BERGISCHE
UNIVERSITÄT
WUPPERTAL

MASTER THESIS

A Cohesive Distillation Architecture for Neural Language Models

Author:

Jan Philip WAHLE

Examiners:

Prof. Dr.-Ing. Bela GIPP

Dr. Terry LIMA-RUAS

A thesis submitted in fulfillment of the requirements

for the degree of Master of Science

in the

Data & Knowledge Engineering Group

Faculty of Electrical, Information, and Media Engineering

Bergische Universität Wuppertal

January 27, 2021

“ *I have no special talents, I am just passionately curious.* ”

Albert Einstein

In the following thesis, the wording “we” will be used rather than “I” as the ideas were discussed with my advisors and fellow researchers.

Abstract

A Cohesive Distillation Architecture for Neural Language Models

by Jan Philip WAHLE

A recent trend in Natural Language Processing is the exponential growth in Language Model (LM) size, which prevents research groups without a necessary hardware infrastructure from taking part in the development process. This study investigates methods for Knowledge Distillation (KD) to provide efficient alternatives to large-scale models. In this context, KD means the extraction of information about language encoded in a Neural Network and Lexical Knowledge Databases.

To test our hypothesis that efficient architectures can gain knowledge from LMs and extract valuable information from lexical sources, we developed two methods. First, we present a technique to learn confident probability distribution for Masked Language Modeling by prediction weighting of multiple teacher networks. Second, we propose a method for Word Sense Disambiguation (WSD) and lexical KD that is general enough to be adapted to many LMs.

Our results show that KD with multiple teachers leads to an improved training convergence. When using our lexical pre-training method, LM characteristics are not lost, leading to increased performance in Natural Language Understanding (NLU) tasks over the state-of-the-art while adding no parameters. Moreover, the improved semantic understanding of our model increased the task performance beyond WSD and NLU in a real-problem scenario (Plagiarism Detection).

This study suggests that sophisticated training methods and network architectures can be superior over scaling trainable parameters. On this basis, we suggest the research area should encourage the development and use of efficient models and rate impacts resulting from growing LM size equally against task performance.

Acknowledgements

Throughout the writing of this thesis, I have received much support and assistance.

First, I want to thank Professor Bela Gipp for inviting me to complete my graduation at his chair, for his open discussions, and for his support towards my goals.

I am especially grateful to my advisor, Doctor Terry Ruas, for supporting me in all my research aspirations. The last six months have truly been a period of beautiful transformation, and I was lucky to find such a great mentor and friend.

I express my special thanks to Norman Meuschke and Tomáš Foltýnek for their support and helpful discussions in our joint research projects. Furthermore, I want to thank all Data & Knowledge Engineering Group members for the valuable discussions and our joint ventures.

Finally, I want to thank my loving family for their wise counsel and sympathetic ear. Without you, I would not be where I am today. Words simply cannot express how much you mean to me.

Contents

List of Figures

List of Tables

List of Abbreviations

ACC	Accuracy	55
AE	Auto Encoding	7
AR	Auto Regressive	7
ALBERT	A Lite BERT	14
BEM	Bi-Encoder Model	26
BERT	Bidirectional Encoder Representations from Transformers	12
BoT	BERT-of-Theseus	21
BP	Back Propagation	8
BPE	Byte-Pair Encoding	6
BPTT	Back Propagation Through Time	9
CC	Common Crawl	13
CNN	Convolutional Neural Network	9
CNN	Convolutional Neural Network	9
D2V	Paragraph Vector Model	44
dEA	denoising Entity Auto-encoder	22
ERNIE	Enhanced Language RepresentatioN with Informative Entities	21
EWISER	Extended WSD Incorporating Sense Embeddings and Relations	26
FT	fastText	44
GloVe	Global Vectors	44
GLU	Gated Linear Unit	26
GLUE	General Language Understanding Evaluation	36
GPT	Generative Pre-trained Transformer	13
GPU	Graphical Processing Unit	33
GRU	Gated Recurrent Unit	9
KAR	Knowledge Attention and Recontextualization	25
KD	Knowledge Distillation	3
KL	Kullback-Leibler	31
LKB	Lexical Knowledge Databases	3

This thesis is dedicated to my mom for never letting me stop dreaming; to my dad for showing me what really matters in life, and to my sister for always making me laugh.

Related Publications

The content included in this master thesis was partially produced in research projects with my fellow researchers. Their respective locations in the master thesis are listed in the following:

Word Sense Disambiguation and lexical Knowledge Distillation methods (Wahle et al., 2021b) (see Sections 3.2 and 4.2).

> Wahle, J. P., Ruas, T., Meuschke, N., and Gipp, B. (2021b). Incorporating word sense disambiguation in neural language models. *arXiv preprint arXiv:2106.07967*

Machine-Paraphrased Plagiarism detection methods (Wahle et al., 2022b, 2021a) (see Sections 3.3 and 4.3).

> Wahle, J. P., Ruas, T., Foltýnek, T., Meuschke, N., and Gipp, B. (2022b). Identifying machine-paraphrased plagiarism. In Smits, M., editor, *Information for a Better World: Shaping the Global Future*, pages 393–413, Cham. Springer International Publishing

> Wahle, J. P., Ruas, T., Meuschke, N., and Gipp, B. (2021a). Are neural language models good plagiarists? a benchmark for neural paraphrase detection. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 226–229

Chapter 1

Introduction

This master thesis addresses a recent Natural Language Processing (NLP) problem: the increasing Language Model (LM) size regarding trainable parameters and computational requirements. Chapter 1 introduces the problem of this current trend in language modeling (Section 1.1), presents the proposed research objective pursued in this thesis (Section 1.2), and outlines the remaining chapters (Section 1.3).

1.1 Problem Setting and Motivation

From the first definition of a LM for speech recognition systems in the early 1980s (Jelinek, 1997), LMs have achieved breakthrough results in many Natural Language Understanding (NLU) tasks as diverse as text-summarization, sentiment analysis, or part-of-speech tagging. Since the introduction of the transformer model (Vaswani et al., 2017), transfer learning approaches with large-scale pre-trained LMs have become a de-facto standard in NLP (Radford et al., 2019; Devlin et al., 2019; Liu et al., 2019a). In September 2020, Google incorporated the large-scale model BERT (Devlin et al., 2019) for almost every English query[1], and Microsoft exclusively licensed[2] the state-of-the-art model GPT-3 (Brown et al., 2020). While these models show significant improvements in downstream tasks, they come at the cost of usually more than a hundred million parameters. Moreover, it appears further increases in the number of parameters, computational budgets, and data often lead to better results (Kaplan et al., 2020).

[1] https://tinyurl.com/y6573ufm
[2] https://tinyurl.com/y3m7s8zz

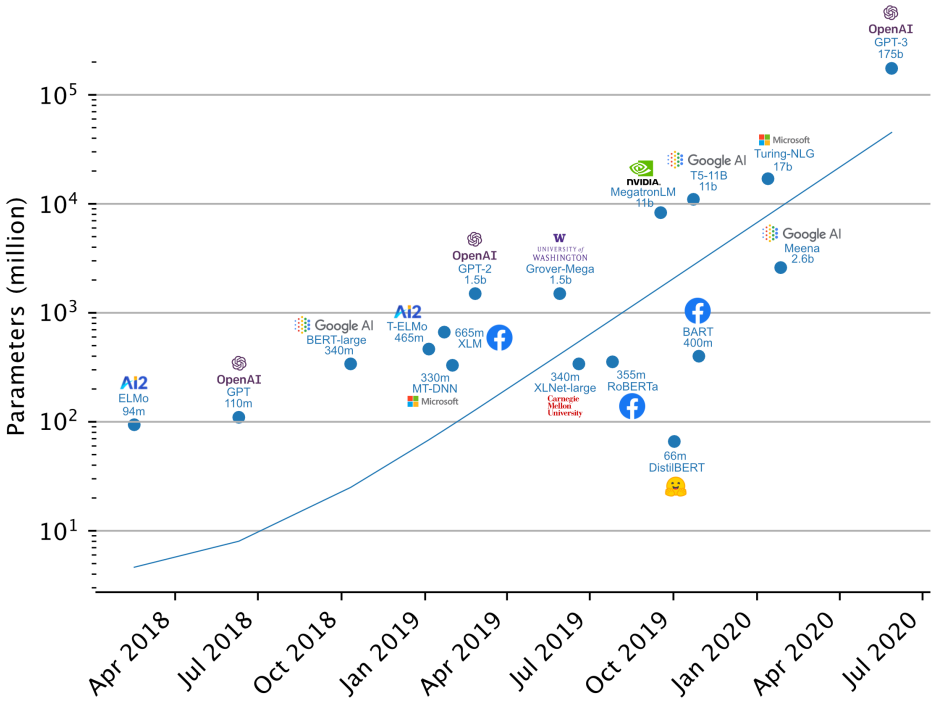

FIGURE 1.1: The increase in Language Model size since the release of ELMo (Peters et al., 2018) with the number of parameters in logarithmic scale and an exponential approximation of two parameters $e^{0.013x} + 3.630$.

This trend of high increases in model size raises three major concerns. First, state-of-the-art models are only reproducible by a small group of organizations and institutions with access to the necessary hardware. Figure 1.1 emphasizes this aspect, where nine large institutions and organizations lead the research with their most impacting models[3]. LM size increasingly prevents researchers without access to the necessary hardware infrastructure from exploring models. At the time of writing, the largest LM contains 175 billion parameters and is only accessible through an official API (Brown et al., 2020). Even if the models were published, researchers would need tremendous amounts of resources to run simple inference in a reasonable time. Second, energy consumption is in linear relation to model size (Schwartz et al., 2019), while model performance seems to follow a logarithmic relation regarding computing budgets (Hestness et al., 2017; Kaplan et al., 2020). The non-linear relation of model performance and energy use makes scaling model size economically and ecologically questionable. Third, computational and memory requirements prevent the deployment

[3]Impact refers to the number of citations to the related research paper.

of neural LMs on-device and inference in real-time.

Studies show that LMs learn redundant features (Voita et al., 2019; Kovaleva et al., 2019; Michel et al., 2019), inspiring the research of alternatives by compression, and Knowledge Distillation (KD) (Sanh et al., 2019; Jiao et al., 2020; Sun et al., 2020b; Gordon et al., 2020). Model compression focuses on parameter optimizations and on reducing computational requirements (Tay et al., 2020). KD leverages the acquired knowledge of large LMs into smaller models while reducing their training time (Romero et al., 2015; Hinton et al., 2015). In this work, we use a compressed model that uses its parameters efficiently. We propose two methods for KD to extract semantic representations of multiple large LMs and to distill the knowledge of Lexical Knowledge Databases (LKB). We release the code to reconstruct our experiments and the pre-trained models. Please refer to Chapter 4 for the corresponding references.

1.2 Research Objective

Motivated by the limitations resulting from large LMs and the trend of further increasing LM size, the following research objective was defined:

Propose, implement, and evaluate a LM training method that distills knowledge from large-scale models and lexical databases into an efficient model to improve its semantic representations.

Therefore, we derived the following research tasks:

Task 1 Perform a comprehensive analysis of state-of-the-art LMs, their strengths and weaknesses, and methods to distill their knowledge.

Task 2 Develop and implement a prototype training architecture to incorporate knowledge of larger models and external sources into an efficient architecture.

Task 3 Evaluate the proposed methods in the tasks of Word Sense Disambiguation (WSD), Machine-Paraphrased Plagiarism (MPP) detection, and general NLU tasks.

1.3 Outline

Chapter 1 introduced the problem of current trends in language modeling (Section 1.1) and presented the proposed research objective for this thesis (Section 1.2).

Chapter 2 addresses research Task 1 by providing fundamental background knowledge and analyzing related work in the domain of language modeling (Section 2.1), WSD (Section 2.3.1), and Plagiarism Detection (PD) (Section 2.3.2). Starting with a definition of LMs, we discuss recent models based on the transformer architecture, efficient alternatives, and methods for KD.

Chapter 3 is concerned with research Task 2 and presents our developed methods. We begin presenting a novel approach for KD with multiple large-scale LMs into an efficient architecture (Section 3.1). Next, we propose a training architecture to incorporate knowledge from LKB and to perform the task of WSD (Section 3.2). We close the chapter by showing how transformer models can be applied to PD for machine-paraphrased text (Section 3.3).

Chapter 4 addresses research Task 3 by testing the models' generalization against related publications. We evaluate WSD methods incorporating LKB on five benchmarks and their generalization on eight NLU datasets (Section 4.2). The final experiment evaluates our architecture in the task of MPP on a recently presented dataset containing text from research papers, graduation Theses, and encyclopedia articles (Section 4.3).

Chapter 5 presents the final considerations to this work. Concluding from the experiments, our methods reduce the amount of computational time and model parameters by using external knowledge and information encoded in large models (Section 5.1). We think the presented systems can have a broader impact on other areas (e.g., Mathematical Information Retrieval) and include more researchers in the development process of LMs (Section 5.2). Finally, we present research directions for future work (Section 5.3).

Chapter 2

Background and Related Work

LMs are a fundamental component of language processing and the main focus of this thesis. This chapter provides the reader with fundamental background knowledge (Sections 2.1.1 to 2.1.3) necessary to explore sophisticated LM training architectures based on transformers (Sections 2.1.5 and 2.1.6). After introducing efficient architectures for LMs (Section 2.1.7), we explore methods to distill the knowledge of a Neural Network (NN) by comparing three important knowledge concepts (Section 2.2.1). We review KD methods for transformer LMs (Section 2.2.2) and conclude that multi-teacher KD and the incorporation of external knowledge appear as promising research directions. Finally, this chapter presents two applications, the task of WSD (Section 2.3.1) and MPP detection (Section 2.3.2).

2.1 Language Models

LMs assign probabilities to parts of unseen text based on prior knowledge of observed text (Jurafsky and Martin, 2009, Chapter 3). For example, a LM might assign a higher probability to “a bit of text” than to “aw pit tov tags” because the words in the former phrase occur more frequently in a text corpus (Hiemstra, 2009).

2.1.1 Fundamentals

One method to estimate a word’s probability in a sentence is by using the *relative count frequency* of sentences (Jurafsky and Martin, 2009, Chapter 3). For example, relative count frequency measures the probability of the word “text” following the sentence “a bit of”, by counting occurrences of “a bit of text” and relating them to all

sentences starting with “a bit of”. Equation (2.1) formalizes the conditional probability expressed by relative count frequencies (where P is a probability measure and C is a counting function over the corpus).

$$P(\text{“text”} \mid \text{“a bit of”}) = \frac{C(\text{“a bit of text”})}{C(\text{“a bit of”})} \tag{2.1}$$

While estimating probabilities from word counts works well in many cases, the corpus size has to be large and unseen variations (e.g., "A bit of *fun*") have no assigned probability. Also, modeling the joint probability of a sentence requires estimating all sentences' probabilities with the same number of words. The poor generalization resulting from sparse representations with many zero probabilities and a high modeling complexity inspired more sophisticated methods for modeling the joint probability of words. Before continuing with these methods, we define a standard set of terms and symbols.

We represent a string of contiguous characters as a *token* and an algorithm that transforms a sequence of characters into tokens as a *tokenizer*. In this work's context, a *token* represents either a word or a word piece (i.e., a sub-word which, together with other sub-words, can construct a word). We use the terms *token* and *word* interchangeably. A popular technique for tokenization in the area of LMs (Radford et al., 2019; Devlin et al., 2019) is the compression algorithm Byte-Pair Encoding (BPE) (Sennrich et al., 2016) that many tokenizers use, e.g., SentencePiece (Kudo and Richardson, 2018) or WordPiece (Schuster and Nakajima, 2012). BPE initializes a set of every character in a corpus to learn efficient merging rules. The algorithm finds frequent (sub-)words and encodes them efficiently using Huffman coding (Furht, 2006).

We represent a sequence of n words as $w_1, ..., w_n$ or $w_{\leq n}$ (with $w_{<n}$ meaning $w_1, ..., w_{n-1}$). The joint probability of words in a sequence having a particular value is $P(w_1, ..., w_n)$. With this notation, we can model the joint probability of a sequence as a forward product by using the chain rule (see Equation (2.2)).

$$P(w_1, ..., w_n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_{<3})P(w_n, w_{<n}) \tag{2.2}$$

$$= \prod_{i=1}^{n} P(w_i \mid w_{<i}) \tag{2.3}$$

LMs can be generally divided into Auto Regressive (AR) and Auto Encoding (AE) models. AR models use the forward product of Equation (2.2), a backward product or a combination of both (Peters et al., 2018). A backward product is analogous to the forward one in but with right-to-left context, and the factorization $P(w1, ..., w_n) = \prod_{i=n}^{1} P(w_i|w_{>i})$. AE models do not perform explicit density estimation but reconstruct corrupted inputs.

AR language models can gain robustness by applying the memoryless Markov property (Markov, 1954, Chapter 4) to the conditional probability estimation. A *bigram* models words' probabilities depending to one previous word, i.e., $P(w_i \mid w_{<i}) \approx P(w_i \mid w_{i-1})$. Bigrams reduce zero probabilities for unseen text as word pairs can model a large fraction of possible sentences. For example, when the bigrams "a bit", "bit of", and "of fun" occured in the corupus, the probability $P($"a bit of fun"$)$ can be modeled although this exact phrase never occured. A generalization of bigrams are *n-grams* which access $n-1$ previous words from the context. Smoothing techniques for n-grams (Kneser and Ney, 1995; Chen and Goodman, 1996) eliminate the problem of zero probabilities by reallocating the probability mass of frequent and infrequent n-grams.

2.1.2 Neural Network Language Models

A profound downside of n-gram LMs is *the curse of dimensionality* that results from modeling the joint probability distribution for all possible n-grams over the *vocabulary* V, which is the set of all (sub-)words in a corpus. The number of possible n-grams grows with $|V|^n$, making the number of parameters to represent large vocabularies and long n-grams infeasibly large. To overcome the curse of dimensionality and to capture long contexts without zero probabilities, Xu and Rudnicky (2000) first introduced a feed-forward NN to learn a bigram model and Bengio et al. (2003) used a NN to

directly approximate each conditional probability distribution from Equation (2.2). The NN receives context words $w_{<i}$ as inputs, which are transformed from their one-hot encoding into a continuous feature vector space by a projection matrix $C \in \mathbb{R}^{|V| \times h}$, with h being the NNs hidden dimensions to learn each conditional distribution. The model predicts $P(w_i \mid w_{<i})$ using the softmax function (Goodfellow et al., 2016, Chapter 6) and optimizes its weights using negative log-likelihood (corresponding to the multi-class cross-entropy (Bishop, 2006, Chapter 4)), the Back Propagation (BP) algorithm (Rumelhart et al., 1986), and Stochastic Gradient Descent (SGD) (Kiefer and Wolfowitz, 1952).

Bi-products of performing language modeling with NNs are high-level semantic features produced in hidden layers with increasing abstractions towards the prediction (Bengio et al., 2013). Hidden representations are often referred to as word embeddings and used in many NLP tasks (Wang et al., 2019c). Compared to *sparse* vectors capturing word frequencies with integers often including many zeros, NN embeddings are *dense* as each value is a real number which results in higher modeling capabilities when the vector size remains equal. Static word embedding methods (Mikolov et al., 2013a; Pennington et al., 2014; Bojanowski et al., 2017) use a unique fixed vector to represent a word's meaning. Context-aware word embedding models (Peters et al., 2018) encode a word with different dense vectors if the context changes.

2.1.3 Recurrent Neural Network Language Models

Using feed-forward NNs as the conditional probability approximator for language modeling has two significant drawbacks. First, fully-connected layers expect a fixed-sized input, making the context size fixed too. Second, feed-forward layers treat inputs as independent and simultaneous features. However, in practice, the context size of sentences varies, and word order can change sentences' semantic content. To introduce higher-level features by capturing longer contextual information and processing variable sized contexts, Mikolov et al. (2011b,a) used Recurrent Neural Networks (RNN)s (Rumelhart et al., 1986) as function approximators for neural LMs. As traditional RNNs tend to be less stable due to the vanishing and exploding gradient problem (Hochreiter,

1991; Bengio et al., 1993, 1994), Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) gained attention for neural LMs (Sundermeyer et al., 2012).

An extension to LSTMs are bidirectional LSTMs (Peters et al., 2018; Howard and Ruder, 2018) which use an additional backward product, analogous to the forward one in Equation (2.2) with the factorization $P(w1, ..., w_n) = \prod_{i=n}^{1} P(w_i|w_{>i})$. Other network architectures include Gated Recurrent Unit (GRU) and Convolutional Neural Network (CNN). When processing character-level information, word embedding models (Peters et al., 2018; Ma et al., 2020) sometimes use GRU networks (Cho et al., 2014) to construct contextual character embeddings serving as inputs to the model. CNNs, the most used architecture in computer vision tasks (LeCun et al., 2015), mainly perform classification tasks in NLP, e.g., sentence matching (Hu et al., 2014), topic categorization (Kalchbrenner et al., 2014), or relation extraction (Nguyen and Grishman, 2015), but are not popular in the language modeling domain (Pham et al., 2016).

2.1.4 Attention in Recurrent Neural Networks

When modeling the probability of a word using its context, intuitively, not every surrounding word is relevant. For example, modeling the probability of the word “python” given a context requires a specific focus on related words such as “snake” or “programming language”. This mechanism is a well known cognitive concept studied in psychology called *attention* (Anderson, 2015, Chapter 3). Our brain selectively concentrates on a specific stimulus to solve a problem while ignoring other perceivable stimuli. Attention for RNNs and LSTMs originated in sequence-to-sequence models, where the model generates one word at a time from an input sequence of words (Bahdanau et al., 2016). Focussing on specific context words while ignoring less important ones, the attention mechanism uses learnable coefficients (Tran et al., 2016; Mei et al., 2016).

Although RNNs and LSTMs using attention show superior performance in many NLP tasks, their recurrent dependence requires a sequential computation of gradients within its layers from the last word to the first one (Goodfellow et al., 2016, Chapter 10) (sometimes referred to as Back Propagation Through Time (BPTT)). BPTT prevents

RNNs and their derivatives to scale with more hardware as sequential parts cannot be parallelized. Modern neural LMs leverage a variation of attention and feed-forward NNs that are highly parallelizable and discussed in the following Section.

2.1.5 The Transformer Model

The introduction of the transformer model (Vaswani et al., 2017) for neural machine translation revolutionized neural language modeling (Radford et al., 2019; Devlin et al., 2019; Liu et al., 2019a). Transformers are feed-forward NNs using an encoder-decoder structure. The encoder maps a sequence of token embeddings to a continuous representation, which is forwarded to the decoder generating an output sequence one word at a time. As the transformer uses feed-forward layers, it overcomes vanishing gradient problems that make learning long-term dependencies difficult and allows for high parallelization, reducing training time. Vaswani et al. (2017) further improve long-term dependency learning by a novel self-attention mechanism that considers the connection of every word to every other word in the input sequence.

Encoder and decoder consist of consecutive modules. The output representations of the final encoder state serve as the input to each decoder module. Each processing step involves mainly three components: fully-connected layers, residual connections, and multi-headed self-attention. Fully-connected layers project the embedding space into a higher dimension to construct more complex features. Residual connections (He et al., 2015) increase the gradient signal for BP by forwarding the identity of the previous layer to the next one achieving higher generalization than chain-like networks (He et al., 2020). The most effective and a well studied architectural component of transformers (Kitaev et al., 2020; Beltagy et al., 2020) is the scaled dot-product attention (which we will refer to as self-attention).

Self-attention estimates a probability distribution over word co-relations within a sequence using softmax. Therefore, the transformer learns three weight matrices, the Key (K), Query (Q), and Value (V). Each matrix is a linear transformation of the word embeddings and has size $s \times d_k$, where s is the sequence length, and d_k is the hidden dimension of the K matrix. Self-attention uses the dot product of Q and K as a similarity measure between word representations. The softmax function applied to

the dot product of Q and K yields an attention score, i.e., a probability distribution representing all word-pairs' influence over the sequence. An additional normalization term prior to the softmax ($\sqrt{d_k}$) reduces large values causing diminishing gradients. The probability distribution multiplied by V selects word representations in proportion to word co-relateions (see Equation (2.4)).

$$Att(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{h}}\right)V \tag{2.4}$$

Compared to RNNs, self-attention and fully-connected layers are large matrices products and can be highly parallelized. Therefore, transformers are scalable and train in a fraction of the time compared to RNNs with the same number of floating-point operations. However, as transformers are feed-forward, they have no time-step dependence like RNNs and require positional information in the input to encode its token position. Transformers typically use a combination of sine and cosine functions of different frequencies added to the token embeddings to represent the word position in the sequence.

In an attempt to capture different aspects of a language, multi-headed self-attention stacks multiple self-attention layers in parallel, similar to kernels in a CNN. Multi-headed self-attention allows the transformer to learn multiple features in each layer. Both methods are illustrated in Figure 2.1.

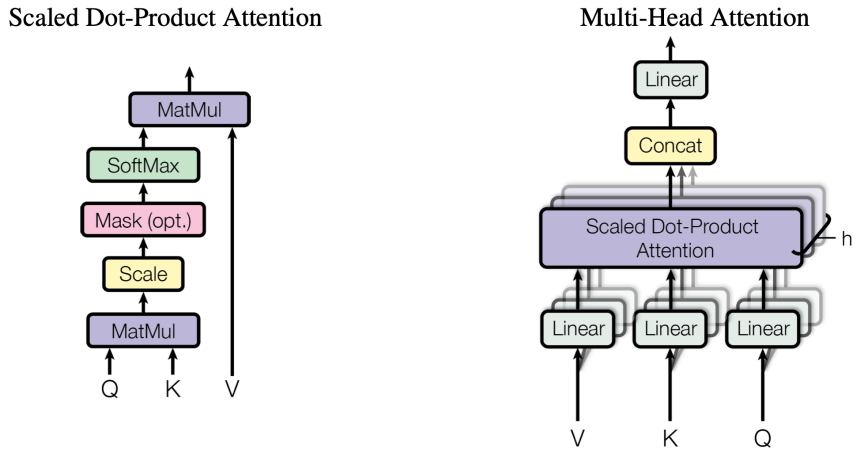

FIGURE 2.1: Scaled Dot-Product Attention (left). Multi-Head Attention (right) (Vaswani et al., 2017).

Self-attention is an effective method to perform neural machine translation, creating semantic encoder representations. These high-level representations inspired many works in the domain of language modeling (Radford et al., 2019; Devlin et al., 2019; Lan et al., 2019; Brown et al., 2020) and shifted the paradigm of how language modeling is approached (see Section 2.1.6).

2.1.6 Transformer-Based Language Models

Universal LM fine-tuning (Howard and Ruder, 2018) introduced an inductive transfer learning paradigm for LMs. Their model first optimizes for the conditional probability distributions over large corpora to capture general language features (pre-training) and then learns new tasks with few additional gradient steps on new datasets (fine-tuning). Pre-training is expensive and typically requires multiple billion tokens but is performed only once. Fine-tuning is cheaper and processes a fraction of the tokens used in pre-training. Howard and Ruder (2018) inspired many models to follow the transfer learning paradigm (Radford et al., 2019; Devlin et al., 2019).

Using the transformer architecture, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) proposes two pre-training tasks to capture general language aspects, i.e. Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM is an AE training method as it reconstructs masked words within a context. For example, given the sentence “I use the programming language python”, words are masked at a certain probability using a special token. The sentence may transform into “I `[MASK]` the programming `[MASK]` python” challenging BERT to reconstruct the masked words using a bidirectional context. NSP predicts whether two sentences are semantically connected. Multiple studies showed the NSP task has little influence on BERT’s performance (Lan et al., 2019; Liu et al., 2019a) which is why we focus on the MLM task in our models (Sections 3.1 and 3.2). For more information about the NSP task, we suggest the original work of Devlin et al. (2019). BERT transforms the final representation of the transformer for each masked word into a probability vector over the vocabulary using softmax and optimizes for the targets using cross-entropy loss.

The same way Word Vector Model (W2V) (Mikolov et al., 2013b) inspired many models in NLP (Bojanowski et al., 2017; Ruas et al., 2019, 2020), BERT echoed in the literature with recent models as well (Yang et al., 2019; Clark et al., 2020). A Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019c) showed additional training data from, a more extensive vocabulary, and more training steps with larger batches, further improve BERT's performance.

XLNet (Yang et al., 2019), an extension of pre-training methods from Transformer-XL (Dai et al., 2019), aims to take advantage of AE concepts for AR language modeling. XLNet explores two of BERT's deficiencies: (1) the corruption of sequences with artificial `[MASK]` tokens, which never occur in regular text, and (2) the prediction of masked words in a single step, assuming independence of words from each other. XLNet removes `[MASK]` tokens and maximizes the expected log-likelihood of a sequence concerning all possible permutations of the factorization order.

ELECTRA (Clark et al., 2020) changes the MLM task of BERT to a generator-discriminator setup. The model substitutes tokens with artificially generated ones from a small masked LM and discriminates them in a noise contrastive learning process (Gutmann and Hyvärinen, 2010). BART (Lewis et al., 2019) pre-trains a bidirectional AE and AR transformer in a joint structure. A two-stage denoising AE first corrupts the input with an arbitrary function (bidirectional) and uses sequence-to-sequence to reconstruct the original input (AR).

Other popular examples of AR models include the three versions of the Generative Pre-trained Transformer (GPT) model, with GPT-3 reaching 175 billion parameters. The training method of GPT-3 mainly relies on predicting consecutive tokens and a large carefully pre-processed dataset composed of Common Crawl (CC) web documents[1]. The T5 model formulates transfer learning tasks as textual questions and performs prediction using text generation of a large AR model. In this work, we exclude the exploration of pure AR models for our models as the targeted downstream tasks do not require text generation. Instead, we focus on the variation of BERT-related AE models which are compatible with each other for KD.

Since the introduction of BERT, a growing area of research is concerned with the question of how to reduce computational requirements and parameter count of LMs

[1] http://commoncrawl.org/

without significant performance loss (Tay et al., 2020). DistilBERT (Sanh et al., 2019) uses the concept of KD (Hinton et al., 2015) with a student-teacher architecture to extract BERT's knowledge into a smaller model. Therefore, DistilBERT initializes with a selection of BERT's layers and optimizes the negative log-likelihood for soft target probabilities, i.e., BERT's logits, with regular introductions of ground truth labels. KD requires fewer training steps compared to training the model from scratch with only ground truth targets and converges to smaller validation losses.

A Lite BERT (ALBERT) (Lan et al., 2019) makes BERT more efficient through factorized embedding parameterizations, cross-layer parameter sharing, and Sentece Order Prediction (SOP). Compared to BERT, RoBERTa, and XLNet which set the hidden layer size h to the WordPiece Embedding size E, i.e. $H = E$, ALBERT minimizes computational requirements for typically large vocabularies V from $\mathcal{O}(V \times H)$ to $\mathcal{O}(V \times E + E \times H)$ with small values for E. ALBERT shares all parameters across layers to reduce parameter count significantly (10% of BERT's parameters for the base model), similar to other strategies for transformers (Dehghani et al., 2018; Bai et al., 2019). ALBERT proposes SOP, as RoBERTa and XLNet found NSP's impact unreliable due to low task difficulty. SOP uses as positive examples two consecutive sentences from the same document, and as negative examples the same sentences but with their order swapped making the task more difficult.

The presented transformer-based LMs rely on self-attention, which is the most computationally expensive layer in the transformer's NN, growing quadratically concerning the sequence length. As one example of exploring new schemes to calculate self-attention with the motivation for capturing larger contexts, Longformer (Beltagy et al., 2020) combines a windowed local-global self-attention scaling linearly with the sequence length in comparison to other models. Section 2.1.7 introduces a taxonomy for efficient transformer architectures which are relevant to explore large-scale LM distillation.

2.1.7 Efficient Transformers

In the first formalization of intelligent machines, Alan Turing described a phenomenon of Machine Learning (ML) with remarkable precision:

"[M]achines of this character can behave in a very complicated manner when the number of units is large."

Alan Turing (1948) "Intelligent Machines", page 6

Supporting Turing's observation, recent studies (Hestness et al., 2017; Kaplan et al., 2020) predict correlations between model size, dataset size, computational budgets, and the problem complexity NNs can solve. Figure 2.2 shows these three factors can accurately predict the test loss of transformer LMs while the model shape and other hyperparameters are negligible.

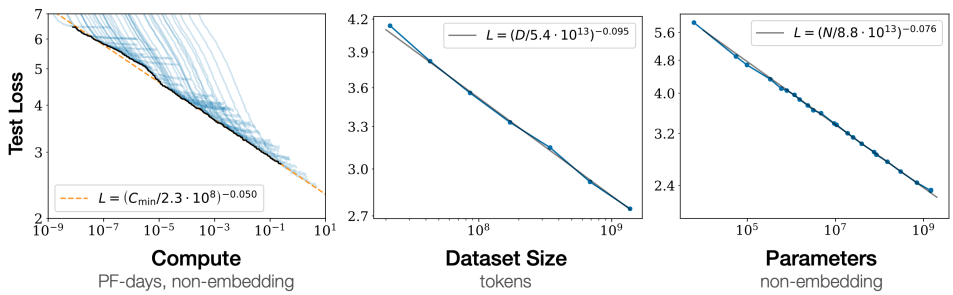

FIGURE 2.2: Test loss for transformer language models with varying compute budgets in Peta Flops (PF), dataset sizes in number of tokens, and parameter counts without embedding layers from Kaplan et al. (2020).

However, in this work, we show how explicit incorporation of external knowledge from LKB (Section 3.2) increases the performance of LMs for downstream tasks while having less computational requirements and an equal dataset size and parameter count. Similar results for the WSD task are present in the literature (Blevins and Zettlemoyer, 2020).

The related fields of regularisation and pruning are concerned with the question of how to reduce model parameters without performance loss. Pruning methods for LMs show that specific attention heads in the transformer architecture are obsolete, and removing them yields marginal performance loss (Voita et al., 2019; Michel et al., 2019). Supporting this hypothesis, KD methods for LMs (Sanh et al., 2019; Wang et al., 2020) extract knowledge of large-scale models into significantly smaller ones while keeping most of their performance.

To distill knowledge into smaller models requires exploring efficient architectures targeting expensive layers regarding the number of parameters and computational budgets. For transformer models, the most computationally expensive component is self-attention. Tay et al. (2020) introduced a taxonomy to divide efficient architectures optimizing attention into five main categories:

Fixed/Factorized/Random Parameter methods reduce the self-attention matrix by limiting the field of view to fixed patterns, e.g., local windows and blocks.

Low Rank/Kernel methods approximate the self-attention matrix by assuming a low rank to decompose it into a smaller dimension. Kernel methods rewrite self-attention to avoid computing large dot-products explicitly.

Recurrence methods forward multiple sequences through the same model keeping previous states.

Memory models are extended with side memory modules to store a temporary context for future processing.

Learnable Patterns approximate the access pattern with data-driven approaches, e.g., clustering methods and hashing functions.

This work constructs a modified version of Longformer (Beltagy et al., 2020) which uses a combination of memory and fixed parameters. We choose Longformer because it performed superior over seven comparable LMs in our related study (Wahle et al., 2022b) and uses MLM, making the model compatible with many transformer-based LMs. We also propose a lexical KD technique that is general enough to be applied to many different LMs (see Section 3.2).

2.2 Knowledge Distillation

In our first years as children, we learn language mainly from observing other humans and interacting with them. We expose ourselves to domain experts during education, whom we call teachers to learn from their knowledge and experiences. In large-scale LMs the trend is different: novel models learn language patterns from scratch instead

of leveraging the acquired knowledge of already trained ones. We argue this course is inefficient and barely adds value to the semantic representations of a LM. Many novel models train on very similar datasets (see Table 2.1 in Section 2.2.2) with small performance gain in downstream tasks.

The field of KD uses a similar teacher-student paradigm to the one employed by humans. When extracting acquired information (knowledge) into a smaller model (distillation), we call the process Knowledge Distillation. A crucial question in the domain of KD is how to better represent knowledge in a NN. In the following Section, we introduce essential knowledge concepts.

2.2.1 Knowledge Concepts

We categorize knowledge concepts using three main categories related to our work:

Logit-Based Knowledge represents information encoded in the final predictions of a model, including semantic information.

Feature-Based Knowledge exists in the hidden representations of a NN that grow with abstraction towards the prediction layer.

External Knowledge consists of structured and explicit sources created by humans to describe concepts.

The different types of knowledge for the transformer architecture are visualized in Figure 2.3. Other types of knowledge include relation-based (Yim et al., 2017; Lee et al., 2018) and self-distilled (Hahn and Choi, 2019; Zhang et al., 2019a) knowledge, which study the relation of features and data to improve representations within the same network. We exclude these types of knowledge because they typically require the extracted network to be of the same size as the pre-trained network contradicting our primary research objective to optimize for a smaller model (see Section 1.2).

Logit-Based Knowledge

The information encoded in the probability distributions of a models' prediction layers represents logit-based knowledge. Logit-based KD uses a soft distribution of a

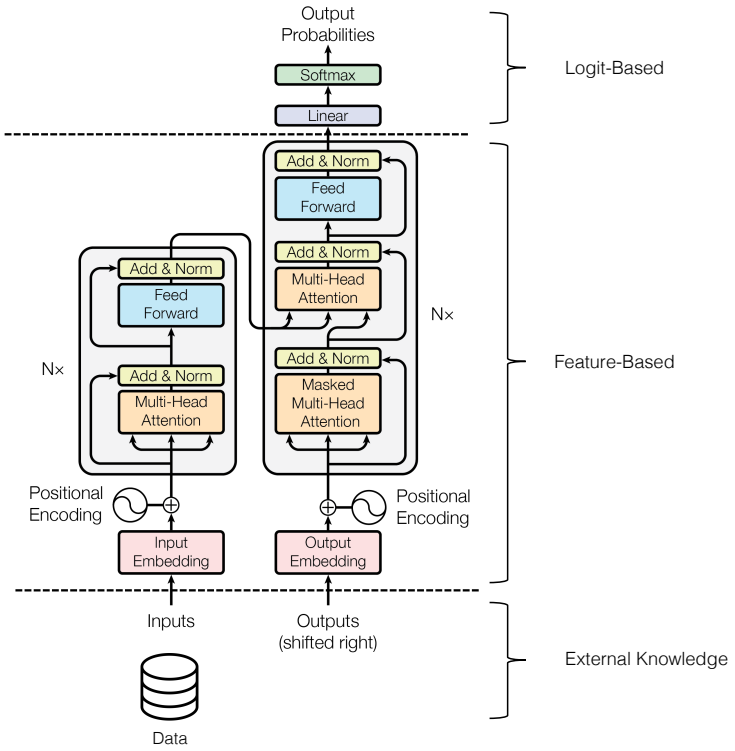

FIGURE 2.3: Knowledge concepts visualized for the transformer architecture. This figure is an adaptation from Figure 1 in (Vaswani et al., 2017).

teacher's prediction layer as the supervision signal for a student (Hinton et al., 2015). For example, in a mutually exclusive multi-class problem to classify whether a text describes one of five classes (e.g., Huskey, Eurasian, cat, car, bus), a trained teacher model assigns a high probability to the actual class (e.g., a Huskey), lower probabilities to similar animals (e.g., the Eurasian), lower but non-zero probabilities to other animals (e.g., cat), but close to zero probabilities to vehicles (e.g., car, bus). This semantic understanding *dark knowledge* and is used to learn semantic features from the teacher model.

The logit-based method proposed by (Hinton et al., 2015) mimics a softer version of a teacher model's predictions. NNs typically learn classification using a softmax layer to produce probabilities q_i from logits z_i with a temperature parameter T as Equation (2.5) shows.

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_i/T)} \tag{2.5}$$

Usually, classification tasks set the temperature T to 1. Larger values of T produce

a softer distribution (i.e., non-maximum logits increase and maximum logits decrease). Logit-based KD uses the soft target distribution from the teacher network with $T > 1$ to estimate a less strict output distribution. When the student model converged to model the teacher's soft distribution, T is set back to 1. Additionally, logit-based methods regularly introduce the ground truth labels, proving to increase the student model's performance.

Logit-based KD uses one part of the teacher network information, the final predictions. However, when working with NNs, all hidden representations are available. Especially in deep learning, additional supervision signals using hidden representations improve training stability and reduce training time (Romero et al., 2015; Jiao et al., 2020).

Feature-Based Knowledge

Hidden representations in NNs encode high-level features that increase in abstraction with layers closer to the final prediction (Bengio et al., 2013). Feature-based KD uses these high-level features with regular supervision signals between hidden layers of the student and teacher model (Romero et al., 2015; Zagoruyko and Komodakis, 2017; Kim et al., 2018). Thin deep networks (Romero et al., 2015) propose hints, which are comparisons of hidden features from the student and teacher model using a regression loss. An extension of thin deep networks related to this work creates supervision signals using an attention score over features (Zagoruyko and Komodakis, 2017). The generalization of these attention scores calculates distributions of selectivity patterns and compares them rather than comparing features directly (Huang and Wang, 2017). Probabilistic knowledge transfer explores the probability distribution of the data in the feature space instead of their sample representation (Passalis and Tefas, 2019).

External Knowledge

Different from implicit knowledge found in trained NNs, we can find knowledge in more explicit external sources, i.e., knowledge acquired and structured by humans to describe natural phenomena. For example, the Wikidata Knowledge Graph[2] contains

[2] https://www.wikidata.org/

91 million entries describing relations of entities, e.g., humans, architectural structures, or chemical elements. Knowledge-based methods for WSD commonly explore the knowledge of LKB. A popular example of a LKB is WordNet (Miller, 1995; Fellbaum, 1998b) which contains, e.g., Part Of Speech (POS) tags for words, word relations, and glosses[3].

External knowledge sources have two significant advantages over knowledge obtained through NNs. First, they are interpretable by humans in a straight-forward manner. Second, their knowledge can be used explicitly to optimize learning a specific aspect of language (e.g., ambiguity). Unsupervised LM pre-training proved to learn features of a language successfully, but with the cost of typically multiple billion tokens of optimization. With explicit knowledge from external sources, we show our models can gain superior semantic understanding while keeping the parameters low.

2.2.2 Language Model Knowledge Distillation

KD has been extensively studied in the NLP domain (Kim and Rush, 2016; Hu et al., 2018). As the usage of large-scale neural LMs with many parameters increased, KD gained much importance for the LM research as well.

DistilBERT (Sanh et al., 2019) uses the logit-based KD algorithm described in (Hinton et al., 2015) to learn soft target probabilities of BERT with the supervised MLM loss. Additionally, DistilBERT optimizes its embedding output similarities with the cosine embedding loss yielding similar embeddings to the teacher model. To transfer prior knowledge of BERT into a smaller model, DistilBERT keeps the weights of selected layers from the original model. As discussed, the hidden size has to remain equal between DistilBERT and BERT when initializing BERT's weights. Therefore, DistilBERT controls model size through the number of layers. The Patient KD algorithm proposed by Sun et al. (2019) focuses on learning aggregate representations of the teacher model, i.e., the `[CLS]`-token in BERT, rather than masked word soft-targets. The aggregate encodes the highest semantic information about a sequence and is typically used for classification tasks.

[3] Gloss: A brief definition of a word sense.

TinyBERT (Jiao et al., 2020) combines feature-based and logit-based approaches to imitate several larger layers into smaller ones (i.e., embedding layer, attention layer, hidden layer, prediction layer). In addition to pre-training, TinyBERT performs KD at the fine-tuning step. LadaBERT (Mao et al., 2020) reduces parameters and computational complexity with Singular Value Decomposition (SVD) over the weight matrices and uses weight pruning (Han et al., 2015; Blalock et al., 2020). Mao et al. (2020)'s KD methods are similar to TinyBERT using logit-based and feature-based knowledge.

MobileBERT (Sun et al., 2020b) proposes bottleneck structures which are smaller alternatives to BERT's modules and balancing between multi-headed self-attention and feed-forward layers. To extract knowledge into the new structure, MobileBERT first trains a modified BERT model, which uses the bottleneck architecture components. The student then distills knowledge from the modified BERT layer-to-layer. Training the modified BERT model adds additional training time, and performing layer-to-layer distillation fixes the number of layers and the teacher and student's hidden size.

MINILM (Wang et al., 2020) focuses on imitating the multi-headed self-attention modules of the teacher's transformer to gain feature-based knowledge. Wang et al. (2020) show high-level attention layers encode the highest semantic information. Thus, imitating layers close to the prediction step appears to yield similar performance with less supervision. Moreover, MINILM uses a teacher assistant (Mirzadeh et al., 2020), which stabilizes the KD loss when the teacher model is much larger than the student model. MINILM has no requirement on the hidden size or the number of layers of the student and does not perform layer-to-layer distillation, making the student model more flexible.

BERT-of-Theseus (BoT) (Xu et al., 2020) creates small substitutes for BERT's large layer modules. The model learns compact substitutes and large modules simultaneously in BERT and increasingly replaces large modules with compact ones with advanced training progress. The smooth transition from large modules to compact modules results in an interaction of original modules and their compact replacements. This interaction proves to be superior to starting with many compact layers.

Enhanced Language RepresentatioN with Informative Entities (ERNIE) (Zhang et al., 2019b) uses entities from the Wikidata Knowledge Graph with a token encoder

and an entity encoder. The entity encoder uses an aggregation sub-layer to perform multi-headed self-attention over the tokens and entities, concatenating their resulting embeddings. The pre-training objective is a denoising Entity Auto-encoder (dEA), which randomly masks token-entity alignments which the transformer needs to predict all corresponding entities based on aligned tokens.

The extension of ERNIE, ERNIE 2.0 (Sun et al., 2020a) explores the incorporation of different pre-training objectives in a multi-task setup using explicit knowledge or self-supervised labels. Replacing training objectives typically reduces previously learned features as they are not needed to perform the new task. However, ERNIE 2.0 maintains knowledge aspects by keeping objectives at a small percentage during training, resulting in semantically richer features. We summarize the most popular presented transformer-based LMs and provide a comprehensive overview in Table 2.1.

TABLE 2.1: Overview of transformer-based language models. Wikipedia refers to the official English Wikipedia Dump. The dash symbol "-" indicates that no official information was provided in the related research paper.

Model	Dataset			Model	
	Corpora	Size / Tokens	Tokens / Epochs	Parameters	Training Procedure
AlBERT (2019)	Books (2015), Wikipedia	13GB / 3.3b	262b / 80	large / xxlarge : 17m / 223m	MLM + SOP
BART (2019)	Books (2015), Wikipedia	13GB / 3.3b	2.2t (large)	base / large : 139m / 406m	Denoising Auto Encoder
BERT (2019)	Books (2015), Wikipedia	13GB / 3.3b	137b / 40	base / large : 110m / 340m	MLM + NSP
ELECTRA (2020)	Books (2015), CC[a], ClueWeb 2012-B[b], Gigaword 5 (2012), Wikipedia	158GB / 33b	419b / 12.7	335m	Replaced Token Detection
ERNIE (2019b)	Wikidata, Wikipedia	- / 4.5m	- / 1	114m	MLM + NSP + dEA
ERNIE 2.0 (2020a)	Books (2015), Discovery Data (2019), Reddit, Wikipedia	19GB / 7.9b	78b / 10	base / large : 110m / 340m	Knowledge Masking + Capital Prediction + Token Document Relation + Sentence Reordering
GPT-2 (2019)	Open Web Text (2019)	40GB / 10.2b	- / -	medium / xl : 345m / 1.6b	Left-to-right LM
Longformer (2020)	Books (2015), CC-Stories (2019), Realnews (2019), Wikipedia	31.6GB / 6.4b	39b / 6	base / large : 149m / 435m	MLM
MT-DNN (2019b)	GLUE (2019b)	4GB / 25m	123m / 5	base / large : 110m / 340m	Multi-Task Finetuning
RoBERTa (2019c)	Books (2015), CC-Stories (2019), CC-News[c] (Sept 2016 - Feb 2019), Open Web Text (2019), Wikipedia	160GB / 33b	2.2t / 66	base / large : 125m / 360m	MLM + large batches + full sentences + large BPE vocab (50k pieces)
T5 (2020)	C4 dataset (2020)	803GB / 165b	34b / 0.2	T5-large / T11 : 770m / 11b	fill-in-the-blank prediction
XLNet (2019)	Books (2015), CC[a], ClueWeb 2012-B[b], Gigaword 5 (2012), Wikipedia	158GB / 32b	2.2t / 66	base / large : 110m / 340m	Permutational LM

[a] http://commoncrawl.org/ [b] https://lemurproject.org/clueweb12/ [c] http://commoncrawl.org/2016/10/news-dataset-available/

This thesis shows the combination of logits and high-level features can extract knowledge into an efficient transformer architecture. Using external knowledge from LKB, our model gains semantic understanding beneficial for many NLU tasks.

2.2.3 Multi-Teacher Knowledge Distillation

Different models provide individual knowledge aspects to perform tasks. The *no free lunch theorem* states, there exists no single model that is best suited for all possible scenarios and data sets (Ho and Pepyne, 2002). Instead, different LM architectures are experts in specific language characteristics.

Hinton et al. (2015) showed that ensembles of expert models are superior to a single generalized model. However, ensembles increase the cost of training and inference as they use multiple models simultaneously. To combine the knowledge of many experts, multi-teacher KD combines the response from many teachers as the supervision signal (e.g., averaging the soft-targets (Hinton et al., 2015)).

Tarvainen and Valpola (2017) average the teacher parameters instead of predictions and use a consistency loss between teacher's and student's predictions. The model keeps a moving average of label predictions on each training example and penalizes predictions inconsistent with the average. You et al. (2017) extend learning soft-targets to features using a triplet loss by increasing the dissimilarity in feature representation between different examples and vice-versa.

In the domain of LMs, some publications explore multi-task KD (Liu et al., 2019a), i.e., learning multiple tasks simultaneously to achieve more general representations, but ignore multi-teacher concepts. We see a gap in the literature to apply multi-teacher KD to large-scale LMs.

In this work, we transfer the concept of multi-teacher KD to large-scale LMs to gain their knowledge aspects. Our method weights the soft-target distribution of multiple teachers focusing on models with the highest prediction confidence.

2.3 Applications

To validate a LM captures the characteristics of language, it needs to produce semantic representations beneficial for many NLU tasks. NLU benchmarks (Wang et al., 2019b; Sarlin et al., 2020) include a wide range of tasks and cover individual aspects of the target language, e.g., polysemy or text similarity.

In particular, WSD is a difficult task to perform. Navigli (2009) showed F1 scores above 80% are difficult to surpass as the inter-annotator agreement (i.e., the percentage of words tagged with the same sense by two or more human annotators) is 67% to 80% for fine-grained, WordNet-style sense inventories. Polysemy frequently occurs in natural language, making it a crucial task to accomplish for any NLP system. Furthermore, WSD appears to be sensitive to changes in the LM architecture and training scheme (Wahle et al., 2021b) which makes it a suitable task to validate our proposed methods against different LM architectures.

Another interesting application for LMs lies in the domain of MPP detection (Foltýnek et al., 2020b; Wahle et al., 2022b). Academic plagiarism has become a pressing problem in our society. We know of more than 30 cases of academic plagiarism in Germany, including former minister of defense Karl-Theodor zu Guttenberg[4]. Modern plagiarists use machine-paraphrasing tools, which typically remain undetected by PD systems using word-based or character-based text comparisons with an indexed corpus (Foltýnek et al., 2019). Real-world forms of plagiarism require detecting re-constructed and replaced parts of the text with probably unknown sources (e.g., Theses material).

2.3.1 Word Sense Disambiguation

WSD seeks to determine the meaning of words given a context and is a fundamental challenge in NLP (Navigli and Ponzetto, 2012). For example, the sentence "I like java" is ambiguous as "java" can refer to the programming language, an island, or coffee[5].

In knowledge-based methods (Camacho-Collados et al., 2015), LKB (e.g., WordNet (Fellbaum, 1998b), ConceptNet (Liu and Singh, 2004), BabelNet (Navigli and Ponzetto, 2012)) are used as a taxonomy to help categorize the relationship between words and their meaning. While unsupervised techniques (Chaplot and Salakhutdinov, 2018) do not rely on annotated data to perform disambiguation, supervised-based ones (Pasini and Navigli, 2020) explore human-labeled or automatically generated annotations.

NN-based models (Bengio et al., 2003; Mikolov et al., 2013b; Bojanowski et al., 2017) have gained much attention in the NLP community, mainly because of their success

[4] https://guttenplag.wikia.org/de/wiki/GuttenPlag_Wiki
[5] according to the LKB BabelNet 4.0 https://babelnet.org/

to capture latent semantic content and superior performance in tasks, such as word similarity (Neelakantan et al., 2014), text classification (Ruas et al., 2020), and topic categorization (Pilehvar et al., 2017). Ruas et al. (2019) combine WordNet (Fellbaum, 1998b) and W2V (Mikolov et al., 2013b) to represent word senses in fixed-sized length vectors. Their Most-Suitable Sense Annotation (MSSA) algorithm combines the vector representation for each word in WordNet's glosses using a general pre-trained word embeddings model. MSSA averages the word representations in a context sliding window to select the meaning of a word with the highest similarity within its adjacent neighbors. However, the technique does not explore the benefits of transfer learning between different tasks. Likewise, this thesis considers the glosses in WordNet but relies on the LM do identify the correct sense of a word given its context with an end-to-end approach.

Using a large-scale LM, GlossBERT (with their best performing method Sent-CLS-WS) (Huang et al., 2019) uses WordNet's glosses to fine-tune BERT for the WSD task. GlossBERT classifies a marked word in a sentence into one of its possible definitions. For each word sense of an ambiguous word, GlossBERT creates a pair of the context and a gloss using the BERT tokenizer. Their method inserts two supervision signals: first, highlighting ambiguous tokens with two unique tokens, and second, repeating the ambiguous tokens with the gloss definition. Du et al. (2019) fine-tune BERT similarly with the encoder and a classifier but without additional supervision.

KnowBERT (Peters et al., 2019) incorporates LKB into BERT (e.g., WordNet) with a Knowledge Attention and Recontextualization (KAR) mechanism. The KAR component uses mention-spans to retrieve entity embeddings from a LKB, update the mention-span embeddings with the linked information and recontextualize the entity embeddings with altered multi-head attention. Their multi-head self-attention is modified so a word-piece can attend to all entity embeddings in the context. Peters et al. (2019) best-performing model, i.e., KnowBERT combining Wikipedia and WordNet (KnowBERT-W+W), achieves better results than both BERT models ($base$ and $large$). However, KnowBERT-W+W adds over 400 million parameters compared to BERT$_{base}$ and is 32% slower. Other approaches combining BERT and WordNet for the WSD task, but less related to our contributions, are also discussed in the literature (Vial et al.,

2019).

Blevins and Zettlemoyer (2020) propose Bi-Encoder Model (BEM), a model to encode a target word with its surrounding context and the gloss of each sense in separate encoders. Their two encoders are learned simultaneously from the WSD objective. Extended WSD Incorporating Sense Embeddings and Relations (EWISER) (Bevilacqua and Navigli, 2020), an extension of (Kumar et al., 2019) uses different word embeddings as input to include semantic relations (e.g., hypernym) in their structure. Language Model Makes Sense (LMMS) (Loureiro and Jorge, 2019) combines $k-$NN using the Most Frequent Sense (MFS) in WordNet with BERT embeddings. Gated Linear Unit (GLU) (Hadiwinoto et al., 2019) integrates contextualized word representations in WSD. The previous techniques enhance semantic representations via context, external knowledge, or MFS, but do not explore generalization to other NLP tasks.

The methods proposed in this thesis do not require recurrent embeddings adjustments from the LKB nor use word-piece attention, resulting in less overhead to the system. We show by choosing the best-suited model for WSD and adjusting the training procedure, our methods outperform preceding techniques in WSD and obtains the highest score in 7 out of 9 NLU tasks.

2.3.2 Machine-Paraphrased Plagiarism Detection

Plagiarism is a severe form of academic misconduct and a pressing problem for educational and research institutions, publishers, and funding agencies (Foltýnek et al., 2019). To counteract plagiarism, many institutions employ PD systems. These tools reliably identify duplicated text yet are significantly less effective for paraphrases, translations, and other concealed forms of plagiarism (Meuschke and Gipp, 2013; Foltýnek et al., 2020a).

Studies (Rogerson and McCarthy, 2017; Prentice and Kinden, 2018) show an alarming proportion of students employs online paraphrasing tools (often referred to as text spinning tools) to disguise text taken from other sources. These tools are expected to employ NN approaches to change a text, e.g., replacing words with potential synonyms (Zhang et al., 2014). Zhang et al. (2014) presented a tool intended to perform Search Engine Optimization (Madera et al., 2014) by inflating a website's PageRank.

The idea is to use the promoted page's content to create bogus links to an advertised website. Paraphrasing tools serve to alter the content, such that search engines do not recognize the fraudulent websites as duplicates and consider them to calculate the PageRank of the promoted site.

In academia, paraphrasing tools help to mask plagiarism, facilitate collusion, and help ghostwriters with producing work that appears original. Paraphrasing tools severely threaten text-matching software effectiveness, which is a crucial support tool for ensuring academic integrity. The academic integrity community calls for technical solutions to identify the machine-paraphrased text as one measure to counteract paraphrasing tools (Rogerson and McCarthy, 2017). The International Journal for Educational Integrity recently devoted a special issue[6] to this topic.

To contribute to the solution for academic integrity, we devise an automated approach that reliably distinguishes human-written from machine-paraphrased text and provide the solution as a free and open-source web application. A recent short paper (Foltýnek et al., 2020b) reports on our fellow researchers' preliminary experiments using one paraphrasing tool and a Wikipedia corpus. In this thesis, we analyze two new collections created from research papers on arXiv[7] and graduation Theses of *English as a second language* students. Additionally, we explore a second paraphrasing tool for generating obfuscated samples and eight neural language models based on the transformer architecture in all datasets. The recent neural language models surpass all preliminary ML techniques in every test set. Compared with two plagiarism detections systems, Turnitin[8], which has the largest market share, and PlagScan[9] which is one of the best-performing systems (Foltýnek et al., 2020a), neural language models detect short paragraphs of machine-paraphrased text with higher confidence.

[6] https://edintegrity.biomedcentral.com/mbp
[7] https://arxiv.org
[8] https://www.turnitin.com/
[9] https://www.plagscan.com/

Chapter 3

Methodology

The main objective of this thesis is to explore KD using different knowledge sources. This chapter presents a method to distill knowledge from large-scale LMs (Section 3.1) into an efficient model and two methods to distill knowledge from lexical databases (Section 3.2). We introduce a general framework to perform WSD with a variety of LMs relying on the transformer architecture and show how their improved semantic understanding can be applied to other NLU tasks (Section 3.2). Finally, we present two new data collections to evaluate MPP detection and explore two paraphrasing tools for generating obfuscated samples. We extend the application of ML techniques using word embeddings and propose a system to detect MPP with transformer LMs (Section 3.3).

3.1 Knowledge Distillation with Multiple Language Models

KD mainly explores the logits and features of a single LM with varying student model sizes (see Section 2.2.2). This work aims to combine the individual knowledge of multiple teacher models with a new prediction weighting approach. The goal of our KD technique is to provide a method for training a small model that incorporates the knowledge of multiple large models and thus, can be used by the community as a replacement for its expensive counterparts. We provide an overview of our method in Figure 3.1.

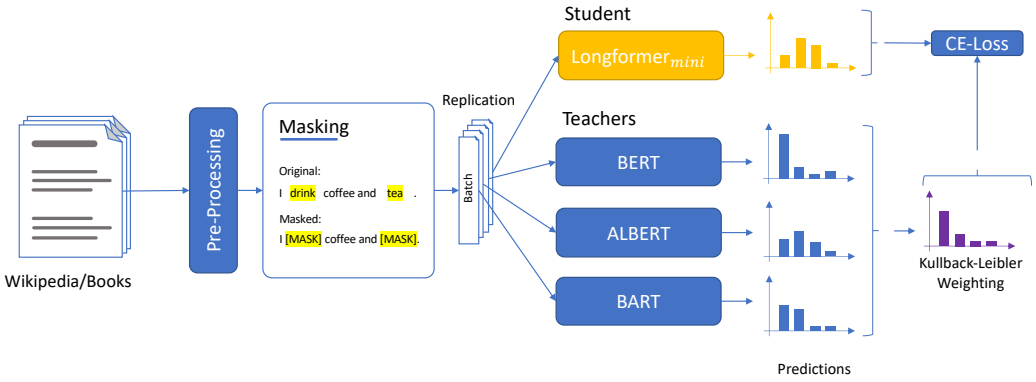

FIGURE 3.1: Overview of the multi-teacher KD method.

3.1.1 Student Architecture and Initialization

Multi-teacher KD requires the student model's training to be compatible with many teachers' training approaches. LMs often use pre-training objectives similar to MLM (see Table 2.1) and output a probability distribution over a standardized sub-word vocabulary for each corrupted token. Furthermore, the student model is smaller than its teachers and needs to use its parameters efficiently.

We initialize the student with the $base$ configuration of Longformer (Beltagy et al., 2020) as the training architecture is compatible with many other transformers (e.g., BERT, ALBERT, RoBERTa), while using an efficient method to calculate attention. Furthermore, Longformer performed superior over other architectures in our related experiments (Wahle et al., 2022b),.

To control the student's trainable parameters, we reduce the number of layers to 6 and retain all remaining configurations of the $base$ model. Following Sanh et al. (2019), we decrease layer count rather than model width or attention head size to initialize pre-trained layers from the $base$ model. We keep the prediction layer of Longformer$_{base}$ and remove every second layer towards the network's input while keeping all weight matrices. We name the resulting model Longformer$_{mini}$.

3.1.2 Distillation Method

The student network gains knowledge about the target language in two ways. First, it mimics the soft probability distribution of masked tokens using a custom weighted

average. Second, we use the last high-level hidden features of each teacher as studies (Wang et al., 2020; Jiao et al., 2020) proved it to be superior over single logit feedbacks.

We propose a logit-based supervision signal weighting the probability of each teacher network by confidence. Teachers with a high probability for the correct class receive a larger proportion in the averaged target distribution, while low probabilities lead to a smaller proportion. We use the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) to compare the classification distribution vector of the i-th teacher model $\hat{y}_i$ and the groud truth label vector y to weight confident and unconfident predictors with high and low proportions in the target distribution respectively (see Equation (3.1)).

$$y_{target} = \sum_{i=0}^{n} KL(\hat{y}_i||y)\hat{y}_i \tag{3.1}$$

$$= \sum_{i=1}^{n} \hat{y}_i \sum_{j=1}^{m} \log\left(\frac{\hat{y}_{i,j}}{y_j}\right) \tag{3.2}$$

The student uses the weighted prediction distribution y_{target} together with regular introductions of the ground truth label y as targets for the students LM objective (Hinton et al., 2015). The model learns features of one layer prior to the prediction using the Mean Squared Error (MSE) loss (Sammut and Webb, 2010). Algorithm 1 details the procedure, where the *forward* function obtains the predictions of a model with given inputs, and *maskfunction* replaces words with mask tokens at a certain probability (see Section 3.1.4 for hyperparameter details).

As our modified version of the Longformer model is smaller than large-scale LM teacher candidates and uses a local windowed and global attention scheme that is different from self-attention, we add no supervision signal for attention layers. Although Wang et al. (2020) showed their feature approximation method overcomes changes in model size by using a teacher assistant, the feature approximator for our method must not only learn a function to map multiple attention layers into a single one but also map self-attention to local and global attention. We presume the approximated target function of attention scores would include larger amounts of noise than distilling the same attention structure, impacting the training convergence.

Algorithm 1: Multi-Teacher KD algorithm for LMs

Data: Training Data $D \sim \mathcal{X} \times \mathcal{Y}, \mathcal{X} \subset \mathbb{N}$ represents the input words as indexes of the vocabulary V. $\mathcal{Y} \subset \mathbb{N}$ represents the target masked words as indexes of the vocabulary V.
Input: Teachers $T = (t_1, ...t_n)$. Student s. Ground truth update step l.
for $(x, y) \in D$; $k = 1$ **to** $|D|$ **do**
  $\widetilde{x} \leftarrow maskfunction(x)$
  $\hat{y}_{student} = forward(s, \widetilde{x})$
  for $t_i \in T$ **do**
    $\hat{y}_i \leftarrow forward(t_i, \widetilde{x})$
    $y_{target} \leftarrow y_{target} + \hat{y}_i \sum_{i=0}^{m} \log\left(\frac{\hat{y}_{i,j}}{y_j}\right)$
  if $k \mod l = 0$ **then**
    $l \leftarrow$CE$(\hat{y}_{student}, y)$
  else
    $l \leftarrow$CE$(\hat{y}_{student}, y_{target})$ + MSE$(\hat{h}_{student}, h_{target})$
  backward(l)
  optimizer.step()

Logit-based and feature-based knowledge in the final layers are not bound to the attention scheme or model size used and do not require a feature approximator. Furthermore, our presented method appears more accessible, as the interpretation of prediction probability distributions is intuitive.

3.1.3 Training and Testing Datasets

Transformer models typically perform training using a combination of Wikipedia articles and books as Table 2.1 emphasizes. Although recent studies show an increase in dataset size (from ≈10 times (Liu et al., 2019c) up to ≈80 times (Raffel et al., 2020)) and model size improves performance, we consider the small gain on average as ineffective. Instead, we construct a compact dataset composed of the official Wikipedia dump from October 2020[1] and the Books Corpus (Zhu et al., 2015) as a cheaper alternative covering long and short documents from different domains. The English Wikipedia contains ≈6.2m articles with ≈3.7 billion words and an average of about 600 words per article. The Books Corpus contains ≈11k books with ≈1b words, with ≈90k words per book on average. We pre-process all documents with the official BERT tokenizer ync split 20% of documents from Wikipedia and books at random as the validation dataset. We provide more details about the full benchmark in Table 3.1.

[1] https://en.wikipedia.org/wiki/Wikipedia:Database_download

TABLE 3.1: Overview of the multi-teacher KD training corpus.

Dataset	Documents	Words	Words per Doc.	Vocab.	Mean Words per Sent.
Wikipedia (Oct. 2020)	6.16m	3.69b	599	550.01k	11
Books Corpus (2015)	11.04k	984.85m	89.22k	1.31m	13

3.1.4 Setup

Our rationale for choosing teachers was to explore models that cover different training architectures but are compatible with our students' pre-training objectives. Student compatibility requires the teacher to a output probability distribution for corrupted tokens and to use the same vocabulary. We choose BERT (Devlin et al., 2019) as a well-researched baseline model, ALBERT (Lan et al., 2019) because of its increased training capacity using shared parameters, and BART (Lewis et al., 2019) due to its different denoising training approach combined with AR language modeling. Furthermore, the three teacher models trained on similar datasets to ours (see Table 2.1), which we assume increases the probability of creating meaningful predictions than models exploring other domains, not including Wikipedia or Books.

We applied best practices for model pre-training proposed by Liu et al. (2019c), using a large batch size of 256 examples with gradient accumulation and dynamic masking. The remaining configuration details are as follows: a sequence length of 512 tokens, the AdamW optimizer with a learning rate of $2 \times 10^{-5}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e-8$, the temperature $T = 2.5$, a masking probability of 0.15, a ground truth update step of $l = 100$, and PyTorch's native automated mixed-precision format. The experiments use 8 NVIDIA Tesla V100 Graphical Processing Unit (GPU)s with 16GB memory per card.

3.1.5 Implementation Details

Emphasizing the problem of model size, a single base-sized LM (e.g., BERT$_{base}$, $L = 12, H = 768, A = 12$) together with the student model barely fits into the GPU memory of a high-end NVIDIA Tesla V100 GPU with 16GB memory and mixed half-precision. When considering multiple teacher models and potentially larger models for future experiments, we need to construct a synchronous parallel setup for training in two ways.

Model Parallelism

Instead of processing teacher and student models on the same GPU, our method dedicates a single processing unit to each model. At each training step, we replicate the input sample once per unit, and each model processes a forward pass. We gather the final predictions in a shared memory pool and process the loss calculation, backward pass, and update rule on the student's side as it uses less memory and computation resources due to its optimized attention scheme and small model size.

Data Parallelism

To further scale training to multiple GPUs yielding larger batch sizes with improved training time and convergence (Liu et al., 2019c), we parallelize the model set up by replicating each model on a second GPU and average their gradients in each step. To decrease training time, we pinned the unallocated memory of each GPU and loaded new training samples asynchronously while the GPU processed the current sample.

3.2 Incorporating Lexical Knowledge into language Models

Humans use many explicit sources to gain knowledge about concepts, e.g., a lexicon describes different natural phenomena in an organized way. Knowledge-based methods for WSD use semantic information to determine word senses. In the following, we present a method using the knowledge about word senses in LKB to improve the semantic understanding of LMs. Our method addresses research Task 2, increasing semantic understanding and boosting task performance (see Section 4.2). We provide an overview of the method using the lexical database WordNet (Miller, 1995; Fellbaum, 1998a) and the annotated SemCor dataset (Miller et al., 1993) in Figure 3.2.

Current state-of-the-art methods in WSD focus primarily on the WSD domain without using the knowledge about ambiguous words for other NLP tasks (see Section 2.3.1). We suggest incorporating two objectives into the training of WSD to maintain LM capabilities while learning the disambiguation of words. Thus, the model obtains superior representations that benefit other tasks and domains as most benchmarks include a high fraction of polysemous words (see Appendix A).

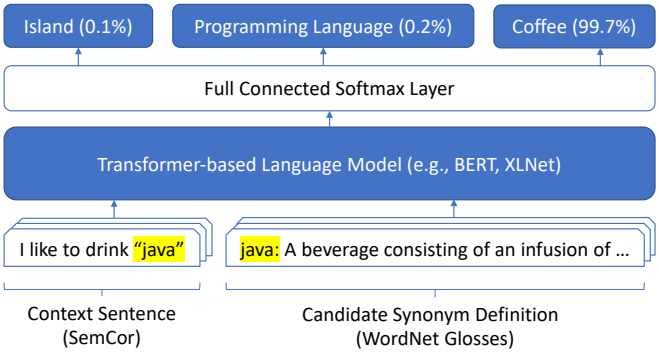

FIGURE 3.2: Overview of the WSD method.

Our literature analysis showed related works in the domain of WSD using neural LMs are typically based on BERT (see Section 2.3.1). BERT is a strong baseline, but recent studies show the model has not reached its full capacity; its training scheme still offers opportunities for improvement (Liu et al., 2019c). We introduce a method to perform WSD with arbitrary LMs and explore architectural changes to increase our model's performance (Section 3.2.2).

Furthermore, related WSD methods (Huang et al., 2019) use a sequential binary prediction head which requires n forward passes with the model for one ambiguous word (with n candidate word senses). With models containing multiple hundred million parameters, sequential processing becomes a bottleneck. We form the prediction step from sequential binary classification to parallel multi-classification to construct a more natural prediction head as previous literature showed (Kågebäck and Salomonsson, 2016).

3.2.1 Training and Testing Datasets

We use the SemCor 3.0 (Miller et al., 1993) dataset as the training corpus for all WSD experiments. SemCor 3.0 is one of the largest manually annotated datasets with approximately 226k word sense annotations from WordNet (Miller, 1995) for all open-class parts-of-speech. Each sentence in SemCor has multiple word annotations, and each word annotation consists of one or more sense annotations. The SemCor 3.0 corpus is well studied in the WSD literature (Huang et al., 2019; Peters et al., 2019). We validated LMs trained on SemCor 3.0 withRaganato et al. (2017)'s evaluation framework, a set of five standardized test sets: Senseval-2 (Edmonds and

Cotton, 2001), Senseval-3 (Snyder and Palmer, 2004), SemEval-2007 (Strapparava and Mihalcea, 2007), SemEval-2013 (Navigli et al., 2013), and SemEval-2015 (Moro and Navigli, 2015). We provide a detailed overview of these datasets in Table 3.2.

TABLE 3.2: SemCor training corpus details: general statistics (left) and class distribution for binary classification of word senses (right).

Dataset	POS Tags					Class dist.	
	Noun	Verb	Adj.	Adv.	Total	Pos.	Neg.
SemCor	87k	88.3k	31.7k	18.9k	226k	226.5k	1.79m
SE2	1k	517	445	254	2.3k	2.4k	14.2k
SE3	900	588	350	12	1.8k	1.8k	15.3k
SE7	159	296	0	0	455	459	4.5k
SE13	1.6k	0	0	0	1.6k	1.6k	9.7k
SE15	531	251	160	80	1k	1.2k	6.5k

To validate our proposed methods achieve high performance in WSD while using the acquired knowledge for other NLP tasks, we used the General Language Understanding Evaluation (GLUE) benchmark. GLUE (Wang et al., 2019b) is a collection of eight language understanding tasks widely used in the language modeling domain (Devlin et al., 2019; Lan et al., 2019; Liu et al., 2019c) to validate transfer learning capabilities of language models. All GLUE tasks are single sentence or sentence pair classification, except STS-B, which is a regression task. All classification tasks are binary classification except for MNLI, which has three classes.

3.2.2 Language Model Gloss Classifiction (LMGC)

With Language Model Gloss Classificaiton (LMGC), we propose a model-independent end-to-end WSD approach to classifying ambiguous words from sentences into one of WordNet's glosses. This approach enables applying different LMs for WSD. LMGC addresses a problem resulting from imbalanced examples by using the focal loss function (Lin et al., 2017), a state-of-the-art method to avoid accumulated gradients from negative examples. By choosing the most suitable model for WSD from eight different LMs, we show superior performance over BERT (Section 3.2.3).

In the LMGC task we pair (i) a sentence containing a polysemous target word with (ii) a prospective gloss definition from a lexical database (e.g., WordNet) for

this word. Let $W = (w_1, \ldots, w_n)$ be a word sequence of a sentence, where w_i is a polysemous word. WordNet provides $S_i = \{s_1, \ldots, s_m\}$ possible senses for word w_i with corresponding glosses $G_i = \{g_1, \ldots, g_m\}$, which are also individual sentences.

We build an annotated corpus using WordNet glosses to perform LMGC. For a sentence W, each polysemous word w_i points to correct WordNet senses $\widetilde{S}_i \subset S_i$. We retrieve the gloss candidates G_i for each synset of the word to create $|G_i|$ pairs of the sentence W and glosses g_j, with $g_j \in G_i$. If g_j corresponds to the desired meaning of our polysemous target word w_i (which is the case if $s_j \in \widetilde{S}_i$), we classify the pair as a positive example, otherwise as negative. As the number of correct senses for a polysemous word is much smaller than the total number of senses, labels are often imbalanced (see Section 3.2.1).

The input sequence used in LMGC follows the same configuration as in its underlying transformer. Each input sequence starts with the aggregate token (e.g., `[CLS]` for BERT), followed by a sentence and a gloss definition. The sentence and gloss are separated with a unique separator token and tokenized using WordPiece (Schuster and Nakajima, 2012).

To perform the classification, Du et al. (2019) used the token embeddings of polysemous words. However, Huang et al. (2019) showed the classification on top of the aggregate representation improves the results in WSD. Therefore, we obtain the final hidden representation of the aggregate token, namely $C \in \mathbb{R}^H$; where H is the embedding size. We apply a weight matrix $W_{LMGC} \in \mathbb{R}^{H \times 2}$ transforming C (together with a bias $B_{LMGC} \in \mathbb{R}^2$) into a binary space and calculate the probability to whether the gloss is appropriate for the ambiguity in context as Equation (3.3) shows.

$$p = \text{softmax}(C\, W_{BGP}^T + B_{BGP}) \tag{3.3}$$

To optimize the model for learning whether a gloss is correct for the annotated ambiguous word, Huang et al. (2019) and Du et al. (2019) used the standard cross-entropy loss function (Equation (3.4)).

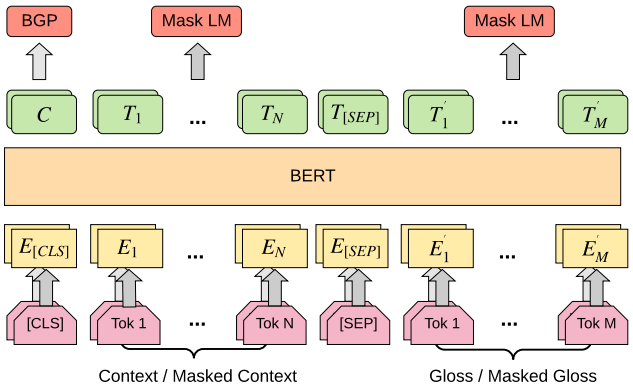

FIGURE 3.3: Language Model Gloss Classificaiton with MLM (LMGC-M) forwards context-gloss pairs and its masked versions through a LM (e.g., BERT) to perform a weakly supervised binary classification LMGC and MLM.

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if y} = 1 \\ 1 - \log(1-p) & \text{otherwise} \end{cases} \tag{3.4}$$

The class imbalance resulting from correct word senses in relation to candidates hinder the cross-entropy loss during training (see Table 3.2). Easily classified negatives comprise parts of the loss, which obstructs the gradient direction. Thus, we propose to apply a weighted loss function, a popular approach to mitigate the class imbalance in object detection from camera images (Ren et al., 2016; Redmon et al., 2016). The focal loss (Lin et al., 2017) (Equation (3.5)) reshapes the cross-entropy loss function, giving less weight to easy examples and forces the training to focus on challenging polysemous words.

$$\mathrm{FL}(p, y) = \begin{cases} -(1-p)^{\gamma} \log(p) & \text{if y} = 1 \\ -(1+p)^{\gamma} \log(1-p) & \text{otherwise} \end{cases} \tag{3.5}$$

We add additional supervision to the input sequence according to Huang et al. (2019) with two signals: (1) highlighting the ambiguous tokens with two special tokens and (2) adding the polysemous word before the gloss.

3.2.3 Language Model Gloss Classification with MLM (LMGC-M)

Pre-trained models serve as a valuable feature extractor that can be optimized to new task-specific objectives in a straight-forward manner (Radford et al., 2019; Devlin et al., 2019). However, optimizing a specific task disregard previously learned LM capabilities (see Section 4.2). The WSD task is special, as we can assume when a LM is capable of disambiguation, its ability to perform well on other tasks may increase. For this reason, we aim to keep language modeling features while learning WSD. Transfer learning between language modeling and WSD increases the likelihood of understanding polysemous words in other related tasks, improving the overall performance. In Appendix A, we discuss the correlation between WSD and other NLU tasks concerning the number of polysemous words.

The task-specific objective of LMGC is to predict whether the appended gloss explains the correct meaning of a word in a sentence. The corresponding objective function is the focal loss function between the network's prediction and the ground truth label as Equation (3.5) shows. In LMGC-M, we perform a parallel forward pass through transformer LM with similar sentence-gloss pairs. The first pass uses the original pair of a context sentence and gloss and applies a weight matrix to the final aggregate embedding resulting in binary classification. In the second pass, we replace words in the sentence and gloss with a mask token, a random token, or the same token with equal probabilities with the same configuration as Devlin et al. (2019). The LMGC-M procedure is exemplified in Figure 3.3.

We sum the two objectives, i.e. the task-specific WSD focal loss and the MLM cross-entropy loss, to obtain the final loss function L_1 described in Equation (3.6)

$$L_1(\hat{y}, y) = \mathsf{FL}(\hat{y}, y) + \sum_{k=1}^{m} \mathsf{CE}(\hat{y}_k^{(mask)}, y_k^{(mask)}) \tag{3.6}$$

where m is the number of masked words, $\widetilde{y}_j \in \{0,1\}^V$ is the one-hot encoding for a masked word w_j, and $\widetilde{p}_j \in \mathbb{R}^V$ is the probability vector for w_j.

3.2.4 Setup

We initialized all models using the base configuration of its underlying transformer (e.g., XLNet$_{base}$, $L = 12, H = 768, A = 12$). Both our methods have $2 * H + 2$ more parameters than their baseline (e.g., LMGC (BERT) has $\approx$ 110M parameters) but equal or fewer parameters compared to related methods (e.g., GlossBERT, KnowBERT W+W). For each polysemous word, we retrieved all possible gloss definitions from WordNet to create sentence-gloss inputs. We increased the hidden dropout probability to $p = 0.2$ as we observed overfitting for most models. FollowingDevlin et al. (2019), we used a batch size of 32 sequences, the AdamW optimizer ($\alpha = 2 \times 10^{-5}, \epsilon = 10^{-5}$), trained three epochs, and chose the best model according to validation loss. We applied the same hyperparameter configuration for all models used in both the SemCor and GLUE benchmarks.

To reduce training time to $\approx \frac{1}{3}$ compared to the approach of (Huang et al., 2019), we reduce the sequence length of all models from 512 to 160[2] as the computational cost of transformers grows quadratic with the sequence length. We treat the class imbalance of positive and negative examples (Table3.2) with focal loss (Lin et al., 2017) ($\gamma = 2$, $\alpha = 0.25$). Our experiments use one NVIDIA Tesla V100 GPU with 16GB memory for training and two GPUs for evaluation (NVIDIA Tesla V100 and NVIDIA TESLA T4) using the official scorer fromRaganato et al. (2017)[3] and Hugginface[4] considering the best results from each evaluation.

We create sentence-gloss inputs by retrieving WordNet's gloss definitions of polysemous words corresponding to the annotated synsets. For LMGC and LMGC-M (Section 3.2.2), we used 8 candidate glosses in the training procedure as a trade-off between memory requirements and model accuracy, as the mean number of synsets for polysemous words in SemCor is approximately 8.94 (Table 3.2). If the number of glosses was smaller than 8, we zero-padded the remaining sequences and disregarded them in the final softmax activation. If the number of glosses exceeded 8 and the word in question had one synset annotation, we randomly chose 9 glosses from the set of possible glosses, including the correct one. If the annotation marked multiple

[2] 99.8% of the dataset's examples contain less than 160 tokens; we truncate the remaining sequences to this limit.

[3] http://lcl.uniroma1.it/wsdeval/

[4] https://tinyurl.com/y4rdkjop

glosses as correct, we sampled multiple times from the set of possible glosses to include each sense once in our training procedure. At inference time, we extended 8 to the number of glosses of each example. LMGC and LMGC-M calculate the argmax along with the senses of the softmax activation from Equation (3.3) to predict the correct sense.

To validate that our model is capable of performing WSD, while still mastering other tasks, we fine-tuned them on GLUE. Therefore, we removed the weight matrices and bias introduced by the heads of each method (i.e., W_{LMGC} and W_{LMGC-M}) and represented the input sequence with the final hidden vector $C \in \mathbb{R}^H$ corresponding to the `[CLS]`-token as the aggregate representation. For all tasks, except for STS-B, we transformed the embedding into a classification vector applying a new weight matrix $W \in \mathbb{R}^{K \times H}$; where K is the number of labels. For STS-B, we applied a new weight matrix $V \in \mathbb{R}^{1 \times H}$ transforming the aggregate representation into a single regression value. We trained on each GLUE task for three epochs with 100 warm-up steps and the same remaining parameters, e.g., learning rate, optimizer, batch size as in SemCor.

3.3 Machine-Paraphrased Plagiarism Detection

The MPP detection methods presented in the following extend our fellow researchers previous study (Foltýnek et al., 2020b) with two new data sources (arXiv and Theses), an additional machine-paraphrase tool (SpinnerChief[5]) and eight state-of-the-art neural language models based on the transformers against our best-performing KD method. We show the general setup of our study in Figure 3.4.

We first perform preliminary experiments with machine learning approaches (e.g., Singular Value Decomposition (SVM), Logistic Regression (LR)) to identify the strongest baselines among the paraphrase tools and data sources. Next, we carry out the best performing techniques to be compared to the ones relying on the transformer architecture, which we believe represent the last advancements in NLP. We describe the paraphrasing tools, datasets, word embedding models, machine learning classifiers, and neural language models used in our experiments in the following sections.

[5] http://www.spinnerchief.com/

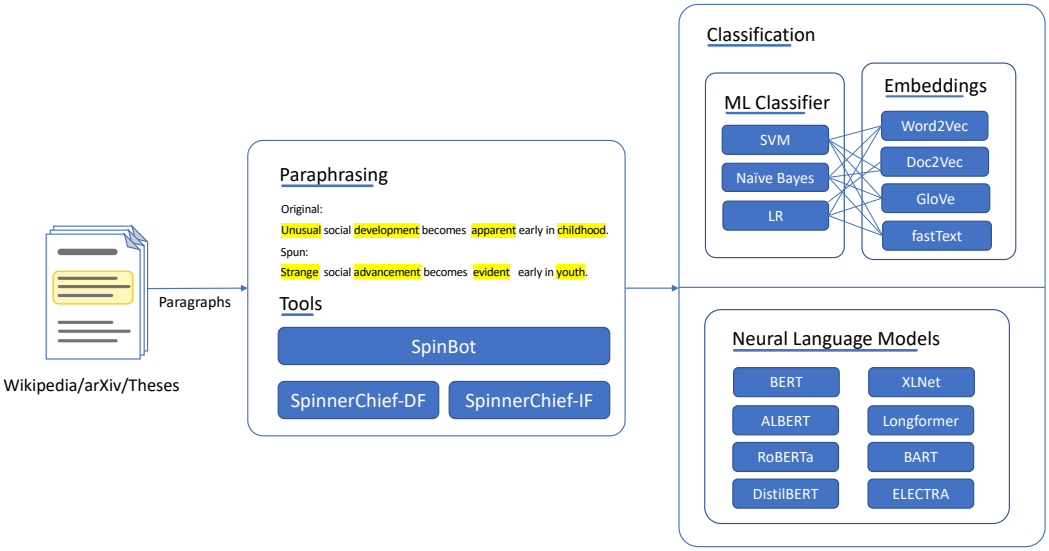

FIGURE 3.4: Overview of the MPP detection methods.

This comprehensive study aims to evaluate neural language models against the best-performing KD method (research Task 3) and to provide a free service that distinguishes human-written from the machine-paraphrased text while being insensitive to the topic, the type of documents, and the paraphrasing tool used. We analyze paragraphs instead of entire documents as it represents a more realistic detection task (Rogerson and McCarthy, 2017; Weber-Wulff, 2019).

3.3.1 Paraphrasing Tools

We employ two commercial paraphrasing services (SpinBot[6] and SpinnerChief[5]) to obfuscate samples in our training and test sets. We use SpinBot to generate the training and test sets, and SpinnerChief only generates test sets for two reasons. First, SpinnerChief is a paid service making paraphrasing on large datasets expensive. Second, we want to provide a realistic scenario where unseen test cases probably employed a different paraphrasing tool.

SpinnerChief allows specifying the ratio of words it tries to change. We experimented with two configurations: the default frequency - attempting to change every fourth word (SpinnerChief-DF) and increasing frequency - attempting to change every second word (SpinnerChief-IF). We found the average ratio of replaced words

[6] https://spinbot.com

is 20.38% for SpinBot, 12.58% for SpinnerChief-DF, and 19.37% for SpinnerChief-IF when analyzing the paraphrased samples. Thus, the actual frequency with which SpinnerChief replaces words is much lower than its settings suggest. The SpinBot API offers no options to influence the paraphrased text.

3.3.2 Training and Testing Datasets

Most paraphrasing tools are paid services, which prevents large-scale experimentation. The financial costs and effort required for obtaining and incorporating tool-specific training data would be immense. Therefore, we employed transfer learning, i.e., used pre-trained word embedding models, trained the classifiers in our study on samples paraphrased using SpinBot, and tested whether the classification approach can also identify SpinnerChief's paraphrased text.

We reused the paragraph training set of our initial study (Foltýnek et al., 2020b) as the training set. We paraphrased all 4,012 *featured articles* from the English Wikipedia using SpinBot. We chose featured Wikipedia articles because they objectively cover a wide range of topics in great breadth and depth[7]. Approximately 0.1% of all Wikipedia articles carry the label featured article. To receive this label, senior Wikipedia editors must confirm the superior quality of the article. Featured articles typically have many authors and revisions. Thus, they are written in high-quality English and unlikely to be biased towards the authors' writing style.

The training set comprises of ≈200.8k paragraphs (≈98.3k original, ≈102.5k paraphrased) extracted from ≈8k Wikipedia articles. We split each Wikipedia article into paragraphs and discarded those with fewer than three sentences, as our earlier findings showed that such paragraphs often represent titles or irrelevant information (Foltýnek et al., 2020b).

Our study uses three test sets that we created from preprints of research papers on arXiv, graduation Theses, and Wikipedia articles. Table 3.3 summarizes the test sets. For generating the arXiv test set, we randomly selected 944 documents from the *no problems* category of the arXiv project[8]. The Wikipedia test set is identical to the one in our preliminary study (Foltýnek et al., 2020b). The paragraphs in the test

[7] https://en.wikipedia.org/wiki/Wikipedia:Content_assessment
[8] https://kwarc.info/projects/arXMLiv/

set were generated analogously to the training set. The Theses test set comprises the paragraphs in 50 randomly selected graduation Theses of *English as a second language* students at the Mendel University in Brno, Czech Republic. The Theses are from a wide range of disciplines, e.g., economics, agronomy, forestry, and computer science, and cover all academic levels (i.e., B.Sc., M.Sc., and Ph.D.). Unlike the arXiv and Wikipedia documents, the Theses were only available as PDF files, thus having to be converted to plain text. We removed all content before the introduction section of each thesis, the bibliography, and all appendices because these parts of the PDF tended to produce content without valuable semantic information.

TABLE 3.3: Overview of the MPP test sets.

No. of paragraphs	**arXiv**		**Theses**		**Wikipedia**	
	Original	Paraphrased	Original	Paraphrased	Original	Paraphrased
SpinBot	20.97k	20.87k	5.23k	3.46k	39.26k	40.73k
SpinnerChief-DF	20.97k	21.72k	2.38k	2.94k	39.26k	39.7k
SpinnerChief-IF	20.97k	21.67k	2.38k	2.94k	39.26k	39.62k

3.3.3 Word Embedding Models

Table 3.4 summarizes the word embedding models used in our experiments: Global Vectors (GloVe)[9] (Pennington et al., 2014), W2V[10] (Mikolov et al., 2013b), fastText (FT)[11](FT-rw and FT-sw) (Bojanowski et al., 2017), and Paragraph Vector Model (D2V) (Le and Mikolov, 2014) which was trained from scratch. Following the hyperparameter recommendations of Lau and Baldwin (2016) for general-purpose applications, D2V uses uses a distributed bag-of-words training, a window size of 15 words, a minimum count of five words, trained word-vectors in skip-gram fashion, averaged word vectors, and 30 epochs. All word embedding models have 300 dimensions. Parameters we do not explicitly mention correspond to the default values in the gensim API[12].

Our rationale for choosing the pre-trained word embedding models was to explore the most prominent techniques regarding their suitability for the plagiarism detection

[9] https://nlp.stanford.edu/projects/glove/
[10] https://code.google.com/archive/p/word2vec/
[11] https://fasttext.cc/docs/en/english-vectors.html
[12] https://radimrehurek.com/gensim/models/doc2vec.html

task. GloVe (Pennington et al., 2014) builds a co-occurrence matrix of the words in a corpus and explores the ratio of the word probabilities in a text to derive its semantic vectors as a count-based model. W2V (Mikolov et al., 2013b) training tries to predict either a word given its context (cbow) or the context given the word (skip-gram).

Even though GloVe and W2V are routinely applied for numerous NLP tasks (Conneau et al., 2017; Ruas et al., 2019, 2020), they do not consider two important linguistic characteristics: word ordering and sub-wording. To explore these characteristics, we also included FT (Bojanowski et al., 2017) and D2V (Le and Mikolov, 2014). FT builds its word representation by extending the skip-gram model with the sum of the n-grams of its constituent sub-word vectors. For the D2V model, two training options exist – named distributed memory (pv-dm) and distributed bag-of-words (pv-dbow). The former is akin to W2V's cbow, while the latter is related to skip-gram. Both options introduce a new paragraph-id vector that is updated for each context window on every timestamp. The paragraph-id vector seeks to capture the semantics of the embedded object. We chose a pv-dbow over pv-dm model because of its results in semantic similarity tasks (Lau and Baldwin, 2016).

TABLE 3.4: Word embedding models for the MPP detection experiments.

Algorithm	**Main Characteristics**	**Training Corpus**
GloVe	Word-word co-occurrence matrix	Wikipedia Dump 2014 + Gigaword 5
W2V	Continuous Bag-of-Words	Google News
pv-dbow	Distributed Bag-of-Words	Wikipedia Dump 2010
fastText-rw	Skip-gram without sub-words	Wikipedia Dump 2017 + UMBC
fastText-sw	Skip-gram with sub-words	Wikipedia Dump 2017 + UMBC

In our experiments, we represent each text as the average of its constituent word vectors by applying the word embedding models in Table 3.4. All models, except for D2V, yield a vector representation for each word. In D2V, the embedded tokens represent the entire text. Thus, a match between a text not part of the external training corpus and the pre-trained D2V model is unlikely. Inferring the vector representations for unseen texts requires an additional training step with specific parameter tuning. For all texts in our test sets, we performed this extra step using the following hyper-parameters for the gensim API: $\alpha = 10^{-4}, \alpha_{min} = 10^{-6}$, and 300 epochs (Lau and

Baldwin, 2016). The resulting pv-dbow embedding model requires at least 7 GB of memory, compared to 1-3 GB required for other word embedding models. The higher memory consumption of pv-dbow can make it unsuitable for some use cases.

3.3.4 Machine Learning Classifiers

After applying the pre-trained models to our training and test sets, we forward the embeddings to three machine learning classifiers: LR, SVM, and Naïve Bayes (NB). We use multiple classifiers to explore the stability of the word embedding models concerning each classifier's characteristics. We employ a grid-search approach for finding the optimal parameter values for each classifier asSection 3.3.4 shows.

TABLE 3.5: Grid-search parameter for ML classifiers considering the scikit-learn[13]package in Python.

Classifier	Parameter	Range
Logistic Regression	solver	newton-cg, lbfgs, sag, saga
	maximum iteration	500, 1000, 1500
	multi-class	ovr, multinomial
	tolerance	0.01, 0.001, 0.0001, 0.00001
Support Vector Machine	kernel	linear, radial bases function, polynomial
	gamma	0.01, 0.001, 0.0001, 0.0001
	polynomial degree	1, 2, 3, 4, 5, 6, 7, 8, 9
	C	1, 10, 100

3.3.5 Neural Language Models

We use the following neural language models based on the Transfomer architecture in our experiments: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019a), ALBERT (Lan et al., 2019), DistilBERT (Sanh et al., 2019), ELECTRA (Clark et al., 2020), BART (Lewis et al., 2019), XLNet (Yang et al., 2019), and Longformer (Beltagy et al., 2020). Our rationale for choosing these models was two-fold. First, we explore models closely related or based on BERT, either by improving it through additional training time and data (RoBERTa) or compressing the architecture with minimal performance loss (DistilBERT, ALBERT). Second, we used contrasting models to BERT, that although

[13] https://scikit-learn.org

relying on the transformer architecture, significantly change the training objective (XLNet), the underlying attention mechanism (Longformer), or employ a discriminative learning approach (ELECTRA, BART).

To distinguish between a paraphrased and human-written paragraph, a randomly initialized linear layer on top of the model's embedding of the `[CLS]`-token performs binary classification with cross-entropy loss. For all models, we use the base version and the official pre-trained weights with the following configurations: a sequence length of 512 tokens, an accumulated batch size of 32, the Adam optimizer with a learning rate of $\alpha = 2 \times 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and PyTorch's native automated mixed-precision format. Using a common sequence length of 512 tokens allows for a fair comparison between models without losing important context information[14]. In Section 4.3.1 we provide more details about these models characteristics.

[14] 99.35% of the dataset's text can be described entirely with less than 512 tokens.

Chapter 4

Evaluation

This chapter evaluates our presented KD methods in the tasks of language modeling, WSD, NLU, and PD with a total of 20 tasks. First, we provide evidence that our weighted multi-teacher KD reduces pre-training time and obtains superior validation performance over large-scale teachers when the dataset size and computational budget remains equal between the models (Section 4.1).

Next, we evaluate methods to incorporate lexical knowledge into LMs on five WSD benchmarks, and eight NLU tasks (Section 4.2). We show LMGC obtains superior performance in the WSD task over related work when using XLNet as the backbone network while keeping the same number of parameters. When transferred to NLU tasks, LMGC-M performs superior on average over BERT, validating its improved semantic representations. The code[1], and pre-trained models[2] to reconstruct our experiments are publicly available.

Finally, we apply our lexical KD models to the task of MPP detection. We compare our method to eight neural LMs and three ML techniques using word-embeddings on encyclopedia articles, student Theses, and research papers (Section 4.3). We show ML methods are superior over a human baseline and accurately predict plagiarism when the data source and paraphrasing tool changes. Compared to two popular PD systems, neural language models detect plagiarism on new sources more accurately, which makes them a suitable extension to PD systems. We release the data[3], code[4], and pre-trained models[2] of our study, as well as a web-based demonstration system[5].

[1] https://github.com/jpwahle/word-sense-disambiguation
[2] https://huggingface.co/models?search=jpwahle
[3] https://doi.org/10.5281/zenodo.3608000
[4] https://https://github.com/jpwahle/iconf22-paraphrase
[5] http://purl.org/spindetector

4.1 Language Modeling

When evaluating a LMs capability of modeling the probability of words, Perplexity (PPL) (Jelinek et al., 1977) is a widely used measure (Jurafsky and Martin, 2009, Chapter 3). PPL is the inverse probability normalized by the number of words in the test corpus as Equation (4.1) shows.

$$PPL(w_1, ..., w_n) = P(w_1, ..., w_n)^{\frac{1}{n}} \tag{4.1}$$

$$= \sqrt[n]{\frac{1}{\prod_{i=1}^{n} P(w_i \mid w_{<i})}} \tag{4.2}$$

To calculate PPL, the LM needs to assign conditional probabilities to consecutive tokens. As masked LMs predict the probability of tokens for corrupted sequences rather than succeeding tokens, the PPL is not well defined. However, when using the cross-entropy loss for MLM, Equation (4.3) shows that perplexity is proportional to the cross-entropy loss by an exponential function.

$$CE(w1, ..., w_n) = -\frac{1}{n} \log(P(w_1, ..., w_n)) \tag{4.3}$$

$$PPL(w_1, ..., w_n) = e^{CE(w_1, ..., w_n)} \tag{4.4}$$

We use cross-entropy to measure a relative gain in language modeling performance during training between the student and teacher models. Therefore, we use the validation loss — which is the cross-entropy for token predictions — to obtain a measure proportional to the inverse probability. This measure provides us with evidence about the language modeling capabilities of the model over the validation corpus.

Another robust way to compare LMs is the validation on general NLU tasks. We evaluate the methods proposed in Section 3.2 in the task of WSD and on a variety of NLU tasks. Due to the computational requirements of large-scale models, we leave their investigation for future work.

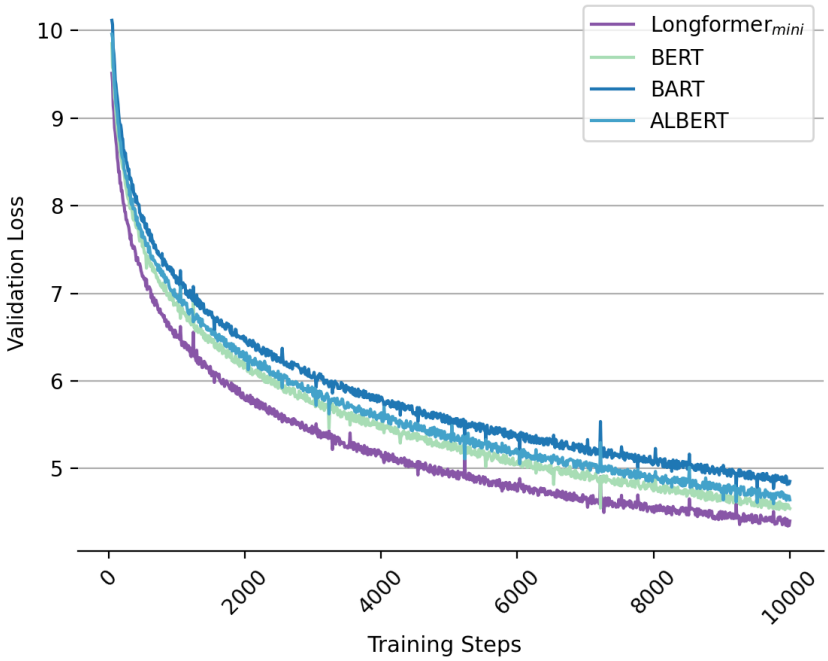

FIGURE 4.1: Validation loss by the number of training steps for student (Longformer$_{mini}$) and teacher (BERT, ALBERT, BART) models. The loss is a smoothed version of the accumulated cross-entropy of MLM.

4.1.1 Results and Discussion

Figure 4.1 reports the validation cross-entropy loss on the ground truth targets for the student model (Longformer$_{mini}$) and each teacher model (BERT, ALBERT, BART) over the number of training steps. Each training step equals 256×512 tokens. Each teacher model curve shows the validation loss for training with the official training procedure and randomly initialized weights.

Our first observation is that the student model Longformer$_{mini}$ initially converges faster than its larger counterparts. We hypothesize the initialization of layers with pre-trained ones increases the convergence rate compared to randomly initialized weights. Furthermore, the weighted target distribution includes highly confident predictions of the pre-trained teachers. As we use similar datasets to the ones used to train the teacher models, we assume confident predictions are also accurate with high probability.

The validation loss of all teachers behaves similarly, with BERT performing marginally better than ALBERT. BART's validation loss converges slower than the other models which we suppose results from BART being a combination of AE and AR models

resulting in more training time to converge to both objectives. We assume different levels of noise in the validation loss at different training steps per model appear due to randomly shuffling the training data leading to batches that sometimes do not represent the data distribution well.

At later training steps (starting at step ≈6000), the student model (Longformer$_{mini}$) is still slightly superior over its teachers, but its advantage in validation loss decreased, supporting our hypothesis about improvements by weight initializations. However, we can interpret the decrease in learning steepness in two ways. First, the student might already have captured a large proportion of the knowledge encoded in the teachers and reached its capacity. Considering the continuation of training, we expect to observe overfitting by a slow change in the curve's direction. Second, the advantage of Longformer$_{mini}$'s layer initialization does not impact the training convergence much in advanced training steps (step ≈6000) and its learning steepness converges to the teacher models.

4.1.2 Limitations

Although the Longformer$_{mini}$ reduced its validation loss faster than its teacher models, we must consider more experiments to conclude its converged model performs better in language modeling than its teachers. These extending experiments must include converging Longformer$_{mini}$ regarding its validation loss and testing the resulting model on NLU benchmarks against its teachers and other KD methods.

We aim to converge the model in future experiments and test its generalization on NLU benchmarks to provide an efficient replacement for large models to the community. In the following, we focus on more promising KD methods using lexical sources that are cheaper to train and prove high performance in WSD, NLU, and PD as the total computational budgets for this thesis are limited.

4.2 Word Sense Disambiguation and Natural Language Understanding

We evaluate our proposed lexical KD methods on two large scale benchmarks, namely SemCor 3.0 (Miller et al., 1993) and the General Language Understanding Evaluation (GLUE) (Wang et al., 2019b). Comparing LMGC and LMGC-M with state-of-the-art systems, we provide an analysis and discussions of experimental results and point out possible limitations of the presented methods.

4.2.1 Results & Discussion

Table 4.1 reports the results of applying LMGC to different transformer models. Our rationale for choosing the models was two-fold. First, we explore models closely related to or based on BERT. Either the chosen models improve BERT through additional training time and data (RoBERTa), or compress the architecture with minimal performance loss using parameter optimizations or KD (ALBERT, DistilBERT). Second, we chose models that significantly change the training objective (XLNet) or employ a discriminative learning approach (ELECTRA, BART). In Table 4.3, we compare our techniques to other contributions in WSD. We report all results of SemCor according to Raganato et al. (2017).

TABLE 4.1: SemCor test results (F1) of LMGC for base transformer models. **Bold** font indicates the best results.

System	SE7	SE2	SE3	SE13	SE15	All
BERT (2019)	71.9	77.8	74.6	76.5	79.7	76.6
RoBERTa (2019c)	69.2	77.5	73.8	77.2	79.7	76.3
DistilBERT (2019)	66.2	74.9	70.7	74.6	77.1	73.5
ALBERT (2019)	71.4	75.9	73.9	76.8	78.7	75.7
BART (2019)	67.2	77.6	73.1	77.5	79.7	76.1
XLNet (2019)	**72.5**	**78.5**	**75.6**	**79.1**	**80.1**	**77.2**
ELECTRA (2020)	62.0	71.5	67.0	73.9	76.0	70.9

RoBERTa shows inferior F1 than the baseline model BERT although it uses more data and training time. DistilBERT and ALBERT perform worse than BERT, which we expected given that they use significantly fewer parameters. However, ALBERT

achieves good results with only $\approx$ 10% of BERT's parameters. ELECTRA and BART results show the models' denoising approach is not suitable for our WSD setup. Besides, BART achieves similar performance as BERT but uses 26% more parameters. XLNet continually performs better than BERT on all evaluation sets while using almost the same number of parameters. Therefore we selected XLNet for our models' variation.

TABLE 4.2: Classification results (F1) of two model sized of BERT ($base$ and $large$) on the three largest SemCor evaluation sets.

System	SE2	SE13	All
BERT$_{base}$ 2019	77.78	76.52	76.59
BERT$_{large}$ 2019	78.26	76.46	76.67

We did not consider larger models than the base configuration as our experiments showed a difference of 0.08% in F1 between BERT$_{base}$ and BERT$_{large}$ (see Table 4.2) for the SemCor datasets, which is in line with Blevins and Zettlemoyer (2020)'s findings. Thus, we consider the base configuration as sufficient for our experiments.

Table 4.3 shows an overall improvement when comparing LMGC to related approaches. LMGC (BERT) generally outperforms Du et al. (2019)'s baseline approach (BERT$_{WSD}$) and KBERT-W+W, which has four times the number of parameters. We outperform GlossBERT in all test sets by using an optimal transformer (XLNet) and adjustments in the training procedure LMGC-M. We exclude EWISER (Bevilacqua and Navigli, 2020) which explores additional knowledge other than gloss definitions

TABLE 4.3: Classification results (F1) on the SemCor test sets compared to state-of-the-art techniques. **Bold** font indicates the best results.

System	SE7	SE2	SE3	SE13	SE15	All
CAN (2018)	-	72.2	70.2	69.1	72.2	70.9
HCAN (2018)	-	72.8	70.3	68.5	72.8	71.1
LMMS$_{BERT}$ (2019)	68.1	76.3	75.6	75.1	77.0	75.4
GLU (2019)	68.1	75.5	73.6	71.1	76.2	74.1
GlossBERT (2019)	72.5	77.7	75.2	76.1	80.4	77.0
BERT$_{WSD}$ (2019)	-	76.4	74.9	76.3	78.3	76.3
KBERT-W+W (2019)	-	-	-	-	-	75.1
LMGC (BERT)	71.9	77.8	74.6	76.5	79.7	76.6
LMGC-M (BERT)	72.9	78.2	75.5	76.3	79.5	77.0
LMGC XLNet	72.5	78.5	75.6	79.1	80.1	77.2
LMGC-M XLNet	**73.0**	**79.1**	**75.9**	**79.0**	**80.3**	**77.5**

(e.g., knowledge graphs). We leave for future work the investigation of BEM (Blevins and Zettlemoyer, 2020), a recently published bi-encoder with separate encoders for context and gloss.

TABLE 4.4: Test results, scored by the GLUE evaluation benchmark. As in BERT, we exclude the problematic WNLI set. We report F1-score for QQP and MRPC, Spearman Correlations (SC) for STS-B, Matthews Correlations (MC) for CoLA, and Accuracy (ACC) for the other tasks (with matched/mismatched accuracy for MNLI). **Bold** face indicates the best result per dataset.

System	Classification		Semantic Similarity			Natural Language Inference			Average
	CoLA (mc)	SST-2 (acc)	MRPC (F1)	STS-B (sc)	QQP (F1)	MNLI m/mm(acc)	QNLI (acc)	RTE (acc)	- -
BERT$_{base}$	52.1	93.5	**88.9**	85.8	71.2	84.6/83.4	**90.5**	66.4	79.6
GlossBERT	32.8	90.4	75.2	90.4	68.5	81.3/80	83.6	47.3	70.7
LMGC (BERT)	31.1	89.2	81.9	89.2	87.4	81.4/80.3	85.4	60.2	74.5
LMGC-M (BERT)	**55.0**	**94.2**	87.1	**88.1**	**90.8**	**85.3/84.2**	90.1	**69.7**	**82.5**

We evaluate our suspicion that WSD training allows language models to achieve higher generalization. Hence, we fine-tune the weights from our converged models in general language tasks from GLUE (Wang et al., 2019b). Table 4.4 shows the results of our proposed methods against the previously best-performing model in WSD (GlossBERT (Huang et al., 2019)), and the official BERT$_{base}$ model on the GLUE datasets. LMGC-M achieves a 2.9% increase in performance on average over BERT. Although LMGC and GlossBERT perform well in WSD, they cannot maintain good performance on other GLUE tasks. Still, LMGC performs better than GlossBERT, which we assume is due to its improved loss function regarding negative examples. LMGC-M outperforms BERT on most tasks and is certainly comparable to the others. Therefore, we conclude the incorporation of MLM adds value from WSD into natural language understanding. We exclude XLNet from the comparison to show that the additional performance is attributable to our method, not to the improvement of XLNet over BERT. In this work, we did not compare LMGC and LMGC-M to the other WSD methods performing worse than Huang et al. (2019) in the WSD task (Table 4.3) because reconstructing their models is computationally expensive.

4.2.2 Limitations

LMGC-M optimizes two objectives simultaneously, which is more expensive than LMGC. However, LMGC-M has the same number of parameters as LMGC during testing, resulting in approximately the same inference time.

Furthermore, we seek to investigate the impact of each component of our presented methods (e.g., weak supervision signals, parallel predictions, short sequence lengths) by extending the ablation. We leave this study to future work as it requires experimenting with the models with many of their variations.

4.3 Machine-Paraphrased Plagiarism Detection

We evaluate the effectiveness of the classification approaches in identifying machine-paraphrased text with three experiments. Section 4.3.1 presents the results of applying word embedding models with machine learning classifiers and neural language models to text from Wikipedia, arXiv, and Theses modified with two paraphrasing tools (SpinBot, SpinnerChief). Sections 4.3.2 and 4.3.3 establish two baselines for the results of the automated classification approaches by indicating how well human experts and two text-matching systems identify machine-paraphrased text, respectively.

4.3.1 Automated Classification

Tables 4.5 to 4.7 show the micro-averaged F1 scores (F1-Micro) for identifying paraphrased paragraphs using SpinBot and SpinnerChief (DF and IF), for machine learning and transformer-based techniques. We show the best performing combination of embedding and classifiers in the row of the test set name for the machine learning approaches. We use the top-ranked results (Tables 4.5 and 4.6) as a baseline against the transformer-based ones in Table 4.7.

Machine Learning Results for SpinBot

GloVe in combination with SVM achieved the best classification performance for all test sets (Table 4.5). The combination of W2V and SVM performed nearly as good as GloVe+SVM for all test sets. For Theses and Wikipedia, the performance difference

between GloVe+SVM and W2V+SVM was less than 2%, but for arXiv the difference was considerably higher with 6.66%. All word embedding models achieve their best results for Wikipedia test cases. This result corresponds to our expectation, since all pre-trained word embedding models, except for W2V, use Wikipedia dumps as part of their training corpora.

TABLE 4.5: Classification results (F1-Micro) for SpinBot. **Bold** face indicates the best score for each combination of dataset and classifier configuration. The highest scores for each classifier are repeated in the dataset name row.

	GloVe	W2V	D2V	FT-rw	FT-sw
arXiv	**86.46**	79.80	72.40	78.40	74.14
LR	76.53	74.82	69.42	75.08	65.92
SVM	86.46	79.80	72.40	76.31	74.15
NB	79.17	74.23	57.99	78.40	64.96
Theses	**83.51**	81.94	61.92	72.75	64.78
LR	68.55	72.89	59.97	69.17	64.03
SVM	83.51	81.94	61.92	72.75	64.78
NB	75.22	74.18	42.30	72.11	61.99
Wikipedia	**89.55**	87.27	83.04	86.15	82.57
LR	80.89	84.50	81.08	85.13	78.97
SVM	89.55	87.27	83.04	86.15	82.57
NB	69.68	69.84	58.88	70.05	64.47

While all classification approaches performed worse for Theses, the drop in performance was smaller than we expected. The F1-Micro score of the best approach for Theses (GloVe+SVM) is 6.04% lower than for Wikipedia and 3.09% lower than for arXiv. All embedding models, except for W2V, perform worse for Theses than for arXiv. This finding suggests writing quality in student Theses mildly affects the detection of machine-paraphrased text.

Although D2V seeks to mitigate the shortcomings of its predecessor W2V, such as word order and variable-length encoding, W2V surpassed D2V for all test sets. One reason for this observation can be the comparably short length of the paragraphs we consider. Lau and Baldwin (2016) found D2V's performance decreases for short documents. The results for paragraphs in Table 4.5 and for documents in our preliminary study (Foltýnek et al., 2020b), where D2V was the best-performing approach, agree with this finding.

When considering FT in Table 4.5, we observe the same behavior as for W2V and

D2V. In theory, the pre-trained FT-sw model is supposed to capture the nuances of the words inner components, i.e., sub-word embeddings. Therefore, we expected a better performance of FT-sw compared to FT-rw, which encodes whole words. However, FT-rw and simpler models, i.e., GloVe and W2V, performed better than FT-sw for all test sets. These results are interesting as both FT-rw and FT-sw have been trained using the same corpus.

Machine Learning Results for SpinnerChief

For test cases using SpinnerChief, we observed a decrease in the classification performance compared to classifying SpinBot test cases. We expected this decrease as SpinnerChief was not used to train these models. The average decrease in the F1-Micro scores was approximately 17% when using SpinnerChief's default setting of attempting to replace every fourth word (Table 4.6) and approximately 13% for increasing the frequency of attempted word replacements to every other word (Table 4.6).

TABLE 4.6: Classification results (F1-Micro) for SpinnerChief. DF - default frequency - attempting to change every fourth word IF - increased frequency - attempting to change every second word. **Bold** face indicates the best score for each combination of dataset, and classifier configuration. The highest scores for each classifier are repeated in the dataset name row.

	SpinnerChief-DF					**SpinnerChief-IF**				
	GloVe	W2V	D2V	FT-rw	FT-sw	GloVe	W2V	D2V	FT-rw	FT-sw
arXiv	58.48	**59.78**	56.46	57.42	59.72	64.34	**65.89**	59.27	63.70	63.66
LR	52.14	55.43	56.46	57.42	58.64	54.92	59.61	59.07	61.74	61.57
SVM	58.42	57.65	56.43	56.43	59.72	64.12	62.77	59.27	62.97	63.66
NB	58.48	59.78	51.58	51.58	55.21	64.34	65.89	52.21	63.70	59.33
Theses	52.63	53.60	**59.09**	53.08	57.25	58.57	58.24	**63.15**	59.13	61.27
LR	48.42	53.60	59.09	52.51	55.63	52.08	57.94	62.88	59.13	60.65
SVM	52.63	51.54	59.00	53.08	57.25	58.57	57.78	63.15	58.12	61.27
NB	50.90	53.32	54.94	52.78	46.99	55.62	58.24	55.09	57.19	50.13
Wikipedia	57.86	**60.30**	55.99	59.19	59.62	64.16	**66.83**	60.94	65.35	66.41
LR	52.97	55.90	55.64	56.40	59.62	55.68	61.32	60.16	62.51	66.41
SVM	57.09	57.48	55.99	57.15	58.72	64.16	64.56	60.94	63.61	64.81
NB	57.86	60.30	51.64	59.19	57.29	63.46	66.83	52.64	65.35	62.06

SpinnerChief's settings for a stronger obfuscation of text (SpinnerChief-IF) increased the success rate with which the automated classification approaches identified the paraphrases. On average, SpinnerChief-DF replaced 12.58% and SpinnerChief-IF 19.37% of the words in the text (see Section 3.3.1). The 6.79% increase in the number of replaced words for SpinnerChief-IF increased the average F1-Micro score of the classification approaches by 5.56%. This correlation suggests the classification approaches can recognize most of the characteristic word replacements of the studied paraphrasing tools.

The word choice and grammatical errors typical for texts of *English as a second language* students decreased the classification performance less than we had expected. The SpinBot Theses test set's highest scores were ≈6% lower than the highest scores for SpinBot overall. For SpinnerChief, the difference between the highest scores for Theses and overall was ≈2%.

Furthermore, text-matching software and our classification approaches have complementary strengths regarding the detection of machine-paraphrased plagiarism. Text-matching software, such as Turnitin, is currently the de-facto standard technical support tool for identifying plagiarism. However, since these tools search for identical text matches, their detection effectiveness decreases when the number of replaced words increases (Weber-Wulff, 2019). Including additional scans with the proposed classification approaches, as part of the detection process of text-matching software, could alleviate current systems' weaknesses.

Nevertheless, the F1-Micro scores for SpinnerChief-IF test cases were, on average, 13% lower for SpinBot cases, although these cases exhibit a similar ratio of replaced words (see Table 4.6). As for the SpinBot test cases, all approaches performed best for Wikipedia test cases and worst for Theses. However, the performance differences were smaller for the SpinnerChief test sets than for SpinBot test sets. For all SpinnerChief test sets, the lowest F1-Micro scores are at most 6.5% below the highest scores for the test set, and the runner-ups are generally within an interval of 2% of the best scores for the test sets.

Our transfer learning approach likely caused the drop in the classification performance and the overall leveling of the F1-Micro scores for SpinnerChief test cases.

TABLE 4.7: Classification results (F1-Micro) for SpinBot and SpinnerChief (DF and IF) using. The first block shows the best results from machine learning methods. The second block is composed of transformer-based optimization techniques, the third of new architectural or training approaches, and the fourth of our best-performing lexical KD method. The highest scores are in **Bold** face.

Techniques	SpinBot			SpinnerChief-DF			SpinnerChief-IF		
	arXiv	Theses	Wiki	arXiv	Theses	Wiki	arXiv	Theses	Wiki
Baseline	86.46[a]	83.51[a]	89.55[a]	59.78[b]	59.09[c]	60.30[b]	65.89[b]	63.15[d]	66.83[b]
BERT	99.44	94.72	99.85	50.74	50.42	43.00	64.59	63.59	57.45
ALBERT	98.91	96.77	99.54	66.88	47.92	50.43	75.57	56.75	59.61
DistilBERT	99.32	96.61	99.42	38.37	45.07	37.05	47.25	51.44	46.81
RoBERTa	99.05	97.34	99.85	57.10	47.40	48.03	66.00	58.24	58.94
ELECTRA	99.20	96.85	99.41	43.83	44.95	56.30	60.77	63.11	**75.92**
BART	99.58	99.66	99.86	69.38	53.39	48.62	76.07	63.57	58.34
XLNet	**99.65**	98.33	99.48	69.90	53.06	50.51	80.56	71.75	61.83
Longformer	99.38	**99.81**	99.87	**76.44**	**70.15**	**63.03**	78.34	74.82	67.11
LMGC-M (XLNet)	99.48	98.86	**99.90**	72.13	61.32	55.75	**82.32**	**75.23**	61.23

[a] GloVe+SVM [b] W2V+NB [c] D2V+LR [d] D2V+SVM

As explained in Section 3.3, we seek to provide a system with high generalization for different document collections and paraphrasing tools. Therefore, we used the machine-paraphrased text samples of SpinBot and applied the pre-trained word embedding models from Table 3.4 to extract the vector representations. We then used these vectors as features for the machine learning classifiers for both Spinbot and SpinnerChief test sets.

Considering the results combining machine learning techniques and word embedding features, for SpinBot (Table 4.5) and SpinnerChief (Table 4.6), we selected the top ranked ones to compose our *Baseline* in the Transformer experiments on Table 4.7.

Results for Transformer-based Architectures

The SpinBot test cases show all transformer models outperformed machine learning counterparts by significant margins, 16.1% for Theses, and 13.27% on average for all three data sources (Table 4.7). Our findings show all neural language models captured SpinBot's intrinsic paraphrasing method almost entirely. We stop the training for each model after one epoch to avoid overfitting to the paraphrasing method.

Longformer achieved the best performance on the SpinnerChief-DF test benchmark, increasing the F1-Micro score over machine learning baselines by up to 16.66% for arXiv pre-prints, and 10.15% on average for all three data sources. Longformer's

architecture is designed to capture short and long dependencies with two main attention mechanisms: a sliding window and global attention. We assume the model generalizes well even when we decrease the word frequency with which words change to capture long dependencies better. However, comparing the results of SpinnerChief-DF in Table 4.7, we see a substantial drop in the F1 score for all approaches in relation to SpinBot and SpinnerChief-IF. Our hypothesis is the ratio of changing words of the paraphrased tools strongly affects our models.

When we increase SpinnerChief paraphrasing frequency to the same one as in SpinBot, most architectures significantly improve baseline performance. This observation is consistent with the discussion and results from machine learning experiments (see Tables 4.5 and 4.6). XLNet, Longformer, and ELECTRA achieve the highest results with an improvement of 14.67%, 11.67%, and 9.09% in F1-score for arXiv, Theses, and Wikipedia, respectively. As ELECTRA was pre-trained using a Wikipedia Dump and the Books Corpus (Zhu et al., 2015), we assume it captured semantic aspects of Wikipedia articles as well. Additionally to Wikipedia, Longformer and XLNet also trained on other larger datasets, e.g., Gigaword 5 (Napoles et al., 2012), CC Stories (Trinh and Le, 2019), and Realnews (Zellers et al., 2019), which are seen more frequently during training. In return, Longformer and XLNet seem to capture unseen semantic structures better (e.g., arXiv and Theses) due to their high diversity in the training data.

When using XLNet as the backbone network for our best performing method (LMGC-M) in WSD and NLU, we observe slight performance gains on average over XLNet. In the SpinnerChief-IF test case, we surpass the highest results for arXiv and Theses texts by 1.13%. LMGC-M generalizes well on the unknown spinning pattern, which is in line with our previous experiments in NLU (Section 4.2). Considering the results of Longformer, we think it is a strong candidate for the backbone network in future versions of LMGC-M.

The high compression rate of DistilBERT with a two-fold reduction in the number of parameters showed a significant decrease in accuracy over BERT in all SpinnerChief test cases. Although we expected a slight decline in F1, the results exceeded our predictions. DistilBERT only performs 2.5% worse than BERT on the GLUE (Wang et al., 2019b) dataset, but it drops by 10.63% F1-Micro on average on the SpinnerChief test

cases, producing results comparable to random guessing. This result intrigued us how our converged Longformer$_{mini}$ model would perform in the task of PD which we aim to address in future work. ALBERT's parameter reduction techniques (e.g., factorized embedding parametrization, parameter sharing) seem to be more robust and outperform BERT with a 4.56% increase on average on SpinnerChief test cases.

RoBERTa performs slightly better than BERT, with an improvement of 0.99% on average. As RoBERTa uses more parameters than most other BERT-related models and has exceptionally high computational requirements for pre-training, we rate this advance as negligible. In most SpinnerChief test cases, the BERT-based approaches performed comparably to the machine learning baselines. Models that are strongly related to BERT are not performing as well on the paraphrase detection task as models that significantly change the attention scheme or training objective.

Given the evidence at hand, spun words' frequency is a strong indicator of our models' performance. However, since the spinning method of SpinBot and SpinnerChief (DF and IF) is unknown and could be different for each case, we can interpret the findings in two ways. First, the models may capture spinning frequency intrinsically and increase their attention to more words, which would mean the methods can better detect highly spun paragraphs. Second, the spinning method in SpinnerChief-IF can be akin to the one used in SpinBot.

At a higher level, the techniques proposing different training architectures performed better than those based on hyperparameter optimization from BERT. We believe the windowed local-global self-attention scheme used in Longformer allowed the model to generalize better between different paraphrasing tools. In 8 out of 9 scenarios, Longformer was either the best or second-best model overall. Also, for almost all cases, the neural language approaches surpassed the machine learning ones, providing an attractive solution for the paraphrased detection problem. Concluding, we can see the results on SpinnerChief-DF as a lower bound for unseen spinning methods, even if the frequency of word replacements is drastically changed.

4.3.2 Human Baseline

To complement our earlier study (Foltýnek et al., 2020b), we conducted a user survey with excerpts from ten randomly selected Wikipedia articles. We paraphrased three excerpts using SpinerChief-DF and SpinnerChief-IF and kept four excerpts unaltered.

Using QuizMaker[6], we prepared a web-based quiz showing the ten excerpts one at a time and asked the participants to vote whether the text had been machine-paraphrased. We shared the quiz via e-mail and a Facebook group with researchers from the academic integrity community. We obtained 32 responses. The accuracy of the participants ranged between 20% and 100%, with an average of 65.59%. Thus, the F1-Micro score of the average human examiner matched the average of the best scores for SpinnerChie-IF test sets (65.29%). Some participants pointed out some excerpts' irregularities, e.g., lowercase letters in acronyms, helped them identify the excerpts as paraphrased. For SpinBot, which we evaluated in our preliminary study, 73 participants answered the survey with an accuracy between 40% and 100% (avg. 78.40%) (Foltýnek et al., 2020b).

Our experiments show that experienced educators who read carefully and expect to encounter machine-paraphrased text could achieve accuracy between 80% and 100%. However, even in this setting, the average accuracy was below 80% for SpinBot and below 70% for SpinnerChief. We expect the efficacy will be lower in realistic scenarios, when readers focus less on spotting machine-paraphrased text.

4.3.3 Text-matching Software Baseline

To quantify the benefit of our automated classification over text-matching software, we tested how accurately current text-matching tools identify paraphrased text. We used two systems - Turnitin[7], which has the largest market share, and PlagScan[8] - one of the best-performing systems according to a test conducted by the European Network for Academic Integrity (Foltýnek et al., 2020a). We created 160 documents to simulate patch-writing - 2x40 from Wikipedia, 40 from arXiv, and 40 from Theses. Each document contains 20 randomly chosen paragraphs. Each set of 40 documents

[6] https://www.quiz-maker.com/
[7] https://www.turnitin.com/
[8] https://www.plagscan.com/

contains 10x4 files. The first dimension captures the paragraph length in sentences (varying from 1 to 10), and the second dimension represents the spinning tool (original, SpinBot, SpinnerChief-DF, and SpinnerChief-IF).

To ensure the text matching-software test is objective and comparable across the data sets, we used solely the overall percentages of text-match reported by a system. In most cases, systems correctly identified the right source. However, in many cases, there were also false positives caused by random matches, which means the systems' actual success is slightly lower than reported in the tables.

The results (see Table 4.8) show PlagScan struggles in the scenario of patch-writing. Even though the PD tool indexes Wikipedia and identifies plagiarism of the whole documents (Foltýnek et al., 2020a), the average text match reported for patch-written documents was 63%. Spinning with SpinBot and SpinnerChief-IF reliably prevented PlagScan from detection. For SpinBot, the average reported percentage of text-match was only 1%, and for SpinnerChief-IF 3%. In the case of SpinnerChief-DF PlagScan managed to identify 19% of plagiarism, which is caused by less portion of altered words. Nonetheless, considering the ground truth for all documents of 100%, we can conclude that patch-writing combined with spinning successfully avoids detection by PlagScan.

TABLE 4.8: Classification results (text-match in %) of two Plagiarism Detection systems: Turnitin and PlagScan.

	Turnitin			**PlagScan**		
	arXiv	Theses	Wikipedia	arXiv	Theses	Wikipedia
Original	84.0	5.4	98.7	44.6	22.3	65.0
SpinBot	7.0	1.1	30.2	0.0	0.1	0.5
SpinnerChief-DF	58.5	4.0	74.5	9.2	12.0	19.1
SpinnerChief-IF	38.8	1.2	50.1	1.8	0.5	3.1

As shown in Table 4.8, Turnitin performed better on patch-written documents than PlagScan. For Wikipedia, Turnitin reported 100% text-match in almost all cases. Also, the text-match reported for spun documents was much higher than for PlagScan — 31% for SpinBot, 74% for SpinnerChief-DF, and 50% for SpinnerChief-IF. Nonetheless, text-spinning prevents Turnitin from identifying a significant portion of plagiarism. Whereas Turnitin copes better with patch-writing, it does not index as many Theses

as PlagScan.

For both systems, we observe longer fragments lead to higher identification rates. This result corresponds without classification, which also yielded higher accuracy for longer passages.

This experiment extended our previous tests (Foltýnek et al., 2020b), in which the documents contained unaltered fragments interspersed with three paraphrased ones. Turnitin identified all unaltered fragments and one of the machine-paraphrased fragments. Of the other two machine-paraphrased fragments, Turnitin incorrectly credited one to another source and missed the other. These results were in line with the findings of Rogerson and McCarthy (2017), who used two paraphrasing tools to obfuscate a paragraph from a prior publication. When Turnitin received original paragraphs as inputs, the system found a 100% match with the source. However, for the two machine-paraphrased versions of the paragraph, Turnitin computed a similarity score of zero.

From these experiments, we conclude if plagiarists copy a few paragraphs and employ a paraphrasing tool to obfuscate their misconduct by spinning, the similarity is often below the text-matching tool's threshold, thus causing the plagiarism to remain undetected. Classification of machine-paraphrased text may be a useful complement of standard text-matching, which can alert users when there is a suspicion of deliberate obfuscation of plagiarism.

4.3.4 Limitations

ML techniques can supplement text-matching software to detect plagiarism from unknown sources and in cases where spun paragraphs do not achieve the PD system's threshold. However, the presented classification techniques provide a classification score with no intuitive interpretation (e.g., similarity to a document, similarity in writing style to an author). Therefore, positive classifications require additional manual validation of the affected paragraphs to avoid misleading plagiarism allegations resulting from false positives.

Chapter 5

Final Considerations

This thesis addressed the problem of increasing LM size, which we see as a risk to the research field. Large model sizes hinder research groups, without access to necessary hardware infrastructures, from experimenting with custom pre-trained models. Furthermore, an increased model size leads to increases in energy consumption and makes on-device deployment difficult. Our presented methods counteract these problems with an efficient architecture, gaining multiple teachers' knowledge and leveraging lexical knowledge sources. This chapter presents the final considerations to this thesis. Starting with a conclusion of the experiments (Section 5.1), we state a broader impact of our research contributions (Section 5.2). Finally, we outline our future work in the domains of KD, WSD, and PD (Section 5.3).

5.1 Conclusion

We introduced a method to perform KD using the knowledge of four large-scale LMs. Our method weights predictions of each teacher by confidence and uses them as targets for our efficient architecture (Longformer$_{mini}$). During training, the student model achieved faster convergence in language modeling than its larger counterparts, although using fewer parameters. However, the experiments conducted in this work did not provide conclusive answers yet. Thus we focussed on lexical KD solutions, which appeared more promising.

We proposed two methods to perform WSD (LMGC and LMGC-M) with eight neural LMs. Our methods distill knowledge from LKB which proved to be beneficial for NLU tasks (e.g., text-similarity). LMGC and LMGC-M perform WSD by combining neural

LMs with lexical resources from WordNet. We exceeded state-of-the-art WSD methods on five representative datasets (+0.5%) and improved the performance over BERT in general language understanding tasks (+1.1%). We release the code[1], and pre-trained models[2] to reconstruct our experiments.

The MPP detection experiments reported in this thesis extended a preliminary study (Foltýnek et al., 2020b) by analyzing two new collections (arXiv, Theses), a new paraphrasing tool (SpinnerChief), and eight neural language models based on the transformer architecture against our best-performing method (LMGC-M). We selected training and test sets reflecting documents particularly relevant for the plagiarism detection use case. The arXiv collection represents scientific papers written by expert researchers. Graduation Theses of non-native English speakers provide writing samples of authors whose style varies considerably. Wikipedia articles represent collaboratively authored documents for many topics and one of the sources from which students plagiarize most frequently. The classification approaches we devised are robust to identifying machine-paraphrased text, which educators face regularly. To support practitioners and facilitate an extension of the research on this important task, the data[3], code[4], and pre-trained models[2] of our study, as well as the web-based demonstration system are openly available[5].

5.2 Broader Impact

Our work focussed on distilling the knowledge of transformer language models and lexical resources to increase semantic representations. We see a potential positive impact with our methods to increase the accessibility for research purposes. Our model contributes to creating high-performing models at no additional hardware overhead, which benefits small research groups. Furthermore, as we have shown, Longformer$_{mini}$ uses an efficient attention scheme and a lower number of parameters than other LMs, resulting in less computational expenses when using the trained model. These computational expenses can translate into greenhouse gas emissions

[1] https://github.com/jpwahle-word-sense-disambiguation/wsd
[2] https://huggingface.co/models?search=jpwahle
[3] https://doi.org/10.5281/zenodo.3608000
[4] https://github.com/jpwahle/iconf22-paraphrase
[5] http://purl.org/spindetector

(Strubell et al., 2019) which would reduce when organizations and institutions use Longformer$_{mini}$ for deployment.

The lexical KD methods presented in this thesis successfully gained semantic understanding about polysemous words, which transferred to tasks beyond WSD. We think polysemy resolution will become crucial to broader domains such as the processing of mathematical formulas. Mathematical documents communicate their information in an ambiguous, context-dependent, and non-formal language (Greiner-Petter et al., 2019). For example, one equation might have a possibly infinite amount of equations with identical meaning. Our methods could disambiguate mathematical equations for information retrieval applications using external knowledge (e.g., using a comprehensive typology for mathematical notations, similar to WordNet for polysemous words).

We think the LM research field focusses more on improving task performance than questioning model size and computational expenses. Therefore researchers using smaller models may, at first, not be competitive against large models running on large hardware infrastructures. We believe with more publications about efficient LMs and the impact of large-scale models; researchers will weight these effects as important as task performance. Aside from this concern, we see no negative ethical or societal impacts of our work beyond what also applies to other core components of LMs.

5.3 Future Work

We aim to extend our multi-teacher KD experiments with converging Longformer$_{mini}$'s training fully. We think the converged student model could achieve comparable performance to its teachers and other KD techniques on general NLU tasks. The GLUE benchmarks is a logical choice as it allows us to compare the model to our lexical KD methods.

With an increasing amount of transformer-based language models proposed in short amounts of time, the choice for KD teachers grows too. This thesis chose teachers by training objective, vocabulary, and the datasets used for training. We expected prediction accuracy to be high when the datasets were similar as teacher

models already trained on similar data. However, another perspective would be to use teachers trained on very different datasets to capture knowledge from many domains. This experiment may ask for a higher penalty for unconfident predictions than we used in our method as some teachers might not produce meaningful predictions.

Our future work in the domain of WSD will include testing generalization on the WiC (Pilehvar and Camacho-Collados, 2019) and SuperGLUE (Wang et al., 2019a) datasets. Furthermore, we want to test discriminative fine-tuning (Howard and Ruder, 2018) against our parallel approach, i.e., whether training MLM on the LKB first, and subsequently optimizing for WSD yields the same effect as LMGC-M.

The lexical KD methods use multiple components to increase the model's semantic understanding. In the future, we seek to investigate which components of our methods are the most impacting by performing a more detailed ablation study. We conducted preliminary experiments with other knowledge bases (e.g., Wikidata) but leave for future work to incorporate knowledge from these sources.

We intend to extend the experimental results from WSD experiments to NLU benchmarks using the trained LMGC and LMGC-M models with XLNet. Although LMGC-M showed superior performance on average in the GLUE, we aim to estimate its classification impact on polysemy.

The MPP detection experiments indicated that obtaining additional training data is a promising approach for improving ML backed approaches for identifying machine-paraphrased text. Additional training data should cover more paraphrasing tools, topics, and languages. We see a community-driven open data effort (or crowdsourcing) as a promising option for generating a comprehensive training set.

Another interesting direction would be to use AE or AR to spin words or generate new text. This setup will be more realistic in the future as LMs are publicly available and generate texts that are difficult to distinguish from human writing.

Considering Longformer's superior performance in most tasks and the improvement of LMGC-M over its backbone network (XLNet), we see two interesting future experiments with our proposed methods. First, using Longformer as the backbone network for LMGC-M, and second, applying the converged Longformer$_{mini}$ model to the task of PD.

Some of these ideas have been realized in later works built on this thesis such as AE (Wahle et al., 2021a) and AR paraphrasing (Wahle et al., 2022c). Other works have used concepts from this thesis for applications such as abstractive text summarization (Kirstein et al., 2022) or misinformation detection about COVID-19 (Wahle et al., 2022a).

Bibliography

Anderson, J. R. (2015). *Cognitive Psychology and Its Implications*. Worth Publishers, New York, eighth edition edition.

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Bai, S., Kolter, J. Z., and Koltun, V. (2019). Deep Equilibrium Models. *CoRR*, abs/1909.01377.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The Long-Document Transformer. *ArXiv200405150 Cs*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Bengio, Y., Frasconi, P., and Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1188 vol.3.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.*, 5(2):157–166.

Bevilacqua, M. and Navigli, R. (2020). Breaking Through the 80% Glass Ceiling: Raising the State of the Art in Word Sense Disambiguation by Incorporating Knowledge Graph Information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864, Online. Association for Computational Linguistics.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. (2020). What is the State of Neural Network Pruning? *ArXiv200303033 Cs Stat*.

Blevins, T. and Zettlemoyer, L. (2020). Moving Down the Long Tail of Word Sense Disambiguation with Gloss Informed Bi-encoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1006–1017, Online. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. *ArXiv200514165 Cs*.

Camacho-Collados, J., Pilehvar, M. T., and Navigli, R. (2015). NASARI: A Novel Approach to a Semantically-Aware Representation of Items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado. Association for Computational Linguistics.

Chaplot, D. S. and Salakhutdinov, R. (2018). Knowledge-based Word Sense Disambiguation using Topic Models. *ArXiv180101900 Cs*.

Chen, S. F. and Goodman, J. (1996). An Empirical Study of Smoothing Techniques for Language Modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). 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)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *ArXiv200310555 Cs*.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. *ArXiv190102860 Cs Stat*.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2018). Universal Transformers. *CoRR*, abs/1807.03819.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv181004805 Cs*.

Du, J., Qi, F., and Sun, M. (2019). Using BERT for Word Sense Disambiguation. *ArXiv190908358 Cs*.

Edmonds, P. and Cotton, S. (2001). SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France. Association for Computational Linguistics.

Fellbaum, C. (1998a). A Semantic Network of English: The Mother of All WordNets. *Comput. Humanit.*, 32(2/3):209–220.

Fellbaum, C. (1998b). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. The MIT Press, Cambridge, Mass.

Foltýnek, T., Dlabolová, D., Anohina-Naumeca, A., Razı, S., Kravjar, J., Kamzola, L., Guerrero-Dib, J., Çelik, Ö., and Weber-Wulff, D. (2020a). Testing of Support Tools for Plagiarism Detection. *ArXiv200204279 CsDL*.

Foltýnek, T., Meuschke, N., and Gipp, B. (2019). Academic Plagiarism Detection: A Systematic Literature Review. *ACM Comput. Surv.*, 52(6):112:1–112:42.

Foltýnek, T., Ruas, T., Scharpf, P., Meuschke, N., Schubotz, M., Grosky, W., and Gipp, B. (2020b). Detecting Machine-Obfuscated Plagiarism. In Sundqvist, A., Berget, G., Nolin, J., and Skjerdingstad, K. I., editors, *Sustainable Digital Communities*, volume 12051 LNCS, pages 816–827. Springer International Publishing, Cham.

Furht, B. (2006). Huffman Coding. In Furht, B., editor, *Encyclopedia of Multimedia*, pages 278–280. Springer US, Boston, MA.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Gordon, M. A., Duh, K., and Andrews, N. (2020). Compressing BERT: Studying the Effects of Weight Pruning on Transfer Learning. *ArXiv200208307 Cs*.

Greiner-Petter, A., Ruas, T., Schubotz, M., Aizawa, A., Grosky, W., and Gipp, B. (2019). Why Machines Cannot Learn Mathematics, Yet. *ArXiv190508359 Cs*.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. and Titterington, M., editors, *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, volume 9 of *JMLR W&CP*, pages 297–304.

Hadiwinoto, C., Ng, H. T., and Gan, W. C. (2019). Improved Word Sense Disambiguation Using Pre-Trained Contextualized Word Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5296–5305, Hong Kong, China. Association for Computational Linguistics.

Hahn, S. and Choi, H. (2019). Self-Knowledge Distillation in Natural Language Processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 423–430, Varna, Bulgaria. INCOMA Ltd.

Han, S., Pool, J., Tran, J., and Dally, W. (2015). Learning both Weights and Connections for Efficient Neural Network. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28, pages 1135–1143. Curran Associates, Inc.

He, F., Liu, T., and Tao, D. (2020). Why resnet works? residuals generalize. *IEEE Trans. Neural Netw. Learn. Syst.*

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *ArXiv150201852 Cs*.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. (2017). Deep Learning Scaling is Predictable, Empirically. *ArXiv171200409 Cs Stat*.

Hiemstra, D. (2009). Language Models. In *Encyclopedia of Database Systems*, pages 1591–1594. Springer US, Boston, MA.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. *ArXiv150302531 Cs Stat*.

Ho, Y. and Pepyne, D. (2002). Simple Explanation of the No-Free-Lunch Theorem and Its Implications. *Journal of Optimization Theory and Applications*, 115(3):549–570.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1).

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. *ArXiv180106146 Cs Stat*.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2042–2050. Curran Associates, Inc.

Hu, M., Peng, Y., Wei, F., Huang, Z., Li, D., Yang, N., and Zhou, M. (2018). Attention-Guided Answer Distillation for Machine Reading Comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2077–2086, Brussels, Belgium. Association for Computational Linguistics.

Huang, L., Sun, C., Qiu, X., and Huang, X. (2019). GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3507–3512, Hong Kong, China. Association for Computational Linguistics.

Huang, Z. and Wang, N. (2017). Like What You Like: Knowledge Distill via Neuron Selectivity Transfer. *ArXiv170701219 Cs*.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. MIT press.

Jelinek, F., Mercer, R. L., Bahl, L. R., and Baker, J. K. (1977). Perplexity—a measure of the difficulty of speech recognition tasks. *J. Acoust. Soc. Am.*, 62(S1):S63–S63.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. (2020). TinyBERT: Distilling BERT for Natural Language Understanding. *ArXiv190910351 Cs*.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

Kågebäck, M. and Salomonsson, H. (2016). Word Sense Disambiguation using a Bidirectional LSTM. *ArXiv160603568 Cs*, Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon (CogALex - V):51–56.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *ArXiv Prepr. ArXiv14042188*.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models. *ArXiv200108361 Cs Stat*.

Kiefer, J. and Wolfowitz, J. (1952). Stochastic Estimation of the Maximum of a Regression Function. *Ann. Math. Statist.*, 23(3):462–466.

Kim, J., Park, S., and Kwak, N. (2018). Paraphrasing Complex Network: Network Compression via Factor Transfer. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31, pages 2760–2769. Curran Associates, Inc.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Kirstein, F., Wahle, J. P., Ruas, T., and Gipp, B. (2022). Analyzing multi-task learning for abstractive text summarization. *arXiv preprint arXiv:2210.14606*.

Kitaev, N., Kaiser, Ł., and Levskaya, A. (2020). Reformer: The Efficient Transformer. *ArXiv200104451 Cs Stat*.

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1.

Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. (2019). Revealing the Dark Secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4364–4373, Hong Kong, China. Association for Computational Linguistics.

Kudo, T. and Richardson, J. (2018). 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*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *Ann. Math. Statist.*, 22(1):79–86.

Kumar, S., Jat, S., Saxena, K., and Talukdar, P. (2019). Zero-shot Word Sense Disambiguation using Sense Definition Embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681, Florence, Italy. Association for Computational Linguistics.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv190911942 Cs*.

Lau, J. H. and Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany. Association for Computational Linguistics.

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lee, S. H., Kim, D. H., and Song, B. C. (2018). Self-supervised Knowledge Distillation Using Singular Value Decomposition. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, volume 11210, pages 339–354. Springer International Publishing, Cham.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *ArXiv191013461 Cs Stat*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal Loss for Dense Object Detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, Venice. IEEE.

Liu, H. and Singh, P. (2004). ConceptNet — A Practical Commonsense Reasoning Tool-Kit. *BT Technology Journal*, 22(4):211–226.

Liu, X., He, P., Chen, W., and Gao, J. (2019a). Improving Multi-Task Deep Neural Networks via Knowledge Distillation for Natural Language Understanding. *ArXiv190409482 Cs*.

Liu, X., He, P., Chen, W., and Gao, J. (2019b). Multi-Task Deep Neural Networks for Natural Language Understanding. *ArXiv190111504 Cs*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019c). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv190711692 Cs*.

Loureiro, D. and Jorge, A. (2019). Language Modelling Makes Sense: Propagating Representations through WordNet for Full-Coverage Word Sense Disambiguation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691, Florence, Italy. Association for Computational Linguistics.

Luo, F., Liu, T., He, Z., Xia, Q., Sui, Z., and Chang, B. (2018). Leveraging Gloss Knowledge in Neural Word Sense Disambiguation by Hierarchical Co-Attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1402–1411, Brussels, Belgium. Association for Computational Linguistics.

Ma, W., Cui, Y., Si, C., Liu, T., Wang, S., and Hu, G. (2020). CharBERT: Character-aware Pre-trained Language Model. *ArXiv Prepr. ArXiv201101513*.

Madera, Q., García-Valdez, M., and Mancilla, A. (2014). Ad Text Optimization Using Interactive Evolutionary Computation Techniques. In Castillo, O., Melin, P., Pedrycz, W., and Kacprzyk, J., editors, *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, volume 547, pages 671–680. Springer International Publishing, Cham.

Mao, Y., Wang, Y., Wu, C., Zhang, C., Wang, Y., Yang, Y., Zhang, Q., Tong, Y., and Bai, J. (2020). LadaBERT: Lightweight Adaptation of BERT through Hybrid Model Compression. *ArXiv200404124 Cs*.

Markov, A. A. (1954). The theory of algorithms. *Trudy Matematicheskogo Instituta Imeni VA Steklova*, 42:3–375.

Mei, H., Bansal, M., and Walter, M. R. (2016). Coherent Dialogue with Attention-based Language Models. *arXiv preprint arXiv:1611.06997*.

Meuschke, N. and Gipp, B. (2013). State of the Art in Detecting Academic Plagiarism. *Int. J. Educ. Integr.*, 9(1):50–71.

Michel, P., Levy, O., and Neubig, G. (2019). Are sixteen heads really better than one? In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. *ArXiv13013781 Cs*.

Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., and Khudanpur, S. (2011a). Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE.

Mikolov, T., Kombrink, S., Deoras, A., Burget, L., and Cernocky, J. (2011b). Rnnlm-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. *ArXiv13104546 Cs Stat*.

Miller, G. A. (1995). WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Miller, G. A., Leacock, C., Tengi, R., and Bunker, R. T. (1993). A semantic concordance. In *Proceedings of the Workshop on Human Language Technology - HLT '93*, page 303, Princeton, New Jersey. Association for Computational Linguistics.

Mirzadeh, S. I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., and Ghasemzadeh, H. (2020). Improved Knowledge Distillation via Teacher Assistant. *AAAI*, 34(04):5191–5198.

Moro, A. and Navigli, R. (2015). SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado. Association for Computational Linguistics.

Napoles, C., Gormley, M., and Van Durme, B. (2012). Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and*

Web-Scale Knowledge Extraction (AKBC-WEKEX), pages 95–100, Montréal, Canada. Association for Computational Linguistics.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):1–69.

Navigli, R., Jurgens, D., and Vannella, D. (2013). SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.

Navigli, R. and Ponzetto, S. P. (2012). BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217–250.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2014). Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar. Association for Computational Linguistics.

Nguyen, T. H. and Grishman, R. (2015). Relation Extraction: Perspective from Convolutional Neural Networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 39–48, Denver, Colorado. Association for Computational Linguistics.

Pasini, T. and Navigli, R. (2020). Train-O-Matic: Supervised Word Sense Disambiguation with no (manual) effort. *Artificial Intelligence*, 279:103215.

Passalis, N. and Tefas, A. (2019). Learning Deep Representations with Probabilistic Knowledge Transfer. *ArXiv180310837 Cs Stat*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). 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)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Logan, R., Schwartz, R., Joshi, V., Singh, S., and Smith, N. A. (2019). Knowledge Enhanced Contextual Word Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Pham, N.-Q., Kruszewski, G., and Boleda, G. (2016). Convolutional Neural Network Language Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1153–1162, Austin, Texas. Association for Computational Linguistics.

Pilehvar, M. T. and Camacho-Collados, J. (2019). WiC: The Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations. In *Proceedings of the 2019 Conference of the North*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Pilehvar, M. T., Camacho-Collados, J., Navigli, R., and Collier, N. (2017). Towards a Seamless Integration of Word Senses into Downstream NLP Applications. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1857–1869, Vancouver, Canada. Association for Computational Linguistics.

Prentice, F. M. and Kinden, C. E. (2018). Paraphrasing tools, language translation tools and plagiarism: An exploratory study. *Int J Educ Integr*, 14(1):11.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv191010683 Cs Stat*.

Raganato, A., Camacho-Collados, J., and Navigli, R. (2017). Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *ArXiv150601497 Cs*.

Rogerson, A. M. and McCarthy, G. (2017). Using Internet based paraphrasing tools: Original work, patchwriting or facilitated plagiarism? *Int J Educ Integr*, 13(1):2.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2015). FitNets: Hints for Thin Deep Nets. *ArXiv14126550 Cs*.

Ruas, T., Ferreira, C. H. P., Grosky, W., de França, F. O., and de Medeiros, D. M. R. (2020). Enhanced word embeddings using multi-semantic representation through lexical chains. *Information Sciences*, 532:16–32.

Ruas, T., Grosky, W., and Aizawa, A. (2019). Multi-sense embeddings through a word sense disambiguation process. *Expert Systems with Applications*, 136:288–303.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sammut, C. and Webb, G. I., editors (2010). *Mean Squared Error*, pages 653–653. Springer US, Boston, MA.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *ArXiv191001108 Cs*.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. (2020). SuperGlue: Learning Feature Matching with Graph Neural Networks. *ArXiv191111763 Cs*.

Schuster, M. and Nakajima, K. (2012). Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2019). Green AI. *ArXiv190710597 Cs Stat*.

Sennrich, R., Haddow, B., and Birch, A. (2016). 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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sileo, D., Van De Cruys, T., Pradel, C., and Muller, P. (2019). Mining discourse markers for unsupervised sentence representation learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3477–3486, Minneapolis, Minnesota. Association for Computational Linguistics.

Snyder, B. and Palmer, M. (2004). The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain. Association for Computational Linguistics.

Strapparava, C. and Mihalcea, R. (2007). SemEval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74, Prague, Czech Republic. Association for Computational Linguistics.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Sun, S., Cheng, Y., Gan, Z., and Liu, J. (2019). Patient Knowledge Distillation for BERT Model Compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332, Hong Kong, China. Association for Computational Linguistics.

Sun, Y., Wang, S., Li, Y., Feng, S., Tian, H., Wu, H., and Wang, H. (2020a). ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding. *AAAI*, 34(05):8968–8975.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. (2020b). MobileBERT: A Compact Task-Agnostic BERT for Resource-Limited Devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Tarvainen, A. and Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1195–1204. Curran Associates, Inc.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2020). Efficient transformers: A survey. *ArXiv Prepr. ArXiv200906732*.

Tran, K., Bisazza, A., and Monz, C. (2016). Recurrent memory networks for language modeling. *ArXiv Prepr. ArXiv160101272*.

Trinh, T. H. and Le, Q. V. (2019). A Simple Method for Commonsense Reasoning. *ArXiv180602847 Cs*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in*

Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc.

Vial, L., Lecouteux, B., and Schwab, D. (2019). Sense Vocabulary Compression through the Semantic Knowledge of WordNet for Neural Word Sense Disambiguation. *ArXiv190505677 Cs*.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Wahle, J. P., Ashok, N., Ruas, T., Meuschke, N., Ghosal, T., and Gipp, B. (2022a). Testing the generalization of neural language models for covid-19 misinformation detection. In Smits, M., editor, *Information for a Better World: Shaping the Global Future*, pages 381–392, Cham. Springer International Publishing.

Wahle, J. P., Ruas, T., Foltýnek, T., Meuschke, N., and Gipp, B. (2022b). Identifying machine-paraphrased plagiarism. In Smits, M., editor, *Information for a Better World: Shaping the Global Future*, pages 393–413, Cham. Springer International Publishing.

Wahle, J. P., Ruas, T., Kirstein, F., and Gipp, B. (2022c). How large language models are transforming machine-paraphrased plagiarism. *arXiv preprint arXiv:2210.03568*.

Wahle, J. P., Ruas, T., Meuschke, N., and Gipp, B. (2021a). Are neural language models good plagiarists? a benchmark for neural paraphrase detection. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 226–229.

Wahle, J. P., Ruas, T., Meuschke, N., and Gipp, B. (2021b). Incorporating word sense disambiguation in neural language models. *arXiv preprint arXiv:2106.07967*.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2019a). Superglue: A stickier benchmark for general-purpose language understanding systems. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019b). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *ArXiv180407461 Cs*.

Wang, B., Wang, A., Chen, F., Wang, Y., and Kuo, C.-C. J. (2019c). Evaluating word embedding models: Methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8:e19.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788. Curran Associates, Inc.

Weber-Wulff, D. (2019). Plagiarism Detectors Are a Crutch, and a Problem. *Nature*, 567(7749):435–435.

Xu, G., Liu, Z., Li, X., and Loy, C. C. (2020). Knowledge Distillation Meets Self-Supervision. *ArXiv200607114 Cs*.

Xu, W. and Rudnicky, A. (2000). Can artificial neural networks learn language models? In *Sixth International Conference on Spoken Language Processing*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *ArXiv190608237 Cs*.

Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7130–7138.

You, S., Xu, C., Xu, C., and Tao, D. (2017). Learning from Multiple Teacher Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1285–1294, New York, NY, USA. Association for Computing Machinery.

Zagoruyko, S. and Komodakis, N. (2017). Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. *ArXiv161203928 Cs*.

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., and Choi, Y. (2019). Defending Against Neural Fake News. *ArXiv190512616 Cs*.

Zhang, L., Song, J., Gao, A., Chen, J., Bao, C., and Ma, K. (2019a). Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Zhang, Q., Wang, D. Y., and Voelker, G. M. (2014). DSpin: Detecting Automatically Spun Content on the Web. In *Proceedings 2014 Network and Distributed System Security Symposium*, San Diego, CA. Internet Society.

Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., and Liu, Q. (2019b). ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

Appendix A

Dataset Details

As our experiments in Section 4.2 showed, the additional learning of WSD in parallel to MLM leverages our model's performance to other natural language understanding tasks. One direct result of incorporating LMGC training methods is the increased results on the GLUE tasks. To investigate the correlation between the number of polysemous words and LMGC, we report how many words from each GLUE tasks can be found as polysemous in WordNet, as Table A.1 shows.

TABLE A.1: The number of polysemous words in relation to the number of total words in the GLUE evaluation datasets except for WNLI. Under each dataset is the number of training examples.

		MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k
Train	Words	11 695k	8 050k	3 818k	633k	65k	114k	160k	130k
	Polysemous	5 205k	3 452k	1 439	366k	23k	51k	65k	50k
	Relation (%)	44.5	42.9	37.7	57.8	35.5	44.9	40.9	39.0
Test	Words	286k	893k	205k	17k	8k	34k	17k	14k
	Polysemous	126k	382k	78k	9k	2k	15k	7k	5k
	Relation (%)	44.2	42.8	8.2	54.1	34.7	46.4	41.9	38.7

Our first observation is the number of polysemous words in the text from different NLU tasks is high in general. We find specific tasks like QQP with 42.8% polysemous words in the test set with a high accuracy boost in LMGC-M, but other tasks like SST-2 with 54.1% in the test set with almost no improvement. This result indicates LMGC-M can improve the performance with an increased number of polysemous words but does not necessarily mean there is a causation. The baseline model BERT$_{base}$ already learns WSD to some extent to resolve polysemy in text, which the results in GLUE tasks show. Also, the difficulty of some tasks is not bound to polysemous words. However,

as our experimental results showed, on average, the performance of LMGC-M is higher than the original BERT$_{base}$ model, which indicates WSD influence general language understanding.