entire document or, more frequently, as a window of words.



	t_1	t_2	t_3	t_4
t_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$
t_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$	$w_{2,4}$
t_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$	$w_{3,4}$
t_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$	$w_{4,4}$

Figure 3.5: An example of term-context matrix.

- Latent features. Although the observed vector spaces can capture interesting relationships between terms, they present a significant disadvantage: the resulting representations are highly sparse and high-dimensional. The number of dimensions, for example, can be, as mentioned, equivalent to the size of the vocabulary, which is cumbersome for most practical tasks. An alternative technique consists of learning lower-dimensional representations that retain the useful features of the original vector spaces. An embedding is a denser representation with fewer dimensions in a new space (latent space), designed to preserve the properties and relationships between elements. It is constructed in such a way that the properties and relationships between elements are preserved. In the context of word embeddings, explicit feature vectors constitute the original representation. An embedding learned in various ways from these features assimilates the properties of the terms and the inter-term relationships observable in the original feature space [78].

  It is important to emphasize that latent space embeddings not only reduce the required space but also improve performance in almost every NLP task compared to sparse vectors, as demonstrated by Baroni et al. in 2014 [8]. Even though not all reasons for this are fully understood, Jurafsky et al. [50] have provided some insights: for example, a smaller number of parameters can facilitate generalization and prevent overfitting.

At this point, it is appropriate to summarize what has been learned, using a terminological distinction suggested by Miura et al. [78]:

- When the vectors are high-dimensional, sparse, and based on observable features, they are referred to as observed (or explicit) vector representations;

- When the vectors are dense, small (k « |V|), and learned from data, they are referred to as latent vector spaces, or embeddings.

But... what are the resulting properties of the embeddings?

- Different types of similarity or association. One of the parameters of vector semantic models is the size of the context window used to gather counts. Shorter context windows tend to produce representations that are somewhat more syntactic, as the information comes from the immediately adjacent words: the words most similar to a target word tend to be semantically similar words with the same part of speech. When vectors are calculated from longer context windows, however, the words closest in multidimensional space to a target word tend to be thematically related but not similar. Therefore, the choice of context window size depends on the goals of the downstream task.
- Analogical/Relational Similarity. One of the fundamental characteristics of semantic embeddings is their ability to capture relational meanings between words. In 1973, Rumelhart et al. [98] proposed the parallelogram model to solve analogy problems like "a is to b as a* is to what?". In this model, the word b* is generated by identifying the vector closest to the vector calculated from the difference between the vectors representing words a and b (i.e., a - b) added to the vector for a*.

$$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}} \text{ is a vector close to } \overrightarrow{\text{queen}}$$

  However, there are some significant limitations to consider. For example, the parallelogram algorithm often tends to return one of the input words, such as b, or a morphological variant of it, rather than the correct word b*. To achieve accurate results, these erroneous outcomes must be explicitly excluded. Additionally, although the parallelogram method is effective for tasks involving frequent words, short distances, and specific relationships (such as correlations between countries and their capitals or between verbs/nouns and their inflected forms), it is not as effective for more complex relationships [50]. In fact, Peterson et al. in 2020 [90] argue that the parallelogram method is generally too simplistic to adequately represent the human cognitive process required to form analogies of this type.

Embeddings are undoubtedly powerful and important tools; however, they also present some significant challenges. While they can learn the meaning of words from text, they unfortunately also reproduce the implicit biases and stereotypes present in the training data. Zhao et al. in 2017 [125] demonstrated that in reflecting the statistics of their inputs, embeddings not only reproduce biases but amplify them. For instance, gender-related terms become even more connoted in the embedding space compared to their original statistical occurrences in the text. This implies that these vector representations also encode the implicit associations characteristic of human reasoning, including the more unpleasant ones [50]. Although there are various debiasing techniques to address the issue, they have not yet been able to completely eliminate biases, often merely masking them, as highlighted by Gonen et al. in 2019 [32]. Consequently, these techniques cannot be considered reliable for creating truly neutral models. The problem of gender bias, for example, is deeper and more systematic than current debiasing metrics suggest, making it clear that more comprehensive solutions are needed to effectively tackle it [32].

3.3.1 Context matters: advancing word representations

In the previous section, the concept of embeddings was introduced, characterizing their properties and discussing some limitations. However, there is an additional issue related to "traditional" embeddings. In these representations, the context is used to generate a word type, which corresponds to an entry in the vocabulary: there is a single vector for each type [50]. However, words can be ambiguous [50]: the same type, as previously noted, may have multiple meanings depending on the context. This refers to polysemous words, meaning words that can have distinct or related meanings in different contexts. For example, the word "pesca" can refer both to the fruit of the peach tree and to the activity of catching fish.

To overcome this limitation, in 2018, Peters et al. [89] introduced the concept of contextual embeddings with ELMo, a word representation technique that uses a bidirectional LSTM model trained on large amounts of text data. Contextual embeddings are vectors that represent the meaning of a word in a specific context [50], unlike previous approaches that used a single vector for each word type. These embeddings are particularly useful for measuring the semantic similarity of words within a given context and are essential for linguistic tasks that require a more sophisticated understanding of meaning [50]. In this vector representation, the meaning of individual word instances, not types, is captured, deriving a vector for each instance: type in a given context [50]. More formally, as described by Jurafsky et al. [50], given a sequence of input tokens $x_1, \ldots, x_n$, it is possible to use the output vector z_i from the final layer of the model as a representation of the meaning of the token x_i in the context of the sentence $x_1, \ldots, x_n$. Alternatively, instead of using only the vector z_i from the final layer, it is common to compute a representation for x_i by averaging the vectors z_i from the last four layers of the model.

3.4 Preprocessing in NLP: why it shouldn’t be ignored

Today, with the advancements in deep neural networks, data preprocessing seems to be a practice that is increasingly overlooked. However, this trend might not be entirely positive. A recent 2023 study conducted by Siino et al. [107] has experimentally demonstrated that text preprocessing remains a crucial practice, capable of leading to significant improvements in downstream tasks. The study examined various tasks related to TC, recording improvements of up to 25% in the models analyzed. The disinterest in preprocessing is mainly due to the growing confidence in the power of the attention mechanism [115] in transformer networks. However, the authors found that in these models, the difference between the best and worst combinations of preprocessing techniques is more pronounced. Therefore, specific preprocessing can still make a difference, significantly improving both the effectiveness and overall performance of the models [107].

Chapter 4

Deep learning approaches in NLP

In this chapter, language models are presented (penultimate step of the presented NLP pipeline), with particular attention to their neural versions, which are crucial tools for understanding and generating natural language. The chapter concludes by addressing the "resource problem", discussing strategies for efficient training and model compression. It will delve into the details of the information useful for generating the models that will be proposed in this dissertation, from fine-tuning to pre-training information.

It is important to note that, unlike the previous chapters where all theoretical concepts were introduced and explained in detail, this section does not delve into the basic elements of machine learning and, specifically, deep learning (for further information, refer to the reading of the text by Goodfellow et al. [33]). Instead, it examines the ML/DL aspects related to natural language processing.

4.1 Foundations and advances in neural language models

In this section, language models are introduced, with a particular focus on their neural versions. The various parts examine the main architectures proposed in the literature and present the state of the art in the field. Additionally, the foundational LLM selected to support the task of this research is presented, highlighting its distinctive characteristics and discussing the methods for adapting it to the specific task.

4.1.1 Language modeling in NLP: traditional approaches

A language model is a type of ML model designed to understand and generate text in natural language. To achieve this, as reported by Goodfellow et al. [33], a language model defines a probability distribution over a sequence of tokens in a natural language: tokens that frequently occur in a given context have a higher probability within the corresponding language model. Consequently, these probabilities implicitly reflect the central themes or topics of the text.

More specifically, as highlighted by Goldberg et al. [31], the goal of language modeling is to assign a probability to sentences in a given language. In practical terms, this means estimating the probability of a token sequence $t_1, \dots, t_n$, or $P(t_1, \dots, t_n)$. Additionally, language models are capable of calculating the probability that a token (or a sequence of words) follows a given sequence. But how is this process carried out? To estimate $P(t_1, \dots, t_n)$, one can apply the chain rule, which allows the formula to be rewritten as follows [1]:

$$P(t_1, \dots, t_n) = P(t_1)P(t_2|t_1) \dots P(t_n|t_1, t_2, \dots, t_{n-1})$$

It is, therefore, possible to think of language modeling as a sequence of token prediction tasks, in which each token is predicted based on the preceding history. In traditional language models, to make everything more computationally feasible, these models rely on the Markov assumption, according to which the future is independent of the past given the present. More formally, the Markov assumption of order k states that the next token in a sequence depends only on the last k tokens:

$$P(t_{i+1} \mid t_{1:i}) \approx P(t_{i+1} \mid t_{i-k:i})$$

This traditional approach represents the simplest language model: the n-gram model. As Jurafsky [50] points out, an n-gram is a sequence of n tokens. However, with some terminological ambiguity, the term "n-gram" can also be used to refer to a probabilistic model that estimates the probability of a token based on the previous n-1 tokens, and thus can assign probabilities to an entire sequence of them [50]. Although the Markov assumption of order k is inaccurate for any value of k, since sentences can have dependencies of arbitrary length, it has nonetheless proven to produce good results for relatively small values of k and has been the dominant approach in language modeling for decades [31].

But how can $p(w_{i+1} = m \mid w_{i-k:i})$ be estimated? A large volume of text can

[1] It is important to observe that the ability to assign a probability for a token following a sequence of tokens $p(t_n|t_1, \dots, t_{n-1})$ and the ability to assign probabilities to arbitrary sequences of tokens $p(t_1, \dots, t_n)$ are equivalent, as one can be derived from the other, because probabilistic sequences can be constructed [31]. Thus, the conditional probability of a token can be expressed as a fraction of two sequences: $p(t_n|t_1, \dots, t_{n-1}) = \frac{p(t_1,\dots,t_n)}{p(t_1,\dots,t_{n-1})}$

be utilized [2]! To do this, the Maximum Likelihood Estimation (MLE) can be used, whose formula is [31]:

$$\hat{P}_{MLE}(w_{i+1} = m \mid w_{i-k}, \ldots, w_i) = \frac{\#(w_{i-k}, \ldots, w_{i+1})}{\#(w_{i-k}, \ldots, w_i)}$$

Although effective, this approach has a major limitation: if a certain event has never been observed in the corpus, the probability assigned will be 0. Consequently, the entire sequence would be assigned a probability of 0, due to the multiplicative nature of sentence probability calculation. There are two main techniques to address this issue [31]:

- Smoothing. This technique assigns a non-zero probability to rare or unseen events in the corpus, distributing a small portion of the total probability to these events. The goal is to prevent an unseen sequence of words from being assigned a probability of zero, which would make the model ineffective on new data.
- Backoff. This technique handles data sparsity by reducing the model's complexity in the case of unseen events. If an n-gram is not present in the corpus, the model "backs off" and uses a lower-order n-gram (for example, from a trigram to a bigram or unigram) to estimate the probability. In this way, the model relies on a smaller context when the more complex one is unavailable.

Language modeling approaches based on smoothed maximum likelihood estimates ("traditional") are easy to train, scale well to large corpora, and work well in practice. However, they present several important limitations, as highlighted by Goldberg et al. [31]:

- Inability to capture long-range dependencies. The main limitation of n-grams is their inability to effectively capture long-range dependencies between words. This issue is not resolved by the use of smoothing or backoff techniques, which, while useful for handling unseen events, adopt a rigid approach, limiting flexibility in adapting to more complex or dynamic contexts.
- Scalability and computational challenges. Scaling to higher-order n-grams represents an inherent problem for maximum likelihood-based language models (MLE). The nature of natural language and the vast number of words in the vocabulary mean that statistics for higher-order n-grams are inevitably sparse. Moreover, increasing the order of n-grams significantly increases memory requirements, making the prediction process computationally expensive.

[2] It is important to highlight that by using vast collections of texts from the web or other sources, it is possible to build extremely large language models, known as Large Language Models.

- Lack of generalization across similar contexts. N-gram-based models do not generalize effectively across similar contexts. For example, if only 'black cat' and 'gray cat' have been observed in the corpus, the model will not be able to deduce that 'white cat' is plausible if it has not been seen before. Each word combination is treated independently, preventing similar events from influencing the probability of new, unseen events.

Before concluding and introducing models that overcome the limitations of traditional ones, it is useful to note, as highlighted by Goldberg et al. [31], that language models can be used not only to understand language but also to generate sentences (NLG). Once trained on a collection of texts, a language model can generate ("sample") random sentences based on the probabilities defined by the model itself, as suggested by Shannon in 1948 [103].

When generating a sentence from a language model, there are various strategies for selecting the sequence [31], for example:

- Deterministic choice. This involves selecting the word with the highest probability at each step.
- Random sampling. This consists of choosing a word randomly, but based on the model's probabilities, introducing more variety.
- Beam search. This technique, invented by Lowerre in 1976 [67], considers multiple possible sequences simultaneously, attempting to identify the entire sentence with the highest overall probability while avoiding getting "trapped" in suboptimal sequences that may arise from decisions based on locally optimal probabilities without considering future outcomes. This approach is particularly useful for handling ambiguous decisions, where later context might clarify the correct choice. This phenomenon is known as "label bias", as discussed in depth by Andor et al. in 2016 [4].

4.1.2 Neural networks and their role in NLP

Neural networks are a fundamental computational tool for language processing [3]. They are called "neural", as highlighted by Jurafsky et al. [50], because their origins trace back to the McCulloch-Pitts neuron, a simplified model of

[3] To understand the importance of using neural networks, it is helpful to highlight their expressive power with a theoretical foundation. The Universal Approximation Theorem states that a neural network with at least one hidden layer, a sufficient number of neurons, and a nonlinear activation function can approximate, under certain assumptions, any continuous function to an arbitrary level of accuracy. In other words, a neural network can fit any function with arbitrary precision, which is why neural networks are referred to as universal approximators. There are various proofs of universal approximation, starting with the works of Cybenko [20] and Hornik et al. in 1989 [42]

the biological neuron as a computational element. The modern use in language processing is no longer inspired by these early biological insights but by their mathematical evolution, which began with the conception of Rosenblatt's perceptron in 1958 [97].

The basic building block of a neural network is a single computational unit that takes an input value and produces a single output value. It receives a set of real numbers as input, performs a calculation on them, and generates an output. Essentially, a neural unit performs a weighted sum of its inputs, with an additional term in the sum called the bias term, introduced by Widrow et al. in 1960 [119]. Formally, given an input vector x, a weight vector w, and a scalar bias b, the weighted sum z can be represented as:

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

Once z is obtained, the computation is not finished; it is necessary to apply a nonlinear function f [4] to z, obtaining the activation value of the unit a [50]. However, since the modeling so far has focused on a single unit, the activation of the node is actually the final output of the network, which will generally be called y [50] (figure 4.1). Therefore, the value y is defined as:

$$y = a = f(z)$$

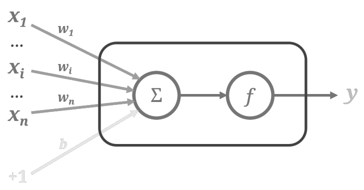


Figure 4.1: Single computational unit in a neural network.

There are several well-known activation functions in the literature; here we introduce those most relevant to the task at hand, as they are used, for example, in reference models.

- The sigmoid/logistic function and the softmax derivative (figure 4.3). The sigmoid $y = f(z) = \frac{1}{1+e^{-z}}$ maps an input value into a range between 0 and 1, allowing the output to be interpreted as a probability. A

[4] If the Universal Approximation Theorem, previously discussed, already highlights the importance of using non-linear functions, it is useful to provide a second theoretical context. A neural network composed of only linear transformations, regardless of its depth (number of layers), is equivalent to a single-layer linear network. Let's consider a deep linear network with L layers. Each layer i performs a linear transformation represented by a matrix W_i. The output of the network for an input vector $\mathbf{x}$ would be $\mathbf{y} = W_L \cdot (W_{L-1} \cdot \ldots \cdot (W_2 \cdot (W_1 \cdot \mathbf{x})))$. Due to the associative property of matrix multiplication, this is equivalent to:$\mathbf{y} = (W_L \cdot W_{L-1} \cdot \ldots \cdot W_2 \cdot W_1) \cdot \mathbf{x}$. Let $W = W_L \cdot W_{L-1} \cdot \ldots \cdot W_2 \cdot W_1$. Then we have:$\mathbf{y} = W \cdot \mathbf{x}$. Which is precisely the form of a single-layer linear network.

significant advantage of the sigmoid is its ability to "squeeze" extreme values towards 0 or 1, making them more manageable. Additionally, being differentiable, it facilitates model learning through the backpropagation algorithm (introduced later). However, very high values of z lead to y values that are saturated, meaning extremely close to 1, which causes derivatives close to 0. These, in turn, lead to learning problems, because when training networks by propagating an error signal backward and multiplying the partial derivatives from each layer of the network, nearly zero gradients cause the error signal to become progressively smaller until it disappears, a problem known as the vanishing gradient problem (highlighted by Bengio et. al in 1994 [9]).

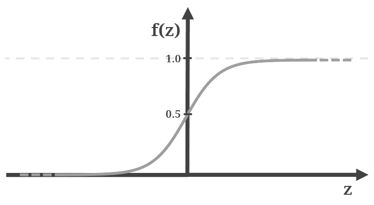


Figure 4.2: Sigmoid function.

If the sigmoid is useful for binary classification problems, the Softmax function represents a generalization of the sigmoid for multiclass problems. This function calculates the relative probabilities of each class, ensuring that the sum of the probabilities is 1.

Typically, the sigmoid function and the softmax function are used in the output layer of a neural network (NN) and are useful in classification problems, binary and multiclass respectively.

- One of the most commonly used activation functions in the hidden layers of neural networks is the ReLU (Rectified Linear Unit) $y = \text{ReLU}(z) = \max(z, 0)$ (figure 4.3). This function returns the input value when it is positive, and 0 when it is negative. ReLU is preferred because it is less prone to the vanishing gradient problem, which affects the sigmoid and softmax functions.

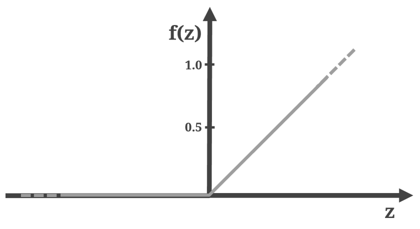


Figure 4.3: ReLU function.

Once the functioning of a single neuron is understood, it is important to take advantage of the representational capacity of more complex networks. As highlighted by Jurafsky et al. [50], the simplest type of neural network is the feedforward network, a multilayer network [5] in which units are connected without cycles: the outputs of the units in each layer are passed to the units in the upper layer, and no output is retransmitted to the lower layers. A feedforward network has three types of nodes: input units, hidden units, and output units (figure 4.4). In the hidden layer, the core of the network, allows to create a new representation of the inputs. This internal representation is crucial because it allows the network to capture complex relationships between the data, which may not be evident at the level of the original inputs. The output layer, on the other hand, takes the new vector produced by the hidden layer and computes the final output. Mathematical operations on these layers can be efficiently performed using matrix operations [50].

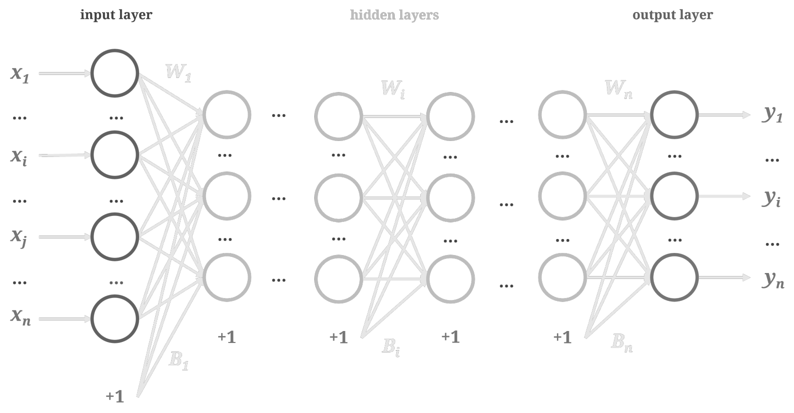


Figure 4.4: A feedforward network.

A feedforward neural network, in detail, is an instance of supervised ma-

[5] A neural network is considered "deep" when it consists of more than one hidden layer between the input layer and the output layer. The founding fathers of deep learning are Hinton, Bengio and Lecun with their respective papers 2006 (Hilton et al. [40], Bengio et al. [11]), 1998 (Lecun et al. [59])

chine learning, where the correct output y is known for each observation x. Through the training process, the network must learn the parameters $W[i]$ and $b[i]$ for each layer i, in order to estimate y as accurately as possible. This ensures that the neural network learns the underlying function and can generalize to new input examples. However, to proceed with training, a loss function is needed to model the distance between the system's output and the GT, specifically cross-entropy [6] [7].

$$L_{CE}(\hat{y}, y) = -\sum_{k=1}^{N} y_i \log \hat{y}_i$$

But which parameters are suitable for the task? Those that minimize the loss function! However, to identify them, an optimization algorithm with gradient descent is necessary. The gradient of the loss function L with respect to the weights $\frac{\partial L}{\partial w_j}$, in fact, indicates the direction in which the loss function increases most rapidly. By following the gradient in the opposite direction, it is possible to find the weights w_i that reduce the error.

$$w_{j,\text{new}} = w_{j,\text{old}} - \eta \left(\frac{\partial L}{\partial w_j} \right)$$

As Jurafsky points out [50], however, for deep networks, calculating the gradients for each weight is very complex, as the derivative is calculated with respect to weight parameters that appear from the very early layers of the network, even though the loss is only obtained at the end of the network after a forward pass. Backpropagation, introduced by Rumelhart et al. in 1986 [99], uses the chain rule in calculus: the intuition behind backpropagation is to propagate the gradients from the final node to all the nodes in the graph (figure 4.5).

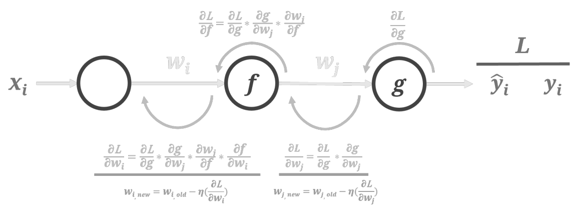

$\frac{\partial L}{\partial f} = \frac{\partial L}{\partial g} * \frac{\partial g}{\partial w_j} * \frac{\partial w_j}{\partial f}$
$\frac{\partial L}{\partial g}$
L
x_i
w_i
f
w_j
g
$\hat{y}_i$
y_i
$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial g} * \frac{\partial g}{\partial w_j} * \frac{\partial w_j}{\partial f} * \frac{\partial f}{\partial w_i}$
$w_{i,new} = w_{i,old} - \eta(\frac{\partial L}{\partial w_i})$
$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial g} * \frac{\partial g}{\partial w_j}$
$w_{j,new} = w_{j,old} - \eta(\frac{\partial L}{\partial w_j})$

Figure 4.5: An example of simple backpropagation.

[6] To understand the cross-entropy loss, it is useful to assume we are dealing with a multiclass classification problem (requiring softmax) with hard classification, where only one of the classes is correct. In this case, the network produces an estimate vector with K elements $\hat{y}$, each of which represents the estimated probability $p(y_k = 1|x)$. The loss function for a single example x is the negative sum of the logs of the K output classes, each weighted by its probability y_k.

[7] Cross-entropy is suitable for classification problems

Having concluded this review of neural networks, it is important to emphasize that their optimization is a non-convex optimization problem, which is very complex [50]. For this and other reasons, many best practices exist for successful learning. To avoid overfitting, various forms of regularization are used. One of the most important is dropout [110]: the random removal of certain units and their connections from the network during training. Hyperparameter tuning is also crucial [50]. The parameters of a neural network are the weights WW and the biases b, which are learned through gradient descent. Hyperparameters, on the other hand, are chosen by the algorithm designer. Although hyperparameters are not presented in detail, they include the learning rate, mini-batch size, model architecture (number of layers, number of hidden nodes per layer, choice of activation functions), regularization method, and so on. Gradient descent itself has many architectural variants, such as Adam [53].

Understanding neural language models

Neural language modeling is the algorithm underlying most modern NLP applications. As highlighted by Goldberg et al. [31], non-linear neural network models address some of the shortcomings of n-gram language models: they allow conditioning on increasingly larger contexts with only a linear increase in the number of parameters, reduce the need to manually design backoff orders, and support generalization across different contexts.

In particular, a feedforward neural language model, first introduced by Bengio in 2003 [10] and used here to introduce neural language modeling, is a feedforward network that takes as input, at time t, a representation of a certain number of preceding words (w_{t-1}, w_{t-2}, etc.) and outputs a probability distribution over the possible next words [50]. Formally, similar to n-gram models, a context window of N-1 words w_i is used to predict the next word w_t at time t.

$$P(w_t \mid w_{t-N+1}, \ldots, w_{t-1})$$

Neural language models also have another distinctive feature: instead of relying on the textual identity typical of n-gram models, they represent words w_i using their embeddings.

But how does the training process for these new models work? As highlighted by Jurafsky et al. [50], the high-level intuition for training neural language models, whether they are simple feedforward models or more powerful language models, is the idea of self-training or self-supervision. In self-training for language, a text corpus is used as the training material, and at each time step t, the model is asked to predict the next word. This model is called self-supervised because no special labels need to be added to the data; the natural sequence of words provides the supervision [50]. By extension, the forward pass works similarly to that of classic feedforward networks used for classification problems. Given $N-1$ preceding words w_i, the goal is to predict the next word at time t, w_{t+1}, from all the words available in the vocabulary V (figure 4.6):

1. The context words w_i are represented as one-hot vectors.
2. The one-hot vectors are multiplied by the embedding matrix E. The embedding weight matrix E has one column for each word, where each column is a d-dimensional vector, so the matrix has dimensionality $d \times |V|$. By multiplying by a one-hot vector, which has only one non-zero element $w_i = 1$, the relevant column for the word i is simply selected, obtaining the embedding for the word i. It is important to emphasize that these embeddings can be trained during the network's training process or pre-trained with other algorithms. In the latter case, the embedding weights will be frozen.

   The use of embeddings in language models represents a significant improvement over traditional one-hot word representations. Embeddings, previously introduced, are lower-dimensional vectors that capture the semantics of words, that is, the relationships of similarity and difference between them. This allows the model to generalize better on sentences or sequences of words never seen before [10].
3. The classic layers of a feedforward network follow. The output layer usually uses a softmax to produce a probability distribution over the words.

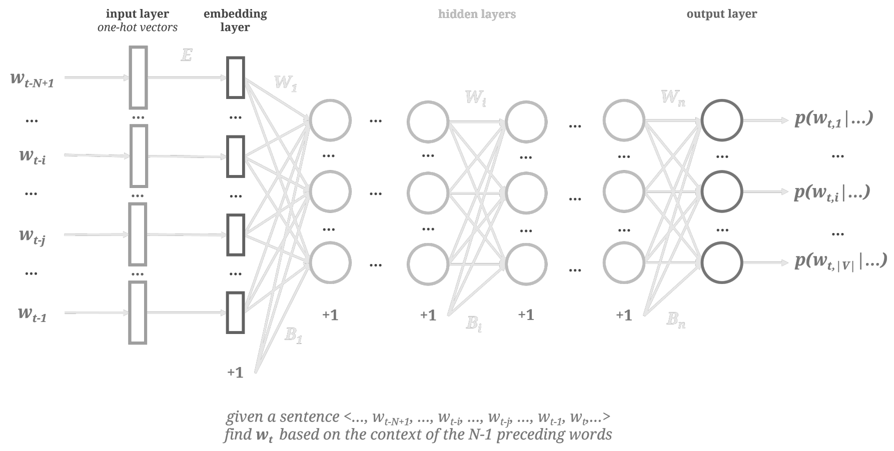


Figure 4.6: A feedforward neural language model.

Once the functioning of neural networks for language is understood, it is useful to highlight some key findings. Goldberg et al. [31] emphasize that, despite the high computational cost due to the large output vocabulary, these models outperform n-gram-based models in terms of prediction accuracy. A significant aspect is that words in different positions share parameters, allowing them to leverage common statistical strengths: the hidden layers identify informative word combinations, recognizing, at least in theory, that only certain sub-parts of the context window are truly relevant for certain w_t [31]. Another interesting property is these networks' ability to generalize across different contexts. The combination of the highlighted results, along with the merely linear dependence of context size in terms of resources, allows for an easy increase in context size without encountering significant issues [31].

However, neural language models in the form presented have some limitations [31]: predicting the probability of a word in a context is much more computationally expensive compared to using an n-gram language model, and working with large vocabulary sizes and training corpus can become prohibitive. Additionally, although the new models are more flexible and capable of better generalizing to word combinations that have not been directly seen in the training corpus, they may generalize incorrectly by "associating" similar words inappropriately. For instance, when observing "black cat", the model might assign a higher probability to "black glass", even though the combination of the two words may not be common or appropriate in a certain context [31].

4.1.3 Understanding transformers: the core of modern NLP

So far, Language Models have been introduced, and in particular, their neural version, useful for overcoming some limitations of n-gram models. Additionally, large language models trained on vast text corpora have been discussed, capable of offering remarkable performance in all natural language tasks thanks to the knowledge acquired during training. The current reference architecture for the latter, as highlighted by Jurafsky [50], is the transformer, a self-trained model that uses an innovative mechanism called self-attention to build contextual word representations by leveraging and managing distant information in sentences [50].

The fundamental module of the Transformer architecture is the transformer block, a multi-layered network that maps sequences of input vectors $(x_1, ..., x_n)$ to sequences of output vectors $(z_1, ..., z_n)$ of the same length.

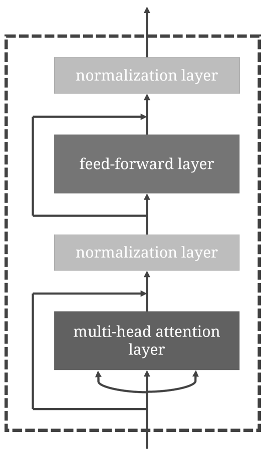


Figure 4.7: A transformer block.

The insight of Vaswani et al. [115] is that, through a series of successive transformer blocks, increasingly richer contextualized representations of the input tokens can be built. At each level of the transformer, to compute the representation of token *i*, the information from the token's own representation and from its neighboring tokens, derived from the previous level, are combined [50]. But what is a transformer block composed of (figure 4.7)?

- Self-attention layer. This layer has the peculiarity of extracting and directly using information from arbitrarily large contexts. The approach was not specifically invented for transformers, but rather by Bahdanau et al. in 2014 [7]. However, until 2017, it had only been applied to recurrent models, limiting its potential. The transformer [115], therefore, is the first model that relies entirely on self-attention to compute the representations of its inputs and outputs without using RNNs or convolutions. This mechanism allows weighing and combining the representations of different words from the context at the previous level k-1 to calculate the representation at the new k (figure 4.8).

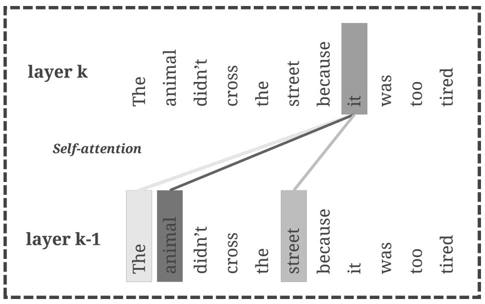

layer k
The animal didn't cross the street because it was too tired
Self-attention
layer k-1
The animal didn't cross the street because it was too tired

Figure 4.8: An example of self-attention

A self-attention layer maps input sequences $(x_1, \ldots, x_n)$ into output sequences of the same length $(a_1, \ldots, a_n)$. As highlighted by Jurafsky et al. [50], it's important to note that the concept of context can be used in two ways within self-attention. In causal, or retrospective self-attention, the context consists of any previous word up to and including the current one. In general bidirectional self-attention, the context can include future words [50]. For now, to maintain clarity in the discussion, it's best to focus on the former case. During the processing of each input element, the model has access to all previous inputs up to the current one, but not to those that follow. Additionally, the calculation for each element is independent of the others, as words at level k "observe" those at level $k-1$. The first aspect allows this approach to be used for building language models and for autoregressive generation, while the second implies that both forward inference and the training of transformers can be easily parallelized [50]. But how does it work? To compare words, represented by vectors, it is possible to use the dot product [8]. Formally, given two words i and j

$$\text{score}(x_i, x_j) = x_i \cdot x_j$$

The resulting value belongs to the range $(-\infty, +\infty)$, where the larger it is, the more similar the vectors are. Thus, for each word a_i, it is possible, using the dot product, to generate a weight towards the words x_j at level $k-1$ up to and including index i [50]. The vector generated for each pair of the word a_i can then be passed through a softmax, which normalizes it and provides a weight vector a_{ij}, indicating the proportion of relevance of each input x_j to the element i under attention. The final score of a_i is the weighted sum of the input using the weight vector.

$$a_i = \sum_{j \leq i} \alpha_{ij} x_j$$

Although this is the basic concept, as highlighted by Jurafsky et al., transformers use a more sophisticated approach to represent the contribution of longer inputs. Specifically, a single input can assume three distinct roles [50]:

- Query, represents the current focus of attention and is compared to other inputs at level k-1,
- Key, is the input compared to the query to calculate their similarity,
- Value, is the actual representation of the token that will be used to calculate the final output. Once the attention weights are calculated (based on the similarity between the query and key), these weights are used to weight the values associated with the keys.

[8] Let vectors A and B in n-dimensional space be: $A = [a_1, \ldots, a_n]$ and $B = [b_1, \ldots, b_n]$. The dot product is given by: $A \cdot B = a_1 b_1 + \cdots + a_n b_n$.

To capture these three different roles W^Q, W^K, W^V., transformers introduce shared weight matrices to project each input value into its role. Given these projections, the score between a current focus of attention x_i and an element in the previous context x_j consists of a dot product between the query vector q_i and the key vectors of the previous element q_j.

$$score(x_i, x_j) = q_i \cdot k_j$$

The subsequent softmax calculation aimed at producing a_{ij} remains the same, but the final output a_i is now based on a weighted sum of the value vectors v_j.

$$a_i = \sum_j a_{ij} v_j$$

At this point, it is important to consider a key aspect. The result of a dot product can take arbitrarily large values: exponentiating such values can lead to numerical issues and cause a loss of gradients during training [50]. To mitigate this risk, the result of the dot product is scaled by dividing it by a factor related to the embedding dimension. A common approach is to divide by the square root of the dimensionality of the query and key vectors (d_k).

$$\text{score}(x_i, x_j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

All this process is aimed at producing a single vector v_j. However, since each output is calculated independently, the entire process can be parallelized by packing the input embeddings (rows) into a single matrix $X \in \mathbb{R}^{N \times d}$ [50]. Thus, X is multiplied by the weight matrices for keys, queries, and values (all of dimension $d \times d$) to produce the matrices $Q = XW^Q; \quad K = XW^K; \quad V = XW^V$ containing the key, query, and value vectors. Following the steps analogous to those presented previously, at this point, it is possible to reduce the entire self-attention step for an entire sequence of N tokens to:

$$A = \text{SelfAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V$$

Proceeding in this way, however, the process is feeding into the calculation the values x_j from the $k-1$ level, following the query x_i, which is incorrect for the way the problem is set up. To resolve this, the upper triangular part of the query matrix is set to $-\infty$ (figure 4.9).

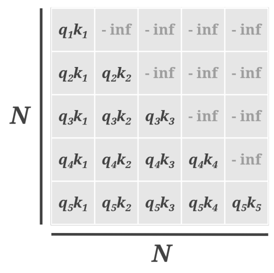


Figure 4.9: An example of query matrix

One final step remains. As Jurafsky et al. [50] highlights, different words in a sentence can be related to each other in multiple ways simultaneously. For this reason, transformers use the multi-head self-attention mechanism, which leverages multiple parallel self-attention layers, called heads. Each head operates at the same depth in the model, with its own independent set of parameters and data. In the original work, Vaswani et al. [115] employed 8 of these attention heads. By using distinct parameters for each head, every block can learn different aspects of the relationships between the inputs at the same level of abstraction.

To implement this concept, each head i in a self-attention layer is provided with its own set of query matrices $W_i^Q \in \mathbb{R}^{d_x \times d_k}$, key matrices $W_i^K \in \mathbb{R}^{d_x \times d_k}$, and value matrices $W_i^V \in \mathbb{R}^{d_x \times d_v}$, which are used to project the inputs into embeddings of key, value, and query separately for each head, while the rest of the self-attention computation remains unchanged [50]. By multiplying these matrices with the input X, we obtain $Q \in \mathbb{R}^{N \times d_k}$, $K \in \mathbb{R}^{N \times d_k}$, and $V \in \mathbb{R}^{N \times d_v}$. The output of each of the h heads, therefore, has a dimension of $N \times d_v$. Once all the heads have computed their results, these results are concatenated and then passed through an additional linear projection (a matrix multiplication called $W^O \in \mathbb{R}^{hd_v \times d}$). This brings the output back to the original input dimensions, ready for the subsequent layers of the model [50].

- Feedforward Layer. After the self-attention layer, a feedforward network is applied, consisting of two fully connected layers with a non-linear activation function between them. The original paper [115] refers to "N position-wise networks", suggesting that there is a different network for each position in the input sequence. However, this interpretation can be misleading. In reality, there is a single network that processes each element of the sequence independently. This is made possible by the use of matrices, where each output vector is computed from a corresponding input vector, without interactions between positions during the process (figure 4.10).

$$FFN(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$

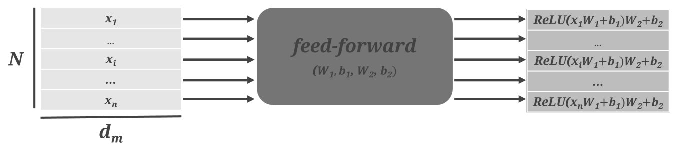

N
x_1
...
x_i
...
x_n
d_m
feed-forward
(W_1, b_1, W_2, b_2)
$ReLU(x_1W_1+b_1)W_2+b_2$
...
$ReLU(x_iW_1+b_1)W_2+b_2$
...
$ReLU(x_nW_1+b_1)W_2+b_2$

Figure 4.10: An example of point-wise network computation

- Residual connections. In the "transformer block", residual connections are present. These connections transmit information from a lower layer to a higher layer, bypassing the intermediate layer [50]. This technique, introduced by He et al. [36] in 2016, allows forward-propagated information and backward gradients to skip a layer. This improves learning and enables upper layers to directly access information from lower layers.

  The central idea is to learn a residual function $H(x) = F(x) + x$ (figure 4.11), instead of directly learning the target function $F(x)$. This approach addresses the performance degradation problem that tends to occur as the network depth increases a key factor in improving model accuracy.

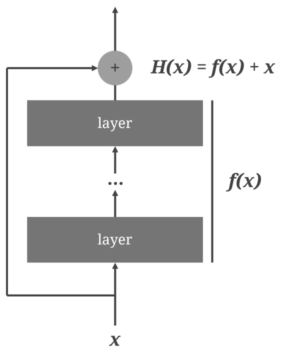


Figure 4.11: An example of residual connection

- Layer normalization. Each output generated by the residual connections and the network, as illustrated in the figure, is normalized using layer normalization, a technique introduced by Ba et al. in 2016 [5]. Layer normalization is one of the various normalization techniques used to improve the training speed in deep neural networks. This method ensures that the values produced by a hidden layer remain within an optimal range, thus facilitating gradient-based training.

The first step in layer normalization involves calculating the mean μ and the standard deviation σ of the elements in the normalized vector. Given a hidden layer x_i, from a layer with dimensionality d_h, these values are calculated as follows:

$$\mu = \frac{1}{d_h} \sum_{i=1}^{d_h} x_i \quad \sigma = \sqrt{\frac{1}{d_h} \sum_{i=1}^{d_h} (x_i - \mu)^2}$$

The components of the normalized vector are obtained by subtracting the mean and dividing by the standard deviation. The result of this operation is a new vector with zero mean and unit standard deviation. Finally, in the standard implementation of layer normalization, two learnable parameters, γ and β, are introduced, which represent the gain and offset term [9], respectively.

$$\text{LayerNorm}(x) = \gamma x + \beta$$

The entire process is carried out for each of the N tokens of length d. Another important consideration concerns the two most common versions of the transformer architecture:

- Post-Norm. Normalization is applied after the attention and feed-forward steps (Vaswani et al. official paper [115]).
- Pre-Norm. Normalization is applied before the attention and feed-forward steps, and it has been shown that this improves performance in many cases [50]. Pre-norm ensures that the token vectors are stabilized before entering the key layers of the model.

Once the structure and functioning of the transformer block are understood, it's useful to move on to the architecture of the network in its entirety. In particular, it is useful to introduce one of the two main components of the architecture proposed by Vaswani et al. [115] in 2017: the decoder (figure 4.12), a module that generates a sequence of text from a latent or encoded representation.

[9] While normalization ensures stable and efficient training, without gain and offset terms, the network could lose some representational power, as all layers would have identical statistical properties. The learnable γ and β parameters allow the network to fine-tune the scale and location of the normalized output, ensuring that the normalization does not overly constrain the model's ability to learn complex features.

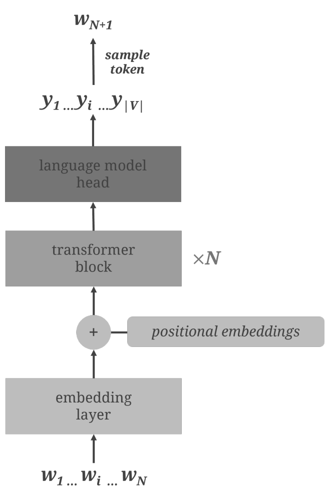


Figure 4.12: A transformer decoder architecture.

First of all, as can be seen from the image, the transformer blocks can be stacked N times, which differentiates various transformer architectures. This is possible because the input and output of the transformer block are both of shape $[N \times d]$. Having understood this, what are the other components of the decoder?

- Embedding Layer. Before passing the vectors through the series of N Transformer blocks, each word in the provided input context must be embedded [115]. This is done using a matrix technique, similar to that used by neural language models described in the previous section, but with a small variation. Token embeddings, so far, do not account for their position in the text. To include this information, token embeddings are combined (summed) with positional embeddings. There are several ways to implement this process [115]. The simplest method, known as absolute positional embedding, involves using randomly initialized embeddings corresponding to each possible position within the input sequence, up to a certain maximum length. These embeddings are learnable during training and are stored in a matrix of size [1 × N]. However, this approach has a limitation: positions at the beginning of the sequence tend to be trained much more than those further away. An alternative is to use static functions that map positions to vectors of real values. These types of embeddings are not directly learnable, as in the case of absolute positional embeddings, but rely on mathematical functions that attempt to capture the relationships between different positions, as proposed in the original Transformer paper [115]. There are even more sophisticated methods for positional embeddings, representing relative rather than absolute position, but these will not be covered here.

- Language modeling head. When decoder-based transformer models are pretrained for various tasks, the term "head" refers to the additional block placed on top of the transformer's base architecture to enable the model to perform a specific task [115].

  Remembering that language models are word predictors, the task of the head is to use the output embedding of the N-th word to predict the N+1-th word. There are several layers involved in this process. The first module is linear and takes the N-th embedding (dimension $1 \times N$) as input, projecting it into the logits vector, or score vector, which will have a score associated with each word in the vocabulary $|V|$. This linear layer can be learned, but more commonly, efforts are made to tie this matrix to the initial embedding matrix E. Thus, in the transformer's input phase, the embedding matrix is used to map a one-hot vector over the vocabulary (of shape $[1 \times |V|]$) to an embedding (of shape $[1 \times d]$). Later, in the language modeling head E^T, the transposed embedding matrix (of shape $[d \times |V|]$) is used to map back from an embedding (of shape $[1 \times d]$) to a vector over the vocabulary (of shape $[1 \times |V|]$). During training, E will be optimized to be effective in both of these mappings. A softmax layer then transforms the logits into probabilities over the vocabulary, useful for assigning a probability to a given text. However, the most important use of the decoder is to generate text, which occurs by sampling a word from these probabilities, as previously introduced.

But how do you train this model? To train a transformer as a language model, the self-supervision algorithm described for neural language models is used: given a text corpus as training material, at each time step t, the model is asked to predict the next word. At each position t of the input word, the model takes the correct sequence of tokens $w_{1:t}$ and uses it to calculate a probability distribution over the possible next words to compute the model's loss. For the next word, what the model predicted is ignored, and instead, the correct token sequence $w_{1:t+1}$ is used to estimate the probability of the token w_{t+2}. This concept, in which the correct historical sequence is always provided to the model to predict the next word, rather than giving the model its best guess from the previous prediction, is called teacher forcing, coined by Williams et al. in 1989 [120].

Exploring the transformer encoder

Once the functioning of the decoder is understood, it is necessary to introduce the concept of the encoder, which is essential for fully exploring the models used in this dissertation. Encoder models produce a representation for each input token, but they are generally not used for generating text. Specifically, this section introduces the bidirectional transformer encoder, which is composed of the same modules as the decoder, but without the language modeling head and which is trained using masked language modeling (MLM), a method proposed by Devlin et al. in 2019 [23], which allows the model to consider both the left and right context of the text.

Up until now, the output of a hidden layer has been calculated based only on the elements preceding the input (inclusive), ignoring potentially useful information found to the right of each labeling decision: bidirectional encoders overcome this limitation. The only difference is that the QK^T matrix, in this case, does not mask the future, allowing the model to contextualize each token using information from the entire input. The first bidirectional encoder model was BERT [23].

But how are these new Transformers trained? A new training scheme is needed. Instead of trying to predict the next word, the model learns to perform a fill-in-the-blank task, technically called a , devised by Taylor in 1953 [112] (figure 4.13): given an input sequence with one or more missing elements, the learning task is to predict those missing elements.

The ________ swiftly climbed the tree.

Figure 4.13: An example of cloze task.

More precisely, during training, the model is deprived of one or more elements from an input sequence and must generate a probability distribution over the entire vocabulary for each of the missing elements. The original approach for training bidirectional encoders is called Masked Language Modeling (MLM): the model is presented with a series of sentences from the training corpus, in which a random sample of tokens from each training sequence is selected for the learning task. Once selected, a token is used in one of the following three ways:

- It is replaced with the unique vocabulary token [MASK] (80% of the tokens in the original paper);
- It is replaced with a random vocabulary token. This process helps the model generalize better since during fine-tuning or real-world usage, it may encounter errors, noise, or variations in language. By training the model with randomly altered data, it reduces dependency on the [MASK] token and makes the model more flexible (10% of the tokens in the original paper);
- It is left unchanged. The idea of leaving some tokens unchanged is to avoid making training too predictable. If the model saw that every selected token was always masked or modified, it would learn to rely on

these alterations to make inferences, which would not be useful in real-world scenarios (10% of the tokens in the original paper).

It is important to note that only the selected tokens (15%) play a role in learning; the other words have no role in the loss function, so in this sense, BERT and its descendants are inefficient [50].

However, there are extensions and modifications of the MLM suited to different use cases.

Transformer encoder-decoder: architecture and application

In the previous section, the transformer architecture and the encoder-decoder models were introduced. The task examined in this dissertation, as previously presented, involves generating an output sequence in a different target style from an input sequence in a given style, in a manner similar to machine translation tasks. Most of the most competitive sequence transduction models use an encoder-decoder structure, introduced by Cho et al. [18] in 2014, which is a combination of the two models previously discussed. Here, the encoder maps a sequence of symbol representations $(x_1, \ldots, x_n)$ into a sequence of continuous representations $z = (z_1, \ldots, z_i)$ of fixed size. Given z, the decoder then generates an output sequence $(y_1, \ldots, y_m)$, one element at a time. At each step, the model is auto-regressive, consuming the previously generated symbols as additional input to generate the next one, using teacher forcing [120]. Consequently, the encoder and decoder of the proposed model are jointly trained to maximize the conditional probability of a target sequence $(y_1, \ldots, y_m)$, given a source sequence $(x_1, \ldots, x_n)$.

$$P(y_1, \ldots, y_m \mid x_1, \ldots, x_n)$$

The state-of-the-art architecture for the encoder-decoder is the transformer encoder-decoder, with the transformer decoder block enriched by a special layer called "cross-attention", which allows each position in the decoder to attend to all positions in the input sequence. In the original paper by Vaswani et al. 2017 [115], the proposed architecture was precisely an encoder-decoder, both composed of 6 transformer blocks (figure 4.14).

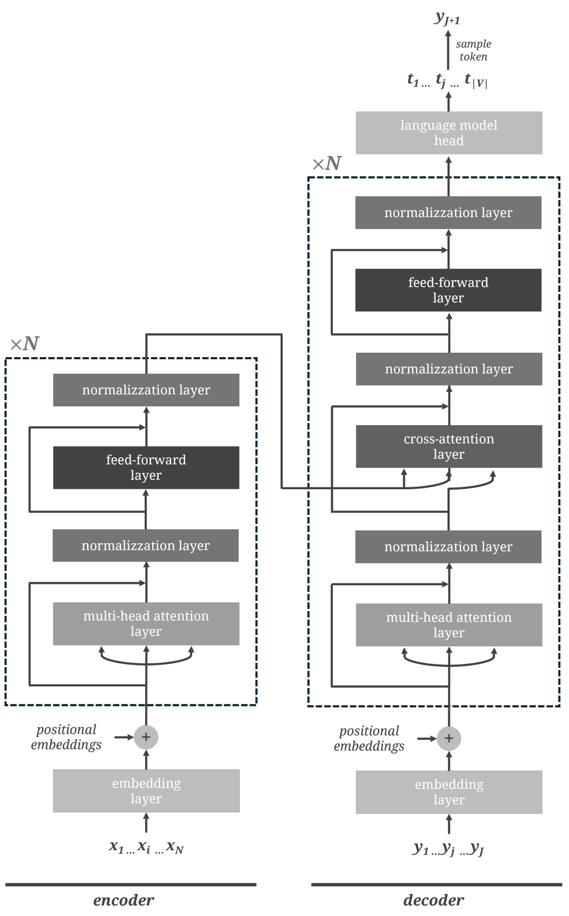


Figure 4.14: A general encoder-decoder transformer.

To go into the details of how the architecture works. The encoder-decoder model operates by transforming an input sequence $x_1, \dots, x_n$ into an output sequence $y_1, \dots, y_m$ through two main phases. The encoder receives a sequence of symbols $x_1, \dots, x_n$ and converts it into a continuous representation that captures the meaning and relationships between the various elements. This representation is then transferred to the decoder, which uses it as context to generate the output. The decoder, starting from the encoder's representation and the portion of the sequence already generated $y_1, \dots, y_j$, produces the final output sequentially, element by element y_{j+1}, while maintaining the context provided by the encoder.

But how does cross-attention, or encoder-decoder attention, or source attention work [50]? First of all, as highlighted, cross-attention is nothing more than an additional layer of the transformer decoder block and has the same form as multi-head attention, except that while the queries come from the previous layer of the decoder, the keys and values come from the encoder's output.

To conclude, it is important to note that training encoder-decoder models requires corpora composed of parallel sequences where, as previously introduced, an input sequence is aligned with an output sequence.

4.1.4 Foundational models and the T5 architecture explained

A foundational model is a type of NLP/AI model that is pre-trained on a large amount of unsupervised data and subsequently adapted to a wide range of specific tasks through fine-tuning. This is because training a machine learning model to perform natural language processing tasks often requires the development of general knowledge that allows the model to "understand" prose before it can be adapted to downstream tasks. The solution to this problem is to pre-train the model on data-rich tasks [93].

T5, a foundational model by Google, can be understood, as highlighted in the official paper by Raffel et al. [93] from 2019, as a framework that converts all text-based problems into a text-to-text format, taking text as input and producing new text as output. Unifying all language problems under a common "text-to-text" format facilitates, after pre-training, the adaptation of the model to subsequent specific tasks, as the architecture remains the same and the transfer of knowledge is smoother and more direct.

Architecturally, the implementation of T5 is based on a transformer encoder-decoder that closely follows the one proposed by Vaswani et al. in 2017 [115]. Except:

- Dropout Applied Widely. Dropout, with a probability of 0,1, is applied within the feed-forward network, as in the model proposed by Vaswani [115], but also on the residual connections, attention weights, and at the input and output of the entire stack. This approach helps prevent overfitting and improves the model's generalization;
- Relative Position Embeddings. The model uses a different method to incorporate information about the position of tokens within a sequence, distinguishing itself from traditional approaches that employ sinusoidal signals or absolute position embeddings. Instead of considering the absolute position of each word, the model focuses on the relative distance between tokens, particularly between the "query" token and the "key" token in the self-attention mechanism. This relative distance is represented through simplified relative position embeddings, which are single scalar values added directly to the attention scores (logits) used in the self-attention mechanism. To optimize efficiency, the relative position embeddings are shared across all layers of the Transformer model. However, within each layer, each attention head uses a different embedding, allowing the model to capture various nuances of positional relationships. The model learns a total of 32 relative embeddings (one

per head), with token offsets increasing logarithmically up to a distance of 128 tokens. This means that smaller distances have a more detailed representation, while larger distances share the same embedding after a certain point. For distances beyond 128 tokens, a single common embedding is used, making each layer unable to distinguish between distances greater than this limit. However, thanks to the model's depth and the combination of information from different layers, the model can still handle long-range dependencies effectively.

- Bias Removal in Layer Normalization. The model removes the bias term b from layer normalization. Since layer normalization normalizes the input to have zero mean, the bias term can be considered redundant, as adding a constant value after normalization does not significantly contribute to the model's expressiveness.
- Layer Norm After Residuals. Layer normalization is moved outside the residual path, meaning it is applied after the addition of the residual connection instead of before. This slightly alters the dynamics of the information flow through the model.

In terms of size, T5 is similar to the $BERT_{BASE}$ configuration, with both the encoder and decoder consisting of 12 transformer blocks each, with 12 head attention mechanisms. In total, the $T5_{BASE}$ model presented here has approximately 220 million parameters. But how was T5 trained? The training dataset is based on Common Crawl [10], a publicly available web archive that provides "text extracted from the web" by removing markup and other non-textual content from the files. Unfortunately, most of the resulting text is not in natural language, but consists of incomprehensible text or boilerplate such as menus, error messages, or duplicate text. For this reason, the authors of T5 used the following heuristics to clean the text and generate the "Colossal Clean Crawled Corpus" (C4) [93] useful for training:

- Only sentences ending with a terminal punctuation mark (i.e., a period, exclamation point, question mark, or a closing quotation mark) were retained.
- Pages with fewer than 3 sentences were discarded, and only lines containing at least 5 words were kept.
- Any page containing a word present in the "List of Dirty, Bad, Obscene, or Otherwise Inappropriate Words" [11] was removed.
- Since many extracted pages contained notices requiring the activation of Javascript, all sentences containing the word Javascript were removed.

[10] https://commoncrawl.org/
[11] https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words

- Pages containing the placeholder text "lorem ipsum" were removed.
- Since some pages inadvertently contained code, any page containing a curly brace "{" was removed because it appears in many programming languages but not in natural text.
- Pages from Wikipedia, with citation markers (e.g., [1], [citation needed], etc.), were filtered to remove such markers.
- Sentences containing standard policy warnings, such as "terms of use", "privacy policy", "cookie policy", "uses cookies", "use of cookies", or "cookie use", were removed.
- To deduplicate the dataset, all sequences of three repeated sentences were discarded.

Additionally, since most downstream tasks focus on english texts, langdetect [12] was used to filter out pages not classified as english with a probability of at least 0,99. During the pre-training phase, in addition to using the C4 dataset, supervised data from downstream tasks were also integrated. This approach, proposed by Liu et al. 2019 [63], allows the model to gain early exposure to task-specific data for which it will later be fine-tuned. This early exposure can help the model develop a more general and transferable representation, improving performance on specific tasks during the fine-tuning phase.

To conclude, it is useful to present the training strategy of T5.

- Tokenization. The tokenization of the text was performed using SentencePiece [56], with a vocabulary of 32.000 tokens;
- Pretraining. The pretraining process utilized 1 million steps, using a batch consisting of 2048 sequences, each with a maximum length of 512 tokens. During this process, a learning rate scheduling based on the "inverse square root" was adopted, maintaining a constant learning rate of 0,01 for the first 10.000 steps, followed by exponential decay until the end of pretraining. The training objective was inspired by the "masked language modeling" of BERT [23], and specifically by that of spanBERT from Joshi et al. in 2019 [48]. The model used a pretraining objective based on the corruption of contiguous spans of tokens. In this approach, 15% of the original text is corrupted by replacing contiguous tokens with "sentinel" tokens, unique for each corrupted block. The average length of these token spans was set to a variable value during the experiments, with the optimal reference value being found to be 3 contiguous tokens.
- Fine-tuning. After pretraining, the authors performed fine-tuning on the various tasks, using 262.144 steps and applying a maximum likelihood objective through the teacher forcing technique. To specify the

[12] `https://pypi.org/project/langdetect/`

task that the model needed to perform, the authors introduced an additional hyperparameter: a specific prefix, associated with the downstream task, that was added to the original input sequence before being processed by the model. During the optimization process, batches consisting of 8 sequences with a length of 512 tokens were used, maintaining a constant learning rate of 0.001. Finally, to monitor the model's progress and identify the optimal version, a checkpoint was created every 1.000 steps.

Both training processes were optimized using AdaFactor (for details, refer to the paper by Shazeer et al. in 2018[106]). There are various configurations of T5 beyond the base version described here, specifically SMALL, LARGE, 3B, and 11B [93].

Finally, it is worth noting that for tasks with long output sequences, the authors observed improved performance by using beam search [67] for translation and summarization tasks.

Advancements in T5: the improved T5.1.1 model

In 2020, the same authors of T5 released T5.1.1, an improved version of T5 that introduces the following enhancements [13]:

- GEGLU activation in feed-forward hidden layer. In 2020, Shazeer [105] introduced the GEGLU activation function, which combines. Gated Linear Units (GLU) with Gaussian Error Linear Units (GELU). GLUs, first proposed by Dauphin et al. in 2016 [21], are activation functions based on an element-wise product between two linear projections, one of which is passed through a sigmoid function. This approach enables the model to act as a gating mechanism, where the flow of information is regulated by a "gate" that decides which inputs are relevant and should be transmitted. In this way, the model can efficiently filter out irrelevant information and focus on the important ones.

  $$\mathrm{GLU}(x, W, V, b, c) = \sigma(xW + b) \otimes (xV + c)$$

  GLUs can be adapted by using various nonlinear functions instead of the sigmoid, such as GELU, which was introduced by Hendrycks et al. in 2016 [38]. GELU is particularly interesting because it allows for a softer and more effective nonlinearity compared to other activation functions, leveraging a Gaussian distribution curve to decide whether to activate a certain input or not, thereby improving the model's ability to learn

[13] `https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md`

complex representations.

$$\text{GELU}(x) = 0,5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0,044715x^3\right)\right)\right)$$

The author [105] who proposed GEGLU tested this function while maintaining the base model of Raffel et al. [93] ($T5_{BASE}$), including the architecture and training task, but without using dropout during pretraining.

$$\text{GEGLU}(x, W, V, b, c) = \text{GELU}(xW + b) \otimes (xV + c)$$

[14] The results showed that GEGLU offers superior performance compared to both GLU and GELU, maintaining an optimal balance between computational efficiency and the ability to model more complex nonlinearities, making it particularly suitable for large-scale models and challenging tasks [105].

- Dropout was turned off in pre-training (quality win). Dropout should be re-enabled during fine-tuning.
- Pre-trained on C4 only without mixing in the downstream tasks.
- No parameter sharing between embedding and classifier layer.

T5.1.1 is an improvement over T5 in terms of stability, efficiency, and performance, making it preferable in many situations, especially when better results are needed for

mT5: multilingual adaptation of the T5 model

For many purposes, as highlighted by Jurafsky et al. [50], a pre-trained multilingual model is more practical than a monolingual model, as it avoids the need to build several different models. Additionally, multilingual models can improve the representations of low-resource languages by leveraging linguistic information from similar languages. However, as the number of languages increases significantly, multilingual models exhibit what has been termed the "curse of multilingualism" introduced by Conneau et al. in 2020 [19]: the performance for each language degrades compared to a model trained on fewer

[14] The Hadamard product (or element-wise product) is an operation between two matrices (or vectors) of the same dimensions. Instead of matrix multiplication (which involves dot products of rows and columns), the Hadamard product multiplies each element in one matrix directly with the corresponding element in the other matrix $GLU(X) = (XW_1 + b_1) \circ \sigma(XW_2 + b_2)$. The GLU allows the model to have fine control over which parts of the signal should be propagated through the network and which parts should not. This control, although implemented with a simple multiplication (Hadamard product), becomes powerful thanks to the sigmoid function, which continuously regulates (from 0 to 1) each individual element of the input vector.

languages. Another issue with multilingual models is that they have "an accent": the grammatical structures of high-resource languages (often english) seep into low-resource languages, making the latter slightly more similar to the former, as noted by Papadimitriou et al. in 2023 [85].

MT5 is a multilingual variant of T5, presented by Xue et al. in 2021 [123], which was pre-trained on a dataset based on Common Crawl covering 101 languages (mC4), introduced in the same paper [123]. To create mC4, the authors used the tool cdl3 [15] to identify over 100 languages, with an accuracy of at least 70%, and leveraged all 71 monthly web scrapes released by Common Crawl up to the time of the study [123]. An important heuristic filtering step in C4 was the removal of lines that did not end with an english final punctuation mark. Since many languages do not use english final punctuation, in mC4 the authors decided to apply a "line length filter" which requires pages to contain at least three lines of text with 200 or more characters. For the rest, the same preprocessing as C4 was performed, deduplicating lines between documents and removing pages that contain inappropriate words. Finally, after applying these filters, Xue et al. [123] grouped the remaining pages by language and included in the corpus all languages with 10.000 or more pages. The fact that the model covers over 100 languages requires a larger vocabulary: the authors increased the vocabulary size to 250000 wordpieces.

The architecture and training procedure that Xue et al. [123] used for mT5 closely follow those of T5. Specifically, they based mT5 on the "T5.1.1" recipe. The pre-training involved 1 million steps and batches of 1024 input sequences, each with a length of 1024 tokens. The learning rate was computed as in T5 [123], using the inverse square root method. The training objective remained the same as well. Among the various models proposed, the base version has about 580 million parameters [123].

For fine-tuning, the authors [123] used a constant learning rate of 0,001 and a dropout rate of 0,1 for all downstream tasks. Additionally, Xue et al. [123] employed a batch size of 2^{17} for most tasks and saved checkpoints every 200 steps.

More information about mT5 can be found in the official paper [123], but only the details relevant to the objectives of this dissertation have been presented here.

4.2 DL and resource efficiency: a path toward sustainability

As highlighted by Menghani et al. in 2021 [76], Deep Learning has revolutionized numerous fields of AI/ML. However, with the progressive improvements

[15] `https://github.com/google/cld3`

in models, the number of parameters, latency, and the resources required for training and inference have significantly increased. Consequently, it has become crucial to consider these impact metrics beyond just the model's quality. According to an analysis by Amodei et al. in 2018 [3], the amount of computation used in major AI training has grown exponentially, with a doubling time of resources every 3,4 months.

In particular, artificial intelligence is undergoing a paradigm shift thanks to the advent of models trained on large amounts of data (e.g., T5), as highlighted by Bommasani et al. in 2021 [13]. The foundational models, introduced in the previous section, are scientifically interesting for their performance and capabilities. However, what makes them fundamental is their rapid integration into real-world implementations of AI systems, with far-reaching consequences for people. While there are positive aspects, such as the final application of the models discussed here, this section explores the downside of these models, identifying techniques useful to mitigate some of the negative effects, particularly those related to intensive resource consumption.

Firstly, training regimes are computationally expensive: they require energy, which results in carbon dioxide emissions into the atmosphere and environmental degradation [13]. For example, the amount of emissions from training a single $BERT_{base}$ model is equivalent to that of a trans-American flight (1438 pounds = 652,27 kg of CO_2e), as calculated by Strubell et al. in 2019 [111]. Moreover, if deployed at scale, foundational models can require a significant amount of energy to handle millions of requests, leading to high emissions. Consequently, the design, implementation, and post-implementation monitoring of foundational models should adequately reflect these risks. Certainly, there are hardware and administrative solutions that can mitigate this impact, but it is preferable to address the issue at its root.

Even from an economic standpoint, the challenges are numerous. Foundational models have the potential to significantly improve overall living standards, thanks to increased productivity and innovation in society [13]. However, as highlighted by Strubell et al. [111], the accuracy improvements of models, which are necessary for societal impact, depend on the availability of exceptionally large computational resources, which require substantial energy consumption, with associated economic implications. To train $BERT_{base}$, with 110 million parameters, it was estimated to require 12.041,52 watts, with a cost ranging from \$3.751 to \$12.571 [111]. Moreover, GPT-3, the 175-billion-parameter model by Brown et al. from 2020 [15], had a training process that cost over \$4,6 million [61].

How could these problems be resolved (with reference to this dissertation)?:

- Fine-tuning. As highlighted by Bommasani et al. [13], consumption factors should be evaluated over the entire lifecycle of the model, not on the basis of individual runs. If a model were to be trained from scratch for every task, resource consumption would be excessive. On the contrary, leveraging fine-tuning to adapt a foundational model to a downstream task could be much simpler and more energy-efficient. Of course, although foundational models are generalist and adaptable to many tasks, they may not always be more efficient than smaller or more specialized models (base models) in certain cases. However, for the task under consideration in this dissertation, general linguistic knowledge is required, which only a foundational model can guarantee.

- Compression. As Bommasani et al. [13] emphasize, for applications requiring strict cost and latency constraints, compression techniques are essential. These techniques allow large models to be transformed into more compact versions while maintaining the desired performance during inference. Originally developed for smaller models and low-memory environments, such as mobile phones, these techniques are now indispensable for managing the enormous scale of modern foundational models in data centers, particularly reducing latency and resource consumption for inference.

As emphasized by Schwartz et al. in 2019 [101], it is crucial to pursue sustainable Artificial Intelligence ("green AI").

4.2.1 PEFT: efficiently adapting pre-trained language models

Once a language model is pre-trained, it can be used for downstream tasks. To do this, as highlighted by Kalyan et al. [51] in 2021, three techniques can be employed:

- Functionality-Based Approach. In this method, pre-trained language models (PLMs) generate contextual word vectors, which are then used as input for models designed for specific tasks. Although these vectors can be integrated into any state-of-the-art architecture designed for particular tasks, this involves the need to train the downstream model from scratch, requiring a large number of labeled instances.

- Fine-Tuning. Pre-training allows the model to acquire general linguistic knowledge. However, to achieve high performance in downstream tasks, the model must possess specific knowledge related to the task itself, which implies adapting the weights to bring them closer to the optimal setting. Fine-tuning provides the model with this task-specific knowledge by optimizing the weights based on the task-specific loss function.

- Prompt-Tuning. As previously stated, fine-tuning aims to adapt PLMs to specific tasks, leveraging the objectives related to the task. However, the discrepancy between pre-training objectives and fine-tuning objectives can negatively affect the model's performance. Prompt-tuning offers an effective solution, especially in few-shot (few instances) and zero-shot (no instances) scenarios. Instead of modifying the internal parameters of the model, prompt-tuning optimizes the input given to the model, called a "prompt", guiding it towards more accurate and relevant responses. This approach closely aligns with the original objective of linguistic modeling in pre-training.

In this section, we focus on fine-tuning. Although fine-tuning can be considered a useful tool for saving resources, as it avoids training a model from scratch, with the increasing size of pre-trained models (PLMs), as highlighted by Ding et al. in 2023 [25], even fine-tuning and managing the storage of all the parameters required for it become prohibitive. Parameter-Efficient Fine-Tuning (PEFT), as noted by Han et al. in 2024 [34], offers a practical solution to this problem, allowing efficient adaptation of large models to specific tasks while minimizing the number of additional parameters introduced and the computational resources required, yet maintaining performance comparable to standard fine-tuning. To better understand, the predominant approach, i.e., full parameter fine-tuning, involves initializing the model with pre-trained weights, updating all parameters, and producing separate instances for each task. However, it has been shown that large-scale models can be effectively optimized even by modifying only a limited number of parameters.

Among the various methods for performing PEFT, those based on re-parameterization are motivated by the hypothesis that adapting pre-trained language models for most downstream tasks is inherently low-rank [16], and therefore, can be accomplished just as efficiently in terms of parameters (as noted by Ding). Specifically, re-parameterized fine-tuning introduces low-dimensional additional parameters during training, which are then integrated with the original model for inference [34].

LoRA: a scalable approach to efficient model adaptation

One of the most well-known reparameterization methods is Low-Rank Adaptation (LoRA), introduced by Hu et al. [44] in 2021. In a neural network, dense layers perform matrix multiplications, where the weight matrices typically have full rank. However, during adaptation to a specific task, Aghajanyan et al. [1] in 2020 demonstrated that pre-trained language models have a low "in-

[16] The rank of a matrix represents the number of rows or columns that are linearly independent, meaning those that cannot be represented as a linear combination of other rows or columns. In practical terms, the rank measures how much distinct and unique information a matrix can contain. Low rank in the context of PEFT is a metaphor: it implies that adaptation to a new task can be done by updating only a small subset of parameters. This subset represents the essential information needed to adapt to the specific task, just as a low-rank matrix can be efficiently represented using only a few independent rows or columns, thereby reducing redundancy.

trinsic dimension" and can learn efficiently even through random projections into lower-dimensional spaces. Inspired by these observations, the authors of LoRA hypothesized that weight updates also exhibit a low "intrinsic rank" during adaptation.

Once fine-tuning is completed, the adaptive weights of LoRA seamlessly integrate with the base pre-trained weights, maintaining model efficiency without adding any extra burden during inference.

In principle, as highlighted by Hu et al. [44], LoRA can be applied to any subset of weight matrices in a neural network to reduce the number of trainable parameters. Compared to GPT-3 [15] 175B fine-tuned with Adam, for instance, LoRA can reduce the number of trainable parameters by 10.000 times and the GPU memory requirement by 3 times [44].

How does it work (figure 4.15)? For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, its update is constrained by representing it with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$ where $B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$ and the rank $r << \min(d, k)$. During training, W_0 is frozen and does not receive gradient updates, unlike A and B. This means that no gradients are calculated for W_0.

The forward pass is then modified as follows:

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

Thus, a random Gaussian initialization is used for A and zero for B, so that $\Delta W = BA$ is zero at the start of training.

Although the formula is almost complete, it is useful to scale ΔW by $\frac{\alpha}{r}$ where α is a constant depending on r. α serves to regulate the magnitude of these updates, and r represents the rank of the matrices, allowing the calibration of how much the updates influence the model in a way proportional to the chosen rank.

$$h = W_0 x + \frac{\alpha}{r} \Delta W x = W_0 x + \frac{\alpha}{r} BAx$$

Hu et al. [44] finally demonstrated experimentally that in a model like transformers with multiple weight matrices, adapting multiple matrices with reduced rank is more effective than adapting only one matrix with a higher rank. Moreover, increasing the rank beyond $r = 4$ does not seem to cover a significantly larger space that could further improve performance [44]. In general, to achieve maximum benefit with a limited budget, it is better to adapt W_q and W_v (respectively, the query matrix and the value matrix) in the transformer architecture with an adequate rank (e.g., $r = 4$) [44].

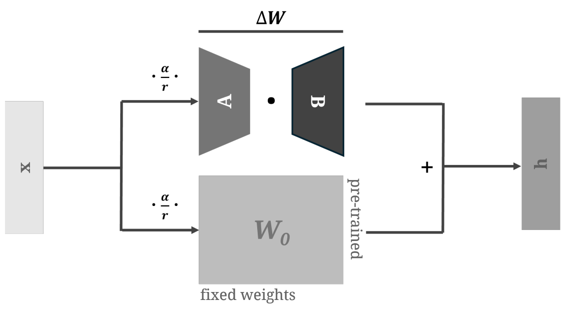

ΔW
$\cdot \frac{\alpha}{r} \cdot$
A
B
x
$\cdot \frac{\alpha}{r} \cdot$
W_0
pre-trained
fixed weights
+
h

Figure 4.15: A simplifying image of how LoRA works.

LoRA has several key advantages, as highlighted by the authors [44]:

- A pre-trained model can be shared and used to create many small LoRA modules for different tasks. It is possible to freeze the shared model and switch tasks efficiently by replacing matrices A and B.
- LoRA makes training more efficient and reduces the hardware barrier by up to 3 times when using adaptive optimizers, as there is no need to calculate gradients or maintain optimizer states for most of the parameters: only the added low-dimensional matrices are optimized.
- The design allows merging the trainable matrices with the frozen weights at deployment, without introducing any inference latency compared to a fully adapted model, by construction.
- LoRA is orthogonal to many compression methods and can be combined with many of them.

LoRA is simple to implement and has been evaluated on models with up to 175 billion parameters, as highlighted by Han et al. [34]

4.2.2 The role of compression in enhancing model efficiency

As highlighted by Menghani et al. [76], there are several general challenges in training and deploying models:

- Sustainable server-side scalability. Training and deploying large deep learning models are costly. While training can be a one-time expense, deploying and maintaining inference over time can still be burdensome in terms of computational resources.
- Deployment on devices. Some deep learning applications must operate on devices with varying computational capacities, making it imperative to optimize models for the specific target platforms.
- Privacy and data sensitivity. It is crucial to use the smallest possible amount of data for training, especially when user data is sensitive.

Training models efficiently with a fraction of the data reduces the need for extensive data collection.

- New applications. New applications impose additional constraints (regarding model quality or impact) that "off-the-shelf" models may not be able to meet.
- Model explosion. Although a single model might perform well, training and/or deploying multiple models on the same infrastructure for different applications can exhaust available resources.

The common thread among these challenges is efficiency. Menghani et al. [76] further break down efficiency into the following aspects:

- Inference efficiency. Mainly concerns those who deploy a model for inference.
- Training efficiency. Concerns those who train a model.

In 1989, LeCun et al. [59], in a study that laid the foundation for modern compression techniques, demonstrated that not all neural network parameters are essential, highlighting the possibility of safely removing certain weights without compromising model performance. This would ensure a reduction in memory and resource consumption, particularly during inference. In fact, although models perform well in the tasks for which they were trained, they may not be sufficiently efficient for direct use in real-world applications". In general, to make a model efficient, there are four techniques according to Menghani et al. [76]:

1. Compression Techniques. These include algorithms that optimize the model's architecture, usually by compressing its layers. A classic example is quantization, introduced by Benoit et al. in 2018 [46], which reduces the precision of the weight matrices of a layer (e.g., switching from 32-bit floating-point values to 8-bit unsigned integers), with minimal quality loss.
2. Learning Techniques. These algorithms focus on a different training approach for the model. The improved quality from this training can be traded for a smaller footprint or a more efficient model, potentially reducing the number of parameters. An example is distillation, introduced by Hinton et al. [39] in 2015, which allows improving the accuracy of a smaller model by training it to mimic a larger model.
3. Automation. These tools aim to improve the main metrics of the model through automation. An example is hyperparameter optimization (HPO), which helps improve accuracy, which can be traded off for a model with fewer parameters. Similarly, architecture search falls into this category, which is useful for optimizing model efficiency through modifications to it.

4. Efficient Architectures. In this case, it refers to neural network building blocks designed from scratch to be efficient. For instance, convolutional layers, presented by LeCun et al. in 1998 [59], introduced parameter sharing for image classification, avoiding the need to learn separate weights for each input pixel and making models more robust to overfitting.

5. Infrastructure. Finally, a solid infrastructure and tools base is necessary to build and leverage efficient models. This includes training frameworks such as PyTorch, designed by Paszke et al. in 2019 [88].

In this dissertation, from model generation to achieving SOTA, both efficient architectures and learning techniques are utilized to make the resulting models as efficient as possible, especially during inference. Once a ’large’ model is generated using PEFT to adapt it to the downstream task, further compression is applied to make inference efficient and enable the use of the compressed model on resource-limited devices. ’Learning techniques’ are exploited during compression as they generally maintain higher fidelity compared to traditional methods. This result was experimentally confirmed by Xu et al. in 2021 [122], who verified that the compressed model preserved:

- Label Fidelity. The authors have examined how the student model behaves in comparison to the teacher model. Label fidelity represents the similarity between the labels predicted by the teacher model and those of the student model. High fidelity indicates a more predictable model and is likely free of additional biases.

- Probability Fidelity. Probability fidelity goes beyond simply predicting the label, comparing the probability distributions assigned to the different classes by the two models. In a classification problem, each model assigns a probability to each class. Even when the compressed model predicts the same final label, the probability distribution, that is the degree of “certainty” in the decision, may differ between the two models.

Compressed models based on learning techniques, such as distillation, outperform in this case those based on compression techniques like pruning or quantization [122].

Understanding knowledge distillation: techniques and benefits

Knowledge transfer was first introduced by Bucila et al. in 2006 [16], aiming to compress large and complex ensemble models [17] into smaller and faster ones without a substantial loss of performance. The main idea behind this type of compression is to use a fast and compact model to approximate the function

[17] An ensemble model is a collection of models whose predictions are combined through weighted averaging or voting.

learned by a larger and slower yet more performant model. By leveraging the universal approximation property of neural networks and considering a network's knowledge as a learned mapping between inputs and outputs, the authors proposed compressing a model by labeling a large set of unlabeled data with it and then training a second neural network using this data [16].

In 2015, Hinton et al. [39], building upon this work, developed knowledge distillation (KD), a technique that allows smaller networks (students) to learn the "dark knowledge" from larger pre-trained models (teachers). Distillation, in particular, uses the original labels from the training set together with the teacher's predictions in the form of soft labels to train the student model. These soft labels are not simple binary labels but represent the probability associated with each class, reflecting not only the correct class but also the model's confidence level in the other classes. Soft labels capture the relationships between different classes that hard labels cannot represent. When soft targets have high entropy, they provide much more information for each training case compared to hard targets and reduce the variance in the gradient between training cases. Consequently, the small model can often be trained with much less data compared to the original complex model and using a much higher learning rate.

Going into detail. During training, the student model seeks to minimize a loss function that combines two terms [39] (fiure 4.16):

- Loss on Original Labels (Hard Labels). The classic cross-entropy loss between the true labels Y and the student's predictions $\hat{Y}^{(s)}$.
- Distillation Loss on Soft Labels. The cross-entropy loss between the teacher's soft labels $\hat{Y}^{(t)}$ and the student's predictions $\hat{Y}^{(s)}$.

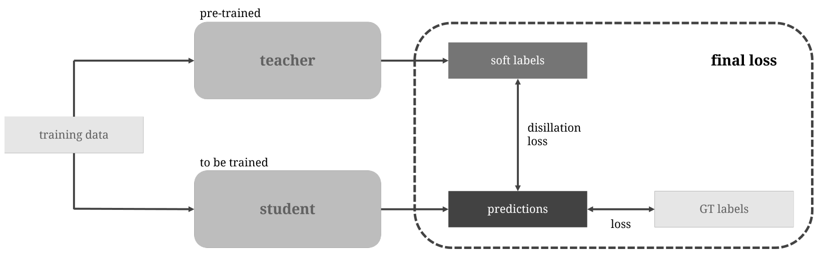


Figure 4.16: A simplifying image of how knowledge distillation works.

The final loss function is:

$$L = \lambda_1 \cdot L_{\text{ground-truth}} + \lambda_2 \cdot L_{\text{distillation}} =$$
$$= \lambda_1 \cdot \text{CrossEntropy}(Y, \hat{Y}^{(s)}) + \lambda_2 \cdot \text{CrossEntropy}(\hat{Y}^{(t)}, \hat{Y}^{(s)})$$

Where λ_1 and λ_2 are hyperparameters that control the relative importance of the two loss terms. Hinton et al. [39] observed that it is preferable to assign a lower weight to the second loss term, as the soft targets convey more detailed information, namely the probabilities associated with all the classes, making them more useful for the distillation process.

At this point, it is essential to introduce a key concept. To generate the soft labels, a "softened" version of the softmax function is used, which incorporates the temperature parameter ($T \geq 1$). The formula for the modified softmax becomes:

$$Y_i^{(t)} = \frac{\exp\left(\frac{z_i^{(t)}}{T}\right)}{\sum_j \exp\left(\frac{z_j^{(t)}}{T}\right)}$$

When a complex model often provides the correct answer with high accuracy, much of the important information actually resides in the relationships between the very small probabilities of the soft labels. The problem is that, since these probabilities are so close to zero, they have very little impact on the cost function during the training phase. As the temperature increases, the differences between the probabilities of the various classes decrease, making the distribution more uniform. A more uniform distribution allows the student model to see which classes the teacher considers similar or related: instead of focusing only on the most probable class, the student model learns how the teacher distributes uncertainty among the classes.

Finally, since the magnitudes of the gradients produced by the soft labels scale as $\frac{1}{T^2}$, it is important to multiply them in the final formula by T^2. This ensures that the relative contributions of the hard and soft targets remain approximately unchanged if the temperature used for distillation is modified during hyperparameter experimentation.

$$L = \lambda_1 \cdot L_{\text{ground-truth}} + \lambda_2 \cdot T^2 \cdot L_{\text{distillation}}$$

In particular, this type of knowledge distillation is called, as suggested by Li et al. [62] in 2023, "response-based", where the main idea is to directly mimic the final prediction of the teacher network. There are also other types of knowledge distillation that, however, will not be discussed here.

As highlighted by Cheng et al. [17] in 2018, in general, Knowledge Distillation (KD) based approaches can simplify deep models, making them more lightweight and contributing to a significant reduction in computational cost. For example, DistillBERT, a distilled version of the well-known model presented by Sanh et al. in 2020 [100], has 40% fewer parameters, retains 97% of the language understanding capabilities, and is up to 605 times faster.

Chapter 5

Evaluating TST models: key metrics and methods

In a previous chapter, the task at the core of this dissertation, Text Style Transfer (TST), was introduced. After presenting the datasets used for the research and delving into the details of state-of-the-art architectures and models, it is essential to understand how the effectiveness of TST approaches should be evaluated (last step of the presented NLP pipeline). This chapter illustrates the various evaluation metrics used in the literature to measure the quality of the results achievable in Text Style Transfers. There are numerous SOTA indicators; in this thesis, the most common ones are presented and used to obtain results comparable with other studies related to the same task.

5.1 The three pillars of Text Style Transfer evaluation

The most effective way to evaluate a Machine Learning model is to test it in action on a specific task (extrinsic or in-vivo assessment). However, this approach can be very costly as it requires building an application to conduct the tests. Therefore, in practice, intrinsic techniques are used, which analyze the model in isolated contexts without the need to implement finished products.

But what parameters need to be considered to evaluate a TST (Text Style Transfer) model? According to Jin et al. [47], the textual output should not only reflect the correct target style but also preserve the original semantics and ensure a certain fluency typical of natural language. Therefore, three main criteria should be considered for the evaluation of TST models [47] [45]:

- Transferred Style Strength;
- Content preservation;
- Fluency.

As emphasized by Hu et al. [45], a model that does not meet these three criteria cannot be considered valid for TST. Evaluations can be conducted using automated techniques or human intervention. The latter provides a more flexible and comprehensive analysis of the textual output, but the cost of applying it is often prohibitive. Additionally, human judgments can be subjective and not easily reproducible, making it difficult to compare the results of different models. Therefore, in practice, automated evaluations are preferred, as they are scalable, cost-effective, and reproducible [47]. In this dissertation, specific metrics for TST are used to measure the capabilities of the models in the three highlighted areas, selected to consider various aspects of the complexity of natural language.

5.2 Criterion 1. Transferred Style Strength

To measure this criterion, it is necessary to determine whether the generated text effectively reflects the target style. To do this, a binary style classifier is typically trained and the final accuracy value can be calculated using the following formula, as suggested by Jin et al. [47]:

$$\frac{\text{correctly classified test samples}}{\text{total test samples}}$$

No further details need to be specified regarding this metric in this section, as the associated code will be discussed in detail later.

5.3 Criterion 2. Content preservation

The objective of this criterion, as highlighted by Hu et al. [45], is to quantitatively measure the amount of original content preserved after the style transfer operation. There are various metrics in the literature, often developed to evaluate tasks related to text style transfer, such as neural machine translation. In this section, four different indices are presented, followed by a discussion on their use in this thesis.

5.3.1 The power of BLEU in automated text evaluation

BLEU is a metric developed by Papineni et al. in 2002 [86], introduced to provide an automatic, fast, language-independent tool that correlates with human evaluations for analyzing Neural Machine Translation (NMT) results. The creators based the metric on the observation that "the closer a machine translation is to a professional human translation, the better it is" [86]. Therefore, BLEU is applicable only in the presence of one or more human-provided

Ground Truths (GTs) (it can consider multiple GTs, but it also works with just one).

In detail, BLEU $\in$ [0, 1] [1], with 1 representing a perfect match between the generated output and the reference translations. Although no translation, not even a human one, necessarily achieves a score of 1, as noted by the authors [86], a higher BLEU score indicates a better translation in terms of adequacy, fluency, and general similarity to human translations.

Commenting on the formula.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

It's possible to view the metric as being composed of four parts:

- N. BLEU works with n-grams, with n $\geq$ 1, which are useful for considering the order in which words are presented in the text when calculating p_n. N refers to the maximum order of n. In fact, Papineni et al. [86] observed experimentally that a combination of the p_n precisions provides a more robust and reliable measure. Therefore, the authors decided to use, in the calculation of the index, the weighted contribution via w_n of the p_n with n $\in$ [1, N]. A good value for N is 4.
- p_n. A good translation or transfer shares many words and phrases with the reference text (Ground Truth). In this regard, Papineni et al. [86] proposed analyzing each n-gram present in the textual output (the candidate sentence) and checking if it matches an n-gram in the ground truth (GTs), assigning a value of 0 or 1 for the match, regardless of whether that n-gram from the GTs has already been used. Precision, indicated as p_n, is calculated by dividing the number of matches found by the total number of n-grams present in the candidate sentence. This solution has an obvious problem [86]: it allows for the repeated use of the same n-gram from the reference corpus to achieve matches. In an extreme case, for example, if the model always produced the same 'reasonable' n-gram, it could achieve very high precision, even as high as 1.

  To resolve this discrepancy, the authors [86] decided to limit the number of possible matches with the GTs. The modified precision is calculated by first counting the maximum number of times a candidate n-gram, taken once, appears in a single reference text. Then, the total count of each unique n-gram in the candidate sentence is reduced (clipped) considering the maximum number of occurrences observed among the

[1] Often BLUE is reported in the range [0, 100] which is equivalent.

reference texts.

$$\text{Count}_{\text{clip}} = \min(\text{Count}, \text{Max\_Ref\_Count})$$

Finally, these clipped counts are summed and divided by the total (non-clipped) number of n-grams in the textual output. This approach rewards the use of words in the correct quantity and penalizes the excessive use of a word relative to its frequency in the reference texts.

The described method applies to a single sentence. By extending this calculation to the entire test set, it is possible to determine the n-gram matches for each textual output. Then, the clipped counts of the n-grams for all candidate sentences are summed and the result is divided by the total number of n-grams present in the test corpus

$$p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{\text{n-gram} \in C} \text{Count}_{\text{clip}}(\text{n-gram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{\text{n-gram'} \in C'} \text{Count}(\text{n-gram'})}$$

- w_n + (log and exp). w_n is the hyperparameter that, along with the logarithm and the exponential function, is used to differently weight the precision of an n-gram of order n. Papineni et al. [86] experimentally observed that the precision p_n tends to decrease as the length of the n-gram increases. In other words, unigrams have higher precision than bigrams, and bigrams have higher precision than trigrams, and so on. This behavior is due to the fact that it is easier to find matches for single words compared to longer sequences. A weighting approach with the geometric mean [2], using the exponential (natural exponential), logarithm (natural logarithm), and w_n in the formula, makes the BLEU score more sensitive to variations in lower precisions. In the original and standard implementation, w_n equals $\frac{1}{N}$: the weights are uniform for all n.

- BP. A candidate output should be neither too long nor too short [86]. While outputs longer than their references are already penalized by the modified n-gram precision measure, shorter generated contents tend to receive inflated precisions: fewer words in the n-grams reduce the chance of errors. The brevity penalty (BP) [86] ensures that short outputs do not receive artificially high scores, which might not accurately

[2] The geometric mean is a measure of central tendency that is used to find the average value of a set of positive numbers. Unlike the arithmetic mean, which sums the values and then divides by the number of elements, the geometric mean is obtained by multiplying all the values together and then taking the n-th root of the result, where n is the number of values. The general formula for the geometric mean of n positive numbers $x_1, \ldots, x_n$ is $G = \sqrt[n]{x_1 \cdot \ldots \cdot x_n} \iff \log(G) = \log\left(\sqrt[n]{x_1 \cdot \ldots \cdot x_n}\right) \iff \log(G) = \frac{1}{n}\log(x_1 \cdot \ldots \cdot x_n) \iff \log(G) = \frac{1}{n}\sum_{i=1}^{n}\log(x_i) \iff G = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log(x_i)\right)$, where $\frac{1}{n} = w_n$, obtaining the formula discussed above.

capture the full semantics. For this reason, BP does not penalize candidate sentences that are longer than the reference sentences (BP=1).

It is also important to highlight a problem [86]: if the brevity penalty were calculated sentence by sentence and the penalties were averaged, length deviations in short sentences would be severely punished. Therefore, BP is calculated on the entire corpus, ensuring some flexibility at the sentence level.

To calculate the discount coefficient, the closest reference length (best match length) is identified for each candidate sentence [86]. The various "best match lengths" of all the sentences in the test corpus are then summed, obtaining a reference length r, which is compared with the total length (sum) of the candidate sentences c.

$$\text{BP} = \begin{cases} 1, & \text{if } c > r \\ e^{(1-\frac{r}{c})}, & \text{if } c \leq r \end{cases}$$

It is important to emphasize that neither this brevity penalty nor the effect of the modified n-gram precision directly consider the length of the source sentence!

It is important to note that the more references per sentence there are, the higher the BLEU score will be [86]: the probability that the n-grams in the candidate output match those in the GTs increases. Therefore, one must be cautious when making comparisons between evaluations with a different number of reference sentences. However, Papineni et al. [86] have experimentally demonstrated that the ranking of different systems based on 4 references or one randomly chosen among them remains unchanged. This result suggests that it is possible to use a large test corpus with a single reference in evaluations, as in the case study of this dissertation.

5.3.2 Measuring translation quality: the advantages of METEOR

In 2007, Lavie et al. [58] developed METEOR with the goal of overcoming some of the limitations of BLEU. One of the main problems with the older metric is that it considers n-gram precision for calculating the final score but does not account for recall. Recall is a measure of the proportion of corresponding n-grams in the candidate relative to the total number of n-grams in the reference, reflecting how well the generated sentence covers the entire content of the GT. Although Papineni et al. [86] introduced brevity penalty to address this issue, they did not fully succeed.

The authors of METEOR, in addition to criticizing the lack of recall in the final score attribution, also raised concerns about the use of n-grams themselves. They argue that a more effective measure is needed to account for word order. The lack of explicit word alignment between the candidate and the GT is another issue [58]: the same word could be used multiple times in different n-grams to create a “match”, artificially inflating the final result. Another problematic aspect, no less significant, concerns the use of geometric mean in the calculation of the score [58]: it results in a "zero" score whenever any of its components is zero, often rendering BLEU scores at the sentence or segment level meaningless. Even though BLEU was designed to be used only for aggregate counts over an entire test set, sentence-level scores could still be useful indicators for evaluating the quality of the generated content.

METEOR addresses the discussed weaknesses. The following formula is presented to calculate the METEOR score for each <candidate_sentence, reference$_i$> pair. Subsequently, at the end of the discussion, it will be highlighted how to calculate the final score.

Commenting on the formula.

$$\text{Score} = F_{\text{mean}} \cdot (1 - \text{Penalty})$$

It’s possible to view the metric as being composed of two parts:

- F_{mean}. One of the added values of the metric by Lavie et al. [58] is that it not only supports an exact match between words in the generated sentence and the GT, but it also takes into account morphological variants (words with the same root) and synonyms. Specifically, given a candidate sentence and a reference, METEOR attempts to create an alignment between the two strings, where an alignment is a mapping between unigrams, such that each unigram in one string maps to zero or one unigram in the other [58]. To achieve the optimal alignment, METEOR employs a series of successive stages, each consisting of two distinct phases [58]:

  1. In phase 1, an external module lists all possible unigram mappings between the two strings. Different modules map unigrams based on different criteria [58]: the “exact” module maps two unigrams if they are exactly the same, the “porter stem” module maps two unigrams if they are the same after they are stemmed using the Porter stemmer and the “WN synonymy” module maps two unigrams if they are synonyms of each other.

  2. In the second phase, the largest subset of these mappings is selected such that each unigram can map to at most one unigram in the other glsstring. If more than one subset constitutes an alignment and also has the same cardinality as the largest set, METEOR selects the one with the fewest crossings. A crossing is defined as follows: given two unigram mappings (c_i, r_j) and (c_k, r_l), where c is the candidate sentence and r is the reference, they are said to be crossed if and only if the following formula evaluates to a negative

number [58]:

$$(\text{pos}(c_i) - \text{pos}(c_k)) \cdot \big(\text{pos}(r_j) - \text{pos}(r_l)\big)$$

The variation between stages is determined by the change in the external module used in phase 1. Each stage maps only the unigrams that were not mapped in the previous ones, and for this reason, the order in which they are executed affects the final score. By default, the first stage uses the "exact" mapping module, the second uses the "Porter stem" module, and the third uses the "WN synonymy" module.

Once all the phases have been executed and an optimal alignment has been produced, the process continues as follows [58]:

- Unigram precision (P) is calculated as the ratio of the number of unigrams in the candidate sentence that are mapped to the total number of unigrams in the candidate.
- Unigram recall (R) is calculated as the ratio of the number of unigrams in the candidate sentence that are mapped to the total number of unigrams in the reference.

F_{mean} is then calculated by combining precision and recall using a harmonic mean [96] that places most of the weight on recall. The authors decided to use a harmonic mean of P and 9R. The resulting formula for F_{mean} is [58]:

$$F_{\text{mean}} = \frac{10PR}{R + 9P}$$

- Penalty. Precision, recall, and the resulting Fmean in METEOR are based on unigram-level matches. However, the authors considered taking into account longer matches as well [58]. The penalty is specifically used to reduce the score assigned when the correctly aligned terms between the generated sentence and the reference are distributed in a fragmented manner rather than being grouped together (adjacent) in coherent phrases. In other words, if the output contains all the correct words but these are arranged in a disordered manner, the system applies a penalty to reflect this disorganization. To calculate this penalty, all the mapped unigrams in the candidate sentence are grouped into chunks (groups of contiguous words), specifically into the fewest number possible, in such a way that the unigrams in each chunk are in adjacent positions in both the candidate sentence and the reference. The longer the chunks, the fewer in number, and in the extreme case where the entire candidate string matches the reference, there is only one chunk (the optimal case). The penalty score, once the chunks are identified, is calculated using the following formula [58]:

$$\text{Penalty} = 0.5 \cdot \left(\frac{\#\text{chunks}}{\#\text{unigrams\_matched}} \right)^3$$

It is observed that the score increases as the number of chunks increases: this has the effect of reducing the F_{mean} by up to 50% if there are no matches longer than unigrams. As the number of chunks approaches 1, the penalty decreases, and its lowerBound is determined by the number of matched unigrams. The parameters of this penalty function were determined experimentally by Lavie et al. [58].

The above calculations are performed for each <candidate_sentence, reference$_i$> pair, as previously highlighted. The score assigned to each candidate is the highest value resulting from these pairs, which is then used to calculate the final METEOR score by aggregating (summing) the scores across the entire test set [58].

The authors have experimentally demonstrated a higher correlation of METEOR with human evaluations compared to BLEU [58], particularly when using the stages in the presented "default" order. However, there is still room for improvement and extendability of the metric. [58]

5.3.3 The power of contextual embeddings in BERTScore

In this section, the BERTScore metric is introduced [124]: an index that utilizes the state-of-the-art BERT model, with its complete description available in the official paper by Devlin et al. [23]. Although it won't be discussed in detail, BERT leverages the transformer architecture [115], previously introduced and already highlighted for its effectiveness in NLP tasks. Therefore, even though the model underlying BERTScore is partially treated as a black box in this section, its utility as a metric can already be understood.

BERTScore [124], like many other metrics in the literature, compares the similarity between tokens in the candidate sentence and the reference sentences. However, the metric introduced by Zhang et al. in 2020 [124], by leveraging BERT's pre-trained contextual embeddings, correlates better with human judgments compared to existing metrics, both at the segment/sentence level and at the system level (as verified experimentally).

To understand the power of contextual embeddings and their "strength", it is essential to briefly recall their "power." Embeddings are numerical representations of words in a vector space, where each word is mapped to a vector of real numbers. These representations capture the semantic and syntactic features of words, allowing words with similar meanings to be represented by vectors that are close to each other in the multidimensional space. Contextual embeddings are an advanced extension of this concept, where the representation of a word is not fixed but varies depending on the context in which the word appears. In other words, a word like "bank" will have a different embedding if it appears in a sentence about a river compared to a sentence about finance. This type of embedding captures not only the intrinsic meaning of the word but also how its semantics change in relation to the surrounding words.

BERTScore [124] overcomes two problems associated with previous metrics [124]:

- N-gram-based metrics often fail to correctly recognize paraphrases, which are sentences that express the same meaning but use different words. This is because indicators like BLEU and METEOR rely primarily on the exact overlap of words (or n-grams) between the generated sentence and the reference sentence. As a result, semantically similar sentences are penalized because they do not share exactly the same words as the reference sentence.
- N-gram-based metrics struggle to capture dependencies between distant words in a sentence and to properly penalize critical changes in word order that affect meaning. For example, if the order of clauses "A because B" is changed to "B because A", BLEU (and similar metrics) tends to penalize this change only slightly, especially for long clauses. This is because such metrics mainly evaluate the proximity of words or phrases that are contiguous, ignoring more complex meaning relationships.

So, how does BERTScore work? The BERTScore for the entire test set is calculated by averaging the BERTScore of each reference-candidate pair [124]. Therefore, it is necessary to understand how to calculate this latter value.

According to the formalization in the official paper [124], given a reference sentence x tokenized into k tokens $\langle x_1, \ldots, x_k \rangle$ and a candidate sentence x' tokenized into l tokens $\langle x'_1, \ldots, x'_l \rangle$, the metric provides a score by applying the function $f(x, x') \in \mathbb{R}$. In this discussion, the basic calculation of BERTScore is presented in point 1, followed by two additional points that lead to the final metric as presented in the official paper, thanks to subsequent refinements [124].

1. The basic calculation of the BERTScore for a couple reference-candidate is based on the following successive steps:
   (a) Token embedding. The tokens of sentences x and x' are transformed into vectors (embeddings) using BERT's pre-trained contextual embeddings [124].
   (b) Calculation of similarity. Given a pair of sentences with embedded tokens, x_i and x_j', the similarity between all pairs of tokens is

calculated using cosine similarity [124].

$$\cos(\mathbf{x_i}, \mathbf{x_j}') = \frac{\mathbf{x_i}^\top \mathbf{x_j}'}{||\mathbf{x_i}|| \, ||\mathbf{x_j}'||}$$

To simplify the calculation, the vectors are first pre-normalized [3], thus reducing the computation to a simple dot product [124].

$$cos(\mathbf{x_i}, \mathbf{x_j}') = \mathbf{X}_i^\top \mathbf{X'}_j$$

(c) Greedy Matching for Precision and Recall. The overall score for a pair of sentences x and x' is based on a greedy matching algorithm [4] useful for calculating precision and recall [124]. In the context of BERTScore, the terms 'precision' and 'recall' are used similarly to their traditional usage, but with a meaning adapted to the nature of the sentence similarity evaluation task [124]:

- Recall. In this context, recall measures how well the tokens of the reference sentence x are represented in the candidate sentence x'. It is calculated by matching each token of x to the most similar token in x' and then averaging the highest similarities. This reflects the traditional concept of recall: it captures how much of the information in the reference sentence is 'recalled' or captured by the candidate sentence.
- Precision. Precision measures how well the tokens of the candidate sentence x' correspond to the tokens of the reference sentence x. Similarly to recall, it is calculated by matching each token of x' to the most similar token in x and then averaging the highest similarities. This reflects the concept of precision because it indicates how much of the information in the candidate sentence is actually relevant compared to the reference sentence.

[3] Normalization refers to the process of adjusting the vector so that it has a unit length of 1 while retaining its direction. Mathematically, for a vector $\mathbf{v}$, its normalized form $\hat{\mathbf{v}}$ is given by: $\hat{\mathbf{v}} = \frac{\mathbf{v}}{||\mathbf{v}||}$ where $||\mathbf{v}||$ is the norm. There are different types of norms: in the official paper [124], the normalization used is based on the L2 norm (Euclidean norm). This is evident because the vectors are described as 'pre-normalized,' which simplifies the calculation of the inner product directly to cosine similarity. The document states that the normalization ensures that the computed scores remain within the numerical range of cosine similarity, which is between -1 and 1.

[4] A greedy algorithm is a problem-solving approach that makes a series of choices by selecting the best option available at each step, with the goal of finding a globally optimal solution. The "greedy" aspect refers to the algorithm's tendency to make the most beneficial choice in the short term, without considering the broader implications or future consequences of that choice. While greedy algorithms are often simple and efficient, they do not always guarantee the most optimal solution in every scenario, as their local optimal choices might lead to a suboptimal global outcome. However, in many cases, they can provide a good enough or even optimal solution, depending on the problem structure

To find the maximum similarity, a greedy algorithm is used, which works by leveraging contextual embeddings in BERTScore. Although the algorithm ignores the order of tokens in sentences, this is not a problem: as previously highlighted, contextual embeddings capture the meaning of words by considering the context in which they are found. Even if the words in two sentences are in a different order, the system is still able to recognize the semantic similarity between them.

Once the precision and recall scores are calculated, the F1 score is computed.

For a reference sentence x and a candidate sentence x', the recall, precision, and F1 scores are as follows [124].

$$R_{\text{BERT}} = \frac{1}{|\mathbf{x}|} \sum_{\mathbf{x}_i \in \mathbf{x}} \max_{\mathbf{x}'_j \in \mathbf{x}'} \mathbf{x}_i^\top \mathbf{x}'_j \quad P_{\text{BERT}} = \frac{1}{|\mathbf{x}'|} \sum_{\mathbf{x}'_j \in \mathbf{x}'} \max_{\mathbf{x}_i \in \mathbf{x}} \mathbf{x}_i^\top \mathbf{x}'_j$$

$$F_{\text{BERT}} = 2 \cdot \frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}$$

2. There are studies in the literature that demonstrate that rare words can be more indicative of sentence-level similarity compared to common words [116] [58]. For this reason, BERTScore allows the incorporation of a word/token weighting technique [124]: the inverse document frequency (IDF) calculated with respect to the test corpus, to evaluate how important a word is to a phrase within a collection of them. Given M reference sentences x_i, with i=1,...,M, the IDF of each token w is computed as follows [124]:

$$\text{idf}(w) = -\log \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}[w \in x_i]$$

where I[·] is the indicator function that results in 1 if w is present in the reference, and 0 otherwise. It is important to note that the authors found it unnecessary to use the tf-idf weighting technique, because the term frequency (TF) for short texts like individual sentences of a term is likely to be 1 [124].

To avoid issues with unseen tokens, in addition, the authors decided to apply plus-one smoothing [124]. Plus-one smoothing, also known as Laplace smoothing, is a technique where you add one to each count in a probability distribution. This ensures that every possible outcome has a non-zero probability.

$$\text{idf}(w) = -\log(\frac{1 + \sum_{i=1}^{M} I[w \in x_i]}{M + 1})$$

Below, the overall metrics of the results are reported [124].

$$P_{\text{BERT-IDF}} = \frac{\sum_{x_j \in x'} \text{idf}(x_j) \max_{x_i \in x} \left(x_i^T x'_j \right)}{\sum_{x_j \in x'} \text{idf}(x_j)}$$

$$R_{\text{BERT-IDF}} = \frac{\sum_{x_i \in x} \text{idf}(x_i) \max_{x_j \in x'} \left(x_i^T x'_j \right)}{\sum_{x_i \in x} \text{idf}(x_i)}$$

$$F_{\text{BERT-IDF}} = 2 \cdot \frac{P_{\text{BERT-IDF}} \cdot R_{\text{BERT-IDF}}}{P_{\text{BERT-IDF}} + R_{\text{BERT-IDF}}}$$

3. BERTScore, by its construction, has the same numerical range as cosine similarity (ranging from -1 to 1) [124]. However, in practice, the authors observed that the resulting scores fall within a more limited range due to how contextual embeddings are distributed in multidimensional space. Although this characteristic does not affect BERTScore's ability to rank text generation systems, it makes the output score less readable [124]. Therefore, Zhang et al.[124] designed a method to rescale BERTScore based on its empirical lower bound b, that needs to be recalculated for each new task (it depends, for example, on the language used). This lower bound b is computed by averaging the BERTScore values obtained from a large number of randomly paired candidate-reference sentences, specifically 1 million pairs. These pairs are created by grouping two random sentences from the common crawl monolingual datasets, ensuring that the sentences in each pair have very low lexical and semantic overlap.

   Once b is calculated, BERTScore can be scaled linearly [124].

$$\hat{P}_{BERT} = \frac{P_{BERT} - b}{1 - b} \quad \hat{R}_{\text{BERT}} = \frac{R_{\text{BERT}} - b}{1 - b} \quad \hat{F}_{BERT} = \frac{F_{BERT} - b}{1 - b}$$

   After this operation, the values typically fall between 0 and 1. This method does not affect BERTScore's classification ability nor its correlation with human evaluations and is intended solely, as previously emphasized, to improve the readability of the score [124].

5.4 Criterion 3. Fluency

Generating fluent sentences is a common goal for Natural Language Generation models [45]. Even if a TST model manages to preserve semantics and maintain the correct target style, it may still produce ungrammatical sentences [84]. Therefore, it is important to introduce an additional metric: perplexity.

According to a definition by Jurafsky [50], the perplexity (PP) of a language model on a test set is the inverse probability of the test set, normalized by the number of words. Given $W = w_1 \dots w_N$ a string obtained by concatenating all the strings in the test set, separated by appropriate separator characters [5], the formula for perplexity on the test set is as follows:

$$\mathrm{PP}(W) = P(w_1 \dots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 \dots w_N)}}$$

Perplexity is based on the principle that the best LM is the one that predicts the test set with the highest accuracy, assigning the highest probability to an 'unseen' sentence: it measures the ability to correctly predict the next word based on the context provided by the previous words. This metric, in particular, reflects the degree of 'surprise' the model has regarding a given sample: the lower the perplexity, the better the model.

Commenting on and extending the formula:

- Normalization. Normalization is the process of making the perplexity independent of the text length, ensured by the Nth root in the formula, where N is equal to the number of words in the string W, including the separator tokens.
- $P(w_1 \dots w_N)$. The joint probability [6] of the set W represents the likelihood of observing the entire sequence W. Since we are dealing with dependent events, it is possible to express the joint probability as the product of conditional probabilities by using the chain rule [7]. Moreover, in the context of an n-gram language model, like the one under examination, the probability of the i-th word w_i depends on the previ-

[5] A "separator token" in language models (LM) is a special symbol used to delimit or separate different parts of a text input. These tokens help the model distinguish between different sections of the text or different types of data. When choosing the separator for the sentences in the string W, it is important to consider the token chosen by the pre-trained language model used to calculate perplexity.

[6] The joint probability is a measure used in statistics and probability theory to describe the likelihood that two or more events occur simultaneously. Formally, if A and B are two events, the joint probability of A and B, denoted as $P(A \cap B)$ or $P(A, B)$, is the probability that both events occur. There are several ways to calculate the joint probability, depending on whether the events are dependent or independent. If A and B are independent events, the joint probability is calculated as $P(A \cap B) = P(A) \times P(B)$. If A and B are not independent, the joint probability is calculated using the conditional probability $P(A \cap B) = P(A) \times P(B \mid A)$, where $P(B \mid A)$ is the probability of B given that A has occurred.

[7] The Chain Rule in probability is a principle that allows us to calculate the joint probability of a sequence of dependent events. The rule is based on the decomposition of the joint probability into a product of conditional probabilities. Suppose we have a sequence of n events or random variables $A_1, A_2, \dots, A_n$. The joint probability of these events can be expressed as $P(A_1, A_2, \dots, A_n) = P(A_1) \cdot P(A_2 \mid A_1) \cdot P(A_3 \mid A_1, A_2) \cdot \dots \cdot P(A_n \mid A_1, A_2, \dots, A_{n-1})$.

ous n-1 words. Therefore [50]

$$\mathrm{P}(W) = \prod_{i=1}^{N} P(w_i \mid w_1 \ldots w_{i-1})$$

The resulting formula is, consequently, the following [8]

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

But why, then, do we use perplexity to measure fluency? As highlighted by Varges [114], for the output of an NLG system to be considered fluent, it must resemble human-written text as closely as possible. Therefore, it is possible to use a foundational model trained on a vast corpus of human-written text, like GPT-2 [92], to calculate perplexity. However, it is important to emphasize that "perplexity", at this point, is a measure that varies not only based on the input text but also depending on the language model used: given a text WW, different language models will yield different "perplexity" values [50].

Despite having lots limitations, as highlight Jin et al. [47], perplexity is the most widely used indicator for evaluate fluency in TST downstream tasks.

[8] Why is the n-th root used for normalization? And why is perplexity, in the calculation, $\frac{1}{P(W)}$? To answer these questions, it is necessary to know that perplexity is related to cross-entropy. In information theory, cross-entropy is a measure of the difference between two probability distributions P (true) and Q (estimated), defined as $H(p,q) = -\sum_x p(x) \log q(x)$. Given a sequence of words $w_1, w_2, \ldots, w_N$, the language model seeks to predict the probability of each word w_i given the previous sequence $w_1, \ldots, w_{i-1}$. The formula becomes $H(p,q) = -\frac{1}{N} \sum_{i=1}^{N} \log q(w_i \mid w_1, \ldots, w_{i-1})$. Now, the perplexity is defined as the exponential of the average cross-entropy $PP(W) = 2^{H(p,q)} = 2^{-\frac{1}{N} \sum_{i=1}^{N} \log q(w_i|w_1,\ldots,w_{i-1})} = 2^{-\frac{1}{N} \log\left(\prod_{i=1}^{N} q(w_i|w_1,\ldots,w_{i-1})\right)} = \left(2^{\log\left(\prod_{i=1}^{N} q(w_i|w_1,\ldots,w_{i-1})\right)}\right)^{-\frac{1}{N}} = \left(\prod_{i=1}^{N} q(w_i \mid w_1, \ldots, w_{i-1})\right)^{-\frac{1}{N}} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$. The presented formula for perplexity has been obtained.

Chapter 6

Introducing J2C datasets

In this chapter, we present the datasets created to support the task defined in this dissertation. These corpora of <journalistic_style, conversational_style> pairs are not only designed to facilitate the automation of the style transfer process in the news market, but they also hold significant potential for application in other NLP tasks or related studies. For this reason, the presentation of the datasets is accompanied by an exploratory analysis aimed at highlighting their characteristics and potential, providing a comprehensive overview of their possible impact in the context of natural language processing.

6.1 Presentation of the datasets

To support the downstream NLP task presented in this dissertation, which involves style transfer of news text from a journalistic style to a conversational one, two datasets have been created. For this purpose, news articles were extracted from various official news agency websites (journalistic style) and aligned with their respective Facebook posts in a conversational style.

The proposed corpora are:

- J2C_news_IT. This dataset, developed in collaboration with ANSA.it, Italy's leading news agency, is designed to support the task in the italian language.
- J2C_news_EN. This dataset was created using publicly available data from various english news agencies and is useful for supporting the task in the english language.

Both corpora have been cleaned of the main sources of noise, following a manual identification process. Therefore, users are left with the option to perform adequate preprocessing according to their own use case.

Datasets available here: `https://github.com/mattiapiazzalunga/MasterThesis/tree/master/corpora`

6.1.1 J2C_news_IT: the italian J2C dataset

The dataset J2C_news_IT, created with the valuable collaboration of ANSA.it, as previously mentioned, is based on the alignment of news articles in the format <journalistic_style, conversational_style> extracted respectively from the ANSA.it website and the official Facebook profile of the agency. Specifically, ANSA.it provided links to the articles in its Facebook posts, enabling the creation of the dataset. Although the corpus is limited to only 1.478 examples, it represents not only a significant starting point for extending the task in a low-resource language (italian), thereby supporting the information market in Italy, but also the collaboration with Italy's leading news agency serves as an indicator of the quality and validity of the produced dataset.

After aligning the news items, to obtain the final dataset consisting of the columns "Journalistic" and "Conversational", pairs containing null values or where the "conversational" news item started with "with a story:" were removed. This formulation was used by the editors to encourage users to click on an external link without providing the content of the news item in the post. Additionally, phrases in the text that did not contribute to the meaning (harmful noise), such as copyright references often present in journalistic prose, and phrases used in Facebook posts to encourage clicks, like "attachments:", "with a video:", "with a photo", "with a story", were removed. The preprocessing of the dataset was intentionally kept light to leave the final user the possibility to make specific adjustments based on the requirements of the ML model or the studies to be conducted.

6.1.2 Style transfer in english news: J2C_news_EN

The J2C_news_EN dataset is based, like the italian version, on the alignment of news in two styles: journalistic and conversational. In this case, an alignment was created using articles from the official websites of 12 U.S. news outlets and the corresponding posts from their Facebook profiles. The final result includes 5.352 data pairs, representing an excellent basis for exploring the task described in this dissertation in the english language. The corpus generation began from Martin's 2016 dataset [1], for which not much technical information about its creation is available, but it was originally intended to extract trends from Facebook posts, using various associated features, regarding [2]:

- Explore the frequency of media coverage and the bias related to "Trump" and "Clinton" across different media sources;

[1] `https://data.world/martinchek/2012-2016-facebook-posts`
[2] https://medium.com/newco/

- Compare social media attention in 2016 (the U.S. election year) with that of the Obama and Romney campaigns;
- Describe other topics raised by mainstream media regarding Trump and Clinton during the 2016 elections;
- Quantify the differences in audience engagement on Facebook for Clinton and Trump.

The original sources for this study were 15, carefully selected to minimize bias in political analysis. Three of these sources were removed to construct J2C_news_EN, as it was not possible to align the news due to website updates and/or copyright issues. The Facebook posts considered in the original research covered the period from January 1, 2012, to November 1, 2016. In the dataset used for this dissertation, which represents an extension of the previous one provided by the same author, the period was extended until November 9, 2016.

Entering into detail, for the creation of the dataset J2C_news_EN, all rows (Facebook posts) that did not have the link associated with the original article, necessary to create the alignment, were removed. Additionally, after downloading the journalistic news, pairs with null values or where the journalistic text started with "Don't want to wait?" were removed, as this indicated a faulty extraction process. Moreover, harmful noise, such as non-Unicode characters, external links, contextual dates, standard/advertising phrases like "24/7 coverage of breaking news and live events" or "watch:", or the opening formula "by < journalistic_style, conversational_style >", which indicated the author of the article, was eliminated. Finally, the J2C_news_EN corpus only includes the columns of interest: journalistic and conversational [3].

The preprocessing of the dataset was, once again, intentionally light, so that the final user could clean the data according to their own needs. Nonetheless, it is essential to ensure that a dataset is of good quality before presenting it. Several years have passed since Martin created the original alignment corpus: the links present in the Facebook posts may no longer point to the correct articles. Moreover, there are often inaccuracies in the creation of the posts: some authors might report an incorrect link to the original news or, furthermore, the few words used in Facebook posts might not reflect the semantics of the news. In this latter case, in particular, the posts might have been designed to attract readers for "clicks", without containing semantically relevant prose. To address the issue, the power of contextual embeddings was leveraged to semantically validate the alignment:

1. A pre-trained model, introduced by Reimers in 2019 [94], was

[3] The order of execution of the preprocessing steps is, obviously, important. In any case, for further information or additional details, please refer to the previously mentioned GitHub repository.

used to generate the embeddings of the journalistic and conversational texts, which helped to capture the news topic (sentence-transformers/paraphrase-mpnet-base-v2 [4]).

2. A similarity threshold was defined to establish when two texts were sufficiently similar to be retained in the final corpus (0,7);
3. Once both texts (journalistic and conversational) were converted into embeddings, cosine similarity was calculated to select the final pairs.

Through these steps, the J2C_news_EN dataset was built.

6.2 An initial exploratory analysis

To understand the usefulness of the generated datasets in scenarios different from the proposed one, and at the same time familiarize oneself with them, a brief exploratory analysis of the "J2C_news_EN" dataset is presented in this section. This dataset was chosen because it is richer and, consequently, more useful for the analysis. This type of study is also valuable in characterizing the two styles involved in the style transfer.

As previously highlighted, J2C_news_EN contains 5.352 text pairs in the format <journalistic_style, conversational_style> and is free of null values. Going into more detail, the journalistic texts have an average length of 539 words, compared to only 27 words in the associated Facebook posts. This data confirms a basic intuition: journalistic prose tends to be more extensive and is oriented towards providing more details, analysis, and contextual information. In contrast, the conversational style is more direct, likely aimed at communicating information in a quick and accessible manner.

A previously proposed hypothesis for these two styles, furthermore, is that journalistic text makes use of more complex, structured, and formal language, whereas the conversational style prefers simple and informal language. By analyzing the dataset, we can partially confirm this intuition (complete data in table 6.1):

- The "Journalistic" column uses 79.692 distinct words to describe events,
- The "Conversational" column uses only 16.979.

	# unique words	**average text length**	**# outliers (length)**
jour	79.692	539	169
conv	16.979	27	248

Table 6.1: Basic text analysis.

[4] `https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2`

Although this discrepancy can be partially attributed to the significant difference in length between the two types of text, it is reasonable to assume that the use of a more limited vocabulary also reflects the simplicity with which information is intended to be conveyed in the conversational style. For a more valid response, it is useful to calculate the text's readability score using the textstat library [5], specifically the Flesch Reading Ease, a readability index proposed by Flesch in 1948 [29], whose purpose is to measure the readability and comprehensibility of a text. This index is based on two parameters to determine the difficulty of comprehension:

- S. The average word length, measured by counting the number of syllables;
- W. The average sentence length, measured by counting the number of words.

$$\text{Ease of Reading} = 206,835 - 0,864S - 1,015W$$

The readability score for the journalistic style is 60,44, very close to that of the conversational style, which is 62,14, both indicating writing that is understandable even for students under 15 years old. This is quite a positive result, as it suggests that, in general, journalists aim to reach a wide audience with both approaches. Although the original articles may offer a greater level of detail, they do not lose clarity, at least for those analyzed, which do not come from agencies specializing in professional fields. In any case, further research would be useful to explore this aspect in more depth.

And what about formality? Using the s-nlp/roberta-base-formality-ranker model [6], presented by Babakov et al. [6], a variant of RobERTA [65] for formality classification, it can be observed that both types of prose are generally formal. Specifically, journalistic style is formal in 95.76% of cases (based on 5125 articles), while conversational style reaches 87,43% (based on 4679 articles). Only 10,65% of the news becomes informal when switching to the conversational style. Therefore, this aspect is not particularly useful for clearly distinguishing between the two styles: authors tend to maintain a formal tone in both cases (complete data in table 6.2).

	formal	**informal**
jour	5.125	227
conv	4.679	673

	no change	**for →inf**	**inf →for**
jour →conv	4658	570	124

Table 6.2: Formality analysis on the text.

[5] `https://pypi.org/project/textstat/`

[6] `https://huggingface.co/s-nlp/roberta-base-formality-ranker`

At this point, it might be interesting to delve deeper into the objectivity of the reported news to understand if and how the two communication styles influence it. Using the GroNLP/mdebertav3-subjectivity-english model [7], fine-tuned by DeBERTaV3 [37] from the University of Groningen, we can observe that 90,92% of journalistic news and 86,81% of conversational-style news are considered objective, with 4866 and 4646 items respectively (complete data in table 6.3). This is a positive finding, as it shows that, in most cases, the news reported, even in a conversational style, tends not to include personal opinions or subjective judgments.

	objective	**subjective**
jour	4.866	486
conv	4.646	706

	no change	**obj →sub**	**sub →obj**
jour →conv	4.534	519	299

Table 6.3: Subjectivity analysis on the text.

Finally, given the formal and objective nature of news, one might think they are devoid of emotions, whether positive or negative. However, using the model cardiffnlp/twitter-roberta-base-sentiment-latest [8] by Loureiro et al. in 2022 [66], based on RoBERTa [65], it appears that 3576 elements (66,82%) in journalistic style and 3023 (56,48%) in conversational style are neutral (complete data in table 6.4). This is not surprising, although the percentage of neutrality might seem lower than expected given the formality and objectivity of news. It is important to remember, however, that sentiment can also be extracted from seemingly objective and formal texts (for example, "The restaurant received five out of five stars" is objective but still conveys a positive sentiment).

An interesting aspect to note is that, regardless of style, negative sentiment prevails in non-neutral news (20,44% of texts in journalistic style versus 12,76%, and 30,55% in conversational style versus 12,97%). This could be due to the fact that "negative" news tends to attract more public attention than positive news. It would be interesting to expand the study to further explore this dynamic. Sentiment is, however, more pronounced in social conversational style, which is not surprising.

	positive	**neutral**	**negative**
jour	683	3.576	1.094
conv	694	3.023	1.635

	no change	**neu → neg**	**neu → pos**	**pos → neu**	**neg → neu**	**pos → neg**	**neg → pos**
jour → conv	3.562	816	335	302	296	31	10

Table 6.4: Sentiment analysis on the text.

[7] `https://huggingface.co/GroNLP/mdebertav3-subjectivity-english`

[8] `https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest`

To summarize, taking into account the inherent errors of the models used, the difference in length between the two styles (shorter texts are often more difficult to classify), and the potential bias introduced by the conversational style, which also aims to monetize through direct interaction with the audience, we can affirm that:

- Journalistic texts are designed to offer in-depth coverage of events, providing detailed information that helps the reader understand the full context and develop critical and informed thinking.
- Conversational texts, on the other hand, primarily aim to engage the reader, often with a more personal tone, trying to elicit responses and comments. Their strength lies in brevity and the ability to focus on key information without details that might distract the reader.

In both cases, news tends to maintain a certain neutrality, formality, and objectivity in most instances. This is surprisingly at odds with the initial hypotheses and the common belief that journalistic style is complex and formal [9]. However, it is important to remember that there can be exceptions: the style can vary significantly depending on the topic covered and the news outlet involved.

Finally, one might wonder whether publishers are truly following the optimal guidelines for creating content that engages the public. In a struggling news market, one would expect more attention to this aspect. Unfortunately, this does not seem to be the case. Although there are no official studies on this subject, some interesting insights emerge from content marketing research:

- An article by AJ Beltis from 2021 [10], former Marketing Manager, Content & Acquisition at HubSpot, highlighted that, based on internal analysis of the best-performing blog posts, the ideal length for Search Engine Optimization (SEO) purposes ranges between 2100 and 2400 words. Although this doesn't strictly concern news articles, it's a significant figure to consider.
- An article by Susan Moeller from 2017 [11], then Business Development Manager at BuzzSumo, revealed that after analyzing over 800 million Facebook posts in 2016, those with fewer than 50 characters received more interactions. Although this doesn't directly relate to news, it's a trend that could be exploited.

Despite these observations, only 54 of the journalistic articles analyzed fall within the ideal length range, and only 12 Facebook posts. Incredibly, no news article adhered to the guidelines in both styles. It would be interesting to ex-

[9] This analysis is based on a limited number of examples, restricted to Facebook and only 9 news outlets. A larger dataset would be needed to generalize more effectively.
[10] `https://blog.hubspot.com/marketing/how-long-should-your-blog-posts-be-faq`
[11] `https://buzzsumo.com/blog/ultimate-guide-facebook-engagement-2017/`

tend the analysis with more recent data and more relevant to the news context.

In conclusion, this exploratory study, while requiring further investigation, offers interesting insights for future analyses, highlighting the usefulness of the data collected.

Code available here: *https://github.com/mattiapiazzalunga/MasterThesis/tree/master/main_code_EN*

Chapter 7

Optimizing J2C models: from theory to practice

In this chapter, the focus shifts from theory to practice, generating TST models to support the new TST task proposed in this dissertation. For each presented corpus (J2C_news_IT and J2C_news_EN), two models are produced:

- A "base" version, using an efficient fine-tuning of T5-base version 1.1 for the english dataset and mT5-base for the italian one;
- A "small" version generated from the small versions of the introduced pretrained models (T5-small and mT5-small).

At the same time:

- To increase the accuracy of the "small" models to the level of the "base" models, the use of knowledge distillation is examined;
- To address the scarcity of training data, the impact of backtranslation, a standard data augmentation technique, is studied.

But how to evaluate the results? The models undergo an effectiveness analysis based on the theoretical concepts discussed in the chapter “Evaluating TST models: key metrics and methods.” In an intrinsic evaluation, the following aspects are studied:

- Transferred Style Strength. To evaluate this characteristic of the generated text, two classifiers (one per language) are trained in this chapter to distinguish between the two styles (journalistic and conversational);
- Content Preservation. The models’ effectiveness in preserving content during style transfer is measured using standard metrics such as sacreBLEU, METEOR, and BERT-score, previously introduced in earlier chapters;

- Fluency. The fluency of the generated texts is analyzed by calculating perplexity, as previously introduced, leveraging state-of-the-art pre-trained models, one for each language. These pre-trained models are capable of evaluating perplexity thanks to their exposure to large amounts of data during training, which enables them to learn syntactic structures, grammatical rules, semantic context, and statistical regularities of the language.

In short, in this chapter, six models are presented:

- *RoBERTa-base-news-style-CLS-journalistic-conversational-EN-v1* and *XLM-RoBERTa-news-style-CLS-journalistic-conversational-IT-v1* to evaluate Transferred Style Strength;
- *T5-v1_1-base-news-style-J2C-EN-v1* and *T5-v1_1-small-news-style-J2C-EN-v1* for J2C style transfer in english, trained on the J2C_news_EN dataset;
- *mT5-base-news-style-J2C-IT-v1* and *mT5-small-news-style-J2C-IT-v1* for J2C style transfer in italian, trained on the J2C_news_IT dataset.

The training and inference phases of these models are discussed, highlighting the results, limitations, and characteristics of the models. Was knowledge distillation and backtranslation helpful in improving their effectiveness?

7.1 Optimizing this study with python libraries and infrastructures

During the experiments, implemented in Python, several libraries and frameworks were essential:

- PyTorch by Paszke et al. [88], a Python framework that allows immediate execution of dynamic tensor [1] computations with automatic differentiation and GPU acceleration, while maintaining performance comparable to the most advanced deep learning libraries.
- Transformers by Huggingface (Wolf et al. [121]), a toolkit designed to support transformer-based architectures and facilitate the distribution of pre-trained models. As noted by the authors, Transformers is an ongoing project maintained by the Hugging Face team of engineers and researchers, with the active contribution of over 400 external collaborators.
- PEFT by Huggingface (Mangrulkar et al. [69]), a module that allows large

[1] Tensors are a generalization of the concepts of scalars, vectors, and matrices to multidimensional spaces. They represent data structures that contain numbers organized in multiple dimensions and are fundamental in the context of machine learning, deep learning, and scientific computing.

pre-trained models to be adapted to various applications without the need to optimize all model parameters, thus reducing computational costs.

- Evaluate by Huggingface [2], a resource that simplifies the evaluation of machine learning models and datasets in a quick and easy manner.

These tools were essential in supporting much of the work. It is also important to mention that the model training was performed on a machine from the DISCo Department at the University of Milan-Bicocca [3], equipped with the Linux #127-Ubuntu SMP operating system, an x86_64 architecture processor, 377 GB of RAM, and an NVIDIA RTX A6000 GPU with 49140 MB of memory.

This details have been provided not only to offer context and ensure a full understanding of the experiments and results, but also to guarantee the reproducibility of the analysis, adhering, as previously highlighted, to the principles of open science.

All additional information about the models training process is available in the commented code at the following `https://github.com/mattiapiazzalunga/MasterThesis/`.

7.2 Adapting datasets for fine-tuning

As highlighted in the chapter dedicated to the presentation of the J2C-TST datasets ("Introducing to novel journalistic to conversational datasets"), the preprocessing of these was deliberately kept minimal to allow for flexible adaptation to different use cases. In the context of this dissertation, it was necessary, however, to prepare the corpora for the fine-tuning of the selected models. For this reason, in order to maintain consistency between the original data, on which the T5 and mT5 models had been trained, and the datasets under study, the preprocessing process adopted for the pre-trained models was largely replicated.

7.2.1 T5 fine-tuning: data cleaning steps

The T5 model, as previously discussed, had been trained on the C4 dataset, created by the same authors. In order to fine-tune this pre-trained model using the english-language dataset, the following data cleaning procedures were applied to J2C_news_EN, which partially mirrored those described in "Foundational Models and the T5 Architecture Explained" for C4:

- Text pairs containing words from the "List of Dirty, Bad, Obscene, or Otherwise Inappropriate Words" were removed;

[2] `https://huggingface.co/docs/evaluate/index`
[3] `https://www.disco.unimib.it/en`

- Lines containing the phrase "lorem ipsum" were eliminated;
- Lines with curly brackets, which are uncommon in natural language but used to delimit "blocks" of code, were removed;
- Citation markers were removed;
- Sentences including policy-related warnings were removed;
- Only text pairs where the content was in english with 99% accuracy, verified using the langdetect tool, were retained.

In contrast to what was described in the original paper, sentences containing the word "javascript" were not removed, as they could be related to news or relevant topics in the field of computer science. Similarly, text pairs where one of the two parts did not contain at least three sentences were not eliminated, as conversational texts tended to be brief by nature, and sentences that did not end with a "final punctuation mark" were also not excluded.

7.2.2 Tailored data cleaning for mT5

For the fine-tuning of the mT5 model, trained on mC4, several data cleaning procedures were applied, mirroring those described in the original paper. However, in this case, it was important to highlight that the J2C_news_IT dataset was created in collaboration with expert editors from ANSA. As a result, for example, it was not necessary to verify the language of the text pairs (italian), unlike with the english corpus used for T5. Instead, the following steps were performed:

- Text pairs containing words from the "List of Dirty, Bad, Obscene, or Otherwise Inappropriate Words" for italian were removed;
- Lines containing curly brackets were eliminated to ensure that no "code blocks" were erroneously extracted during data retrieval.

Unlike the fine-tuning process for the T5 model, it was therefore not necessary to remove citations, lines containing the phrase “lorem ipsum,” or policy-related warnings, as the accuracy of the dataset was guaranteed by ANSA.

7.3 Custom style classifiers for english and italian texts

For a proper style transfer, as described in this study, it is essential that the TST model generates text that adheres to the desired target style: the "Transferred Style Strength" specifically evaluates this. Since there is currently no classifier in the literature that can precisely distinguish journalistic style from conversational style, it is necessary to develop a specific one. In particular, this section introduces two style classifiers, each designed for one of the languages analyzed in this dissertation. These classifiers are used in the evaluation phase of the TST models, but their application can be extended to new downstream tasks.

The classifiers are available at the following link: `https://github.com/mattiapiazzalunga/MasterThesis/tree/master/models`.

7.3.1 RoBERTa for english style classification

To obtain a state-of-the-art style classifier for the english language, the foundational model RoBERTa by Liu et al. [65] was used as a starting point. RoBERTa is a bidirectional encoder pre-trained with the objective of Masked Language Modeling. The model was then fine-tuned using the english dataset J2C_news_EN to distinguish between two classes: journalistic style and conversational style. During hyperparameter optimization for training, some of the key parameters that emerged were: an input size of 256 tokens, the Adafactor optimizer, a learning rate of 3e-5, and a batch size of 8. Additionally, a linear learning rate scheduler was employed, which allows for a linear decrease in the learning rate based on the number of training steps. To prevent overfitting and improve the model's performance on test or validation data, the early stopping technique was applied. Early stopping automatically halts training when the model's performance on the validation set stops improving. In this case, "patience" was set to 5, meaning that training would stop if the monitored metric did not improve for 5 consecutive epochs. The best model was selected based on the F1 score [4] (results summarized in a table 7.1).

	precision	**recall**	**F1-score**	**support**
journalistic	99%	99%	99%	487
conversational	99%	99%	99%	451
accuracy			99%	938

Table 7.1: Results of RoBERTa-base-news-style-CLS-journalistic-conversational-EN-v1.

When evaluating the final classifier on the test set, an accuracy of 99,15% and an F1 score of 99,11% were obtained, highlighting the model's effectiveness in analyzing Transferred Style Strength. The model is called *RoBERTa-base-news-style-CLS-journalistic-conversational-EN-v1*.

[4] F1 Score is a measure of a model's accuracy that balances precision and recall. It is defined as the harmonic mean of precision and recall, $F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

7.3.2 The italian classifier: fine-tuning XLM-RoBERTa

The italian classifier, trained on the J2C_news_IT dataset, was fine-tuned starting from the XLM-RoBERTa-base model by Conneau et al. [19], a multilingual variant of RoBERTa. During training, the same parameters as the english version were maintained, with the exception of the learning rate, which was reduced to 1e-5 (results summarized in a table 7.2).

	precision	**recall**	**F1-score**	**support**
journalistic	100%	99%	99%	138
conversational	99%	100%	99%	138
accuracy			99%	276

Table 7.2: Results of XLM-RoBERTa-news-style-CLS-journalistic-conversational-IT-v1.

Evaluation on the test set yielded an accuracy of 99,28% and an F1-score of 99,28%, once again confirming the effectiveness of the model. The model is called XLM-RoBERTa-news-style-CLS-journalistic-conversational-IT-v1.

Given the extremely high performance of both models, there are not many significant comparisons to be made between them. The high accuracy, considering the hyperparameter optimization work, suggests that the classification task in question is relatively simple, likely due to the characteristics of the text, which make the two classes easily distinguishable. However, it is important to highlight that the stylistic transfer of text represents a much more complex challenge, which is addressed next.

7.4 Optimizing J2C style transfer: the new english models

In this section, the fine-tuning process of version 1.1 of T5-base is introduced, aimed at generating an english sequence-to-sequence model for J2C-TST, trained on the J2C_news_EN dataset.

Delving into the details, the input to the model was limited to 512 tokens, which represents the maximum length used for training the T5-base model. Although journalistic texts can often be longer, this choice proved to be a good compromise between output accuracy and computational efficiency. T5, in fact, utilizes a relative memory mechanism provided by relative positional embeddings, which would allow the model to handle input of arbitrary length. However, the memory required quadruples each time the input size doubles, making it crucial to maintain a cost-efficiency balance. The maximum length of the output text, on the other hand, was set to 64 tokens, considering the average brevity of conversational texts. The pairs, after being adapted to the model's requirements, were tokenized and, by extension, prepared for the training of the final model.

After the text preprocessing, fine-tuning was performed, made more efficient through the use of the Hugging Face PEFT (Parameter-Efficient Fine-Tuning) library. This allowed the application of the LoRA technique during training. LoRA, as underlined in the theoretical chapter, is based on the intuition that adapting to a new task can be achieved by modifying only a subset of the model's parameters. To do this, it exploits a low-rank decomposition of the weight matrix, updating only the smaller matrices resulting from this decomposition. This approach significantly reduces computational costs and memory requirements while maintaining high performance. After various experiments to optimize the configuration, an ideal compromise was identified with a rank value of $r = 16$:

- A lower rank would have required less computational complexity, at the expense of the model's adaptation capability;
- A higher rank could have ensured better performance, in exchange for higher memory consumption and the risk of overfitting.

The LoRA updates were applied exclusively to the Q (query), K (key), V (value), and O (output) modules of the transformer network. Reducing the number of target modules would have, once again, improved memory efficiency at the expense of the model's adaptability, while adding additional modules, such as the feed-forwards, would have had the opposite effect. Additionally, it was decided to exclude the biases of the selected modules from the training process. The LoRA-alpha parameter, which controls the contribution of LoRA updates relative to the pretrained weights, was set to 32 (updates scaled by a factor of 32), and a 10% dropout was applied to mitigate the risk of overfitting (code in listing 7.1).

```
peft_config = LoraConfig(
    task_type=TaskType.SEQ_2_SEQ_LM,
    inference_mode=False,
    r=16,
    lora_alpha=32,
    target_modules=['q', 'k', 'v', 'o'],
    lora_dropout=0.1,
    bias="none"
)
```

Listing 7.1: LoRA configurations

Thanks to LoRA, the trainable parameters of T5 decreased from 251,116,800 to 3,538,944, representing a reduction of 98,59%, paving the way for more efficient fine-tuning.

But how was the fine-tuning performed? To train sequence-to-sequence Transformer networks, the Seq2SeqTrainer module from the Transformers library can be used. However, its original version presented a significant limitation for J2C-TST. By default, most models use greedy decoding, a method that generates the output text token by token, selecting each time the one with the highest probability based on the previous context. Although this approach allows for fast generation, it can lead to suboptimal choices, especially in complex tasks like TST, where it is crucial to produce a fluid text consistent with the target style, while preserving the informational content. The results obtained with greedy decoding were unsatisfactory. The adoption of beam search, a method that simultaneously explores several generation alternatives, led to initial improvements. However, to achieve optimal performance, it was necessary to adjust other generation parameters that could not be directly modified through the Seq2SeqTrainer class. This required adapting and customizing the module to allow for finer control of the generation process (code in listing 7.2).

```python
class CustomSeq2SeqTrainer(Seq2SeqTrainer):
    def __init__(self, generation_config=None, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.generation_config = generation_config

    def evaluate(self, *args, **kwargs):
        return super().evaluate(generation_config=self.generation_config, *args, **kwargs)

    def predict(self, *args, **kwargs):
        return super().predict(generation_config=self.generation_config, *args, **kwargs)
```

Listing 7.2: CustomSeq2SeqTrainer class

The constructor of the new class accepts, as shown in the listing, an optional parameter called generation_config, which contains the settings for text generation. The evaluate and predict methods are then overridden to incorporate the new configurations.

Once these modifications are implemented, it is essential to understand how the text generation has been customized (see the code in the listing 7.3):

- Maximum output sequence length. The generation length (max_new_tokens) has been limited to 50 tokens, an ideal value for creating conversational-style sentences.
- Beam search hypotheses. The beams number has been set to 10. This means that 10 paths are kept active during the search for the optimal text, considering the 10 best options at each step, allowing for greater exploration of the possible solution space.

- Sampling. This parameter enables sampling-based generation [5]. When set to true, the model selects tokens randomly from the probability distribution, instead of always choosing the most likely token. This approach introduces more creativity and makes the text less predictable. It's important to note that sampling still bases its choices on the calculated probabilities for each token, and there are other parameters that regulate the level of randomness in do_sample, such as temperature and top_p.
- Temperature. The temperature has been set to 0,8, which moderates the level of randomness in the generation. Values below 1 make the probability distribution more "focused," amplifying the differences between token probabilities and limiting the variety of the model's choices. Conversely, values greater than 1 increase randomness.
- Top-p (nucleus sampling). The parameter has been set to 0,85, which means that the model will only "select" from tokens that cumulatively represent 85% of the total probability. This reduces the influence of less likely tokens, improving the quality of the generation.
- Length penalty. The length penalty affects the length of the generated sequence. Values below 1 encourage the creation of shorter sequences, which is useful for the task at hand. This parameter has been set to 0,85. It is important to note that max_new_tokens imposes a hard limit on the length of the generated text: once this limit is reached, the generation stops, regardless of the length_penalty. The length_penalty only influences the preference for longer or shorter sequences within the limit set by max_new_tokens, but can never force the model to exceed that limit.
- Repetition penalty. This reduces the likelihood of repeating tokens that have already been generated. A value of 1,5 means that the probability of reusing an already chosen token is reduced by 1,5 times, preventing unwanted repetitions.

This configuration has been designed to generate high-quality responses, balancing creativity and coherence. At the same time, it prevents repetitions and keeps the text length under control, improving the appropriateness of the responses.

```
generation_config = GenerationConfig(
    max_new_tokens=50,
    num_beams=10,
    temperature=0.8,
    top_p=0.85,

```

[5] Enabling beam search and/or sampling means disabling greedy decoding. However, it is possible to combine beam searching with techniques such as core sampling to improve the diversity and quality of the generated text, as performed in this thesis.

```
    repetition_penalty=1.5,
    do_sample=True,
    length_penalty=0.85
)
```

Listing 7.3: Generation parameters

After customizing the generation configuration, training was conducted with a batch size of 16, using AdaW as the optimizer, a learning rate of 5e-4, and a linear learning rate scheduler. Again, the Early Stopping technique was applied, with patience set to 5.

But how was the best model selected? During training time, content preservation, which has more impact than fluency and Transferred Style Strength, was the aspect of style transfer evaluated to choose the most performant neural network. In addition, it is important to highlight that, although metrics such as BERT-score and METEOR address some of the limitations of BLEU, the latter still remains today the standard reference for evaluating machine translation and content preservation tasks in text style transfer. For this reason, although all three indices were monitored during training, only BLEU influenced the model selection during training time.

At test time, finally, all aspects, including those beyond content preservation, were analyzed to determine the validity of the J2C-TST model produced:

- Content Preservation. This aspect was analyzed through the previously generated classifier, RoBERTa-base-news-style-CLS-journalistic-conversational-EN-v1;
- Transferred Style Strength. Content preservation was evaluated based on the BLEU, METEOR, and BERT-score metrics;
- Fluency. The fluency of the generated texts was analyzed, as described in the chapter "Evaluating TST models: key metrics and methods", by calculating the "perplexity", a measure of how well a model predicts data. For the english version of the J2C-TST model, GPT-2 was used, which, thanks to its exposure to large amounts of data, is capable of learning syntactic structures, grammatical rules, semantic contexts, and statistical regularities of the language, allowing for an accurate assessment of fluency.

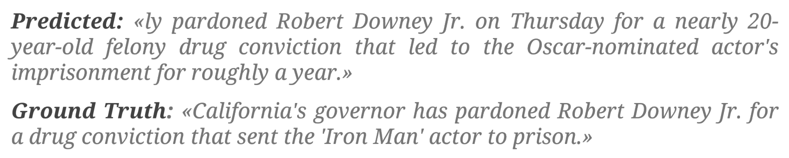
The final model, named T5-v1_1-base-news-style-J2C-EN-v1, achieved the following results on the test set 7.3 [6].

BLEU	METEOR	BERT-score	Perplexity	% conversational
27,27	43,00	89,77%	50,64	100%

Table 7.3: Results of T5-v1_1-base-news-style-J2C-EN-v1.

These results demonstrate the model's good ability to generate fluent and coherent sentences, both with the initial content and the desired target style.

The performance of the T5-v1_1-base-news-style-J2C-EN-v1 model has been analyzed through various metrics, but an interesting aspect to explore is the actual output produced. Although no human evaluation has been conducted, at least for now, one might still wonder whether it is truly functioning as expected. The figure 7.1 shows the neural network's output given a news article, randomly selected from the test set [7] [8].



Predicted: *«ly pardoned Robert Downey Jr. on Thursday for a nearly 20-year-old felony drug conviction that led to the Oscar-nominated actor's imprisonment for roughly a year.»*

Ground Truth: *«California's governor has pardoned Robert Downey Jr. for a drug conviction that sent the 'Iron Man' actor to prison.»*

Figure 7.1: An example of T5-v1_1-base-news-style-J2C-EN-v1 output.

By observing the image and comparing the network's output with the ground truth, it can be seen that the model manages to capture the essential information correctly, though with some minor issues. Among these, initial errors in form and a tendency to enrich the text with non-essential details stand out. Despite this, the generated text exhibits good fluency and the brevity typical of conversational language. These results indicate the validity of the neural network as a first J2C-TST model. Certainly, there is still room for improvement.

[6] Although there is no universal standard for evaluating the scores obtained, as they heavily depend on the specific task, considering the demands of style transfer and the complexity of the task, this dissertation adopts the following classification based on commonly accepted practices found on the web: (1) a BLEU score between 20 and 30 is considered acceptable, while a score between 30 and 40 is deemed good; (2) a METEOR score between 20 and 40 is considered acceptable, while a score between 40 and 60 is considered good; (3) for BERT-score, a value between 50 and 70 is considered acceptable, while a value between 70 and 90 is considered good; (4) finally, a Perplexity score between 100 and 50 is considered acceptable, while a score between 50 and 20 is considered good. It is important to emphasize that overly high scores in content preservation metrics may indicate that the style has not been effectively transferred.
[7] The input was not included due to its length, but it is almost irrelevant for the analysis. However, it is available in the J2C_news_EN dataset.
[8] The element was taken randomly, the model generates both better and worse style transfers.

7.4.1 The small variant for english J2C-TST

In the previous section, the T5-v1_1-base-news-style-J2C-EN-v1 model was introduced, a fine-tuning of T5-v1_1-base for J2C-TST. However, as already mentioned, the resources required for training and inference of large models represent a significant issue in everyday practice. The use of more complex and deeper models certainly leads to better performance, but it also entails higher energy consumption, resulting in significant economic costs and a substantial environmental impact in terms of carbon emissions.

In this section, a "small" variant of the T5-v1_1-base-news-style-J2C-EN-v1 model is introduced, aimed at making the inference phase more efficient and enabling the use of the english J2C-TST model in resource-constrained contexts while minimizing the loss of accuracy. To this end, a fine-tuning phase was performed on the "small" version of T5 v1.1, which contains 78,337,408 parameters, representing a 68,80% reduction compared to the "base" version. It is important to note, however, that even a smaller model still has a computational cost that cannot be overlooked. To optimize the training, the LoRA technique was once again used, reducing the trainable parameters of T5-v1_1-small by 98,24%, bringing them down to "only" 1,376,256.

Keeping the training and generation configurations unchanged and using the same evaluation technique as the base model, with an early_stopping set to patience 5, the following results were obtained 7.4.

BLEU	**METEOR**	**BERT-score**	**Perplexity**	**% conversational**
24,53	36,45	88,84%	65,59	100%

Table 7.4: Results of T5-v1_1-small-news-style-J2C-EN-v1.

These values, very close to those of the base model, confirm the validity of the new model, called *T5-v1_1-small-news-style-J2C-EN-v1*. The choice between the small model and the base model will therefore depend on the available resources, budget constraints, the required level of precision, and, more generally, the needs of the task. But is it possible to further improve T5-v1_1-small-news-style-J2C-EN-v1?

7.4.2 Small J2C-TST model via knowledge distillation

In chapter "Deep Learning approaches in NLP", the issue of resource consumption was introduced, highlighting, as in the previous section, the need to adopt smaller models that require fewer resources. From a theoretical standpoint, the technique of knowledge distillation was discussed, which is useful for transferring knowledge from a large pre-trained model to a smaller one, guiding it during the training process. Although T5-v1_1-small-news-style-J2C-

EN-v1 achieved performances similar to those of its "base" counterpart, it was still worthwhile to attempt to further reduce the performance gap.

As explained earlier, in the context of knowledge distillation, the student model seeks to incorporate additional information during training by leveraging the more detailed and informative predictions of the teacher model. This is achieved by minimizing a loss function that combines two main components:

- The "distillation loss", which measures the difference between the student's predictions and the teacher's "soft" predictions (non-binary and richer);
- The traditional loss function based on the ground truth.

In the case under consideration, the teacher model is T5-v1_1-base-news-style-J2C-EN-v1, already trained for the J2C-TST task in english, while the student model is T5-small. Will the fine-tuned version of T5-small achieve better performance than that obtained during an independent finetuning?

To perform knowledge distillation and incorporate the new loss function into the training (of the student), it was necessary to once again modify the already adapted Seq2SeqTrainer. Starting from the CustomSeq2SeqTrainer, the new module integrated the following changes:

- IF condition. In the computation of the loss, the behavior of the module changes through an if condition, which distinguishes whether the student model is in the training or evaluation phase. This approach allows, during evaluation and testing, to focus exclusively on the model's performance, utilizing the loss from the parent class, without including the distillation component, which is only relevant during training.
- New loss. The new loss, created following the theoretical guidelines, is as follows.

  $$\mathcal{L}_{\text{totale}} = \alpha_{\text{ce}} \cdot \mathcal{L}_{\text{ce}} + \alpha_{\text{distill}} \cdot T^2 \cdot \text{KL}\left(\text{softmax}\left(\frac{\text{logits}_{\text{teacher}}}{T}\right) \parallel \text{log}\_softmax\left(\frac{\text{logits}_{\text{student}}}{T}\right)\right)$$

  The formula for the total loss is nearly identical to the one presented in the chapter "Deep Learning approaches in NLP", which is why the utility of the individual parameters will not be re-explained here. The only difference lies in the calculation of the distillation loss, which in this case was chosen to be computed using the KL divergence, a measure that quantifies how much a probability distribution diverges from a reference distribution. Although this choice is common, it is not the one introduced in this dissertation nor the one indicated in the original paper by Hinton et al. [39].

  It is important to emphasize that all operations involved in the loss calculation (sum, cross-entropy, KL divergence, softmax, log-softmax) are differentiable. This means that the entire loss function is differentiable with respect to the model parameters, allowing for the calculation of gradients and optimization of the model through techniques such as gradient descent [9].

- New attributes. Each trainer object requires the following below attributes:
  - Teacher model. The network used to provide its predictions (logits) as a guide for training the model.
  - Temperature. Temperature controls how 'smooth' or 'sharp' the probabilities produced by the softmax function applied to the logits of both the student and teacher models are. When the temperature is high (value > 1), the probabilities become more uniform, and the differences between the values of the various classes are reduced, providing more information to the student model. When the teacher model is extremely confident about a class, it tends to assign nearly zero probabilities to the others. However, by increasing the temperature, the student is able to better perceive the differences between the classes and understand how the teacher evaluates them in relation to the correct one. Conversely, when the temperature is lower than 1, the probability distribution becomes sharper and more concentrated: the softmax function behaves more 'rigidly,' further increasing the probabilities for the correct class.

[9] In the original paper, cross-entropy is used to calculate the distillation loss, but it is closely related to the Kullback-Leibler (KL) divergence. The KL divergence measures how much a predicted probability distribution $Q(x)$ diverges from a true or reference distribution $P(x)$. Its formula is $D_{\mathrm{KL}}(P \parallel Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} = \sum_x P(x) \log P(x) - \sum_x P(x) \log Q(x)$. This expression can be rewritten as the difference between two terms: the first term, $\sum_x P(x) \log P(x)$, is the negative entropy of the distribution $P(x)$, denoted as $-H(P)$, while the second term, $-\sum_x P(x) \log Q(x)$, represents the cross-entropy $H(P,Q)$ between the true distribution $P(x)$ and the predicted distribution $Q(x)$. Therefore, the KL divergence can be expressed as $D_{\mathrm{KL}}(P \parallel Q) = H(P,Q) - H(P)$. Here, $H(P,Q)$ measures the distance between the predictions $Q(x)$ and the true distribution $P(x)$, while $H(P)$ represents the intrinsic entropy of $P(x)$, which remains constant with respect to $Q(x)$ and does not affect the optimization process. Consequently, minimizing the KL divergence is equivalent to minimizing the cross-entropy $H(P,Q)$. Thus, in terms of optimization, both cross-entropy and KL divergence lead to the same gradient descent updates for the student model, even when soft labels are used. Why is KL divergence preferred in Knowledge Distillation? Cross-entropy measures not only the distance between $P(x)$ and $Q(x)$ but also incorporates the uncertainty inherent in $P(x)$. This uncertainty is captured by the entropy term $H(P)$, which is subtracted in the KL divergence. As a result, even if $Q(x)$ perfectly aligns with $P(x)$, the cross-entropy will still equal the entropy of $P(x)$, which is generally positive. This residual term can introduce fluctuations that depend on the batch, potentially destabilizing the training process. In classical classification problems with hard labels, this issue does not arise because the entropy of $P(x)$ is zero. A one-hot distribution assigns a probability of 1 to a single class and 0 to all others, eliminating any uncertainty about the correct class.

Knowing that T5-small was able to achieve high performance, close to that of the teacher, the optimal choice was to use a low temperature (0,8). This allowed the student model to focus on the teacher model's more accurate predictions, avoiding confusion from less relevant classes.

– Alpha_ce and alpha_distill. Alpha_ce and alpha_distill are two fundamental parameters in knowledge distillation because they control where the student model derives its learning from. Specifically, alpha_ce determines how much the student model learns from the real labels, while alpha_distill influences how much the student model learns from the teacher model's predictions.

  In many cases, it is preferable to balance the classical loss and the distillation loss to achieve the best performance. For this reason, in the experiments associated with the dissertation, a balanced combination of alpha_ce = 0,5 and alpha_distill = 0,5 was chosen.

```python
class DistillationTrainer(CustomSeq2SeqTrainer):
    def __init__(self, teacher_model, tmp=1.0, alpha_ce=0.5, alpha_distil=0.5, **kwargs):
        super().__init__(**kwargs)
        self.teacher_model = teacher_model
        self.tmp = tmp
        self.alpha_ce = alpha_ce
        self.alpha_distil = alpha_distil

    def compute_loss(self, model, inputs, return_outputs=False):
        if model.training:
            labels = inputs.get('labels')
            inputs_no_labels = {k: v for k, v in inputs.items() if k != 'labels'}
            outputs_student = model(**inputs_no_labels, labels=labels)
            student_loss = outputs_student.loss
            logits_student = outputs_student.logits

            with torch.no_grad():
                outputs_teacher = self.teacher_model(**inputs_no_labels)
                logits_teacher = outputs_teacher.logits

            min_length = min(logits_student.shape[-1], logits_teacher.shape[-1])
            logits_student = logits_student[:, :, :min_length]
            logits_teacher = logits_teacher[:, :, :min_length]
            logits_student = logits_student.reshape(-1, logits_student.size(-1))
            logits_teacher = logits_teacher.reshape(-1, logits_teacher.size(-1))

            loss_fct =
    torch.nn.KLDivLoss(reduction="batchmean")
            loss_distillation = loss_fct(
                torch.nn.functional.log_softmax(logits_student
    / self.tmp, dim=-1),
                torch.nn.functional.softmax(logits_teacher /
    self.tmp, dim=-1)
            )

            loss = self.alpha_ce * student_loss +
    self.alpha_distil * loss_distillation
            return (loss, outputs_student) if return_outputs
    else loss
        else:
            return super().compute_loss(model, inputs,
    return_outputs)
```

Listing 7.4: DistillationTrainer class

Training T5-base with the help of T5-v1_1-base-news-style-J2C-EN-v1 as a teacher, thanks to the new trainer and using the same parameters and techniques (e.g., LoRA) as in the training of T5-v1_1-small-news-style-J2C-EN-v1, yields the following results 7.5.

BLEU	METEOR	BERT-score	Perplexity	% conversational
25,01	38,70	89,24%	57,66	100%

Table 7.5: Results of T5-v1_1-small-news-style-J2C-distilled-EN-v1.

These results not only demonstrate an improvement, albeit small, in the small model in terms of content preservation and fluency but also, to some extent, highlight the effectiveness of the knowledge distillation technique in the considered environment. Therefore, KD proves to be an excellent method for compressing models, ensuring efficiency in terms of computational and energy resources, reducing emissions, and simultaneously speeding up inference. The new base model (T5-v1_1-small-news-style-J2C-EN-v1) is therefore the one derived from distillation.

Considering the improvements compared to independent fine-tuning, the one obtained through KD is the new small model for J2C-TST in english, T5-v1_1-small-news-style-J2C-EN-v1. This model, with a 68,80% reduction in parameters compared to the base version, has achieved a slight decrease in terms of BLEU (8,29%), METEOR (10,00%), BERT-score (0,59%), and an increase of "only" 13,86% in perplexity. Certainly, in some use cases, these percentage variations in metrics can be significant, but in the presence of limited resources, small models have proven to be a valid alternative.

7.5 Building italian models: fine-tuning mT5 for J2C-TST

In the previous section, the base and small models for the J2C-TST in english were presented. Here, however, the process of generating their italian counterparts, trained on the J2C_news_IT dataset, is described, starting from the "base" model named mT5-base-news-style-J2C-IT-v1.

For fine-tuning, the training parameters of the multilingual model (mT5-base) were kept almost identical to those used for adapting T5. The input to the model was limited to 512 tokens, despite the model being pre-trained with input lengths of 1024 tokens. Although journalistic texts are often longer, this choice once again proved to be a good compromise between output accuracy and computational efficiency. The maximum length of the output text was set to 64 tokens, considering the average brevity of conversational texts. After being adapted to the model's requirements, the pairs were tokenized and subsequently prepared for the final model training, as was done with T5-base.

After preprocessing, fine-tuning was made more efficient, once again, thanks to the use of LoRA, configured with rank = 16, lora_alpha = 32, targeting the Q, K, V, O modules (excluding biases), and with a dropout rate of 10%. Thanks to LoRA, the trainable parameters of the mT5 model were reduced from 585,940,224 to 3,538,944, achieving a 99,40% reduction. A keen observer might notice that the total number of parameters in mT5 is greater than in T5. This is due to the fact that, as highlighted in the official paper, mT5 uses a much larger vocabulary to handle multilingualism. As a result, when training modules whose size does not depend on the vocabulary, the number of trainable parameters remains the same as in T5-base.

The generation configuration was also kept similar to that of the english counterpart, with num_beams=10, temperature=0,8, top_p=0,85, repetition_penalty=1,5, do_sample=True, and length_penalty=0,85. The only difference introduced was an increase in the maximum number of output tokens from 50 to 64, due to the slightly longer average length of conversational texts in italian compared to english.

Finally, the training was conducted using the AdaW optimizer and a linear learning rate scheduler, as with T5-v1_1-base-news-style-J2C-EN-v1, but with a batch size of 8 and a learning rate of 1e-3 [10], considering the smaller size of the italian dataset. Monitoring the BLEU metric during training to select the best model and using early stopping with patience set to 5, at test time, the selected model produced the following results 7.6.

[10] A smaller batch size was chosen because, with a reduced dataset, this can introduce greater stochastic noise in the gradient estimates, helping to avoid local minima and potentially improving the model's generalization ability. The learning rate was adjusted accordingly, and after performing hyperparameter optimization, 1e-3 was identified as the optimal value.

BLEU	**METEOR**	**BERT-score**	**Perplexity**	**% conversational**
32,84	45,16	77,54%	56,24	100%

Table 7.6: Results of mT5-base-news-style-J2C-IT-v1.

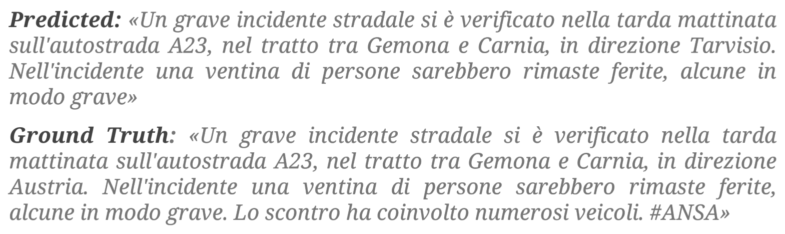
It is important to highlight, however, that a small change was introduced in the evaluation of Fluency and Style Transfer Strength, compared to the english model:

- Content Preservation. This aspect was analyzed through the previously generated classifier, but the one for italian, namely XLM-RoBERTa-news-style-CLS-journalistic-conversational-IT-v1;
- Fluency. Fluency was measured, once again, through perplexity, but an italian version of GPT-2, called GePpeTto by De Mattei et al. [22], was used. This is because GPT-2 was primarily trained on english, and the fine-tuning in italian that led to the creation of GePpeTto allowed it to acquire knowledge useful for calculating perplexity for the language in question.

In any case, the results demonstrate, once again, the model's good ability to generate fluid and coherent sentences, both with the initial content and with the desired target style. Of course, there is room for further improvement, but mT5-base-news-style-J2C-IT-v1 represents a solid starting point for the italian J2C-TST as well.

To conclude with a global consideration, drawing a parallel with the baseline model in english, although it is difficult to justify the small variations in the obtained scores, which could depend on multiple factors, the slightly higher BLEU and METEOR values could be attributed to the greater homogeneity of the italian dataset. The latter was trained exclusively on <journalistic_style, conversational_style> pairs from ANSA, unlike the english dataset, which was composed of texts from nine different news outlets. On the other hand, the lower BERT-score might be due both to BERT's lower effectiveness in working with italian and to the fact that the semantic alignment between the pairs of english texts was verified during the creation of the J2C_news_EN dataset: only pairs with a high degree of similarity were retained, which could have influenced the BERT-score. Finally, the higher perplexity observed in italian might be due to both the greater morphological complexity of the language and the fact that the training corpus for the italian language in GePeTo is less extensive compared to the english one in GPT-2.

The model's results were evaluated once again using automatic metrics. Although no human evaluation was conducted, it would be interesting to directly examine the generated output, as was done for the English model. In the figure 7.2, one can see the neural network's output, obtained in response to a journalistic text randomly taken from the test set.



Predicted: *«Un grave incidente stradale si è verificato nella tarda mattinata sull'autostrada A23, nel tratto tra Gemona e Carnia, in direzione Tarvisio. Nell'incidente una ventina di persone sarebbero rimaste ferite, alcune in modo grave»*

Ground Truth: *«Un grave incidente stradale si è verificato nella tarda mattinata sull'autostrada A23, nel tratto tra Gemona e Carnia, in direzione Austria. Nell'incidente una ventina di persone sarebbero rimaste ferite, alcune in modo grave. Lo scontro ha coinvolto numerosi veicoli. #ANSA»*

Figure 7.2: An example of mT5-base-news-style-J2C-IT-v1 output.

By comparing the resulting text in the image with the ground truth, it can be observed that the model's output is consistent in key points, but it shows some shortcomings in terms of geographical accuracy and informational completeness. The higher correspondence of certain parts of the text between the ground truth and the output, yet accompanied by lower semantic coherence, reflects the results of the metrics: METEOR and BLEU show higher scores, while the BERT-score, which is more suitable for capturing semantics thanks to its use of contextual embeddings, is lower compared to the English counterpart. Despite this, the generated text exhibits, once again, good fluency and the brevity typical of conversational language. These results indicate the validity of the neural network as a first J2C-TST model. Certainly, there is still room for improvement.

7.5.1 Introducing finetuned mT5-small for italian J2C-TST

Also for the italian J2C-TST task, a "small" model was trained, suitable for contexts with limited resources and aimed at reducing economic and environmental impact. To achieve this result, as done for the english model, a finetuning of the "small" version of mT5 was performed.

Specifically, mT5 small consists of 301.553.024 parameters, with a reduction of 48,54% compared to the base counterpart (mT5-base). However, as already highlighted for the adaptation of T5, a lower number of parameters does not necessarily imply that the computational cost of training can be disregarded. To optimize the latter, the LoRA technique was once again applied, reducing the number of trainable parameters to only 1.376.256 (-99,54%).

Keeping the training and generation configurations unchanged and using the same evaluation technique as the base model, with an early_stopping set to patience 5, the following results were obtained 7.7.

BLEU	METEOR	BERT-score	Perplexity	% conversational
30,48	42,85	76,38%	55,28	100%

Table 7.7: Results of mT5-small-news-style-J2C-IT-v1.

These values, very close to those of the base model, confirm, as with the english counterpart, the validity of the new model, called mT5-small-news-style-J2C-IT-v1. The choice between the "small" model and the base model will, as always, depend on the task requirements, respecting the accuracy-resource trade-off.

7.5.2 Leveraging KD for enhanced italian J2C-TST models

Given the usefulness of knowledge distillation for compressing T5-v1_1-base-news-style-J2C-EN-v1 into T5-v1_1-small-news-style-J2C-EN-v1, why not leverage it to also improve the performance of the J2C-TST small italian model? Although mT5-small-news-style-J2C-IT-v1 achieved performances similar to those of its "base" counterpart, it was still worthwhile to attempt to further reduce the performance gap.

Delving into the implementation details of KD. First of all, the previously modified training module for knowledge distillation (DistillationTrainer) was utilized. Furthermore, once again, the temperature was set to 0,8 to favor the teacher's accurate predictions during training, and both Alpha_ce and alpha_distill were set to 0,5 to balance the contribution, in the final loss, of the distillation loss and the original loss. The small version of mT5 was then trained using the supervision of mT5-base-news-style-J2C-IT-v1 as the teacher. For the training, the same parameters and techniques (e.g., LoRA) were used as in the adaptation of mT5-small-news-style-J2C-IT-v1, yielding the following results at test time 7.8.

BLEU	METEOR	BERT-score	Perplexity	% conversational
30,29	42,16	76,00%	53,65	100%

Table 7.8: Results of mT5-small-news-style-J2C-distilled-IT-v1.

Unfortunately, in this case, KD did not prove to be helpful in improving the student model's performance. Although it is difficult to justify this behavior, it is useful to draw a parallel with KD on english models and try to understand the differences in results. First of all, for T5 models, the gap between the teacher and student model was larger, so the potential improvement margin, which was then guaranteed by KD, was also higher. Additionally, the multilingual training of mT5 may lead to output representations that are less specialized for italian compared to english representations in T5, making it more difficult for the student model to extract relevant information. Finally, the smaller amount of data in the italian dataset may have prevented the teacher model, even if overall more performant, from learning deep and varied representations. KD is more effective when the teacher model has learned knowledge that the student model does not possess, but with limited data, the teacher model doesn't have much to transfer.

7.6 Exploring backtranslation for J2C-TST

As a concluding section of the chapter on experiments, an attempt to improve the accuracy of the previously defined models is presented. Due to the scarcity of data in the J2C-TST datasets, data augmentation (DA) techniques were adopted to enrich them. Data augmentation, as highlighted by Feng et al. [28], includes strategies to increase the diversity of training data without the need to collect new data [11]. The most common DA techniques involve creating modified copies of existing data or, in general, generating synthetic data.

In the work associated with this dissertation, the backtranslation technique, popularized by Sennrich et al. [102], was employed, following the following process to create artificial data:

1. The original text from the dataset to be augmented is taken in a source language;
2. It is translated into another language (pivot language);
3. Finally, the text is back-translated from the pivot language to the source language.

These steps allow for variations of the original prose while preserving its meaning, thus increasing the diversity of the training data.

For this experiment, the italian dataset was increased by 75% and the english dataset by 25%, considering the larger number of text pairs present in *J2C_news_EN*. These percentages were chosen as a compromise to improve the quality of the data without introducing too much noise, which could have compromised the model's ability to separate content from style, and the pivot

[11] Data augmentation, of course, is applied only to the training set because the test set must remain independent and represent real, unseen data for the model.

language selected for backtranslation was german [12] [13]. The backtranslation was applied to the training set before performing the fine-tuning of the base models, keeping the training configurations (e.g., LoRA configurations) and generation settings unchanged. The results are shown in the table 7.9.

	BLEU	METEOR	BERT-score	Perplexity	% conversational
T5-base-news-style-J2C-EN-v1	25,15	40,89	89,43%	49,84	97,65%
mT5-base-news-style-J2C-IT-v1	32,16	44,56	77,30%	56,02	97,65%

Table 7.9: Results of base models using backpropagation.

However, as shown in the table, the backtranslation technique did not yield the expected improvements. This highlights how some advanced state-of-the-art techniques, useful for improving models, are not always suitable for all use cases. The choice of such techniques strongly depends on the models, datasets, the specific task, and a range of other factors.

Why, then, were no improvements observed? Although it is not possible to provide a definitive answer, it can be hypothesized that the quality of the translations was insufficient, introducing excessive noise into the data. Furthermore, although german is a commonly used pivot language, it may not have been the most suitable choice for this specific task. The augmentation percentages adopted may also have played a role: were the additional data sufficient, or perhaps more were needed? These are questions that should be further investigated in future analyses.

Since the base models did not show significant improvements, tests on the "small" models were not continued.

All J2C-TST models are available here: `https://github.com/mattiapiazzalunga/MasterThesis/tree/master/models`

[12] German is a good pivot language for backtranslation in data augmentation for several reasons. First of all, there are high-quality machine translation models between german and many other languages, thanks to the vast amount of available bilingual data. Additionally, german has a different syntactic and grammatical structure compared to languages like italian or english. This difference can introduce interesting variations in the text when translated back and forth, enriching the dataset with new expressions and formulations. Lastly, using a language with distinct linguistic characteristics can help make the model more robust and generalizable, improving its performance on unseen data.
[13] The integration of the translation functionality was implemented using Google Translate APIs through a third-party library.

Chapter 8

Final remarks: J2C-TST and news

8.1 Conclusion

The war in Ukraine and the resulting economic shocks have encouraged publishers to accelerate their transition to digital, embracing new business models, different types of storytelling, and new forms of distribution. While numerous paths are possible, innovation, flexibility, and a relentless focus on the audience will be some of the key ingredients for success. Quality information is crucial to shaping thoughtful citizens and improving the future of society as a whole. However, news agencies must align with current trends to disseminate it. Of course, as previously highlighted, business models will need to be reviewed, but change is to be welcomed.

The "desire" for access to accurate and reliable news is present in society [81], but the information market must actively and correctly contribute to its dissemination. Furthermore, the lack of trust in news by the population is a significant problem but offers an opportunity where reputable news agencies can intervene to disseminate verified quality information, restore trust, and build user loyalty. The keyword must be simplicity: new generations are accustomed to simple and "social" language to grasp essential concepts without wasting time on comprehension. Society is accustomed to personalization, to having constant stimuli linked to individual interests, and although there is widespread fear of excessive personalization of news, this aspect is intrinsic to today's reality and cannot be overlooked. News agencies must also exploit this aspect to their advantage: they cannot ignore new communication channels but can work to use algorithms to their advantage, integrating them with editorial choices curated and selected by expert journalists.

Research, therefore, could not remain indifferent. In this dissertation, in support of the news market, a new task has been defined in the literature, underlying Text Style Transfer, the Journalistic to Conversational TST (J2C-TST). This task aims to stylistically modify text at a pragmatic level, transforming it from journalistic (standard) style to conversational (social) style, in order to automatically disseminate news on social channels, without requiring costly human intervention in terms of time and resources needed for article review. After formally defining J2C-TST and understanding its potential to support the information market, its complexity was also highlighted. The chapters alternated between theoretical and practical notions to define and understand which architectures could support the new task. The state-of-the-art transformer architecture was identified as a reference, and after understanding its potential and limitations, as well as drawing a general framework for how to assign meaning and work with a string in ML, we moved on to defining J2C-TST models. However, as this is a new task, there were no datasets available for training the models; for this reason, *J2C_news_IT* was introduced, consisting of 1.478 text pairs <journalistic_style, conversational_style> in italian, produced through collaboration with ANSA.it [1], and *J2C_news_EN* for english, consisting of 5.352 examples. To understand the usefulness of these corpora in other contexts, an independent analysis from the models was also conducted regarding the formal difference between conversational and journalistic style on the english corpus, noting that, with equal formality, subjectivity, and sentiment, the conversational style tends to be shorter and more useful in providing the reader with essential information and cues.

Given the complexity, working at a pragmatic level, of J2C-TST, which requires an intense knowledge of the language, a training from scratch of the models was not sufficient. Therefore, pre-trained T5 v1.1 [93] and mT5 [123] (multilingual variant) models were used, respectively for english and italian. Pretraining allowed these neural networks to learn general linguistic knowledge that could be applied to the task. Through fine-tuning, the foundational models were then adapted to J2C-TST, efficiently using LoRA [44]. LoRA led to a reduction in trainable parameters for T5-base by 98,59% and for mT5-base by 99,40%. Despite this, valid neural networks were generated: *T5-v1_1-base-news-style-j2c-EN-v1* and *mT5-base-news-style-j2c-IT-v1*. To make the latter usable in resource-limited contexts, speed up inference, and move toward green AI, small counterparts were also generated(*T5-v1_1-small-news-style-j2c-EN-v1* and *mT5-small-news-style-j2c-IT-v1*), again leveraging LoRA. For these small models, knowledge distillation [39], a well-known compression technique, was used to reduce the gap between their performance and that of the base models. For the english model, KD, which required a modification to the Huggingface transformers library's training module [121], proved effective, but it was less so for the italian one, probably due to the less specialized representations for the language in question of the mT5 model, considering its multi-

[1] `https://www.ansa.it/`

lingual pretraining. In all cases, however, customization of existing libraries was essential to integrate advanced output generation parameters.

In an attempt to further improve the performance of the produced neural networks, efforts were made to expand the corpora using backtranslation [102] and leveraging the German language as a pivot. The english dataset was increased by 25%, while the italian one by 75%. However, these efforts were not successful even for the base models, probably due to the inconsistency of the translations.

The models were then evaluated during the testing phase based on three characteristics [47]:

- Transferred Style Strength. This criterion evaluated whether the transferred style matched the conversational style. Since there was no suitable classifier for this task in the literature, two models were trained on the J2C-TST datasets. For english, RoBERTa was fine-tuned to produce the classifier (*RoBERTa-base-news-style-CLS-journalistic-conversational- EN-v1*), while for italian, XLM-RoBERTa was adapted (*XLM-RoBERTa-news-style-CLS-journalistic-conversational- IT-v1*). Both models achieved accuracy and F1 scores above 99%, demonstrating, on the one hand, the presence of discriminative features in the text of the two classes, and on the other hand, the effectiveness of the classifiers in evaluating Transferred Style Strength.
- Content Preservation. The effectiveness of the models in preserving content during style transfer was measured using standard metrics such as BLEU, METEOR, and BERT-score. BLEU [86] is the traditional metric for content preservation in machine translation, METEOR [58] represents an improved version, and BERT-score [124] exploits the power of contextual embeddings for more accurate evaluation. Since BLEU is the standard, it was the metric monitored in selecting the best model during training time.
- Fluency. The fluency of the generated texts was analyzed by calculating the "perplexity", a measure of how well a model predicts test data, using pre-trained state-of-the-art neural networks specific to each language. In particular, GPT-2 [92] was used to evaluate the english version, while GePpeTto [22], an italian variant of GPT-2, was employed for the italian one. Thanks to exposure to large amounts of data, these foundational models can learn syntactic structures, grammatical rules, semantic context, and statistical regularities of the language, effectively evaluating the fluency of the texts.

The results are reported in the table 8.1.

	BLEU	METEOR	BERT-score	Perplexity	% conversational	parameters
T5-base-news-style-J2C-EN-v1	27,27	43,00	89,77%	50,64	100%	251.116.800
mT5-base-news-style-J2C-IT-v1	32,84	45,16	77,54%	56,24	100%	585.940.224
T5-small-news-style-J2C-EN-v1	25,01	38,70	89,24%	57,66	100%	78.337.408
mT5-small-news-style-J2C-IT-v1	30,48	42,85	76,38%	55,28	100%	301.553.02

Table 8.1: Final characteristics of the models presented for the J2C-TST.

These results highlight the validity of the produced models, but there is still room for further improvement.

In summary, the thesis, alternating between detailed theoretical and practical notions, defined a new task in Text Style Transfer literature, the J2C-TST, useful to support the free information market. Furthermore, it introduced two datasets, one in italian and one in english, to support the task, studying them and making them available for further analysis. In the context of defining accurate and resource-efficient J2C-TST models, along with careful hyperparameter tuning and modifications to existing libraries, the validity of using SOTA compression techniques and data augmentation for the task was explored. Finally, six models were presented: two style classifiers (for evaluating Transferred Style Strength), two models for italian J2C-TST, and two for the english task.

Surely, the cutting-edge task presented in this dissertation can help reach a wide audience, especially the younger generations, reducing costs and accelerating the dissemination of information. Social networks represent one of many tools to be exploited and used unconventionally for the news market, by adopting a simpler and more conversational language. This work certainly represents just a small step in the attempt to restore trust and improve the dissemination of information, but it is nevertheless a concrete and significant tool.

As Brad Smith, president of Microsoft, stated: "Our future success depends on our ability to embrace artificial intelligence and use it for good". Contributing to the field of open information through AI, in my opinion, represents an excellent starting point. Therefore, I will extend these studies during my PhD at the University of Milan Bicocca.

Models, code and additional information are available here: https://github.com/mattiapiazzalunga/MasterThesis/

8.2 Future researches

In this dissertation, the analysis focused on how to leverage new platforms to disseminate reliable and accurate news in the highly digital society we live in. Techniques and state-of-the-art deep neural approaches were studied in an attempt to translate timely news published by expert journalists on official apps and websites into simpler language suitable for the broad social media audience, while still maintaining the accuracy of the information produced. This was accompanied by a thorough literature review in an effort to achieve the best results and utilize SOTA techniques and technologies.

The automatic translation approach can certainly help publishers reduce the costs and time associated with writing and publishing news across numerous communication channels, not limiting them to a narrow selection and facilitating the dissemination of quality information. But what else can be done? How can techniques and technologies be improved?

This chapter outlines the possible future directions of research, which emerge as natural extensions of the topics discussed and the results obtained in this dissertation. The areas for improvement of technologies, models, and approaches include both NLP in its generality and the specific downstream task of Text Style Transfer, with particular attention to the innovative application of TST to support the information industry in the textual transfer from standard journalistic style to conversational style, the novelty presented in this dissertation. The ideas for these future extensions are rooted in three main sources: the analysis of current trends in the news sector, the limitations of the models identified in this dissertation, and the in-depth study of the existing literature.

8.2.1 Future research to extend the studied task

This initial section outlines some potential directions for improving the downstream task of TST analyzed in this dissertation. To fully understand the possible future developments, it is essential to consider both the collaboration with ANSA, Italy's leading news agency, in an effort to support the news industry, and the current trends that could influence and transform the information sector.

New benchmarks: pathways for further analysis

In this dissertation, two datasets have been presented to support the new task of text style transfer in italian and english. The datasets consist of 1.478 and 5.352 pairs of texts respectively, in <journalistic style, conversational style>. Considering the importance of this task for the information market, which promotes the free flow of news, the next steps could include:

- Expanding the italian dataset, through the ongoing collaboration with ANSA, to further support the news industry in Italy;
- Initiating collaborations with english news agencies in order to increase the number of pairs in the *J2C_news_EN* dataset, even with news more recent than 2016, while ensuring the validity of the alignments with the

contribution of expert editors, as has already been done with ANSA.

These developments should be accompanied by close collaboration with journalists, to more precisely define the concept of "conversational style" and adapt the corpora accordingly. This style, in fact, should not be limited to encouraging readers to click through to the official site, but should offer a concise overview of the news for a social media audience, then encourage further exploration and allow news agencies to monetize. This would help sustain both free information and the news industry.

Moreover, the use of larger and richer datasets could significantly improve the performance of Machine Learning models. It would be useful for these new corpora to include texts from platforms other than Facebook, in order to better generalize the conversational style. Alternatively, specific datasets could be considered for creating personalized language models based on the "social" linguistic style of the platform in question, since, after all, LinkedIn does not use the same language as Twitter.

With larger datasets, it would also be possible to conduct a more detailed study of the distinctive characteristics of the two styles, building on the exploratory analysis of the english dataset presented in this thesis. In any case, *J2C_news_EN* and *J2C_news_IT* represent a solid starting point for establishing a benchmark for this innovative task, while also enabling additional analyses on <journalistic, conversational> style pairs.

Extrinsic evaluation: key to real performance

In the chapter on evaluation, it was highlighted how extrinsic evaluation is essential, as intrinsic evaluation is too costly, requiring the construction of an entire application to perform the tests. However, although automatic evaluations offer numerous advantages, such as providing an objective benchmark to compare the effectiveness of different models on the same task, it is crucial to gain a clear understanding of how the model actually performs in real-world contexts.

For this reason, in the future, it may be appropriate to focus on extrinsic evaluations, with the collaboration of industry experts, such as those from ANSA.it, and with the contribution of end users. The models developed in this dissertation aim to support the news industry and promote the free circulation of information, accessible to all, without linguistic barriers. Therefore, a future analysis of this kind is essential for the concrete validation of the results obtained.

Enhancing output quality through models and techniques

The models obtained in this dissertation were fine-tuned starting from T5 and its multilingual version, mT5. However, other reference models in the literature exist that could be tested to further improve the outputs produced. For example, Lewis et al. in 2020 presented BART [60], a sequence-to-sequence model alternative to T5, which has a multilingual variant called mBART, proposed by Liu et al. [64]. The choice between T5 and BART, as always, depends on the specific use case; therefore, it might be interesting to experiment with the use of mBART for the J2C-TST task.

In addition, it could be useful to explore new data augmentation techniques to increase the amount of data in the training datasets. Alternatively, another possibility is to improve the backtranslation already used, by testing new percentages of dataset augmentation and utilizing pivot languages other than German. Moreover, state-of-the-art neural models for translation, such as MarianMT models [2], could be used. These models are pre-trained for machine translation and were developed using the Marian framework by Junczys-Dowmunt et al.[49] [3]. MarianMT models are highly specialized and offer high-quality translations thanks to the use of specific optimizations, such as advanced tokenization techniques and targeted attention mechanisms. This focus allows MarianMT models to be extremely efficient and fast, making them ideal for real-time applications or hardware with limited resources. Furthermore, these models support a wide variety of pre-trained language pairs, facilitating translation between numerous languages without the need to train the models from scratch. These characteristics make them an interesting option to consider in the context of the J2C-TST task, especially for improving the quality and quantity of the data used.

New trends to leverage

In one of the previous chapters, the Digital News Report 2023 by Newman et al. [81] was cited, providing valuable insights into trends within the news industry. The data presented in the report offer significant points that can guide new research directions to support the news market.

[2] `https://huggingface.co/docs/transformers/model_doc/marian`

[3] Marian is a deep learning framework specifically designed for neural machine translation (NMT), written entirely in C++ to maximize efficiency and speed. Compared to more general frameworks like PyTorch, Marian focuses on optimized performance for training and inference of translation models, with advanced features such as efficient memory management, native multi-GPU support, and translation algorithms like batched beam search. The Huggingface Transformers library has incorporated MarianMT optimizations when using its models in frameworks like PyTorch, allowing their efficiency and speed to be leveraged in these environments.

A text-to-video approch to news Across all the countries analyzed in the Newman et al. [81] report, the majority of online users prefer to read the news rather than watch or listen to it, as text offers greater speed and control in accessing information, emphasizing the importance of the task studied in this dissertation. However, in some markets, respondents express a preference for video formats, with younger people showing a stronger tendency toward consuming this type of content. This suggests that future habits of online news consumption could change radically over the next decade. Indeed, until now, access to news has long been dominated by two giants: Google and Meta, which, at their peak, represented almost half of the online traffic to news sites. Although the so-called "duopoly" remains highly influential [81], the Digital News Report 2023 highlights how this position is weakening in many markets, with a growing number of providers competing. The increasing popularity of digital audio and video content has introduced new platforms, and these changing habits outline a "new normal" in which publishers must operate.

In any case [81], Facebook remains one of the most-used social networks (28% of users), but its influence on journalism is in decline, with a 14-point drop from its 2016 peak for news (42%). Social networks, by their nature, are constantly evolving, and today we observe a decline in interaction with traditional networks in favor of the rise of TikTok and other video-based platforms. The use of TikTok is particularly high among young people and in some countries in the Asia-Pacific, Latin America, and Africa, where it has played a role in the spread of both information and misinformation, as demonstrated by recent elections in Kenya and Brazil, and the Ukrainian conflict that has increased its visibility in Eastern Europe.

In light of these dynamics, why not further leverage artificial intelligence to support the news industry? A rapidly expanding area of research concerns the generation of video from text (text-to-video, T2V). Studying the literature on this topic and investing in research could lead to the creation of videos that tell news stories in a more interactive way, suitable for platforms like TikTok. As highlighted by Singh [108], rapid advances in deep learning and NLP techniques in recent years have made text-to-image and text-to-video generators powerful tools for automating content creation, making it faster, more efficient, and more cost-effective.

A significant example of a T2V model is SORA, an OpenAI project from 2024, which made headlines for its ability to generate realistic videos from textual descriptions. SORA by Brooks et al. [14] is an AI model capable of creating visually realistic scenes aligned with users' requests, with a duration of up to one minute. Exploring such solutions and tools could offer new opportunities to the news market, facilitating the dissemination of news on emerging platforms and representing a promising innovation.

A text-to-audio approch to news A second significant trend, closely related to the one discussed in the previous section, concerns the growth of audio news consumption. Newman et al. [81] indicate that this increase is driven by changes in audience preferences, higher content quality, and more effective monetization. Publishers are investing in podcasts, a format that, in addition to being relatively low-cost, fosters audience loyalty and particularly attracts younger people. Data shows that about one-third of the sample (34%) accesses podcasts monthly in a basket of 20 countries, and one-third of these (12%) regularly listens to news podcasts, with the most significant growth in the United States and Australia.

Despite this [81], the majority of respondents still prefer reading the news (57%) rather than watching it (30%) or listening to it (13%). However, among young people under 35, a higher percentage (17%) is inclined to choose listening, driven by the opportunity to multitask while consuming news, thanks to the widespread use of smartphones and headphones.

The use of AI-based tools could offer a competitive advantage in reaching this new market segment. Kumar et al. [57] highlight that text-to-speech (TTS) systems have made significant advancements in recent years and are now a major research topic in developing human-computer interaction technologies. These systems could be used to generate audio versions of news to be distributed via social channels and digital platforms.

In conclusion, although visual and audio formats will not replace written online text, it is likely that these new formats will become an increasingly important part of the information landscape over the next decade.

8.2.2 Future research to improve TST

The task conceived and analyzed in this dissertation is a sub-task of text style transfer. To apply state-of-the-art techniques and technologies in TST to perform style transfer from a standard journalistic style to a conversational one, with the aim of facilitating the dissemination of news on social networks, an in-depth analysis of the existing literature on TST was conducted. During this study, significant limitations emerged, and potential improvements for this well-known task were identified.

A non-parallel TST

The necessity of parallel corpora limits the exploration of new stylistic attributes in text for Text Style Transfer and its applicability to low-resource languages. Although, as highlighted in the previous section, for the specific task examined in this dissertation, a possible generation of benchmark datasets in languages other than english is anticipated through collaboration with ANSA, this is not a definitive and scalable solution. Therefore, improvements in non-parallel supervised techniques, which already exist, or the development of new non-parallel approaches would not only benefit all downstream tasks but also allow for the extension of TST to new and/or emerging tasks in various application contexts. Below are some examples of possible emerging application contexts where TST could be applied:

- Emotion-specific TST. TST can be used to model the emotions expressed in prose (e.g., sadness, happiness, etc.) in an attempt to better reach the target audience of the discourse. For instance, one could imagine a chatbot capable of expressing emotions, even if artificially, thanks to the use of TST, to engage more effectively with a younger audience.

  There is existing work in this direction in the literature. For example, in 2021, Sharma et al. [104] explored an approach to enhance empathy in online mental health support conversations. In the case of the paper, the goal was to transform responses in online conversations from a low-empathy style to a higher-empathy style: this is certainly an excellent starting point, as well as a potential source of inspiration for future improvements.

- Neutral style TST. TST can, by projecting text into a neutral style, remove biases and detoxify the text, making the online environment overall less toxic. A neutral style would also eliminate sensitive data from the text, reducing privacy-related issues.

  For instance, in 2018, Nogueira dos Santos et al. [82] worked on unsupervised TST to translate offensive sentences into non-offensive ones based on social media texts, leaving room for possible improvements.

- TST to break down language barriers. TST can be employed to reduce language barriers through the downstream task of Text Simplification, for which several works exist in the literature. In the medical field, TST could simplify the formal language used in prescriptions or medical reports, thereby reducing the gap between doctors and patients.

These are just a few of the possible emerging and/or innovative tasks that could be further explored thanks to non-parallel TST techniques, which would reduce the costs associated with creating datasets with < source_style, target_style > pairs, not to mention the advantage of more easily extending TST to low-resource languages.

A resource-efficient TST

Resource consumption is an important issue in the field of Machine Learning, particularly Deep Learning, and it is crucial for adopting an ethical approach to research. As highlighted by Menghani [76] from Google Research, Deep Learning has revolutionized various fields of AI/ML; however, with the progressive improvements in models, the number of parameters, latency, and resources required for training have significantly increased. Consequently, it has become important to pay attention to these impact metrics of a model, not just its quality.

Training a deep neural network from scratch is costly in every respect: money, time, and environmental impact. Even fine-tuning large models is often prohibitive, either due to a lack of data or the computational resources required. It is therefore necessary to explore resource-efficient techniques for TST that can achieve comparable results, in some downstream tasks, to those obtainable from state-of-the-art models.

An interesting solution thanks to Activation Engineering A possible solution for resource-efficient Text Style Transfer that, moreover, does not require parallel pairs for training, is based on activation engineering, a technique that focuses on studying the internal activations of a neural network to find useful tools for modifying and adapting existing models. In 2024, Konen et al. [54] sought to derive style vectors by leveraging the power of pre-trained LLMs with the goal of modifying the output of a generalist model.

But how does this approach work [54]? By inputting various documents in the target style into a state-of-the-art model, it is possible to record the average activation values of one or more layers, thereby constructing one or more style embeddings. These vectors are then added, as a difference from a generic vector, to the activation values of the generalist model in an attempt to influence the style of the generated content (example in figure 8.1).

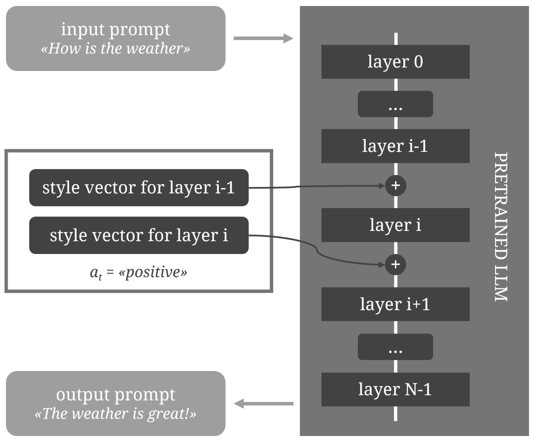


Figure 8.1: Figure derived from the paper [54]. The LLM output is steered by adding style vectors to selected layers (e.g., layers i-1 and i) during a forward pass. For example, the answer of the LLM to the input prompt “How is the weather?” is steered towards a positive style, with a sample answer of “The weather is great!”, a positive answer [54].

Konen et al. [54] demonstrated not only that style vectors can be easily calculated from the recorded activations of layers for input texts in a specific style, but also the effectiveness of activation engineering in subtly and parametrically influencing the style of generated text, distinguishing it from prompt engineering, which is often specific to a particular task and requires prompt formulation.

This approach to TST not only requires minimal computational resources but also a minimal amount of data for training: it is certainly a technology worth exploring further.

Context-aware TST, a multimodal approach

The importance of context, in the era of LLMs, is evident. Nevertheless, what could be a turning point is the modification of TST models to integrate context with features from multi-modalities (e.g., images, audio, video, etc.). The potential applications are numerous. For example, the style could be inferred directly from a visual context, enabling TST without manual intervention to define the target style.

8.2.3 Future research in the world of NLP

In the previous section, some potential improvements for Text Style Transfer were identified. However, this thesis includes a literature review that is not limited specifically to this area but broadly encompasses the entire field of Natural Language Processing. During this analysis of the use of state-of-the-art NLP models and techniques to support the task explored in this thesis, numerous limitations in current NLP approaches emerged. Overcoming these limitations is essential to significantly improving the outcomes in all downstream NLP tasks, making the discipline not only more effective but also more ethical.

Study the impact of pre-processing

It has previously been highlighted how text preprocessing represents a fundamental step in preparing words and phrases before representing them in a machine-readable format, improving model accuracy. However, advancements in deep neural networks, as previously noted, have shifted the focus away from preprocessing; but has it really become an optional step? As emphasized in this dissertation, studies such as the one by Siino et al. [107] have experimentally demonstrated that proper data preparation can still lead to significant improvements in downstream tasks: targeted preprocessing can still make a difference, enhancing the overall effectiveness and performance of models.

Therefore, exploring the impact of preprocessing more deeply is essential to understand how to improve the performance of state-of-the-art models. Furthermore, most studies on the impact of preprocessing have focused on specific tasks, mainly related to Text Classification. It is, therefore, essential to analyze how this practice affects other tasks, such as Natural Language Generation, thereby expanding the understanding of its impact on various downstream tasks.

Removing bias in embeddings

In a previous chapter, embeddings were introduced, highlighting their properties, advantages, and limitations as reported in the literature. These embeddings were subsequently used in this thesis to develop models for text style transfer.

One of the main issues with these vector representations lies in the fact that, while learning the meaning of words from the textual context, embeddings tend to reproduce and amplify the implicit biases and stereotypes present in the training dataset, as emphasized by Zhao et al. [125]. Moreover, although various debiasing techniques exist that aim to address this problem, they are still unable to completely eliminate such biases; often, they merely mask them [32]. As a result, these methods cannot be considered entirely reliable for creating neutral models. The problem of bias is deeper and more systemic than current metrics suggest: there is a need for more comprehensive and sophisticated solutions to tackle it effectively. Furthermore, the debiasing techniques proposed so far often focus solely on gender bias, even though embeddings reflect various types of bias, including those related to religion, ethnicity, and other social constructs. Therefore, a potential future research direction could focus on developing new debiasing techniques that respect the bias-accuracy trade-off: not only would they help make the task presented in this dissertation more ethical and neutral, but they would also benefit many downstream NLP tasks.

List of acronyms

AI	Artificial Intelligence.
BPE	Byte-Pair Encoding.
C4	Colossal Clean Crawled Corpus.
DA	Data Augmentation.
DL	Deep Learning.
GELU	Gaussian Error Linear Unit.
GLU	Gated Linear Unit.
GT	Ground Truth.
HPO	Hyperparameter Optimization.
IDF	Inverse Document Frequency.
IR	Information Retrieval.
J2C	Journalistic to Conversational.
J2C-TST	Journalistic to Conversational TST.
KD	Knowledge Distillation.
LLM	Large Language Model.
LoRA	Low-Rank Adaptation.
ML	Machine Learning.
MLE	Maximum Likelihood Estimation.
MLM	Masked Language Modeling.
NLG	Natural Language Generation.
NLP	Natural Language Processing.
NLU	Natural Language Understanding.
NN	Neural Network.
OCR	Optical Character Recognition.
PEFT	Parameter-Efficient Fine-Tuning.
PLM	Pre-trained Language Model.
PP	Perplexity.
REG	Referring Expression Generation.
ReLU	Rectified Linear Unit.
RNN	Recurrent Neural Network.
SEO	Search Engine Optimization.
SOTA	State Of The Art.
T2V	Text-to-Video.
TC	Text Classification.
TF	Term Frequency.
TST	Text Style Transfer.
TTS	Text-to-Speech.

Glossary

activation engineering A field within machine learning and artificial intelligence focused on designing and optimizing the activation functions used in neural networks. Activation engineering aims to enhance model performance, improve convergence, and ensure stability by selecting and fine-tuning appropriate activation functions for different layers and tasks.

ANSA An Italian news agency.

auto-regressive A type of model that generates data by predicting each element sequentially based on the previously generated elements, commonly used in text and time series generation.

backpropagation algorithm A method used in training neural networks to adjust weights by propagating the error backward through the network to minimize the loss.

backtranslation A data augmentation technique in machine translation where a text is translated from a source language to a target language, and then translated back to the source language to check for accuracy and consistency of meaning between the original and translated texts.

batch In ML, a batch refers to a subset of a dataset that is processed at one time during training or evaluation.

beam search A search algorithm used in sequence generation tasks that keeps track of the top scoring sequences at each step, balancing exploration and efficiency to find the most likely output.

bias A predisposition or inclination, often involuntary, towards a particular point of view, individual, group, or outcome, which can be considered unfair.

bidirectional self-attention A self-attention mechanism that considers both past and future parts of a sequence, allowing the model to understand the context more completely by attending to all elements simultaneously.

bigram A type of n-gram where n=2, meaning it consists of sequences of two contiguous items from a sequence of text.

black box A system or device whose internal workings are not known or accessible, but its input and output are observable.

boilerplate text Non-informative or repetitive text that appears on every page of a document, such as copyright notices or navigation links.

bottom-up algorithm A computational approach that builds a solution to a problem by starting with the simplest or smallest components and combining them incrementally to solve larger and more complex parts of the problem.

cardinality The "number of elements" in a set or other mathematical structure.

chain rule A rule in calculus used to find the derivative of a composite function by differentiating each part step-by-step.

chunk In NLP, a group of continuous words that together form a meaningful unit or phrase within a sentence.

clause A group of words that contains a subject and a predicate.

cloze task A type of language task where words are removed from a text, and the model or reader must fill in the missing words based on the surrounding context.

Colossal Clean Crawled Corpus A large dataset of text collected from the internet, filtered and cleaned for training large-scale language models. Primarily used to train T5.

common crawl A non-profit organization that provides a large-scale, openly accessible repository of web crawl data.

computer vision A field of artificial intelligence and computer science that focuses on enabling computers to interpret and understand the visual world.

conditional probability The probability of an event occurring given that another event has already occurred.

content In the context of a data-driven definition of style, the essential information or meaning conveyed by a text, excluding stylistic elements.

context The information surrounding a particular element or event and influencing its interpretation or behavior.

context window In the context of NLP, a context window refers to the range of words or tokens surrounding a target word that a model considers when analyzing or predicting the meaning, relationship, or function of that word.

contextual embedding In NLP, an advanced extension of embedding, where the representation of a word is not fixed but varies depending on the context in which the word appears.

conversational A style of communication that mimics natural, everyday spoken dialogue. It is characterized by being informal, interactive, and often spontaneous, with a focus on creating a comfortable and engaging exchange between participants .

convolution An operation in neural networks where a filter slides over input data to detect specific patterns or features, commonly used in image and signal processing.

corpus A large collections of texts or documents that are used for training and testing natural language processing models.

correlation Correlation is a statistical measure that describes the extent to which two or more variables fluctuate together.

cosine similarity A measure used to determine the similarity between two non-zero vectors in an inner product space.

cross-attention A mechanism in neural networks where attention is applied across different sequences, allowing a model to align and relate information from one sequence to another, commonly used in encoder-decoder architectures.

curse of multilingualism A challenge faced in multilingual models where the model's capacity must be shared across many languages, often resulting in decreased performance compared to monolingual models due to limited resources per language.

data augmentation A technique used in NL to increase the diversity of a dataset by applying various transformations, such as rotation, scaling, flipping, or noise addition, to the original data, thereby improving the performance and generalization of models.

debiasing technique A method or approach used to reduce or eliminate bias in machine learning models, data, or algorithms.

decoder A component in neural networks that generates output sequences by processing encoded information, commonly used in tasks like language translation and text generation.

degenerate In ML, a task in which the model consistently produces the same output, regardless of the input provided.

differentiable A function or model is said to be differentiable if its derivative exists at all points in its domain..

discourse In linguistics, it is the study the use of language in extended forms of communication.

disinformation Misleading, inaccurate, or outright false information that is disseminated with the explicit intent to deceive.

domain entity A specific term or phrase within a text that is relevant to a particular field or subject area.

downstream task A specific application or task that utilizes the output of a previous process, model, or system to achieve a particular goal. .

dropout A regularization technique used in neural networks to prevent overfitting by randomly setting a fraction of the neurons' outputs to zero during training.

early stopping A regularization technique in machine learning used to prevent overfitting by stopping the training process when the model's performance on a validation set starts to degrade, indicating that further training may lead to overfitting.

effectiveness The degree to which a model accurately and reliably performs its intended task.

embedding In NLP, a numerical representations of words in a vector space, where each word is mapped to a vector of real numbers.

emoji A small digital image or icon used to express an idea, emotion, or concept in electronic communication.

emoticon A textual representation of a facial expression or emotion using combinations of keyboard characters such as punctuation marks, letters, and numbers.

empirical Knowledge, definitions, and methods based on direct experience or observation of facts, rather than on theory or hypothesis.

encoder A component in neural networks that processes input data and transforms it into a fixed representation, often used in sequence-to-sequence models to extract key features.

equivalence class A set of elements that are considered equivalent under a given relation or criteria, meaning that all members of the set share a specific property or satisfy a particular condition.

exploratory analysis The process of analyzing datasets to summarize their main characteristics using statistical graphics and data visualization methods..

F1 score A performance metric in ML that represents the harmonic mean of precision and recall. It provides a balanced measure, especially in cases of imbalanced datasets, by taking both false positives and false negatives into account.

feedforward network A type of artificial neural network where connections between the nodes do not form a cycle, and information moves in one direction from input to output.

fine-tuning A technique used in transfer learning where a pre-trained model, which has been trained on a large and general dataset, is further trained on a domain-specific dataset to adapt it to a particular task.

fluency The ability to produce or comprehend language smoothly, accurately, and with ease, mimicking the proficiency of a native speaker.

foundational model A type of LLM that is trained on a vast amount of data across a broad range of tasks and domains.

functional word A word that primarily has a grammatical role in the text rather than carrying substantial lexical meaning.

generalist model A machine learning model designed to handle a wide range of tasks or domains rather than being specialized for a single specific task.

generalization In the context of machine learning, generalization refers to the ability of a model to perform well on new, unseen data that was not part of the training set.

generator A component or system that produces human-like text from some form of input data.

geometric mean A measure of central tendency that is used to find the average of a set of numbers. It is especially useful for sets of positive numbers whose values are not all at the same scale or when the numbers are multiplicative in nature (e.g., rates of growth). The geometric mean is calculated by multiplying all the numbers together and then taking the nth root (where n is the total number of values).

gradient A vector that represents the direction and rate of the fastest increase of a function, often used to adjust model parameters in machine learning.

Green AI A branch of AI that focuses on reducing the environmental impact of AI research and development, emphasizing energy efficiency and sustainable practices in both the design and deployment of AI systems.

Ground Truth The accurate, real-world data used as a reference to train, validate, and evaluate the performance of a model.

hyperparameter In Machine Learning, a parameter whose value is set before the learning process begins.

hyperparameter optimization The process of finding the best set of hyperparameters for a machine learning model to improve its performance, typically using techniques like grid search, random search, or Bayesian optimization.

inference The process of drawing a conclusion or making a logical judgment based on evidence, reasoning, or data, rather than direct observation.

intonation A variation in pitch while speaking.

invariance That which does not vary or remain constant during a transformation.

inverse document frequency A statistical measure used in information retrieval and text mining to evaluate how important a word is to a document within a collection of documents, also known as a corpus.

in-vivo assessment The evaluation of a machine learning model's performance and behavior in a real-world, operational environment.

joint probability The joint probability is a measure used in statistics and probability theory to describe the likelihood that two or more events occur simultaneously.

knowledge distillation A model compression technique in machine learning where a smaller model (student) is trained to replicate the behavior of a larger, more complex model (teacher) by learning from its outputs, aiming to retain accuracy while reducing computational requirements.

language model A probabilistic framework that defines a probability distribution over sequences of words in a language.

language modeling head A component added to the end of a neural network to predict the next word or token in a sequence, commonly used in language models for tasks like text generation.

latent features In the context of machine learning, latent features refer to hidden or underlying variables that are not directly observed in the data but are inferred by the model during training.

latent space In the context of NLP, latent space refers to a continuous, multi-dimensional space where textual data (such as words, sentences, or documents) are represented as vectors that capture their underlying semantic meanings or relationships. The term "latent" here emphasizes that these semantic features are not directly observable in the original data but are inferred or discovered by the model during training.

learning rate A hyperparameter in ML algorithms that controls the step size at each iteration while moving toward a minimum of the loss function. It determines how quickly or slowly a model updates its weights during training.

lemma The base or dictionary form of a word, from which all its inflected forms are derived.

lemmatization The process in natural language processing (NLP) of reducing a word to its base or dictionary form, known as the lemma.

likelihood A function that, given a set of parameters of a statistical model, returns a numerical value indicating how compatible the observed data are with that model. It is based on probability, but while probability calculates the likelihood of a certain event occurring given a fixed set of parameters, the likelihood, on the contrary, evaluates how likely it is that a particular set of parameters is correct given a fixed set of observed data. A higher likelihood value indicates greater plausibility of the considered parameters in relation to the observed data.

logits The raw, unnormalized output values produced by the last layer of a neural network, which are typically passed through an activation function like softmax to obtain probabilities.

LoRA A technique used in ml to fine-tune large pre-trained models by adapting only a few low-rank matrices, reducing the number of trainable parameters while maintaining model performance.

loss function A function used to measure how well a machine learning model's predictions match the expected outcomes.

lowerBound A value that is less than or equal to every element in a given set or a function's output.

Low-Rank Adaptation A method used for fine-tuning large language models that reduces the number of trainable parameters by decomposing weight updates into low-rank matrices, improving efficiency and reducing computational costs.

low-resource language A language that has limited digital resources available, such as data, tools, and computational models, which makes it challenging to apply natural language processing or other advanced linguistic technologies.

low-resource languages Languages that have limited availability of linguistic resources and tools, such as large-scale corpora, annotated datasets, and natural language processing tools.

Markov assumption An assumption that the future state of a system depends only on its current state, not on how it arrived at that state.

markup A system of symbols or codes inserted in a text document to control its structure, formatting, or the way it is displayed, often used in languages like HTML or XML to define elements within a webpage or document.

masked language modeling A training technique where certain words in a sentence are masked, and the model learns to predict these missing words based on the surrounding context.

Maximum Likelihood Estimation A method to find the best parameters for a model that make the observed data most likely.

message In a NLG system, a formal representations defining entities, concepts, and relationships.

meta-information Data about data, providing context or additional details about the primary information.

meta-informative data Data that provides context or additional information about other data, often used to describe attributes, structure, or relationships of the primary data.

misinformation Information that, while misleading, inaccurate, or completely false, is disseminated without the explicit intent to deceive.

model In linguistics, it is a formalized representation that captures the underlying patterns and structures of language.

morphology In linguistics, it is the study of the structure and meaning of words.

mT5 A multilingual version of the T5 model that is designed to handle a wide range of languages, allowing for text-to-text processing across diverse linguistic inputs. Primarily used to train mT5.

multi-head self-attention An extension of self-attention where multiple attention mechanisms run in parallel, allowing the model to focus on different parts of the input sequence simultaneously and capture a richer set of relationships.

multi-modalities In Machine Learning, the integration and processing of multiple types of data or sensory inputs, such as text, images, audio, and video, within a single model or system.

natural language Any language that has developed naturally through use and evolution among people, as opposed to artificial or constructed languages. It is used as a synonym to refer to human language.

neuron A cell in the nervous system that transmits information by electrical and chemical signals.

news agency An organization dedicated to the collection, verification, and distribution of news to the press and the public.

n-grams Contiguous sequences of n items from a given sample of text or speech.

NLP pipeline A structured sequence of computational operations applied to a text, designed to transform and analyze textual data to extract meaningful information and support natural language processing applications.

noise In the context of NLP, unwanted or irrelevant content within textual data that interferes with the accurate interpretation or processing of the text, potentially obscuring its true meaning.

non-convex optimization A type of optimization where the objective function has multiple peaks and valleys, making it difficult to find the global minimum.

norm A function that assigns a non-negative length or size to vectors in a vector space.

normalization the process of converting text to a standard or canonical form to ensure consistency and improve the effectiveness of subsequent processing steps.

Open Science A set of practices that promote transparency and the open sharing of the scientific process.

optical character recognition A technology that converts different types of documents, such as scanned paper documents, PDFs, or images captured by a digital camera, into editable and searchable data by recognizing and extracting text characters from the images.

order In n-grams, it refers to the number of contiguous items (words, characters, etc.) in each sequence considered by the model or analysis.

overfitting A situation in machine

learning where a model learns the training data too well, including noise and details, resulting in poor performance on new data.

Parameter-Efficient Fine-Tuning A technique used to adapt pre-trained models to specific tasks with minimal additional parameters, reducing computational and storage requirements while maintaining good performance.

paraphrases Sentences that express the same meaning but use different words.

pattern In linguistics, it is a regularity or repetition observable in linguistic data.

perceptron A simple type of artificial neuron used in machine learning, which makes decisions by weighing inputs and applying a threshold.

phonetics In linguistics, it deals with the study of language sounds from a physical and acoustic perspective.

phonology In linguistics, it is the study the sounds of language from the perspective of their role within a conversation.

pipeline A series of data processing steps designed to automate and streamline the workflow of developing, evaluating, and deploying machine learning models.

podcast A digital audio program that is made available for streaming or download via the internet, typically produced as a series of episodes and often focusing on a specific topic or theme.

polysemous word A word that has multiple meanings or senses.

pragmatics In linguistics, it is the study the use of language in real contexts, both from a linguistic and situational perspective.

precision A metric in ML and IR that measures the proportion of true positive predictions out of all positive predictions made by a model. It indicates how accurate the positive predictions are.

prescriptive An approach, definition or methodology that dictates rules or norms to be followed.

probability distribution A mathematical function that provides the probabilities of occurrence of different possible outcomes in an experiment.

prompt A piece of text or question given to a language model to guide its response, often used to generate specific types of outputs or answer questions.

prose It is a way of expressing oneself, both verbally and in writing, that follows a natural and common grammatical structure.

protocol It is set of standardized rules and procedures that establish how specific activities, interactions, or operations are conducted.

query A question, especially one expressing doubt or requesting information.

rank In the context of a matrix, the rank is the maximum number of linearly independent rows or columns in the matrix, indicating the matrix's dimensionality or the amount of independent information it contains.

recall A metric in ML and IR that measures the proportion of true positive predictions out of all actual positive instances in the dataset. It indicates how well a model can identify all relevant instances.

referential form In linguistics, how words or phrases refer to objects, actions, or ideas in the world.

regularization A technique used in machine learning to prevent overfitting by adding a penalty to the model's complexity.

retrospective self-attention A self-attention mechanism that focuses (only) on previous parts of a sequence to improve understanding of current elements, enhancing the model's ability to learn long-term dependencies.

sarcasm It is a form of humor that involves the use of expressions that usually mean the opposite of what is intended to be communicated.

self-attention A mechanism in neural networks that allows the model to focus on different parts of the input sequence to better understand the relationships between elements, crucial for tasks involving language and sequence processing.

self-trained model A model that has been trained using a process where it learns by predicting parts of its own input data, such as predicting the next word in a sequence from a text corpus.

semantics In linguistics, it is the study the formation of the meaning of sentences.

smoothing In statistics refers to techniques used to remove noise or irregularities from data, enabling better identification of patterns and trends.

social media Internet-based platforms that enable users to create, share, and exchange content.

soft label A probabilistic label that represents the likelihood of belonging to each class, instead of assigning a definite class, often used to provide richer information during model training.

source problem The problem of identifying the actual input of an NLG system.

stemmed Refers to the form of a word that has undergone stemming, a common text preprocessing technique in NLP that involves reducing a word to its base or root form. The goal is to remove inflectional or derivational endings, making it easier to analyze the text.

stemming A process in natural language processing (NLP) that reduces a word to its root or base form by removing prefixes, suffixes, and inflectional endings. Unlike lemmatization, stemming may not always produce a linguistically correct root word, but it effectively reduces variations of a word to a common base, facilitating easier text processing.

stop words Commonly used words in a language, such as "the," "is," "in," and "and," that are often filtered out during text processing in natural language processing (NLP) tasks.

string A sequence of characters, typically used to represent text.

style It is a distinctive manner of expression.

suffix A letter or group of letters added to the end of a word to change its meaning, grammatical function, or form.

supervised machine learning A type of machine learning where a model is trained on labeled data to make predictions or classify new data.

syntax In linguistics, it is the study the grammatical structure of sentences.

T5 A transformer-based model developed by Google that treats every NLP task as a text-to-text problem, allowing for a unified approach to diverse language tasks.

term (or index term) A type that has been normalized and included in the vocabulary for processing or analysis.

term frequency A measure used in information retrieval and text mining to evaluate how frequently a term appears in a specific document.

text-to-speech A machine learning technology that converts written text into spoken words using synthetic voice.

text-to-video A machine learning technology that generates video content from textual descriptions.

token A unit of text that is treated as a single entity for processing purposes.

Transformer architecture A neural network architecture designed for handling sequential data, using mechanisms like self-attention to capture relationships between different parts of the input without relying on recurrence.

trigram A type of n-gram where n=3, meaning it consists of sequences of three contiguous items from a sequence of text.

type The class or category of all tokens that share the same character sequence.

uniform distribution A type of probability distribution in which all outcomes are equally likely, meaning that every value within a specified range has the same probability of occurring.

unigram A type of n-gram where n=1, meaning it consists of single items (words, characters, etc.) from a sequence of text.

vanishing gradient problem A challenge in training deep neural networks where gradients become very small, causing the learning process to slow down or stop.

Bibliography

[1] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 'Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning'. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 7319–7328. DOI: `10.18653/v1/2021.acl-long.568`. URL: `https://aclanthology.org/2021.acl-long.568`.

[2] Khetam Al Sharou, Zhenhao Li, and Lucia Specia. 'Towards a Better Understanding of Noise in Natural Language Processing'. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Ed. by Ruslan Mitkov and Galia Angelova. Held Online: INCOMA Ltd., Sept. 2021, pp. 53–62. URL: `https://aclanthology.org/2021.ranlp-1.7`.

[3] Dario Amodei and Danny Hernandez. 'AI and Compute'. In: (2018).

[4] Daniel Andor, Chris Alberti, David Weiss, et al. 'Globally Normalized Transition-Based Neural Networks'. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Katrin Erk and Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 2442–2452. DOI: `10.18653/v1/P16-1231`. URL: `https://aclanthology.org/P16-1231`.

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: `1607.06450 [stat.ML]`. URL: `https://arxiv.org/abs/1607.06450`.

[6] Nikolay Babakov, David Dale, Ilya Gusev, et al. 'Don't lose the message while paraphrasing: A study on content preserving style transfer'. In: *Natural Language Processing and Information Systems: 28th International Conference on Applications of Natural Language to Information Systems, NLDB 2023, Derby, UK, June 21-23, 2023, Proceedings*. Derby, United Kingdom: Springer-Verlag, 2023, pp. 47–61. ISBN: 978-3-

031-35319-2. DOI: 10.1007/978-3-031-35320-8_4. URL: https://doi.org/10.1007/978-3-031-35320-8%5C%5F4.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 'Neural Machine Translation by Jointly Learning to Align and Translate'. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1409.0473.

[8] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 'Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors'. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Kristina Toutanova and Hua Wu. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 238–247. DOI: 10.3115/v1/P14-1023. URL: https://aclanthology.org/P14-1023.

[9] Y. Bengio, P. Simard, and P. Frasconi. 'Learning long-term dependencies with gradient descent is difficult'. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[10] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, et al. 'A neural probabilistic language model'. In: *Journal of machine learning research* 3.02 (2003), pp. 1137–1155.

[11] Yoshua Bengio, Pascal Lamblin, Dan Popovici, et al. 'Greedy layer-wise training of deep networks'. In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. Nips'06. Canada: MIT Press, 2006, pp. 153–160.

[12] Douglas Biber and Susan Conrad. *Register, Genre, and Style*. Cambridge Textbooks in Linguistics. Cambridge University Press, 2009.

[13] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, et al. 'On the Opportunities and Risks of Foundation Models'. In: *CoRR* abs/2108.07258 (2021). arXiv: 2108.07258. URL: https://arxiv.org/abs/2108.07258.

[14] Tim Brooks, Bill Peebles, Connor Holmes, et al. 'Video generation models as world simulators'. In: (2024). URL: https://openai.com/research/video-generation-models-as-world-simulators.

[15] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[16] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 'Model compression'. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Kdd '06. Philadelphia, PA, USA: Association for Computing Machinery, 2006, pp. 535–

541. ISBN: 1595933395. DOI: 10.1145/1150402.1150464. URL: https://doi.org/10.1145/1150402.1150464.

[17] Yu Cheng, Duo Wang, Pan Zhou, et al. 'Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges'. In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 126–136. DOI: 10.1109/msp.2017.2765695.

[18] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, et al. 'Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. URL: https://aclanthology.org/D14-1179.

[19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, et al. 'Unsupervised Cross-lingual Representation Learning at Scale'. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747.

[20] G. Cybenko. 'Approximation by superpositions of a sigmoidal function'. In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (Dec. 1989), pp. 303–314. DOI: 10.1007/bf02551274. URL: http://dx.doi.org/10.1007/BF02551274.

[21] Yann N. Dauphin, Angela Fan, Michael Auli, et al. 'Language Modeling with Gated Convolutional Networks'. In: *CoRR* abs/1612.08083 (2016). arXiv: 1612.08083. URL: http://arxiv.org/abs/1612.08083.

[22] Lorenzo De Mattei, Michele Cafagna, Felice Dell'Orletta, et al. 'GePpeTto Carves Italian into a Language Model'. In: *CoRR* abs/2004.14253 (2020). arXiv: 2004.14253. URL: https://arxiv.org/abs/2004.14253.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[24] Oxford English Dictionary. *information, n.* Sept. 2023. URL: https://doi.org/10.1093/OED/2626286341.

[25] Ning Ding, Yujia Qin, Guang Yang, et al. 'Parameter-efficient fine-tuning of large-scale pre-trained language models'. In: *Nature Machine Intelligence* 5.3 (Mar. 2023), pp. 220–235. ISSN: 2522-5839. DOI: 10.1038/s42256-023-00626-4. URL: https://doi.org/10.1038/s42256-023-00626-4.

[26] Dimitrios Effrosynidis, Symeon Symeonidis, and Avi Arampatzis. 'A comparison of pre-processing techniques for twitter sentiment analysis'. In: *Research and Advanced Technology for Digital Libraries: 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings 21*. Springer. 2017, pp. 394–406.

[27] Susan E. Feldman. 'NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval.' In: *Online* 23 (1999). URL: https://api.semanticscholar.org/CorpusID:60899789.

[28] Steven Y. Feng, Varun Gangal, Jason Wei, et al. 'A Survey of Data Augmentation Approaches for NLP'. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 968–988. DOI: 10.18653/v1/2021.findings-acl.84. URL: https://aclanthology.org/2021.findings-acl.84.

[29] Rudolf Franz Flesch. 'How to write plain English : a book for lawyers and consumers'. In: 1979. URL: https://api.semanticscholar.org/CorpusID:107137765.

[30] Albert Gatt and Emiel Krahmer. 'Survey of the state of the art in natural language generation: core tasks, applications and evaluation'. In: *J. Artif. Int. Res.* 61.1 (Jan. 2018), pp. 65–170. ISSN: 1076-9757.

[31] Yoav Goldberg and Graeme Hirst. *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers, 2017. ISBN: 1627052984.

[32] Hila Gonen and Yoav Goldberg. 'Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them'. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 609–614. DOI: 10.18653/v1/N19-1061. URL: https://aclanthology.org/N19-1061.

[33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618.

[34] Zeyu Han, Chao Gao, Jinyang Liu, et al. *Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey*. 2024. arXiv: 2403.14608 [cs.LG]. URL: https://arxiv.org/abs/2403.14608.

[35] Zellig S Harris. 'Distributional structure'. In: *Word* 10.2-3 (1954), pp. 146–162.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. 'Deep Residual Learning for Image Recognition'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90.

[37] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 'DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing'. In: *CoRR* abs/2111.09543 (2021). arXiv: 2111.09543. URL: https://arxiv.org/abs/2111.09543.

[38] Dan Hendrycks and Kevin Gimpel. 'Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units'. In: *CoRR* abs/1606.08415 (2016). arXiv: 1606.08415. URL: http://arxiv.org/abs/1606.08415.

[39] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML]. URL: https://arxiv.org/abs/1503.02531.

[40] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. 'A Fast Learning Algorithm for Deep Belief Nets'. In: *Neural Computation* 18.7 (July 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. eprint: https://direct.mit.edu/neco/article-pdf/18/7/1527/816558/neco.2006.18.7.1527.pdf. URL: https://doi.org/10.1162/neco.2006.18.7.1527.

[41] Alexander Hogenboom, Daniella Bal, Flavius Frasincar, et al. 'Exploiting emoticons in sentiment analysis'. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. Sac '13. Coimbra, Portugal: Association for Computing Machinery, 2013, pp. 703–710. ISBN: 9781450316569. DOI: 10.1145/2480362.2480498. URL: https://doi.org/10.1145/2480362.2480498.

[42] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 'Multilayer feedforward networks are universal approximators'. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[43] Eduard Hovy. 'Generating natural language under pragmatic constraints'. In: *Journal of Pragmatics* 11.6 (1987), pp. 689–719. ISSN: 0378-2166. DOI: https://doi.org/10.1016/0378-2166(87)90109-3. URL: https://www.sciencedirect.com/science/article/pii/0378216687901093.

[44] Edward J. Hu, Yelong Shen, Phillip Wallis, et al. 'LoRA: Low-Rank Adaptation of Large Language Models'. In: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685. URL: https://arxiv.org/abs/2106.09685.

[45] Zhiqiang Hu, Roy Ka-Wei Lee, Charu C. Aggarwal, et al. 'Text Style Transfer: A Review and Experimental Evaluation'. In: *SIGKDD Explor. Newsl.* 24.1 (June 2022), pp. 14–45. ISSN: 1931-0145. DOI: 10.1145/3544903.3544906. URL: https://doi.org/10.1145/3544903.3544906.

[46] Benoit Jacob, Skirmantas Kligys, Bo Chen, et al. 'Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2704–2713. DOI: 10.1109/cvpr.2018.00286.

[47] Di Jin, Zhijing Jin, Zhiting Hu, et al. 'Deep Learning for Text Style Transfer: A Survey'. In: *Computational Linguistics* 48.1 (Mar. 2022), pp. 155–205. DOI: 10.1162/coli_a_00426. URL: https://aclanthology.org/2022.cl-1.6.

[48] Mandar Joshi, Danqi Chen, Yinhan Liu, et al. 'SpanBERT: Improving Pre-training by Representing and Predicting Spans'. In: *Transactions of the Association for Computational Linguistics* 8 (2020). Ed. by Mark Johnson, Brian Roark, and Ani Nenkova, pp. 64–77. DOI: 10.1162/tacl_a_00300. URL: https://aclanthology.org/2020.tacl-1.5.

[49] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, et al. 'Marian: Fast Neural Machine Translation in C++'. In: *Proceedings of ACL 2018, System Demonstrations*. Ed. by Fei Liu and Thamar Solorio. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 116–121. DOI: 10.18653/v1/P18-4020. URL: https://aclanthology.org/P18-4020.

[50] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. Usa: Prentice Hall PTR, 2000. ISBN: 0130950696.

[51] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sangeetha Sivanesan. 'AMMUS : A Survey of Transformer-based Pretrained Models in Natural Language Processing'. In: *CoRR* abs/2108.05542 (2021). arXiv: 2108.05542. URL: https://arxiv.org/abs/2108.05542.

[52] Diksha Khurana, Aditya Koli, Kiran Khatter, et al. ‘Natural language processing: state of the art, current trends and challenges’. In: *Multimedia Tools and Applications* 82.3 (July 2022), pp. 3713–3744. ISSN: 1573-7721. DOI: 10.1007/s11042-022-13428-4. URL: http://dx.doi.org/10.1007/s11042-022-13428-4.

[53] Diederik P. Kingma and Jimmy Ba. ‘Adam: A Method for Stochastic Optimization’. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[54] Kai Konen, Sophie Jentzsch, Diaoulé Diallo, et al. *Style Vectors for Steering Generative Large Language Model*. 2024. arXiv: 2402.01618 [cs.CL]. URL: https://arxiv.org/abs/2402.01618.

[55] Taku Kudo. ‘Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates’. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 66–75. DOI: 10.18653/v1/P18-1007. URL: https://aclanthology.org/P18-1007.

[56] Taku Kudo and John Richardson. ‘SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing’. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by Eduardo Blanco and Wei Lu. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 66–71. DOI: 10.18653/v1/D18-2012. URL: https://aclanthology.org/D18-2012.

[57] Yogesh Kumar, Apeksha Koul, and Chamkaur Singh. ‘A deep learning approaches in text-to-speech system: a systematic review and recent research perspective’. In: *Multimedia Tools Appl.* 82.10 (Sept. 2022), pp. 15171–15197. ISSN: 1380-7501. DOI: 10.1007/s11042-022-13943-4. URL: https://doi.org/10.1007/s11042-022-13943-4.

[58] Alon Lavie and Abhaya Agarwal. ‘Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments’. In: *Proceedings of the Second Workshop on Statistical Machine Translation*. StatMT ’07. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 228–231.

[59] Y. Lecun, L. Bottou, Y. Bengio, et al. ‘Gradient-based learning applied to document recognition’. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[60] Mike Lewis, Yinhan Liu, Naman Goyal, et al. 'BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension'. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703.

[61] Chuan Li. 'Demystifying GPT-3'. In: *Lambda Labs* (2020). Accessed: 2024-09-26. URL: https://lambdalabs.com/blog/demystifying-gpt-3.

[62] Zhuo Li, Hengyi Li, and Lin Meng. 'Model Compression for Deep Neural Networks: A Survey'. In: *Computers* 12.3 (2023). ISSN: 2073-431X. DOI: 10.3390/computers12030060. URL: https://www.mdpi.com/2073-431X/12/3/60.

[63] Xiaodong Liu, Jianfeng Gao, Xiaodong He, et al. 'Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval'. In: Naacl, May 2015. URL: https://www.microsoft.com/en-us/research/publication/representation-learning-using-multi-task-deep-neural-networks-for-semantic-classification-and-information-retrieval/.

[64] Yinhan Liu, Jiatao Gu, Naman Goyal, et al. 'Multilingual Denoising Pre-training for Neural Machine Translation'. In: *Transactions of the Association for Computational Linguistics* 8 (2020). Ed. by Mark Johnson, Brian Roark, and Ani Nenkova, pp. 726–742. DOI: 10.1162/tacl_a_00343. URL: https://aclanthology.org/2020.tacl-1.47.

[65] Yinhan Liu, Myle Ott, Naman Goyal, et al. 'RoBERTa: A Robustly Optimized BERT Pretraining Approach'. In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

[66] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, et al. 'TimeLMs: Diachronic Language Models from Twitter'. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Ed. by Valerio Basile, Zornitsa Kozareva, and Sanja Stajner. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 251–260. DOI: 10.18653/v1/2022.acl-demo.25. URL: https://aclanthology.org/2022.acl-demo.25.

[67] Bruce T. Lowerre. 'The HARPY speech recognition system'. In: 1976. URL: https://api.semanticscholar.org/CorpusID:61409851.

[68] H. P. Luhn. 'A statistical approach to mechanized encoding and searching of literary information'. In: *IBM J. Res. Dev.* 1.4 (Oct. 1957), pp. 309–317. ISSN: 0018-8646. DOI: 10.1147/rd.14.0309. URL: https://doi.org/10.1147/rd.14.0309.

[69] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, et al. *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods*. `https://github.com/huggingface/peft`. 2022.

[70] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Usa: Cambridge University Press, 2008. ISBN: 0521865719.

[71] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. Cambridge, MA, USA: MIT Press, 1999. ISBN: 0262133601.

[72] David D. McDonald. *Natural Language Generation*. Report Available for Public Release. Amherst, MA: Massachusetts University Amherst Department of Computer and Information Science, Feb. 1986.

[73] David D. McDonald. 'Natural language generation'. In: *Handbook of Natural Language Processing*. Ed. by Nitin Indurkhya and Fred J. Damerau. 2nd. London: Chapman and Hall/CRC, 2010, pp. 121–144.

[74] David D. McDonald and James D. Pustejovsky. 'A computational theory of prose style for natural language generation'. In: *Proceedings of the Second Conference on European Chapter of the Association for Computational Linguistics*. Eacl '85. Geneva, Switzerland: Association for Computational Linguistics, 1985, pp. 187–193. DOI: `10.3115/976931.976958`. URL: `https://doi.org/10.3115/976931.976958`.

[75] Chris Mellish, Donia Scott, Lynne Cahill, et al. 'A Reference Architecture for Natural Language Generation Systems'. In: *Nat. Lang. Eng.* 12.1 (Mar. 2006), pp. 1–34. ISSN: 1351-3249. DOI: `10.1017/s1351324906004104`. URL: `https://doi.org/10.1017/S1351324906004104`.

[76] Gaurav Menghani. 'Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better'. In: *ACM Comput. Surv.* 55.12 (Mar. 2023). ISSN: 0360-0300. DOI: `10.1145/3578938`. URL: `https://doi.org/10.1145/3578938`.

[77] Merriam-Webster. *Style*. Accessed 1 May 2024. Merriam-Webster. 2024. URL: `https://www.merriam-webster.com/dictionary/style` (visited on 05/01/2024).

[78] Bhaskar Miutra and Nick Craswell. 'An Introduction to Neural Information Retrieval'. In: *Found. Trends Inf. Retr.* 13.1 (Dec. 2018), pp. 1–126. ISSN: 1554-0669. DOI: `10.1561/1500000061`. URL: `https://doi.org/10.1561/1500000061`.

[79] Lili Mou and Olga Vechtomova. 'Stylized Text Generation: Approaches and Applications'. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. Ed. by Agata Savary and Yue Zhang. Online: Association for Computational Linguistics, July 2020, pp. 19–22. DOI: 10.18653/v1/2020.acl-tutorials.5. URL: https://aclanthology.org/2020.acl-tutorials.5.

[80] United Nations. *Universal Declaration of Human Rights*. Dec. 1948.

[81] N Newman, R Fletcher, K Eddy, et al. *Digital news report 2023*. Tech. rep. 2023.

[82] Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 'Fighting Offensive Language on Social Media with Unsupervised Text Style Transfer'. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 189–194. DOI: 10.18653/v1/P18-2031. URL: https://aclanthology.org/P18-2031.

[83] Charles E. Osgood, George J. Suci, and Percy H. Tannenbaum. *The Measurement of Meaning*. Urbana, IL: University of Illinois Press, 1957.

[84] Richard Yuanzhe Pang and Kevin Gimpel. 'Unsupervised Evaluation Metrics and Learning Criteria for Non-Parallel Textual Transfer'. In: *Proceedings of the 3rd Workshop on Neural Generation and Translation*. Ed. by Alexandra Birch et al. Hong Kong: Association for Computational Linguistics, Nov. 2019, pp. 138–147. DOI: 10.18653/v1/D19-5614. URL: https://aclanthology.org/D19-5614.

[85] Isabel Papadimitriou, Kezia Lopez, and Dan Jurafsky. 'Multilingual BERT has an accent: Evaluating English influences on fluency in multilingual models'. In: *Findings of the Association for Computational Linguistics: EACL 2023*. Ed. by Andreas Vlachos and Isabelle Augenstein. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 1194–1200. DOI: 10.18653/v1/2023.findings-eacl.89. URL: https://aclanthology.org/2023.findings-eacl.89.

[86] Kishore Papineni, Salim Roukos, Todd Ward, et al. 'BLEU: a method for automatic evaluation of machine translation'. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Acl '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://doi.org/10.3115/1073083.1073135.

[87] Nirali Parekh, Siddharth Trivedi, and Kriti Srivastava. 'Text Style Transfer: A Comprehensive Study on Methodologies and Evaluation'. In: *Big Data, Machine Learning, and Applications*. Ed. by Malaya Dutta Borah et al. Singapore: Springer Nature Singapore, 2024, pp. 269–284. ISBN: 978-981-99-3481-2.

[88] Adam Paszke, Sam Gross, Francisco Massa, et al. 'PyTorch: an imperative style, high-performance deep learning library'. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[89] Matthew E. Peters, Mark Neumann, Mohit Iyyer, et al. 'Deep Contextualized Word Representations'. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Ed. by Marilyn Walker, Heng Ji, and Amanda Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: `10.18653/v1/N18-1202`. URL: `https://aclanthology.org/N18-1202`.

[90] Joshua C. Peterson, Dawn Chen, and Thomas L. Griffiths. 'Parallelograms revisited: Exploring the limitations of vector space models for simple analogies'. In: *Cognition* 205 (2020), p. 104440. ISSN: 0010-0277. DOI: `https://doi.org/10.1016/j.cognition.2020.104440`. URL: `https://www.sciencedirect.com/science/article/pii/S0010027720302596`.

[91] Martin F. Porter. 'An algorithm for suffix stripping'. In: *Program* 40 (1997), pp. 211–218. URL: `https://api.semanticscholar.org/CorpusID:6093716`.

[92] Alec Radford, Jeffrey Wu, Rewon Child, et al. 'Language models are unsupervised multitask learners'. In: *OpenAI blog* 1.8 (2019), p. 9.

[93] Colin Raffel, Noam Shazeer, Adam Roberts, et al. 'Exploring the limits of transfer learning with a unified text-to-text transformer'. In: *J. Mach. Learn. Res.* 21.1 (Jan. 2020). ISSN: 1532-4435.

[94] Nils Reimers and Iryna Gurevych. 'Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks'. In: *CoRR* abs/1908.10084 (2019). arXiv: `1908.10084`. URL: `http://arxiv.org/abs/1908.10084`.

[95] Ehud Reiter and Robert Dale. 'Building applied natural language generation systems'. In: *Nat. Lang. Eng.* 3.1 (Mar. 1997), pp. 57–87. ISSN: 1351-3249. DOI: `10.1017/s1351324997001502`. URL: `https://doi.org/10.1017/S1351324997001502`.

[96] C. J. Van Rijsbergen. *Information Retrieval*. 2nd. Usa: Butterworth-Heinemann, 1979. ISBN: 0408709294.

[97] F. Rosenblatt. 'The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain'. In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: `10.1037/h0042519`.

[98] David E Rumelhart and Adele A Abrahamson. 'A model for analogical reasoning'. In: *Cognitive Psychology* 5.1 (1973), pp. 1–28. ISSN: 0010-0285. DOI: `https://doi.org/10.1016/0010-0285(73)90023-6`. URL: `https://www.sciencedirect.com/science/article/pii/0010028573900236`.

[99] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 'Learning representations by back-propagating errors'. In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. DOI: `10.1038/323533a0`.

[100] Victor Sanh, Lysandre Debut, Julien Chaumond, et al. 'DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter'. In: *CoRR* abs/1910.01108 (2019). arXiv: `1910.01108`. URL: `http://arxiv.org/abs/1910.01108`.

[101] Roy Schwartz, Jesse Dodge, Noah A. Smith, et al. 'Green AI'. In: *Commun. ACM* 63.12 (Nov. 2020), pp. 54–63. ISSN: 0001-0782. DOI: `10.1145/3381831`. URL: `https://doi.org/10.1145/3381831`.

[102] Rico Sennrich, Barry Haddow, and Alexandra Birch. 'Neural Machine Translation of Rare Words with Subword Units'. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Katrin Erk and Noah A. Smith. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: `10.18653/v1/P16-1162`. URL: `https://aclanthology.org/P16-1162`.

[103] C. E. Shannon. 'A mathematical theory of communication'. In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: `10.1002/j.1538-7305.1948.tb01338.x`.

[104] Ashish Sharma, Inna W. Lin, Adam S. Miner, et al. 'Towards Facilitating Empathic Conversations in Online Mental Health Support: A Reinforcement Learning Approach'. In: *Proceedings of the Web Conference 2021*. Www '21. Ljubljana, Slovenia: Association for Computing Machinery, 2021, pp. 194–205. ISBN: 9781450383127. DOI: `10.1145/3442381.3450097`. URL: `https://doi.org/10.1145/3442381.3450097`.

[105] Noam Shazeer. 'GLU Variants Improve Transformer'. In: *CoRR* abs/2002.05202 (2020). arXiv: `2002.05202`. URL: `https://arxiv.org/abs/2002.05202`.

[106] Noam Shazeer and Mitchell Stern. 'Adafactor: Adaptive Learning Rates with Sublinear Memory Cost'. In: *CoRR* abs/1804.04235 (2018). arXiv: `1804.04235`. URL: `http://arxiv.org/abs/1804.04235`.

[107] Marco Siino, Ilenia Tinnirello, and Marco La Cascia. 'Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers'. In: *Inf. Syst.* 121.C (May 2024). ISSN: 0306-4379. DOI: 10.1016/j.is.2023.102342. URL: https://doi.org/10.1016/j.is.2023.102342.

[108] Aditi Singh. 'A Survey of AI Text-to-Image and AI Text-to-Video Generators'. In: *2023 4th International Conference on Artificial Intelligence, Robotics and Control (AIRC)* (2023), pp. 32–36. URL: https://api.semanticscholar.org/CorpusID:264977095.

[109] We Are Social and Meltwater. *Digital 2024 Global Overview Report*. 2024. URL: https://datareportal.com/reports/digital-2023-global-overview-report.

[110] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, et al. 'Dropout: a simple way to prevent neural networks from overfitting'. In: *J. Mach. Learn. Res.* 15 (2014), pp. 1929–1958. URL: https://api.semanticscholar.org/CorpusID:6844431.

[111] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 'Energy and Policy Considerations for Deep Learning in NLP'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3645–3650. DOI: 10.18653/v1/P19-1355. URL: https://aclanthology.org/P19-1355.

[112] Wilson L Taylor. '"Cloze procedure": A new tool for measuring readability'. In: *Journalism quarterly* 30.4 (1953), pp. 415–433.

[113] European Union. *Charter of Fundamental Rights of the European Union*. Vol. 53. Brussels: European Union, 2010, p. 380.

[114] Sebastian Varges. 'Fluency and completeness in instance-based natural language generation'. In: *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*. Coling '02. Taipei, Taiwan: Association for Computational Linguistics, 2002, pp. 1–7. DOI: 10.3115/1072228.1072233. URL: https://doi.org/10.3115/1072228.1072233.

[115] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. 'Attention is all you need'. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Nips'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[116] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 'CIDEr: Consensus-Based Image Description Evaluation'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[117] Douglas Walton. *Fallacies Arising from Ambiguity*. Vol. 1. Jan. 1996. ISBN: 978-90-481-4717-5. DOI: `10.1007/978-94-015-8632-0`.

[118] Hao Wang and Jorge A. Castanon. 'Sentiment expression via emoticons on social media'. In: *2015 IEEE International Conference on Big Data (Big Data)*. 2015, pp. 2404–2408. DOI: `10.1109/BigData.2015.7364034`.

[119] Bernard Widrow and Marcian E. Hoff. 'Adaptive Switching Circuits'. In: *1960 IRE WESCON Convention Record, Part 4*. Institute of Radio Engineers. New York: Institute of Radio Engineers, Aug. 1960, pp. 96–104. URL: `http://www-isl.stanford.edu/~widrow/papers/c1960adaptiveswitching.pdf`.

[120] Ronald J. Williams and David Zipser. 'A Learning Algorithm for Continually Running Fully Recurrent Neural Networks'. In: *Neural Computation* 1.2 (June 1989), pp. 270–280. ISSN: 0899-7667. DOI: `10.1162/neco.1989.1.2.270`. eprint: `https://direct.mit.edu/neco/article-pdf/1/2/270/811849/neco.1989.1.2.270.pdf`. URL: `https://doi.org/10.1162/neco.1989.1.2.270`.

[121] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. 'Transformers: State-of-the-Art Natural Language Processing'. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Ed. by Qun Liu and David Schlangen. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: `10.18653/v1/2020.emnlp-demos.6`. URL: `https://aclanthology.org/2020.emnlp-demos.6`.

[122] Canwen Xu, Wangchunshu Zhou, Tao Ge, et al. 'Beyond Preserved Accuracy: Evaluating Loyalty and Robustness of BERT Compression'. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens et al. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10653–10659. DOI: `10.18653/v1/2021.emnlp-main.832`. URL: `https://aclanthology.org/2021.emnlp-main.832`.

[123] Linting Xue, Noah Constant, Adam Roberts, et al. 'mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer'. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Kristina Toutanova et al. Online: Association for Computational Linguistics, June 2021, pp. 483–498. DOI: `10.18653/v1/2021.naacl-main.41`. URL: `https://aclanthology.org/2021.naacl-main.41`.

[124] Tianyi Zhang, Varsha Kishore, Felix Wu, et al. ‘BERTScore: Evaluating Text Generation with BERT’. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: `https://openreview.net/forum?id=SkeHuCVFDr`.

[125] Jieyu Zhao, Tianlu Wang, Mark Yatskar, et al. ‘Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints’. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2979–2989. DOI: `10.18653/v1/D17-1323`. URL: `https://aclanthology.org/D17-1323`.

[126] George Kingsley Zipf. *The psycho-biology of language : an introduction to dynamic philology*. The psycho-biology of language: an introduction to dynamic philology. Oxford, England: Houghton Mifflin, 1935.