Universiteit Leiden

Master Computer Science

Integration of BERT and WordNet for improving Natural Language Understanding

Name:	Mohamed Barbouch
Student ID:	s2129116
Date:	10/03/2021
Specialisation:	Computer Science and Advanced Data Analytics
1st supervisor:	Suzan Verberne
2nd supervisor:	Tessa Verhoef

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Preface

”

The moment you start thinking about the meaning of a word,
the word becomes meaningless!

Back in 2014, on my way home from work in Shiphol-Rijk, I started repeating a word in my head just before entering The Hague. Gradually I began to think about the meaning of that word, trying to figure out the connection between the word-name and the meaning it represents. To my surprise, at some point I completely lost the (connection with the) meaning of that word. The word sounded so dry in my head, without being able to imagine anything about it.

That has changed my perception about how we use words in the formation of concepts. I experienced a chain-breaking between word names as labels and everything they represent as concepts. An experience that made me question our understanding of the language we use, the means of communication that is unique to us humans, and something on which our connection as well as the shaping of our present, past and future strongly depend.

Why is it that exchange of words intuitively goes quite smoothly and automatically, enabling a sensible communication process in which information is transferred and possibly new thoughts arise, while if we consciously think about the used vocabulary (as a means) and try to find out the connection between them, we then lose control over

the entire process – including concept formation and understanding? Would our thinking, word storage and retrieval and concept formation be separate? Why can't we rationally comprehend the underlying model, given that thinking is part of us, which also uses the same vocabulary as in the language we communicate with? If we have a thinking system and an intuitive system, why can the second reasonably easily connect all parts of the communication process and quickly provide us with conception while we do not (yet) reach the same level of comprehension with the first one?

All these questions make it interesting to uncover the secrets behind human language. Success in this would provide enormous opportunities to connect the power of the computers to assist us in processing the large amounts of data generated as a result of our natural language usage, something that together hopefully would bring us a step closer to our understanding of ourselves.

It is therefore my pleasure to now guide the completion of my master's program I started a few years ago in Computer Science and Advanced Data Analytics in the direction of natural language understanding and deep learning.

Acknowledgements

The journey of taking my learning to this stage would not have been possible without a number of great people who have meant a lot to me.

First of all, I want to thank myself for making the decision to quit a permanent job as a software developer and go back to school. Flowing from one project to another where ready-to-use products, tools, and frameworks are used has made me discover how techniques work, but it also left me stuck in frameworks where I missed the answer to why questions. For this it was necessary to think on a large scale, transcending to fundamental concepts that enable many of the applications that make our daily life a little easier.

I would like to thank my father who was understanding in making my choice and wished me success. I really appreciate the support of my mother, who took care of a lot of the daily household. It was a great favor to come home and find a hot meal, washed clothes, and a clean room. Thanks mom. :)

In my experience, Leiden Institute of Advanced Computer Science (LIACS) was the right place where I could follow my master's program, especially because it focuses on people, where there are short lines of communication with the lecturers and the organization as a whole, having a vision in which artificial intelligence is put at the service of people and society.

I learned a lot from Suzan Verberne about data science in general and about natural language processing and text mining in particular. I would like to thank her very much for her valuable supervision, patience and guidance during this thesis project. While I sometimes go into details, she was the one who kept me focused on the big picture, interpretation and the end goal. I would also like to thank Tessa Verhoef for fulfilling the role of second supervisor, where she was still critical of adding nuances to a number of points. Together with Suzan and her, we co-authored a paper that is now for review at DeeLIO 2021 Workshop.

Of course I would also like to thank everyone from the LIACS staff who was part of my learning process at the institute so far. They all complete the entire LIACS community and they make it a great place to belong to.

Abstract

In this thesis we propose an integration of BERT and WordNet for improving natural language understanding (NLU). While BERT has shown superiority in several NLU tasks, it, however, also turned out to be limited in making and 'understanding' semantic relationships related to, for example, abstraction, inference and reasoning. We believe that the implicit way in which the model learns in context limits the model to obtain the required knowledge about language semantics that are not necessarily present in given text. In this regard, we connect BERT with WordNet, an external semantic lexicon that is explicitly constructed by humans, which provides information about words at different abstraction levels.

We represent the semantic knowledge from WordNet as embeddings using *path2vec* and *wnet2vec*, and realize the integration following two different strategies: *external combination*, using a multi-layer perceptron (MLP) meta learner, and *internal inclusion*, building upon VGCN-BERT. We evaluate the performance on sentiment analysis (SA) and sentence similarity, using SST-2 and STS-B from GLUE benchmark, respectively.

We found internal inclusion of BERT with wnet2vec to be the best model. However, our model did generally not outperform the state-of-the-art benchmarking results, although it was slightly better on

SST-2. The limitations of lower WordNet coverage, moderate WSD in path2vec and inclusion of irrelevant synsets in wnet2vec, would have prevented the model from outperformance. Nevertheless, we did find cases were the integrated model was better than BERT-only model. Moreover, analysing the multi-head self-attention of BERT has shown that WordNet embeddings, especially of wnet2vec, have a mutual influence with BERT embeddings and have eventually strongly contributed to the final output. An observation is promising for future work to improve on the aforementioned limitations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

The task of making human language understandable for computers has increasingly attracted the attention of Artificial Intelligence (AI) research in recent decades [41]. With the advent of social media and smartphones, people all over the world are generating billions and billions of records of textual and audiovisual data[1] (Figure 1.1). While computers can handle these enormous amounts of data quite well, human intervention is still required to unfold the inner message. Due to the asymmetry between the fast processing and (relatively) 'poor understanding' by computers, and the fast understanding and slow processing of humans, a gap has emerged between the amount of raw data generated and the ability to do something meaningful with that data (e.g. Fig. 1.2). This has both emphasized the necessity and created the opportunity to explore and exploit a great potential for bridging the gap between human and artificial understanding of natural language. By joining the two forces, research in the field can be accelerated forward,

[1] https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read

possibly bringing the technological singularity [44] steps closer.

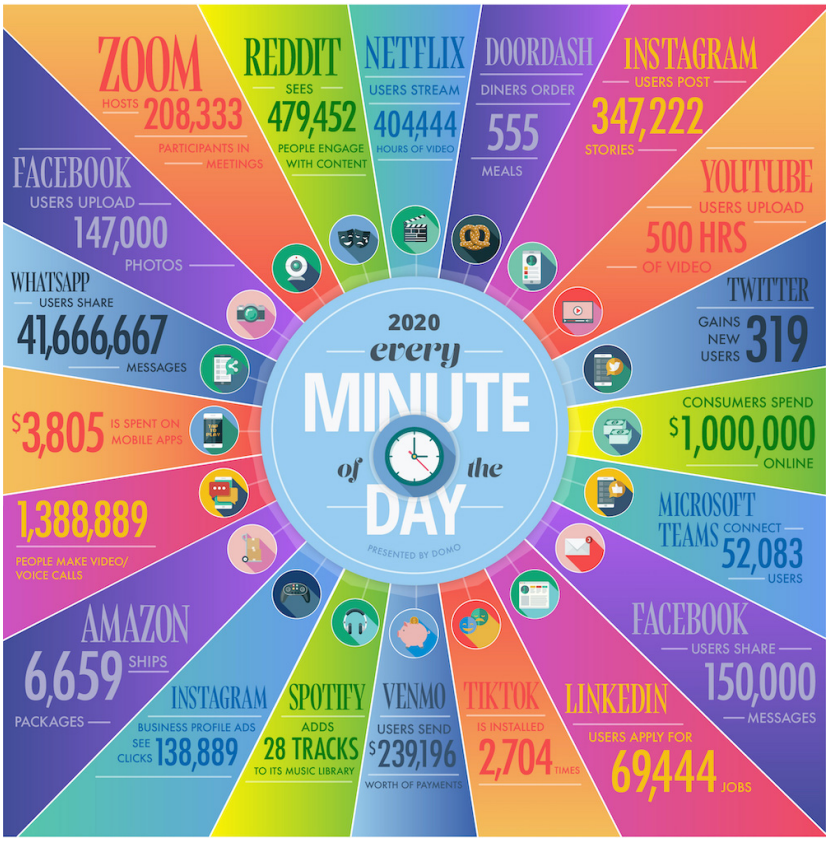


Figure 1.1: How much data is generated each minute in 2020.[2]

The problem of making human language comprehensible for computers is therefore the general motivation behind this research. Although this is an interdisciplinary problem between computer science, linguistics and psychology, for us the computational side is the most interesting. This is mainly concerned by natural language understanding NLU, a sub-discipline within natural language processing (NLP) (Fig. 1.3). Although 'understanding' is the starting point, it is formally not clear what is meant by it. This is a challenging question, since the concept has not yet been solved psycholinguistically [10]. So far, NLU has

[2] https://www.domo.com/learn/data-never-sleeps-8
[3] https://towardsdatascience.com/machine-learning-with-big-data-86bcb39f2f0b

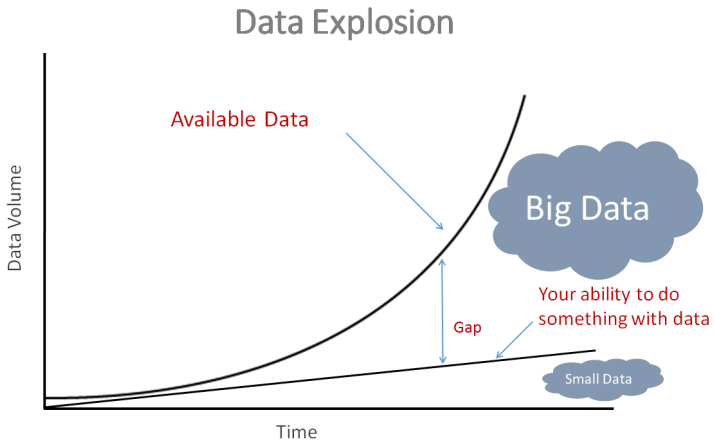


Figure 1.2: The exponential increase of generated data and the gap with the (linear) ability to handle that data meaningfully.[3]

approached this in a black-box way where the computer is being tested based on the behavior that it shows in various language tasks that require abstract language understanding [45]. This means, if the expected output for a given language-data-input is given then the inference is made that the computer has 'understood' the task, relative to the degree of shown performance. The assumption is that 'understanding' would then be implicitly captured in the underlying models that have led to the results. The tasks include, among many others, Question Answering (QA), Sentiment Analysis (SA) and Natural Language Inference (NLI). Being able to handle these kinds of tasks well has a lot of interest in practice, such as in stock markets, where bots place buy and sell orders based on sentiment and conclusions from various news sources [48]. Big search engines like Google Search[4] and Bing[5] keep improving their services by understanding the daily billions of search queries better linguistically. Other commercial organizations increase the effectiveness of their marketing around their products and

[4] https://www.blog.google/products/search/search-language-understanding-bert/

[5] https://azure.microsoft.com/en-us/blog/bing-delivers-its-largest-improvement-in-search-experience-using-azure-gpus/

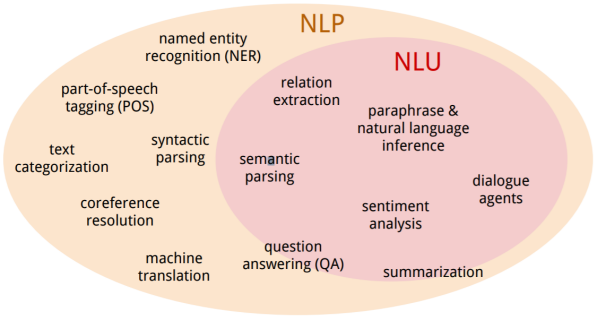


Figure 1.3: NLU and the different tasks it covers.[6]

services by personalizing the offerings based on data generated by the users. Online retailers like Amazon advocate in future directions for more user involvement and understanding when recommending products [39]. And there are countless applications where NLU models are useful.

1.2 Context of the Study

The models that drive NLU research nowadays are based on language modeling (LM) and deep learning (DL). *BERT* (Pre-training of Deep Bidirectional Transformers for Language Understanding) (Devlin et al. [8]) as one of these models forms, with its outperforming results on various NLU tasks, the most successful paradigm for deep language modeling in the recent years. The strength of BERT lies in combining *pre-training* on large language corpora, such as Wikipedia and BookCorpus, with the *attention* mechanism of *Transformers* (Vaswani et al. [42]), where word meanings are learned *bidirectionally* in sentences, i.e. from left to right and from right to left contexts. Pre-training allows general language knowledge to be *transferred* to downstream tasks through *fine-tuning*, with relatively less effort, yet achieving high performance. However, further analysis has shown that the model still struggles with certain cases. These are mainly related to

[6] https://nlp.stanford.edu/~wcmac/papers/20140716-UNLU.pdf

semantics, such as reasoning, number representation, named entity replacement, commonsense and pragmatic inference (Rogers et al. [37]). Syntactically there are also some cases, but those are limited, for example, to negations (Ettinger [9]) and malformed input.

For understanding, the *meaning* of words and word combinations is very important. For example, in the lexical approach [21] for learning a language, Lewis M. argues that lexical phrases (called 'chunks') that frequently appear, such as "*We'll see*", "*Would you like a cup of coffee?*", "*I'll get it*", etc, are key in language acquisition [29]. The way in which BERT learns about language is done *implicitly*. The model captures patterns from text itself by updating its internal structure weights during training, based on word masking and next sentence prediction (NSP) tasks, without exposure to any rules about the language. Although this is a major advantage where the model learns independently of human supervision, in this way it remains also a disadvantage (as we have seen in the previous section) to miss familiar (and sometimes basic) relationships, from a human point of view. The patterns are stored in the so-called *word embeddings*. These are contextualized vector representations with continuous numerical values for each word. The embeddings have proven to be capable of absorbing different language patterns. This includes syntactical, structural as well as semantic relationships. With this property we would be able to customize the representation and therefore also influence the type of patterns that are included.

We aim to extend BERT's implicit semantic knowledge with *explicit* knowledge extracted from external human-constructed semantic lexicons. In particular, we involve WordNet[7] to fill the second role. This is a lexical database (Miller et al. [28]) where words are semantically connected. The construction is based on psycholinguistic principles about how the human brain deals with language. In this

[7] https://wordnet.princeton.edu/

regard, we aim to examine the impact of exploiting the relationships between words from WordNet in BERT, and find out for what type of tasks they are influential and for what not. Considering understanding-based tasks, we hypothesize that semantics play a major role. We consider the enrichment to be our point of connection to test the hypothesis.

The reasoning behind the choice to enrich BERT with explicit semantic knowledge from WordNet goes as follows:

- ❖ *Natural language* is the basic and distinctive means of communication of humans.
- ❖ In order to involve an *external actor* in the communication process of humans, *understanding* of natural language is necessary.
- ❖ Given that computers are superior in (data) *processing*, we aim to take advantage of this property for the benefit of humans.
- ❖ We therefore regard a *computer* as an external actor.
- ❖ *Deep learning* has made breakthroughs in natural language understanding in recent years, making deep neural models become state-of-the-art in the field.
- ❖ Deep neural models learn by capturing language patterns *implicitly*.
- ❖ Understanding requires association, abstraction and inference between and from words to form *meaning*.
- ❖ Meaning is determined by *semantics*.
- ❖ Although implicit learning has the advantage of independence and catching patterns that are (still) unknown to humans, it, however, has the disadvantage of missing explicit semantics, that are not mentioned in text, but are necessary for full understanding of the text.
- ❖ Thus, we use *explicit* semantic constructs to transfer, enrich or integrate knowledge related to human language understanding into deep neural models.

1.2.1 Research Questions

The intended integration as well as the test of its contribution leads us to pose the following research questions.

1. **How to represent semantics from WordNet as embeddings?**
   WordNet is a network with nodes of textual descriptions about words. BERT represents words as embedding vectors. To integrate the two, WordNet structure needs to be converted to a compatible format with BERT.

2. **How to combine WordNet embeddings with BERT embeddings?**
   Obviously, in theory there would be several possible options to realize the integration. We will consider some to find a good approach. 'Good', be it one with best performance and minimal limitation.
3. **Does including explicit semantic knowledge from WordNet improve the performance of BERT on NLU tasks?**
   After integration, the task is to test whether the enriched knowledge from WordNet contributes to improvement of BERT on natural language understanding. This will require evaluation on task-related datasets.
4. **How do WordNet embeddings affect BERT model?**
   Apart from improvement or deterioration, it is also important to gain insight into the behavior of BERT after adding WordNet embeddings. It might be important to, for example, further analyze the influence on this behavior.
5. **What kind of NLU tasks can be solved with WordNet combined with BERT?**
   We need to diversify the types of tasks – that are driven by semantics – to see if there are differences between tasks that can be improved and those that cannot.

1.3 Objectives and Contributions

By conducting this research, we intend to make the following contributions.

- Integrating explicit semantic relationships knowledge from *WordNet* [28] with state-of-the-art pre-trained language model of *BERT* [8]. (Building, in particular, on the (proposed future) work of Lu et al. [23]).
- Applying *path2vec* (Kutuzov et al. [19]) and *wnet2vec* (Saedi et al. [38]) as WordNet embeddings representation suitable for WordNet-BERT integra-

tion.

- Evaluation of created WordNet-BERT models on sentiment analysis (SST-2) and sentence similarity (STS-B) NLU tasks [45].
- Training both path2vec and wnet2vec on entire WordNet and making these models available for the NLP research community[8].
- Additional: evaluation of VGCN-BERT on STS-B task.

1.4 Overview of the Thesis

For the remainder of this document, the thesis is structured as follows.

After this introduction, related work follows in Chapter 2. We will take you through the developments that have led to the latest successes in deep language modeling with BERT [8], where we pay particular attention to relevant works we (directly) build upon.

We describe our method in Chapter 3. In the order of the research questions, we first select two methods, *path2vec* [19] and *wnet2vec* [38], with which we convert WordNet [28] to embeddings. For the combination with BERT we follow two different ensemble strategies: *internal inclusion* and *external combination*, making use of VGCN-BERT [23] and meta multi-layer perceptron (MLP) [31] architectures, respectively. Furthermore, we address the limitation of WordNet coverage, and explain how we manage to get the coverage as high as possible.

For evaluating our method, we present our experimental results in Chapter 4. This includes datasets, experimental setup, results as well as further analysis. We use relevant datasets for NLU, including ones from reference work and from the GLUE [45] benchmark. Our results are compared to baselines of referenced papers. In the analysis we examine the development of attention scores in the last

[8]Models are published at: https://github.com/mbarbouch/WN-BERT.

layer as well as across all layers of BERT, both integrated with and disconnected from WordNet, and try to gain insights for why observed behavior – w.r.t. the task – is shown.

Points that require additional attention as a result of the research are discussed in Chapter 5. In particular, we discuss the limitation of WordNet coverage and performance of synset selection in case of ambiguous words, a problem related to word sense disambiguation (WSD) [1].

In Chapter 6 we wrap up and conclude on the findings of the study. We answer each research question separately. Furthermore, we provide directions for future work, with suggestions for related work that might be useful.

Chapter 2
Related Work

Over the years there has been a shift in how NLU tasks are tackled. Last century, the models were mainly rule-based and statistical in nature [17]. For example, features were extracted from text using Bag-of-Words, term and document frequencies, and distances between words. These managed to catch targeted language patterns to a certain extent, but the imposed constraints, on the other hand, were limiting to include other patterns that are not covered in the rules.

With the breakthrough of deep learning in the last decade [11, 18], it became possible to learn word and text meanings from context [25], free of rules and constraints. The inner weights in which the network ends after training on a textual input, e.g. word or sequence of words, gives a unique numerical representation for each textual unit, e.g. tokens or entire sentences. In fact, this representation is a vector with a series of continuous values (i.e. decimal numbers) called 'embeddings'. The vector space is able to catch any form of (non-linear) patterns present in the language. However, the task remains to find an 'optimal' combination of the decimal numbers for any given textual unit.

Mikolov et al. [25] and Pennington et al. [34] have proposed Word2Vec and

GloVe for word vector representations learnt unsupervised on large corpuses of data. These made it possible, for example, to find out relationships between words by applying arithmetic operators to the vectors. A calculation like $King - Man + Woman$ would give $Queen$ as answer. Strictly speaking: a vector that is closest to the vector for 'Queen'. However, when words become ambiguous, such as the words 'bank' and 'stick', which have multiple meanings, it becomes difficult to link the correct representation of the meaning. In addition, these models were limited to single word representations. This is where language modeling came into place the last few years.

ULMFiT [13] and ELMo [35] learn different word meanings from context, first pre-trained on large language corpora to grasp general language knowledge, and then fine-tuned for downstream tasks to transfer the learning. The word embeddings became *contextualized*. Yet, it remains that when words are combined in a sentence, some become more important than others for catching the message from the text, depending on where the emphasis is placed. In a sentence like "*The animal didn't cross the street because it was too tired*"[9], it is for traditional language models confusing where the word 'it' is referring to; is it 'animal' or 'street'?

Transformers (Vaswani et al.) [42] tackle this by using attention mechanisms to favor some words over others by giving them more weights. The network is trained using encoder and decoder layers to go back and forth between input and output in order to learn word meanings and determine their importances. In machine translation where the same terms appear in two different languages, these terms are very helpful for the model to determine connections.

OpenAI Generative Pre-trained Transformer (GPT) (Radford et al.) [36] builds on the ideas of Transformers of attentions and decoder component for improving

[9] http://jalammar.github.io/illustrated-transformer/

language understanding in pre-trained language models. Its network is trained uni-directional, meaning that text input is passed forward from left to right context.

More recently, BERT (or Pre-training of Deep Bidirectional Transformers for Language Understanding) (Devlin et al. [8]) has further developed this by training the Transformers network bi-directionally. That means that in addition to left-to-right context, right-to-left context is also included during the training process. With this idea, BERT achieved state-of-the-art results on various NLU tasks in 2018, taking the power of pre-trained language models to the next level, comparable to AlexNet's breakthrough in computer vision in 2012 [18].

However, as stated in Context of the Study (Section 1.2), BERT turned out to have great difficulty with understanding language semantics [37]. For this reason we aim to investigate to what extent this problem can be alleviated by adding human-known semantic knowledge. We use WordNet, a main semantic knowledge base that came to a standstill before the flourishing of deep learning due to its limitations with traditional methods. Now we can test its potential with BERT and examine whether a hybrid integration between deep language models and external knowledge bases can be complementary.

2.1 WordNet

WordNet [27] is a semantic lexical database built by psychologists and linguists from Princeton University between 1985 and 2005. It represents a hierarchical network of words with their semantic relationships, building upon theories from psycholinguistics. The words cover nouns, verbs, adjectives and adverbs as parts-of-speech. Senses between words are defined in sets and synonyms called *synsets*.

The synsets make up an inheritance system in which common properties of words are stored once in a super synset, which could be inherited by one or more sub-synsets that have one or more additional attributes. Nouns are distributed across 25 unique beginners. These are the top-level generic synsets. The relationships are modeled as *hyponymy*, *antonymy* and *meronymy* (Figure 2.1). Hyponymy denotes relations of the form word A *'is a'* word B, e.g. sparrow 'is a' bird. The other way down, hypernymy is the super synset (bird). Antonymy speaks for itself; it indicates contradictions. Meronymy, on the other hand, indicates relationships of word A *'is part of'* word B. For example, a handle 'is part of' a door. Princton WordNet 3.0[7] contains about 150k words, organized in 170k synsets (as some words can have multiple meanings), making up around 200k word-sense pairs.

Semantic similarities between two words could be determined by, for example, calculating the distance between associated synsets [33]. Leacock & Chodorow (*LCH*) [20] algorithm is used to compute the shortest path, with respect to '*is a*' relationship hierarchy; WuPalmer (*WuP*) [46] computes the path to the most specific root node that two synsets share; while in a third approach an inverse of the shortest *path* is calculated. For word relatedness, in addition to finding a balanced path with Hirst & St-Onge (*HsO*) [12] between path length and relationship type that doesn't change often, also synset descriptions (called '*glosses*') are used. *Lesk* [2] measure determines the relatedness by computing overlap scores between synset glosses. *Vector* [32] does this by computing the cosine between gloss vectors using a co–occurrence matrix consisting of gloss tokens.

Although these methods were powerful in finding out semantic relationships, they were, however, limited to computations between two words, by method constraints (e.g. assuming the 'is a' relationship in LCH [4]), and by having the

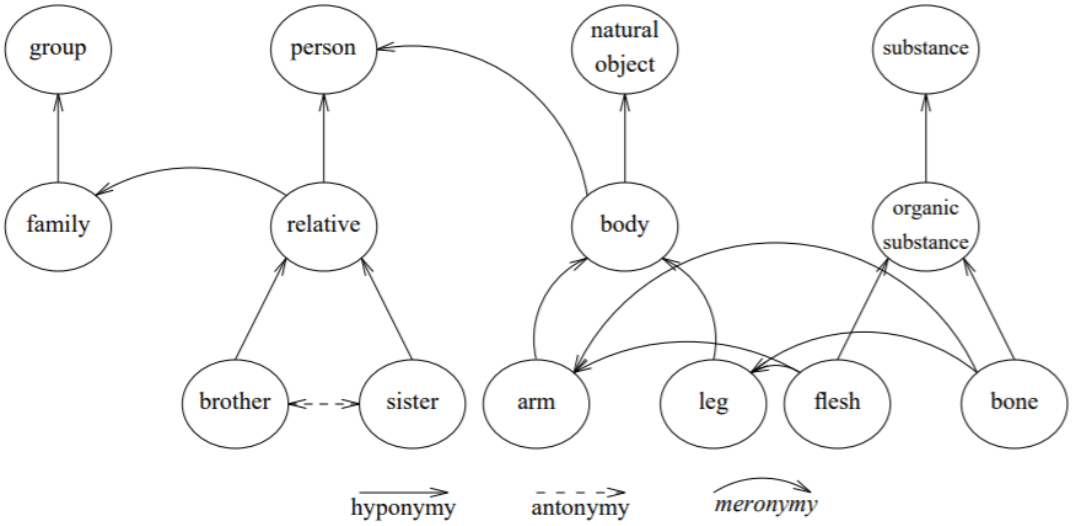


Figure 2.1: WordNet semantic relations with hyponyms, antonymy and meronymy [26].

same pair similarity when path length is equal [24]. With the power of deep learning and language modeling nowadays, we are investigating the combination of WordNet with BERT and see what impact this has on improving performance on natural language understanding. This brings us to the next question to see what the options are for combining the two models.

2.2 WordNet-based embeddings

To make the integration possible, it is essential that knowledge from WordNet is first converted into a representation that is suitable for feeding into a (deep) neural network. We seek the connection at the edge of embeddings.

Path2vec, proposed by Kutuzov et al. [19], is one of the methods that fullfils this task. It re-encodes nouns from WordNet as synset embeddings using dot products between pairs of synset vectors. The approach estimates WordNet similarities, computed by Jiang-Conrath (JCN), LCH, shortest path (ShP) and WuP,

and user defined similarities in SimLex999. For evaluation, Spearman correlation is taken between the estimated score and the ground truth. The highest scores were achieved with ShP, reaching correlations up to 0.952 and 0.512 for WordNet similarities and SimLex999, respectively.

Wnet2vec is another method that is proposed by Saedi et al. [38]. In contrast to path2vec, wnet2vec covered all parts-of-speech from WordNet, i.e. nouns, verbs, adjectives and adverbs. The embeddings are induced using Point-wise Mutual Information (PMI) matrix transformation, L2 normalization, and Principal Component Analysis (PCA) for dimensionality reduction. The method is evaluated against SimLex-999 dataset, achieving a Spearman correlation of 0.50 with a sample containing 60k synsets.

For combining embeddings from different models, we rely on the approach proposed by Ostendorff et al. [31] and VGCN-BERT, proposed by Lu et al. [23]. For a classification task of German books into 8 general categories (Task A) and 343 more detailed categories (Task B), Ostendorff et al. [31] used next to textual content such as book titles and blurbs, also metadata (e.g. ISBN number and publication data) and author information. The content part was fed into BERT, while metadata was represented as vectorized features, and additional author identity information was extracted as author embeddings from Wikidata using PyTorch BigGraph. The authors enriched BERT with the metadata vectors and author embeddings by adding a 2-layer multilayer perceptron (MLP) top-classifier. They achieved a micro-F1 score of 87.20 for Task A and 64.70 for Task B; 0.55 and 4.21 percent points more than BERT-German, where only titles and blurbs were used.

VGCN-BERT, on the other hand, follows a completely different approach. Lu et al. [23] start with the discussion that although BERT alleviates the problem of losing long-range information in Recurrent Neural Networks (RNNs) using self-

attention, it, however, does not completely solve it. They argue – similar to our problem statement in Section 1.2 – that BERT is especially good at absorbing local information as a result of learning embedding representations from context. To provide global information, they construct a vocabulary graph (VG) of word co-occurrences and use a convolutional network (CN) to convolve over the nodes to represent the information by embeddings. They concatenate these embeddings with BERT word embeddings and feed them into BERT starting from the first layer. This way local and global information would interact with each other through all layers, resulting in a co-influence of the output in the final layer. The authors applied the model to classification tasks concerning sentiment analysis (SST-2 & MR), binary single-sentence classification (CoLA) and hate speech (ArangoHate & FountaHate). The following weighted average F1-Scores over 5 runs were presented: 91.93 (SST-2), 86.49 (MR), 83.68 (CoLA), 88.43 (AH) and 81.26 (FH) for VGCN-BERT, vs. 91.49, 86.24, 81.22, 87.99, 80.59 for BERT-only execution, respectively. So with VGCN inclusion the performance is slightly better, but in most cases it is lower than 1 percent point. In future work the authors leave room for using other types of vocabulary graphs, like WordNet, which is well in line with the work we started.

Chapter 3

Method

In this chapter we describe our method. As introduced before, we mainly focus on the integration of BERT and WordNet. However, as we rely on other methods for integration, i.e. VGCN-BERT [23], we take advantage of a third type, text graphs, used by these methods for investigating a wider integration. The relationship between the three different model parts is shown in the context view in Figure 3.1.

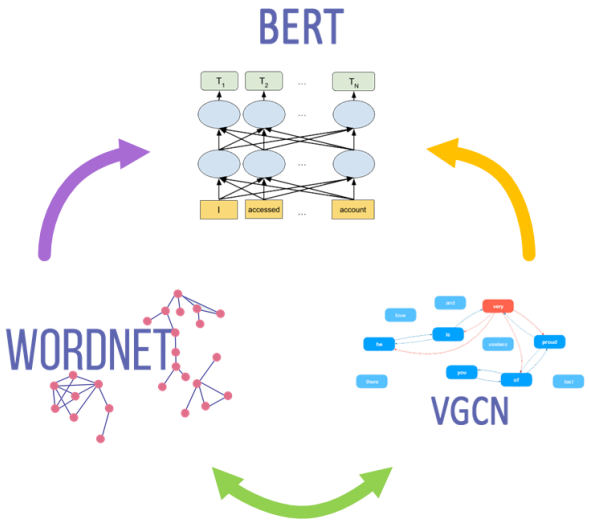


Figure 3.1: Context view of our integrated method where WordNet is combined with BERT and additionally with the text graph of VGCN.

From a language point of view, we determine the strengths and shortcomings of each model based on the type of knowledge that each model covers and the way in which this is incorporated (see Table 3.1). We distinguish between *local* and *global* knowledge. The one is task-related and is extracted from a task-specific dataset. The other is general knowledge about a language, regardless of any task. In both cases learning or modelling this knowledge can be done *implicitly* or *explicitly*.

Typically, BERT is considered to incorporate local knowledge Lu et al. [23]. It is limited to including language patterns in given text. That makes it very good at figuring out the details of a specific task during fine-tuning, but it (partly) lacks the coverage of word concepts at the generic level. This is partly solved by pre-training the model on large amounts of data that is richer in vocabulary, not all of which is necessarily linked to specific tasks. For that reason, we consider the absorption of knowledge during pre-training to be semi-global. The dependence still remains on given text.

WordNet, on the other hand, provides known semantic knowledge about words, regardless of text and context in which they occur. This is so generic that it can be translated to any kind of text. We consider this form as global. That does not apply to VGCN that builds a network of word occurrences in given text. We see that approach as a variant of semi-global models.

Regarding representation and learning, in the implicit case meanings of and

Table 3.1: Knowledge type and knowledge representation of each model.

Model	Knowledge			Representation	
	Global	**Semi-Global**	**Local**	**Implicit**	**Explicit**
BERT		x	x	x	
WordNet	x				x
VGCN Graph		x			x

relationships between words are derived from given unstructured input texts. This has the advantage that knowledge extraction and vocabulary management (i.e. adding, updating and deleting words) are done completely automatically, without human intervention. This is characteristic of BERT's learning process. However, as we have stated in Section 2, we see this also as a limitation where knowledge about words, especially at higher abstraction levels, is incomplete. Think about text in which a dog and a cat are mentioned, but the generalisation that both are animals is not explicitly made. The same applies to a car and a truck where it is not given that they are both vehicles. Obviously, the more training data, the greater the chance that such connections can be made, but overall you cannot say a priori that the connections are complete.

That gap is filled in the explicit representation, where words or text sequences are explicitly constructed based on certain known language rules to humans. Supplementing this knowledge in a deep language model such as BERT, would therefore provide the model with knowledge in advance that could possibly be missing in text. However, this approach can be time-consuming and is not dynamically scalable; any vocabulary management has to be done manually. Eventually, this limits vocabulary coverage.

3.1 From WordNet to Word Embeddings

WordNet in its original form offers a hierarchical network of words and their semantic relationships. In order to transfer the knowledge to BERT, we need to convert WordNet structure to a suitable format for the combination. Since BERT is our base (trainable) deep model, we consider its word representation to be leading. That means that we are looking for a method to convert WordNet structure to BERT word representation, and not the other way around. BERT

represents word tokens by vectors of continuous values between $[0, 1]$, trained by its deep neural network. These learned vector representations are called *word embeddings*. WordNet, on the other hand, represents words in a graph, with nodes containing synonym and sense descriptions about the words, called *synsets*, and edges between these nodes representing semantic relationships. Thus, the question is how can we convert word synsets in WordNet into similar BERT word embeddings that capture WordNet's semantic relationships?

In recent years, a number of works have been published about expressing synsets as embeddings. We consider the two methods *path2vec* and *wnet2vec*.

3.1.1 Path2Vec

Kutuzov et al. [19] proposed path2vec for representing WordNet synsets as node embeddings. The representation is learned as dense vectors based on pairwise synset similarities. Four graph distance measures from WordNet in NLTK and a user defined similarity are used for the ground truth. The four measures are: Leacock- Chodorow (LCH); Jiang-Conrath calculated over the SemCor corpus (JCNS); Wu-Palmer (WuP); and Shortest path (ShP). As of human defined similarities, Sim-Lex999 gold standard was used. Path2vec learns word similarities based on a dot product between pairs of corresponding synset node vectors, such that the value is close to the ground truth. The ShP approach used by path2vec has outperformed other methods like Node2Vec and Deepwalks. Moreover, working with dot products has also contributed to faster training. We use the model published by the authors based on shortest paths[10] – as this showed the best estimations. However, the published model covers only nouns from WordNet of about 82k synsets. According to the author, nouns are better represented than

[10] https://github.com/uhh-lt/path2vec/#pre-trained-models-and-datasets

other parts-of-speech[11].

3.1.2 Wnet2Vec

Saedi et al. [38] presented yet another method for re-enconding WordNet semantincs to word embeddings. They build on the intuition that the more edges between two synset nodes are and the shorter the edges are, the stronger the semantic similarity between two words is. The model constructs an adjacency matrix of words where the cells are set to 1 if there is a an edge between two word nodes, and 0 otherwise. Then the matrix vectors for each word are iteratively adapted until convergence to an inverted matrix, using Positive Point-wise Mutual Information transformation (PMI). The evaluation was done using six testsets, three of which for semantic similarity, i.e. SimLex-999, RG1965, and WordSim-353-Similarity; and three for semantic relatedness, i.e. WordSim-353-Relatedness, MEN, and MTurk-771. In contrast to path2vec, wnet2vec covers all parts-of-speech. However, the published pre-trained model[12] contains only about 60k word embeddings, due to memory limitations for training the entire matrix on all words from WordNet. Initially, we will proceed with this version of the model.

3.2 WordNet Coverage

WordNet 3.0 provides a database with about 150k words. This is a number far below the total number of English words (which is approximated between 600k

[11] Probably for the fact that adverbs and adjectives do not have pairwise synset connections at all. However, training the model on other parts of speach could be done analogously to the approach used for nouns.

[12] https://github.com/nlx-group/WordNetEmbeddings

and 1mil[13]). Oxford Dictionary, for example, contains around 600k word-forms[14]. So there is a good chance that words in the input text will not be covered. This indeed turned out to be the case after doing some experiments with WordNet, where for each word in a given sentence (including function and content words) it was checked whether there is a corresponding synset or not.

When using the initial pre-trained models of path2vec and wnet2vec, i.e. containing each 82k and 60k words respectively, the word count coverage for SST-2 dataset in both cases is around 30% (Figure 3.2). Although path2vec did only cover nouns, while wnet2vec included also verbs, adverbs and adjectives, path2vec found a little more synsets. Probably the size was more determining. Yet there is another difference that path2vec finds the words through lemmas in NLTK's WordNet version[15], while wnet2vec searches for exact words in a txt file. For example, the lemmatization mechanism of NLTK returns the same result of synsets for the words 'bicycle' and 'bicyles', and a search in the wnet2vec file – as it is fixed – does not. Nevertheless, to get a first global coverage w.r.t. the input, we consider all tokens in the first plots.

(a) (b)

Figure 3.2: Path2vec and wnet2vec coverage using initial models.

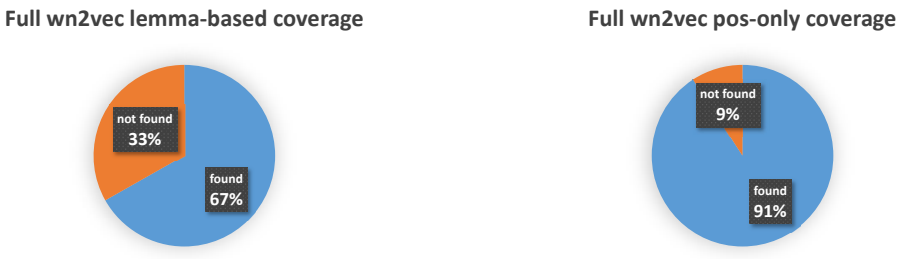
[13] https://englishlive.ef.com/blog/language-lab/many-words-english-language

[14] https://en.wikipedia.org/wiki/Oxford_English_Dictionary#Entries_and_relative_size

[15] See *def synsets()* function in: https://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html

3.2.1 Increasing Coverage

In order to increase the coverage we follow two strategies. First, if a word is not found directly in wnet2vec, we fall back to using lemma embeddings. We first perform a search through `synsets()` function of NLTK's WordNet[16] and, if any result is found, we retry to get the embeddings from wnet2vec. Second, we retrain the model to cover all 150k words from WordNet, instead of the original 60k. As path2vec was already lemma-based, we only retrained the model on entire WordNet. However, the model's vocabulary size only increased to 88k, made up of 75k noun and 13k verb synsets. After further investigation[17] it turned out that a) adjectives and adverbs are not connected in WordNet, b) path2vec sets neighborhood and similarity thresholds to prune the intermediate matrix of synset pairs to reduce time and space complexity, and c) the difference between 82k nouns in the published model and 75k in our trained one is due to the fact that more than 7k noun synsets do not have any neighbors, these are included with their initial (random) weights in the 82k version.



(a) **(b)**

Figure 3.3: Wnet2vec coverage when including lemmas and pre-trained on the entire WordNet **(a)** vs. coverage when only covered parts-of-speech are considered.

[16]NLTK does not explicitly document which lemmatizer is used under the hood. However, there is a *WordNetLemmatizer* which can be found at: https://www.nltk.org/_modules/nltk/stem/wordnet.html.

[17]Including a discussion with the authors: https://github.com/uhh-lt/path2vec/issues/27.

The new coverage became 67% (Figure 3.3 (a)). A good increase of more than 100% when compared to the initial model. When only the supported parts-of-speech by the model are considered, the coverage becomes no less than 90% (Figure 3.3 (b)).

3.3 WordNet-BERT

After obtaining word embeddings from WordNet, the next step is to combine these with BERT. We distinguish two types of ensembles: 1) *'external combination'* and 2) *'internal inclusion'*. The external approach first uses each model independently and then combines the outputs of two or more models in an additional level at the top. As for internal inclusion, as the name suggests, it incorporates a certain structure produced by one model into the internal structure of the other model.

3.3.1 External Combination

We can further subdivide the external ensembles into different types, such as a) combining the embedding vectors of two models, b) combining the embedding of one model with the prediction of the other model, or c) simply stacking the predictions of both models as features in a low-dimentional matrix. Option a) seems promising as the embeddings of both models are kept intact. With c) we have previously experimented in the work [3] about multi-class tweet classification of natural disasters, and overall the ensemble did not outperform a BERT-only model. Regarding b), the values are not proportional to each other, i.e., a relatively high dimensional embedding vector combined with single prediction values, making the integrated model prefer embeddings over prediction features.

To combine embedding vectors on the outside, we rely on the idea of Ostendorff et al. [31]. They have done this by the addition of a top multilayer perceptron

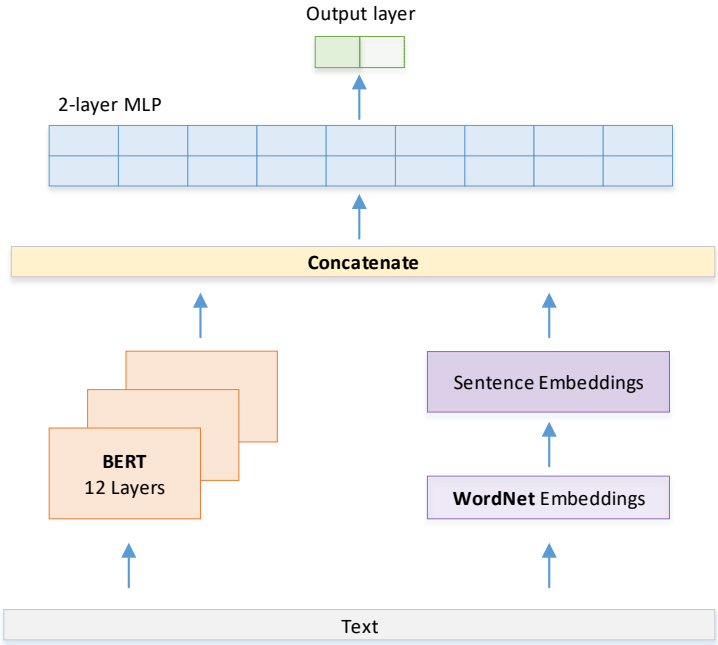


Figure 3.4: The model with external learning of embeddings combination using a 2-layer MLP top classifier.

(MLP) classifier. For a book classification task they have concatenated BERT word embeddings of book texts to author embeddings extracted from knowledge graphs. The MLP would then learn the combination. This is similar to horizontal concatenation in combination with meta-embedding learning from Section 3.3.3. In this case, the concatenation does not suffer from the length variation and length explosion problem, as the additional embeddings are learned at document level with fixed size. However, in our case we are getting word embeddings as input. Therefore we need first to turn these to sentence embeddings.

We adjust the input layer, such that in addition to text input, it gets also the embeddings from WordNet. After the adjustment, the model looks like as shown in Figure 3.4.

3.3.2 Internal Inclusion

Besides the external approach, embedding of a second model can be integrated with a language model by including it in its internal structure. Lu et al. [23] did this in VGCN-BERT by enriching BERT with embeddings of a vocabulary graph starting from the first layer of its network. The benefit this model has over external combination is the utilization of attention heads that BERT has. This allows the included embeddings to influence the attention scores. Important words will then be given higher weight if the added embeddings provide additional information.

In VGCN, the graph is built on word co-occurrences in the entire corpus using normalized point-wise mutual information (NPMI). A convolutional neural network convolves over the graph for learning node embeddings by looking at their neighborhood. When classifying a document, only relevant embeddings to the input text are extracted from the graph. A vocabulary graph would provide additional information about explicit word relationships at corpus-level, similar to WordNet that encodes semantic relationships at language-level.

We therefore use the VGCN-BERT model for projecting embeddings from WordNet into BERT. For this we slightly adjust the model architecture to integrate WordNet embeddings, see Figure 3.5. First, we feed input text with n tokens and get initial word embeddings of m dimensions from BERT and WordNet. Then, we combine both and send the new representation through the network of BERT, starting from the first layer[18].

Embedding combination within BERT. Since BERT works with word embeddings, we must also keep our WordNet embeddings in their original shape. This allows the embeddings to be combined at word level and take advantage of

[18]The 'base-uncased' version with 12 self-attention layers is used.

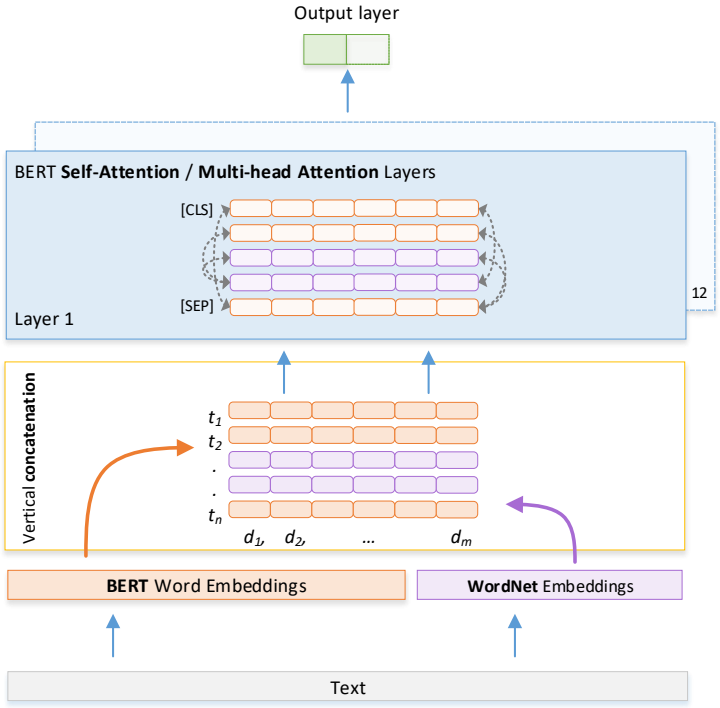


Figure 3.5: Model of internal inclusion of WordNet embeddings in BERT, based on the approach of VGCN-BERT [23].

the attention mechanism in BERT. Since not all words in the sentence will have an embedding in WordNet, options such as vector averaging and summation are not suitable in this case. Instead, we apply vertical concatenation.

For example, consider a sentence like *"This is a nice orange"* for which we want to combine the embeddings for a classification task. BERT first tokenizes the sentence as *[[CLS], this, is, a, nice, orange, [SEP]]*. Next to word split, two [CLS] and [SEP] tags are added. The first indicates the type of classification task we are doing and the second highlights the end of the sentence. The model then calculates the embeddings for the seven individual tokens, e.g. Figure 3.6 **(a)**.

Suppose that for the same sentence WordNet embeddings were found for only the words *'is'* (lemmatized as *'be'*), *'nice'* and *'orange'*. As we follow the strategy of vertical concatenation the three vector embeddings can be simply added at the

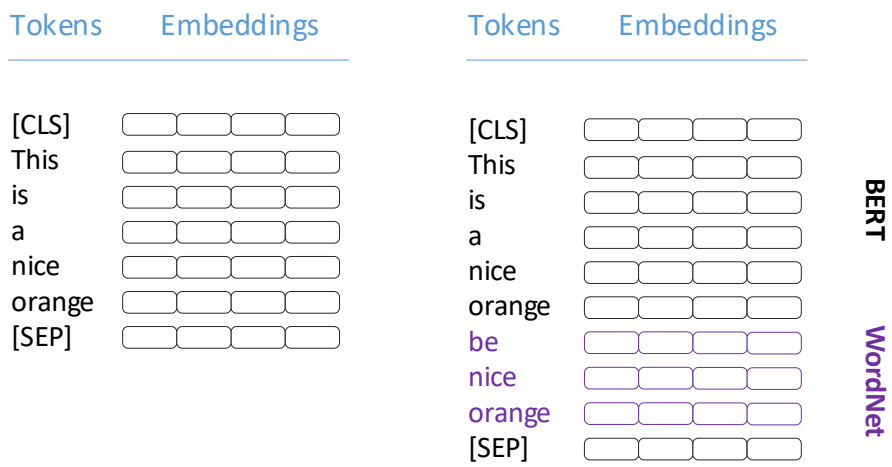


Figure 3.6: BERT word embeddings combined with WordNet embeddings.

tail of word embeddings matrix -1, Figure 3.6 **(b)**[19].

3.3.3 Sentence embedding representation

Both path2vec and wnet2vec provide the embeddings for individual words. In external combination, however, keeping the embeddings in their raw format will lead to a horizontal dimensional explosion. We therefor suggest to apply dimensionality reduction. To stay in line with [31]'s approach, it is best to convert the words to sentence embeddings. There are several options for representative combinations. We consider word vector concatenation, averaging, summation, and meta-learning methods.

1. Concatenation is a naive yet effective approach [7], [16]. It preserves all the information from input word embeddings. This is very useful when passing the embeddings to a next model or when doing analysis at word level. However, the longer the sentences are the bigger the dimensional space becomes, making

[19]The -1 is for shifting the embedding of *'[SEP]'* tag to the end in order to consider the added WordNet embeddings within the scope of the sentence.

training inefficient.

Concatenation can be seen from *horizontal* and *vertical* points of view. In a horizontal representation (Figure 3.7 **(a)**) multiple word embedding vectors are concatenated in a new single vector. This has the advantage of working with just one vector, making it easy to feed them to neural nets. Still this has the downside, next to training inefficiency, as the length can vary between sentences – depending on the number of words in a sentence –, additional customization (e.g. padding) is required to fix the length when the new representation is used as input to a model. Vertical concatenation on the other hand, prevents this issue by managing the word size vertically. The vectors are added consecutively to a multiple stacked vector (Figure 3.7 **(b)**) representation instead. This makes it in addition very suitable for neural nets with fixed input sizes, while allowing sentences of different lengths.

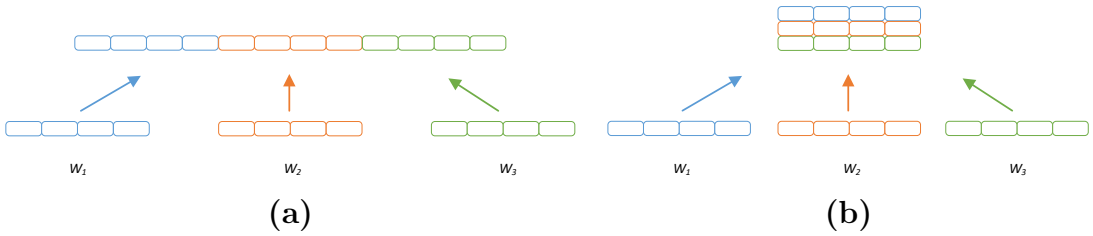


Figure 3.7: Horizontal **(a)** and vertical **(b)** word embedding concatenation.

2. Averaging of input embeddings also proved to be a good representation in various models [7],[15][20]. This maintains the range of word vectors. In addition, the dimensional space does not grow as more words are added, making it, in contrast to concatenation, an efficient strategy. Nevertheless, minor nuances can be lost after averaging.

3. Summation: is yet another naive approach for approximating sentence embeddings. This is similar to averaging, except that it has a different range for

[20]First step of the Deep Averaging Network (DAN).

vector values, as for each word vector from a different model the value in the summed vector becomes larger.

4. Meta-learning

In meta-embedding learning, the task of finding a good and lower-dimensional representation is left to a learner. This is usually a shallow or deep neural net feeding multiple (and possibly different) word embeddings while being trained on a certain task. After finishing, the internal layer weights are considered to be the learned embeddings of the combined words. This approach is mainly used in recent years for learning a combined representation of pre-trained word embeddings produced by different models with different dimensional spaces ([30], [47], [15]).

For external combination we combine averaging with meta-learning, and for internal inclusion we do vertical conactenation.

Chapter 4

Experimental Results

4.1 Datasets

To evaluate the quality of our model, we use relevant datasets from the General Language Understanding Evaluation (GLUE) [45] benchmark. This is a collection of nine datasets covering different NLU tasks. The tasks are divided into three different categories: Single-Sentence Tasks, Similarity and Paraphrase Tasks, and Inference Tasks. In this research we limit us to the first two categories. From each we select a task that is strongly based on semantic understanding in text. These are Sentiment Analysis (SA) from the first category and Sentence Similarity (SS) from the second . The corresponding datasets are SST-2 and STS-B, resprectively.

4.1.1 SST-2

The Stanford Sentiment Treebank 2 (SST-2) [40] is a dataset from Stanford University for binary sentiment classification. The authors considered 10,662 different movie review-sentences to extract partial phrases. Amazon Mechanical Turk was

used for labeling, and a subset of only positive and negative phrases was selected. The original dataset contains 9,613 examples in total, 6,920 of which are for training, 872 for dev, and 1,821 for test. GLUE benchmark provides a much bigger version of the dataset, where training-set is extended to 67,349 instances[21], while dev and test sets are kept the same. We will refer to the original SST-2 as 'SST-2' and to the GLUE version as 'SST-2 (GLUE)'.

4.1.2 STS-B

The Textual Similarity Benchmark (or STS-B) [5] provides a collection with 8,628 pairs of sentences extracted from different text sources, divided into train, test and dev sets of sizes 5,749, 1,500 and 1,379, respectively. The corresponding task is to express the similarity between two sentences on a scale of $[0, 5]$. The models are evaluated using Pearson and Spearman correlations with predefined human scores.

4.2 Experimental Setup

For the experimental setup we basically follow the same hyperparameter and training settings as used in the reference papers, i.e., [23] for *internal inclusion* and [31] for *external combination*, unless there is another more suitable value. See Table 4.1. Since BERT is not deterministic in each run, we perform 5 runs in total to average out and get a better picture of the output. In the external approach, we first fine-tune BERT's model in 3 epochs before combining the embeddings. For the training of the concatenated embeddings, we fix the number of epochs at 20 and enable early stop to use the best model on the dev set for the test. We stop the training if the performance on the dev set does not improve after 3 consecutive

[21] After inspection we found (additional) partial phrases split from original sentences.

Table 4.1: Settings of the experimental setup used for different models and datasets.

Setting	SST-2		SST-2 (GLUE)		STS-B	
	Internal	External	Internal	External	Internal	External
runs	5					
epochs	9	20	3	20	3	20
early stop	False	True	False	True	False	True
batch size	16				12	
optimizer	Adam					
learning rate	$1e-5$				$2e-5$	
$L2$ decay	0.01					
dropout	0.2				0.1	
loss	Cross Entropy				MSE	
output dim	2	1	2	1		
activation	Linear	Sigmoid	Linear	Sigmoid	ReLU	
GPU	Nvidia GeForce GTX 980 Ti / RTX 2080 — (11GB)					

epochs. On everage, 5 epochs were sufficient. In the *internal included* model, we use 3 epochs for SST-2 (GLUE), the same number as used by BERT [8]. The sentence pairs in STS-B led to a high dimensionality, making the 11GB GPUs used ran out of memory. For this we reduced the batch size to 12. Furthermore, we set up this task as regression with 1 output node, using ReLU activation function with MSE loss. Since the scores are between 0 and 5, ReLU ensures that at least the minimum is 0, and max is learned from observed upper bound in the training data.

4.3 Results

In this section we present and describe the results of the proposed models in Chapter 3, following the experimental setup from Section 4.2. The results are given in Table 4.2. We take BERT and VGCN-BERT as baselines to compare

Table 4.2: Results of the different models. SST-2 is reported in F1; the GLUE version in accuracy; and STS-B in Pearson and Spearman correlations. For the baseline, both the published results and the output of our 5 runs are included. The results of the integrated models with WordNet – following *internal inclusion* and *external combination* strategies – follow at the bottom. (Standard deviation between brackets.). * As reported in paper [8]; and ** in paper [23].

	Model	SST-2	SST-2 (GLUE)	STS-B
	Metric	F1	acc.	P/S corr.
Baseline	BERT (ref.) *	-	93.50	- / 85.80
	VGCN-BERT (ref.) **	91.93	-	-
	BERT (own)	**91.56** (0.13)	92.94 (0.35)	**83.66** (0.22) **82.55** (0.28)
	VGCN-BERT (own)	91.33 (0.15)	92.99 (0.19)	83.53 (0.24) 82.32 (0.23)
Internal	P2V-BERT	91.22 (0.32)	92.92 (0.17)	82.98 (0.62) 81.70 (0.69)
	P2V-VGCN-BERT	91.51 (0.29)	92.98 (0.20)	83.19 (0.36) 81.98 (0.39)
	WN2V-BERT	91.36 (0.45)	**93.23** (0.37)	82.91 (0.29) 81.65 (0.40)
	WN2V-VGCN-BERT	91.42 (0.29)	93.10 (0.11)	83.12 (0.33) 81.95 (0.39)
Ext.	P2V-BERT	90.41 (0.29)	92.53 (0.19)	-
	WN2V-BERT	90.34 (0.10)	92.51 (0.10)	-

our own combined models with. Instead of using reported results in reference papers (indicated in the table with (ref.)), we first run an (own) execution of both models with the same experimental setup for a fair comparison. The ensemble models we use of WordNet-BERT are combinations of path2vec (*P2V*) with BERT and VGCN in *internal inclusion*, and wnet2vec (*WN2V*) with BERT in *external combination*.

The scores of our implementation of the baselines are on average slightly lower

than reported in the respective papers. This could be a result of the random behavior of BERT and the average we take over different runs. Of the three datasets, SST-2 (GLUE) is the only one where we achieved an average performance above the baseline, i.e., with a 93.23% accuracy of our *WN2V-BERT* combined model; 0.29 and 0.24 percent points higher than *BERT (own)* and *VGCN-BERT (own)*, respectively. The other results on the two versions of SST-2 datasets are reasonably close to the baselines. On the other hand, the correlations between sentence pair similarities in STS-B are about 3 points lower with internal models than the baselines[22]. For the BERT-only baseline model we did the experiments with the same architecture used for VGCN-BERT, but we only use BERT embeddings. This architecture, with the experimental setup from Section 4.2, would have influenced the lower score, but it has also made the comparison with our WordNet-BERT models more fair.

4.4 Analysis

To gain insight into the behavior of the different models, we investigate how they deal with solving the selected NLU tasks. At metric level, the scores were in most cases not higher with the addition of WordNet embeddings. We therefore look further at whether there are individual differences at the data point level that could be affected by the different types of embedding. In the analysis we continue with the models of *internal inclusion*, as these have given better results.

4.4.1 Multi-head self-attention

One of the challenges of deep neural networks nowadays is that they are a black-box for their workings. Although they are outperforming traditional methods on

[22]We did not apply the external model to STS-B as a MLP is not suitable for regression tasks.

the output, it remains a mystery how the internal structure leads to obtained results. Explainable AI tries to give substance to this by making the abstract parameters, weights and states interpretable.

In recent years more and more research has been devoted to the explainability of BERT's success in NLP. For example, it is examined which language knowledge and patterns the model absorbs, what the impact of the network architecture is on the output, and which cases the model still has difficulty with [37]. In our case, we analyze the multi-head self-attention heads, as these are the core of the Transformers architecture behind BERT.

The $\text{BERT}_{\text{BASE}}$ model we are using has 12 layers, and each layer has 12 attention heads, making up 144 unique heads in total. These all seem to pick up different patterns [6]; either syntactically or semantically. The latter is especially interesting for us, as we also inject external semantic knowledge into the model – from the semantic lexicon of WordNet. However, examining all heads is overwhelming. It also makes it extra difficult because it appears that the type of patterns that are captured cannot be found in one head, but are spread across the entire network [37]. The few studies that looked at this estimate the type of patterns by performing a qualitative analysis of the attentions.

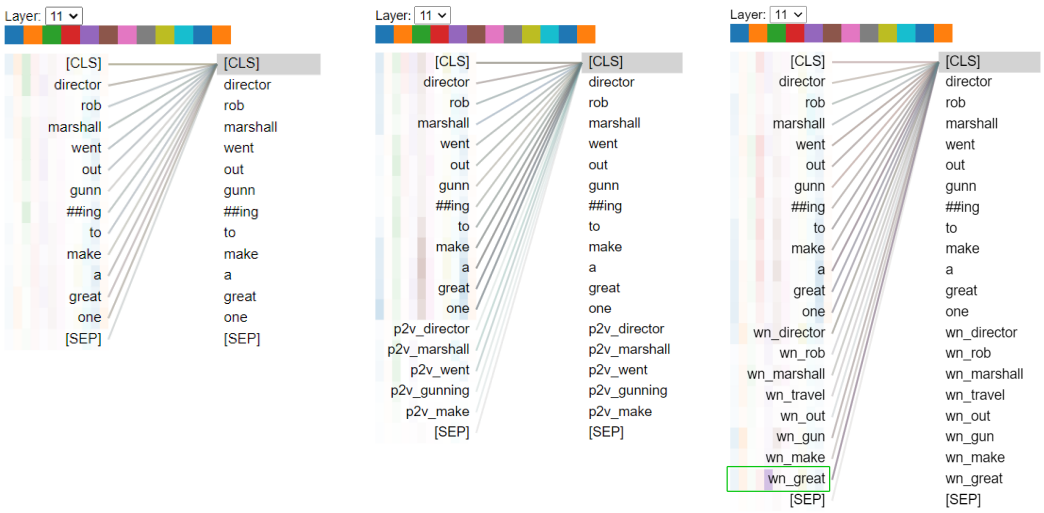
In our case we examine the attention contribution of all tokens to the [CLS] tag in the last layer. The embeddings of this tag are used by BERT for final classification. In this way we implicitly include the contribution of all heads in the previous layers as well, and we target the differences in the [CLS] tag with the addition of WordNet embeddings. The assumption is that any difference between the BERT-only model and the *BERT-WordNet* integrated model is due to the addition of WN embeddings. Given that WordNet is semantically based, we can also consider the differences to be affected by its semantics.

For the analysis we compare the models *BERT*, *WN2V-BERT* and *P2V-*

BERT, as they form a one-to-one comparison between *BERT* and WordNet. We focus on the task of SST-2, as attention between tokens in the same sentence is relevant to sentiment; unlike STS-B where sequences of two different sentence structures are constructed. For this we look at three types of cases. 1) in which *BERT-WordNet* integrated models perform better than BERT-only model; 2) where WordNet is neutral for the output, i.e., giving the same output as BERT; and 3) where WordNet is worse. We visualize the attentions using BertViz [43].

Positive Contribution

In the following example a *positive* output is expected: *"director rob marshall went out gunning to make a great one."*. In the 5 runs of the models, BERT-only model classified the test example as *negative* in 4/5 cases, while WN2V-BERT and P2V-BERT were respectively in 5/5 and in 4/5 cases correct. The attention to [CLS] is shown in Figure 4.1.



(a) **(b)** **(c)**

Figure 4.1: Token attention to the output [CLS] mask in BERT **(a)**, path2vec **(b)** and wnet2vec **(c)** models.

The figures give us an indication of the importance of the tokens for the

Table 4.3: Ranked token attentions for the example *"director rob marshall went out gunning to make a great one."*

Rank	BERT		P2V-BERT		WN2V-BERT	
1	[CLS]	0.0678	one	0.0917	**wn_great**	**0.0866**
2	director	0.0548	[CLS]	0.0912	a	0.0767
3	one	0.0546	a	0.0911	wn_director	0.0665
...	...	...	...	...	...	...
last - 2	make	0.0393	[SEP]	0.0129	wn_rob	0.0262
last - 1	gunn	0.0317	p2v_director	0.0126	wn_out	0.0213
last	[SEP]	0.0267	p2v_make	0.0117	[SEP]	0.0138

Table 4.4: Top tokens for BERT-wn2vec for the examples from Fig. 4.2.

Rank	**(a)**		**(b)**		**(c)**	
1	wn_caricature	2.89	wn_forgive	2.72	[CLS]	2.48
2	by	2.57	[CLS]	2.48	wn_manhattan	2.39
3	[CLS]	2.53	wrote	1.84	you	1.31
...	...	...	...	...	...	...
last - 2	##ica	0.76	be	0.34	wn_judgment	0.11
last - 1	but	0.53	[SEP]	0.14	see	0.11
last	[SEP]	0.17	,	0.13	wn_ben	0.11

[CLS] tag, but it is difficult to determine which ones are the most important. To quantify attention, we sum over all attention weights (of the 12 heads in the last layer) per token, normalize by the number of heads, and rank them by highest value. The top and lowest tokens are given in Table 4.3.

In the given example, token embeddings from wnet2vec (prefixed with 'wn_') are leading, with 'wn_great' having the highest attention score. Eventually, this probably affected [CLS] embeddings to output the positive label.

Neutral Contribution

Following are examples of WordNet top ranked tokens with the most attention contribution, but without any difference in the prediction, as BERT-only model already had predicted the correct labels. A negative sentiment (0) was expected

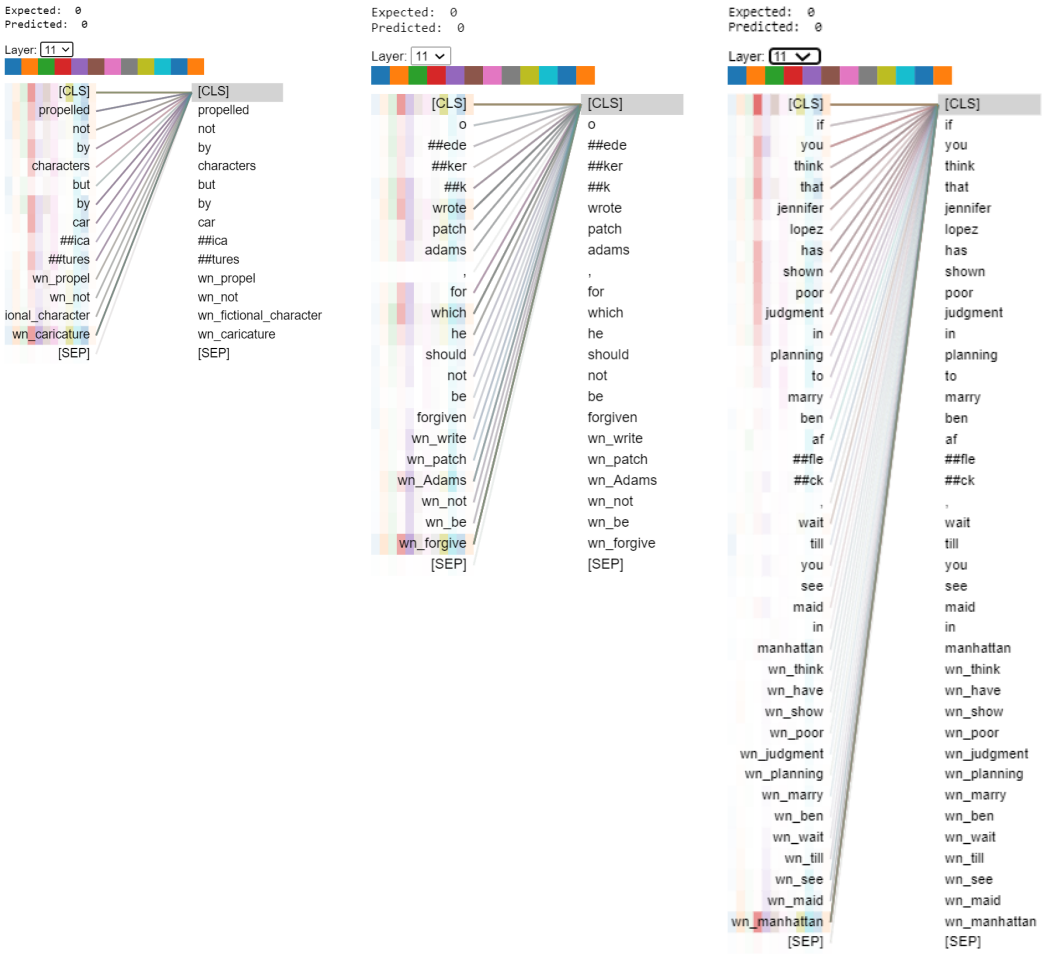
and a negative output was given by the model. Of the dark highlighted tokens in Figure 4.2 we find out in Table 4.4 that 'wn_caricature' and 'wn_forgive' are the highest ranked in the first two sentences, while 'wn_manhatten' is positioned the second in the last one. *Caricature* is usually satire in nature and *forgive* in given context is negative. The last one remains to be disputed; on the one hand Manhattan is a place and on the other it seems to be about the movie of the same name.

Negative Contribution

In the example shown in Figure 4.3 path2vec and wnet2vec had a wrong prediction in comparison to BERT's. The sentence is considered to be negative, while a positive sentiment was assigned by the model. This example is tricky, as it consists of a positive first part and a negative second one. Path2vec with a very low attention contribution only seems to confuse the model here. Wnet2vec on the other hand gives the most attention to 'wn_tuna' (Table 4.5) – which in itself can be neutral or positive –, while it can be inferred from context that this combined with 'canned' gives a sarcastic negative sentiment. The negativity is determined by the word combination of "*canned tuna*". In both models this combination is not highlighted. Wnet2vec gave the most attention to 'wn_tuna', which is a neutral word in itself. So in both cases, WordNet embeddings only confused BERT to deviate from the expectation.

4.4.1.1 Global token attention

To determine the overall attention contribution of all tokens of path2vec and wnet2vec vs. BERT, we calculate this for all tokens in the test set and rank the



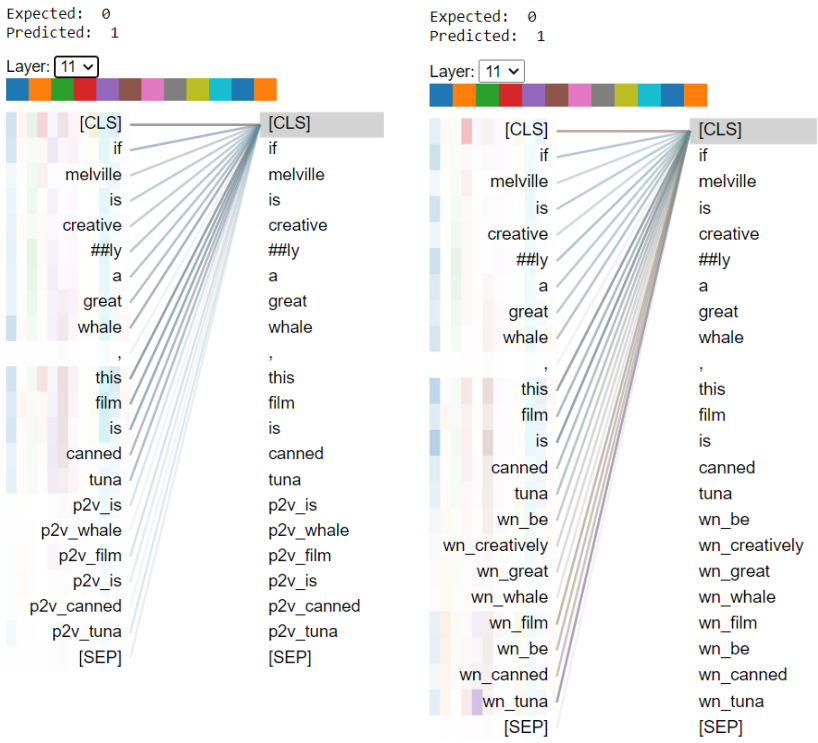
(a) **(b)** **(c)**

Figure 4.2: Wnet2vec top token attention to output [CLS] mask. Caricature **(a)**, forgive **(b)**, manhatten **(c)**.

values in descending order (Eq. 4.1).

$$Att_{cum}(D, H_l, T, A) = \sum_{i=1}^{D} \sum_{h=1}^{H_l} \sum_{t=1,a}^{T_i, A_{T_i}} \frac{a}{|H_l|} \tag{4.1}$$

where Att_{cum} is the cumulative attention, D the testset, i each instance in D, H_l all heads in the last layer l, h each head in H_l, T_i all tokens in i, A_{T_i} all token attentions in i, t each token in i, a each token attention in i, and $|H_l|$ the number



(a) **(b)**

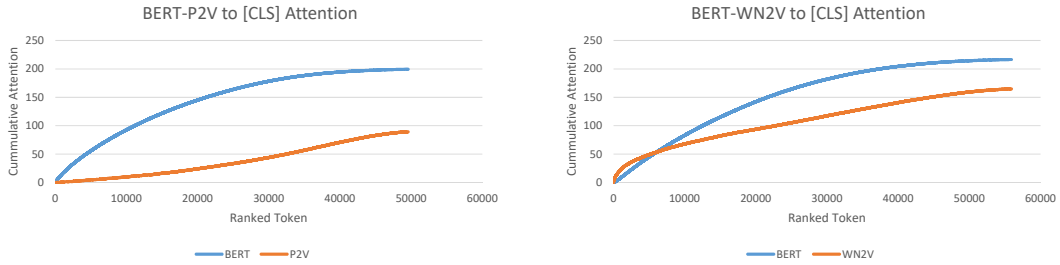
Figure 4.3: Path2vec and wnet2vec [CLS] token attention when a negative output is expected, while a positive one is given.

Table 4.5: Top tokens for the examples from Fig. 4.3 when path2vec and wnet2vec perform worst in P2V-BERT and WN2V-BERT models.

Rank	path2vec		wnet2vec	
1	[CLS]	0.1141	wn_tuna	0.0865
2	this	0.0953	is	0.0834
3	film	0.0874	this	0.0799
...	...	...	...	...
last - 2	,	0.0083	wn_whale	0.0164
last - 1	p2v_canned	0.0069	[SEP]	0.0075
last	p2v_whale	0.0065	,	0.0071

of heads, which is equal to 12,

The ranking is visualized cumulatively in Figure 4.4. From **(b)** it can be clearly seen that the first 500 top tokens of wnet2vec provide more attention to



(a) **(b)**

Figure 4.4: Path2vec **(a)** and wnet2vec **(b)** cumulative attention of ranked tokens to [CLS] token in comparison to BERT's.

Table 4.6: Top ranked tokens across SST-2, using P2V-BERT and WN2V-BERT.

Rank	path2vec		wnet2vec	
	Unique	Avg.	Unique	Avg.
1	ridiculous	propelled	proves	wn_caricature
2	point	dreadful	wn_movie	wn_manhattan
3	extremely	confusing	wn_ridiculous	wn_table
4	a	brit	wn_caricature	wn_infomercial
5	extremely	extremely	wn_idea	wn_elsewhere
6	extremely	moderately	wn_boring	wn_Nash
7	confusing	model	wn_forgive	wn_security
8	dumb	##pha	wn_actress	wn_aberration
9	proves	substitutes	wn_more	wn_unmolested
10	good	##zard	wn_make	wn_bothersome

the output than BERT. Path2vec **(a)**, on the other hand, lags far behind. The latter could be a matter of wrong synset selection[23].

In Table 4.6 we select top 10 tokens for *P2V-BERT* and *WN2V-BERT* models. We do this both for all tokens uniquely, and for tokens with weights averaged over all occurrences, when there are multiple occurrences found of the same token. Here the image is indeed confirmed that wnet2vec tokens get much more attention than the path2vec tokens.

[23]See Discussion in Chapter 5.

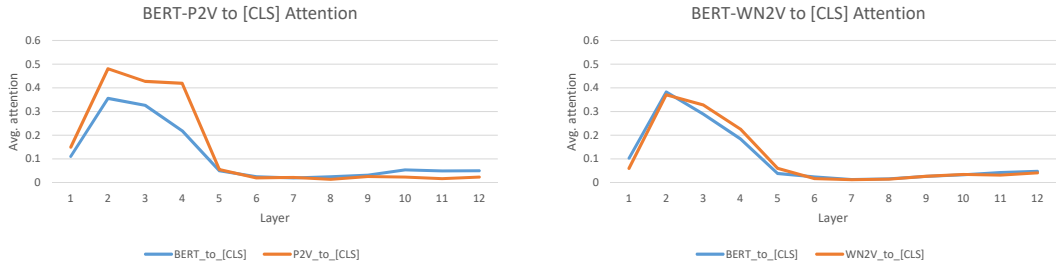
For further analysis, we monitor attention development across all layers of BERT network. For this we reduce the token score per type of model to a single value, normalizing over all tokens and over all attention heads per layer. We also take into account the coverage ratio per model, i.e., the proportion of BERT tokens in % vs. the proportion of WordNet tokens (Eq. 4.2). This is around 0.7 vs. 0.3.

$$Att_l(D, H_l, T, A) = \sum_{i=1}^{D} \sum_{h=1}^{H_l} \sum_{t=1,a}^{T_i, A_{T_i}} \frac{a_i}{|H_l| \cdot |D|} \cdot r \ , \ \text{with } r = \frac{|T_m|}{|T_{BERT}| + |T_{WN}|} \quad (4.2)$$

where Att_l is the normalized attention in layer $l \in L$, with $L = \{1, 2, 3, ..., 12\}$, D the testset, i each instance in D, H_l all heads in layer l, h each head in H_l, T_i all tokens in i, A_{T_i} all token attentions in i, t each token in i, a each token attention in i, $|H_l|$ the number of heads, $|D|$ the size of the testset, and r the token ratio of each model $m \in M$, with $M = \{BERT, WN\}$.

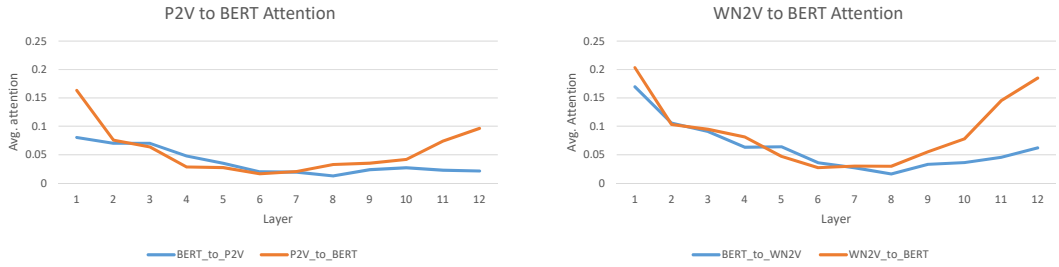
First, we track the contribution of BERT, path2vec and wnet2vec tokens towards [CLS] tag (Figure 4.5). Path2vec has a higher attention up to layer 5, then it goes down and becomes slightly lower than BERT in the last layers. Wnet2vec and BERT are reasonably neck and neck across all layers. For the attention to [CLS], we could consider wnet2vec as more stable than path2vec.

Second, we look at how BERT tokens and those of WordNet models attend towards each other (Figure 4.6). Here we see that path2vec gives relatively more attention to BERT, especially in the beginning, and wnet2vec especially at the end. BERT itself seems to pay relatively less attention to both models. This means that WordNet tokens have more influence on the last attention delivered to [CLS] in the final output than path2vec tokens.



(a) **(b)**

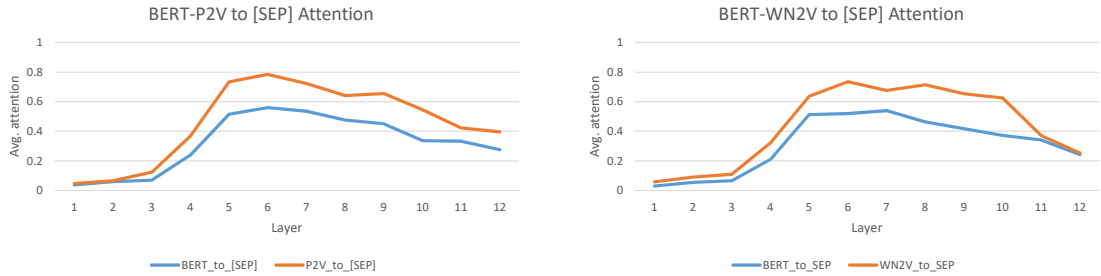
Figure 4.5: Path2vec **(a)** and wnet2vec **(b)** normalized attention to [CLS] across all 12 layers of $\text{BERT}_{\text{BASE}}$, excluding [CLS] and [SEP] from-tokens.



(a) **(b)**

Figure 4.6: Path2vec **(a)** and wnet2vec **(b)** to BERT token attention.

Finally, we also check the attention towards [SEP] tag (Figure 4.7). According to literature [37], this is used as a 'no-opt' in the attention mechanism. That is, if



(a) **(b)**

Figure 4.7: Path2vec **(a)** and wnet2vec **(b)** normalized attention to [SEP] token across all 12 layers, excluding [CLS] and [SEP] from-tokens.

the model does not know where to attend to for a given token from the sentence, it prefers the [SEP] token. Path2vec seems to tend towards this relatively more in the last layer, while wnet2vec's tendency is similar to BERT's.

Chapter 5
Discussion

While we have established a model integration where WordNet affects the functioning of BERT well, the question remains why the presented results in Section 4.3 are not superior to BERT. There could be several reasons for this, but the following are at least known limitations.

With a WordNet coverage of 67% (including lemmas), we keep missing a lot of terms from the input text. Although the assumption is that the terms found are complementary, there are cases where the gaps can lead to completely different meanings, such as missing word negations. However, BERT itself, on the other hand, does have complete word coverage. Theoretically, this leaves the possibility to prevent, or rather alleviate, potential deterioration with the present information.

A more severe case for path2vec is the wrong synset selection for ambiguous words, e.g., for bank, table, note, etc. The model has an F1 of .53 for word sense disambiguation (WSD) [19]. When a synset is selected that, for example, has an opposite sentiment, or changes the sentence construction, it can consistently lead to undesirable results at the end. In addition, although the pruning thresholds set for neighborhood and similarity speed up the training process of path2vec

enormously, they would exclude synsets that do not fall in the thresholds' range. However, Kutuzov et al. [19] have shown that the overall impact is very small. A more concerning aspect is the disconnected synsets, which applies to all adjectives and adverbs and partly to nouns and verbs.

Wnet2vec prevents WSD by expressing all found synsets for a term in one embedding. The advantage is that you have the certainty that the synset of the searched word is in any case represented in the embeddings. However, at the same time the disadvantage is that irrelevant synsets are also included. Overall, this approach appears to perform slightly better than path2vec.

Furthermore, sentiment words are strongly context-based. A word can be positive in one context and negative in another. By design BERT uses context and WordNet does not. However, the idea was that with the *internal inclusion* approach, WN embeddings would become contextualized as well. Given the attention contribution shown, this is very likely, but we only include WN embeddings during fine-tuning, while BERT is pre-trained with BERT tokens only. In a complete scenario, WN embeddings should also be included during pre-training.

Chapter 6
Conclusions

This study is motivated by the idea that semantics are leading in natural language understanding (NLU) and that explicit projecting of this knowledge in state-of-the-art pre-trained language models (PLMs) can potentially improve the models on various NLU tasks. We have used BERT as PLM and WordNet as additional semantic lexicon. After integrating these two, performing experiments, and analyzing the results, we conclude on the research questions with the following.

1. **How to represent semantics from WordNet as embeddings?**

   We have represented semantic relationships in WordNet as synset embeddings using path2vec [19], and as word embeddings using wnet2vec [38].

2. **What is a good way to combine WordNet embeddings with BERT embeddings?**

   We find that *internal inclusion*, inspired by [23] gives better results than *external combination*, inspired by [31]. In the internal approach we project WordNet embeddings (i.e. path2vec or wnet2vec embeddings) into BERT, starting form the first layer, and let them interact with BERT word embeddings through all layers by the self-attention mechanism. The projection is accomplished by *vertical concatenation* of both types of embeddings, i.e.,

adding WordNet embeddings to the tail of BERT word embeddings w.r.t the token order.

3. **Does including explicit semantic knowledge from lexical databases improve the performance of pre-trained language models on NLU tasks?**

   We found that adding explicit semantic knowledge from WordNet does not outperform BERT-only models on sentiment analysis (SST-2) and sentence similarity (STS-B). However, we achieved a slightly better accuracy on SST-2 (GLUE) dataset, around .3 percent point better than the baseline.

4. **How do WordNet embeddings affect BERT model?**

   Analysing multi-head self-attentions of BERT has shown a substantial degree of attention contribution from WordNet embeddings to BERT. Wnet2vec was found to have the most contribution. This means that we were able to influence the model.

5. **What kind of NLU tasks can be solved with WordNet combined with BERT and semantic knowledge bases and what not?**

   At datapoint level we see differences both in attention weights as in output. Although wnet2vec has a great contribution to the attentions, it, however, did not make much difference for the output, as BERT-only model, with an F1 and accuracy > 91.5 for SST-2, already predicts the correct label in most cases. The cases in which WordNet was determining are limited. In these cases embeddings for positive or negative tokens (e.g. 'great') from WordNet get the most attention.

All in all, we can conclude that we have set up a WordNet-BERT integrated model, in which the two influence each other via embeddings and attentions, without compromising performance. This leaves room for further optimization of the embeddings combination in subsequent research.

Future Work

As for future work, there is room for improvement on word sense disambiguation for synset selection in path2vec and on WordNet coverage in general (see Discussion 5). The works of Loureiro and Jorge [22] and AlMousa et al. [1] may be interesting for WSD. Regarding coverage, Ilievski et al. [14] presents a good overview of different semantic (and other) knowledge bases that could be combined for increasing the word coverage.

Furthermore, for the cases where WordNet was of negative influence, we suggest the following idea to experiment with. During the training phase of fine-tuning, if a WN-BERT example does not give the expected output, or it does not improve on BERT-only model, the step for updating network weights has to be skipped. However, since this is done in a batch with multiple examples, it must be considered how the separation can be made.

References

[1] Mohannad AlMousa, Rachid Benlamri, and Richard Khoury. A Novel Word Sense Disambiguation Approach Using WordNet Knowledge Graph. *arXiv preprint arXiv:2101.02875*, 2021. 10, 51

[2] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Ijcai*, volume 3, pages 805–810. Citeseer, 2003. 14

[3] Mohamed Barbouch, Frank W Takes, and Suzan Verberne. Combining Language Models and Network Features for Relevance-Based Tweet Classification. In *International Conference on Social Informatics*, pages 15–27. Springer, 2020. 25

[4] Alexander Budanitsky and Graeme Hirst. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and other lexical resources*, volume 2, pages 2–2, 2001. 14

[5] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi:

10.18653/v1/S17-2001. URL https://www.aclweb.org/anthology/S17-2001. 33

[6] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://www.aclweb.org/anthology/W19-4828. 37

[7] Joshua Coates and Danushka Bollegala. Frustratingly Easy Meta-Embedding – Computing Meta-Embeddings by Averaging Source Word Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2031. URL https://www.aclweb.org/anthology/N18-2031. 29, 30

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423. xi, 4, 8, 9, 13, 34, 35

[9] Allyson Ettinger. What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models. *Transactions of the Association for Computational Linguistics*, 8:34–48, 2020. doi: 10.1162/

tacl_a_00298. URL https://www.aclweb.org/anthology/2020.tacl-1.3. 5

[10] Fernanda Ferreira and Zoe Yang. The problem of comprehension in psycholinguistics. *Discourse Processes*, 56(7):485–495, 2019. 2

[11] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012. 11

[12] Graeme Hirst, David St-Onge, et al. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, 305:305–332, 1998. 14

[13] Jeremy Howard and Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://www.aclweb.org/anthology/P18-1031. 12

[14] Filip Ilievski, Pedro Szekely, Jingwei Cheng, Fu Zhang, and Ehsan Qasemi. Consolidating Commonsense Knowledge. *arXiv preprint arXiv:2006.06114*, 2020. 51

[15] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computa-*

tional linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers), pages 1681–1691, 2015. 30, 31

[16] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. Dynamic Meta-Embeddings for Improved Sentence Representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1176. URL https://www.aclweb.org/anthology/D18-1176. 29

[17] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019. 11

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 11, 13

[19] Andrey Kutuzov, Mohammad Dorgham, Oleksiy Oliynyk, Chris Biemann, and Alexander Panchenko. Learning Graph Embeddings from WordNet-based Similarity Measures. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 125–135, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-1014. URL https://www.aclweb.org/anthology/S19-1014. 8, 9, 15, 21, 47, 48, 49

[20] Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998. 14

[21] Michael Lewis. *The lexical approach*, volume 1. Language teaching publications Hove, 1993. 5

[22] Daniel Loureiro and Alípio Jorge. 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, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1569. URL https://www.aclweb.org/anthology/P19-1569. 51

[23] Zhibin Lu, Pan Du, and Jian-Yun Nie. VGCN-BERT: Augmenting BERT with Graph Embedding for Text Classification. In *European Conference on Information Retrieval*, pages 369–382. Springer, 2020. ix, xi, 8, 9, 16, 18, 19, 27, 28, 33, 35, 49

[24] Lingling Meng, Runqing Huang, and Junzhong Gu. A review of semantic similarity measures in wordnet. *International Journal of Hybrid Information Technology*, 6(1):1–12, 2013. 15

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013. 11

[26] George A Miller. Nouns in WordNet: a lexical inheritance system. *International journal of Lexicography*, 3(4):245–264, 1990. ix, 15

[27] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. 13

[28] George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and

Katherine J Miller. Introduction to WordNet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990. 5, 8, 9

[29] Olga Moudraia. Lexical Approach to Second Language Teaching. ERIC Digest. 2001. 5

[30] James O' Neill and Danushka Bollegala. Angular-based word meta-embedding learning. *arXiv preprint arXiv:1808.04334*, 2018. 31

[31] Malte Ostendorff, Peter Bourgonje, Maria Berger, Julian Moreno-Schneider, Georg Rehm, and Bela Gipp. Enriching BERT with Knowledge Graph Embeddings for Document Classification. *arXiv preprint arXiv:1909.08402*, 2019. 9, 16, 25, 29, 33, 49

[32] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *International conference on intelligent text processing and computational linguistics*, pages 241–257. Springer, 2003. 14

[33] Ted Pedersen, Siddharth Patwardhan, Jason Michelizzi, et al. WordNet:: Similarity-Measuring the Relatedness of Concepts. In *AAAI*, volume 4, pages 25–29, 2004. 14

[34] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 11

[35] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North Ameri-*

can Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202. 12

[36] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018. 12

[37] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*, 2020. 5, 13, 37, 45

[38] Chakaveh Saedi, António Branco, João Rodrigues, and Joao Silva. Wordnet embeddings. In *Proceedings of the third workshop on representation learning for NLP*, pages 122–131, 2018. 8, 9, 16, 22, 49

[39] Brent Smith and Greg Linden. Two decades of recommender systems at Amazon. com. *Ieee internet computing*, 21(3):12–18, 2017. 4

[40] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013. 32

[41] Amirsina Torfi, Rouzbeh A Shirvani, Yaser Keneshloo, Nader Tavvaf, and Edward A Fox. Natural Language Processing Advancements By Deep Learning: A Survey. *arXiv preprint arXiv:2003.01200*, 2020. 1

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you

need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4, 12

[43] Jesse Vig. A Multiscale Visualization of Attention in the Transformer Model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3007. URL https://www.aclweb.org/anthology/P19-3007. 38

[44] Vernor Vinge. Technological singularity. In *VISION-21 Symposium sponsored by NASA Lewis Research Center and the Ohio Aerospace Institute*, pages 30–31, 1993. 2

[45] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://www.aclweb.org/anthology/W18-5446. 3, 9, 32

[46] Zhibiao Wu and Martha Palmer. Verbs Semantics and Lexical Selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, page 133–138, USA, 1994. Association for Computational Linguistics. doi: 10.3115/981732.981751. URL https://doi.org/10.3115/981732.981751. 14

[47] Wenpeng Yin and Hinrich Schütze. Learning meta-embeddings by using ensembles of embedding sets. *arXiv preprint arXiv:1508.04257*, 2015. 31

[48] Wenbin Zhang and Steven Skiena. Trading strategies to exploit blog and news sentiment. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 4, 2010. 3