

https://dspace.org/

Knowledge-driven language modelling for text embedding and semantic similarity

Bhana, Nimesh

2023
https://hdl.handle.net/10539/35731

Downloaded from a DSpace repository. https://dspace.org/

RESEARCH REPORT

Knowledge-Driven Language Modelling for Text Embedding and Semantic Similarity

Author:
Nimesh BHANA (2371061)

Supervisor:
Prof. Terence VAN ZYL



UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

submitted to
the Faculty of Science, in fulfilment of the requirements for the degree of
Master of Science

School of Computer Science and Applied Mathematics
University of Witwatersrand

March 14, 2023

Declaration of Authorship

I, Nimesh BHANA (2371061), declare that this research report titled, "Knowledge-Driven Language Modelling for Text Embedding and Semantic Similarity" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this research report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this research report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the research report is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 15/03/2023

UNIVERSITY OF WITWATERSRAND

Abstract

Faculty of Science
School of Computer Science and Applied Mathematics

Master of Science

Knowledge-Driven Language Modelling for Text Embedding and Semantic Similarity

by Nimesh BHANA (2371061)

Language Models such as BERT have grown in popularity due to their ability to be pre-trained and perform robustly on a wide range of Natural Language Processing tasks. Often seen as an evolution over traditional word embedding techniques, they are capable of producing representations of text, useful for tasks such as semantic similarity. However, state-of-the-art models often have high computational requirements and lack global context or domain knowledge which is required for complete language understanding. To address these limitations, an investigation of the benefits of knowledge incorporation into the fine-tuning stages of BERT is done. An existing K-BERT model, which enriches sentences with triples from a Knowledge Graph, is adapted for the English language and extended to inject contextually relevant information into sentences. Given the appropriate knowledge, Knowledge-enabled BERT (K-BERT) outperforms similar models, USE & SBERT, suited for text embedding and semantic similarity. Performance is based on the STS-B and ag_news_subset datasets. Knowledge ablation studies conducted indicate that injected knowledge causes noise. When this noise is minimised, we see statistically significant performance improvements for knowledge-driven tasks. Results show evidence that, given the appropriate task, modest injection, with relevant, high quality knowledge is most performant. However, achieving successful integration autonomously is non-trivial.

Acknowledgements

A sincere thank you to my project supervisor, Prof. Terence van Zyl. Your guidance and support motivated me and allowed me to persevere through times of distress and dejection. I respect your strong work ethic and high standards as it has driven me to produce the best work I'm capable of. My accomplishments would have not been possible without you.

To the people who've supported me the most in my life, my parents. Thank you for the sacrifices and effort you make to ensure of my success in academia. I will forever be grateful for the privileges I have been blessed with.

This research required the use of machines with significant computational power. I would like to acknowledge the Mathematical Sciences Support unit at the University of Witwatersrand for providing the High Performance Computing infrastructure I needed.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1

2 Background 3
2.1 Knowledge Graphs . . . 3
2.2 Embedding Techniques . . . 5
2.2.1 One Hot Encoding . . . 5
2.2.2 Term Frequency-Inverse Document Frequency (TF-IDF) . . . 5
2.2.3 Word2Vec . . . 7
2.2.4 BERT . . . 9
Transformer . . . 9
BERT Architecture . . . 13
2.3 K-BERT . . . 14
2.3.1 Knowledge Layer . . . 14
2.3.2 Embedding Layer . . . 14

3 Related Work 16
3.1 Introduction . . . 16
3.2 String-based Methods . . . 16
3.3 Knowledge-based Methods . . . 16
3.3.1 Edge-based Methods . . . 17
3.3.2 Feature-based Methods . . . 17
3.4 Information-based Methods . . . 17
3.5 Corpus-based Methods . . . 18
3.6 Deep Neural Network-based Methods . . . 20
3.7 Hybrid Methods . . . 22
3.8 Conclusion . . . 25

4 Wikidata Knowledge Graph 27
4.1 Introduction . . . 27
4.2 Data Structure . . . 27
4.3 Motivation . . . 27
4.4 Data Retrieval . . . 27
4.4.1 Data Processing . . . 29
4.4.2 Data Storage . . . 29

5 Methodology **30**
5.1 Architecture Overview . . . 30
5.2 English Knowledge Graph . . . 30
5.3 Term-based Sentence Tree . . . 30
5.4 Contextualised Knowledge Injection . . . 32
5.5 Corrections & Optimisations . . . 32
5.5.1 Sequence Truncation . . . 32
5.5.2 Memory Usage . . . 33

6 Experiments **34**
6.1 Baselines . . . 35
6.2 Fine-tuning . . . 35
6.2.1 STS-B . . . 35
6.2.2 Ag_news_subset . . . 36
6.3 Experiment Outline . . . 36
6.3.1 Experiments . . . 37
Manual Knowledge Injection . . . 38
6.4 Research Limitations . . . 38

7 Results **39**
7.1 Baseline Comparison . . . 39
7.2 Comparison between BERT and K-BERT . . . 39

8 Analysis **41**
8.1 STS-B Analysis . . . 41
8.1.1 STS-B Sentence Pair Ablation Study . . . 41
8.2 AG News Subset Analysis . . . 42
8.3 Discussion . . . 44

9 Conclusion **46**

A Appendix A **48**

List of Figures

2.1 Knowledge Graph example . . . 4
2.2 Facts in triple notation . . . 4
2.3 One hot encoding vectors . . . 6
2.4 The Neural Probabilistic Language Model . . . 8
2.5 CBOW and Skip-gram model . . . 9
2.6 Skipgram illustration . . . 10
2.7 Skipgram example . . . 11
2.8 Transformer architecture . . . 12
2.9 BERT input representation . . . 14
2.10 K-BERT sentence tree . . . 14
2.11 K-BERT visible matrix . . . 15

4.1 Wikidata data item example . . . 28
4.2 Wikidata data items for "race" . . . 28

5.1 K-BERT architecture . . . 31
5.2 Comparison of truncation methods where $max_length = 16$. The words highlighted in red are truncated, while the green words are kept. . . . 33

6.1 Baseline regression architectures . . . 36
6.2 Baseline classification architectures . . . 36

8.1 STS-B bias analysis . . . 43

List of Tables

2.1 One hot encoding example 5
2.2 TF-IDF example 6

3.1 Embedding algorithm comparison 19
3.2 STS-B Literature Model Performance 23

6.1 Examples from STS-B dataset 34
6.2 Examples from ag_news_subset dataset 34
6.3 Evaluation Datasets 37
6.4 Hyperparameters 37

7.1 Overall results 39
7.2 Knowledge Ablation results on STS-B 40
7.3 Knowledge Ablation results on ag_news_subset 40

8.1 MSE Loss & Correlation coefficient t-test 42
8.2 STS-B Ablation results 42
8.3 Accuracy t-test 43

A.1 Pre-trained models used 48

List of Abbreviations

NLP	**N**atural **L**anguage **P**rocessing
LSA	**L**atent **S**emantic **A**nalysis
KB	**K**nowledge **B**ase
KG	**K**nowledge **G**raph
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
CBOW	**C**ontinuous **B**ag **O**f **W**ords
LSTM	**L**ong **S**hort-**T**erm **M**emory
RNN	**R**ecurrent **N**eural **N**etworks
LM	**L**anguage **M**odel
DNN	**D**eep **N**eural **N**etwork
NLPM	**N**eural **P**robabilistic **L**anguage **M**odel
YAGO	**Y**et **A**nother **G**reat **O**ntology
NELL	**N**ever-**E**nding **L**anguage **L**earning
NMT	**N**eural **M**achine **T**ranslation

Chapter 1

Introduction

The digital divide has been waning with every passing year. Five billion people now have access to the internet in some capacity [Union 2021], leading to huge throughput of information sharing between people. While access to zettabytes of digital data is beneficial, it makes finding pertinent information arduous. Search techniques have fortunately been assisting with this problem for decades. Furthermore, in situations that require providing suggestions to the user, we have Recommender Systems [Bobadilla *et al.* 2013] which have been very beneficial in commercial applications.

However, such systems typically rely on user data to make optimal recommendations. In cases where users feel uncomfortable providing such data, these systems must rely on approaches that strictly consider item similarity. For instance, when viewing an "office chair" on an e-commerce website such as Amazon, related items such as "desks" or other "office chairs" will also be suggested for users to view. Recommendations make the process of finding items of interest as seamless as possible for the user. While item similarity has been implemented successfully for various types of content, one major challenge is applying this to textual data [Nguyen *et al.* 2019].

Measuring the degree of similarity between pairs of text is an open research problem in the field of Natural Language Processing (NLP). Since natural language is both versatile and ambiguous, it is difficult to define rule-based methods for measuring the semantic similarity between pairs of text [Marsi and Krahmer 2013; Marneffe and Manning 2008; Schuster and Manning 2016]. For example, the sentences "The bank increased its interest rate" and "The boat that passed the river bank, piqued his interest" both contain the same words "bank" and "interest", however, their meanings vary greatly. The converse is also true, i.e. words may not be syntactically identical but may be synonyms. Disambiguation of these meanings requires knowledge about the language and, possibly, the domain to which the concepts belong - e.g. finance, medicine, etc.

Knowledge sources such as ontologies, thesauri, dictionaries, and knowledge bases contain general information about the world. These sources of knowledge are useful for uncovering facts and providing additional context to fill the gaps in one's understanding. While this idea has been extended to machines [Rodriguez and Egenhofer 2003; Majumder *et al.* 2016], it requires feature engineering and causes a tight coupling between the NLP system and a given knowledge source. Since the structure of various knowledge sources is inconsistent with each other, major changes are necessary within the NLP system to accommodate a new knowledge source. Consequentially, the changes cause performance inconsistencies within the NLP system. A different approach to computing similarity between pairs of text would be to create a representation of the text which incorporates its meaning. Neural networks are a popular architecture to produce these representations of text.

Advances in computational power and accessibility thereof, have catalysed a move toward more complex neural architectures. An abundance of free text has allowed these architectures to learn in a self-supervised manner. Language Modelling exploits this capability by capturing statistical patterns and nuances over a large corpus of text. This information is represented as a series of real-valued numbers corresponding to a word. One such pattern could recognise that "bank" occurs frequently with terms like "money" and "interest". Its representation is therefore affected by the words that surround it, also known as its context. These representations are known as contextualised embeddings and have been used to produce state-of-the-art (SOTA) results over a range of NLP tasks [Song *et al.* 2020; Devlin *et al.* 2019; Cer *et al.* 2018].

While there are various approaches explored to improve the quality of embeddings, a consensus is that model size and performance are linearly related [Brown *et al.* 2020]. Most state-of-the-art models have grown from millions to billions of parameters. As this number continues to grow, computational requirements become exorbitant for the average person as well as contributes to the massive energy crisis [Yang *et al.* 2017]. It is therefore important to make models as efficient as possible while maintaining maximal performance.

One method to achieve efficiency is to leverage existing knowledge to complement the understanding of the Language Model. Leveraging existing knowledge is beneficial, especially for tasks that require domain-specific knowledge such as medicine [Ormerod *et al.* 2021]. Knowledge sources are an ideal repository of universal information, containing both general and domain-specific knowledge that can be exploited by the Language Model. There have been various attempted hybridisations, one of which, is K-BERT [Liu *et al.* 2020].

The K-BERT model by Liu *et al.* [2020] combines the language representation model BERT with a Knowledge Graph. This work extends K-BERT by adapting it to the English language. I explore the possibility and related benefits/detriments of incorporating information from a Knowledge Graph into the fine-tuning stages of the Language Modelling process. The aim is to provide additional contextual information, to represent text in greater fidelity, ultimately leading to more accurate text comparisons. Additionally, I propose and test the effectiveness of a simple contextualised mechanism to query entities from the Knowledge Graph. The new query mechanism forms an extension of the existing K_{Query} by Liu *et al.* [2020].

Experimental results indicate that injected knowledge may introduce noise which hurts performance. When noise is minimised and injected in the appropriate context, we observe a 0.7% statistically significant performance improvement (p-value < 0.001) over the BERT baseline for the appropriate knowledge-driven task.

The main contributions of this research are:

- A K-BERT model adapted for the English language
- Integration of the model with a new Knowledge Graph
- A modified K_{Query} to inject contextually relevant information

The rest of this report will first provide some background information about Knowledge Graphs and text embedding techniques in Chapter 2. Related work is discussed and critiqued in Chapter 3. Details of a new Knowledge Graph, data retrieval, and pre-processing are explained in Chapter 4. Changes and enhancements to the K-BERT model are outlined in Chapter 5. Experiment setup, results and analysis are shown in Chapter 6, 7 and 8 respectively. Lastly, Chapter 9 contains conclusions of the current study and avenues for future work.

Chapter 2

Background

2.1 Knowledge Graphs

A Knowledge Graph is a structured representation of knowledge about the real world [Yan *et al.* 2016]. It contains information about entities (e.g. organizations, movies, people, locations, fictional characters) and relations between them (e.g. birthplace, director). Additional information such as entity properties, labels, or values is also included. Figure 2.1 shows a simple Knowledge Graph with entities as nodes and relationships represented as directed edges between nodes.

The term Knowledge Graph is synonymous with Knowledge Base [Frost 1986]. Most work refers to them interchangeably. The primary difference is that Knowledge Graphs must be able to be viewed as a graph. Because Knowledge Bases contain factual information in a triple format (subject, predicate, object), under the resource description framework (RDF) [Swick and Lassila 1999], they can be visualised as a graph in most cases.

Since facts are written in propositional logic [Ji *et al.* 2022] inference is also possible through the use of Knowledge Bases. A simple example is shown in Figure 2.2. We can infer that BARACK OBAMA was born in the country UNITED STATES since the city of HONOLULU is LOCATEDIN UNITED STATES. Even though this appears to be a weak inference, real-world Knowledge Bases may contain thousands of properties for a single entity, leading to more accurate and specific inferences.

For example, entity UNITED STATES may have property INSTANCEOF COUNTRY and/or HAWAII INSTANCEOF U.S. STATE. These inherent properties and features illustrate that Knowledge Bases can be thought of as intelligent databases to manage, store, and query information. As such, many existing Knowledge Bases are provided in a format which supports queries through a query-language such as SPARQL [Prud'hommeaux and Seaborne 2008]. Using these query-languages, we can answer questions such as "How many states does United States have?" or, a more complex question, "Which U.S. presidents have had parents who were also politicians?".

Knowledge Bases are useful in a wide range of applications such as search engines [Noy *et al.* 2019; Sinha *et al.* 2015], question-answering, fact-checking, and chatbots, among other tasks. Notable Knowledge Bases include: YAGO [Pellissier Tanon *et al.* 2020], DBPedia [Auer *et al.* 2007], BabelNet [Navigli and Ponzetto 2012], NELL [Carlson *et al.* 2010], KnowItAll [Etzioni *et al.* 2004], Wikidata [Vrandečić and Krötzsch 2014].

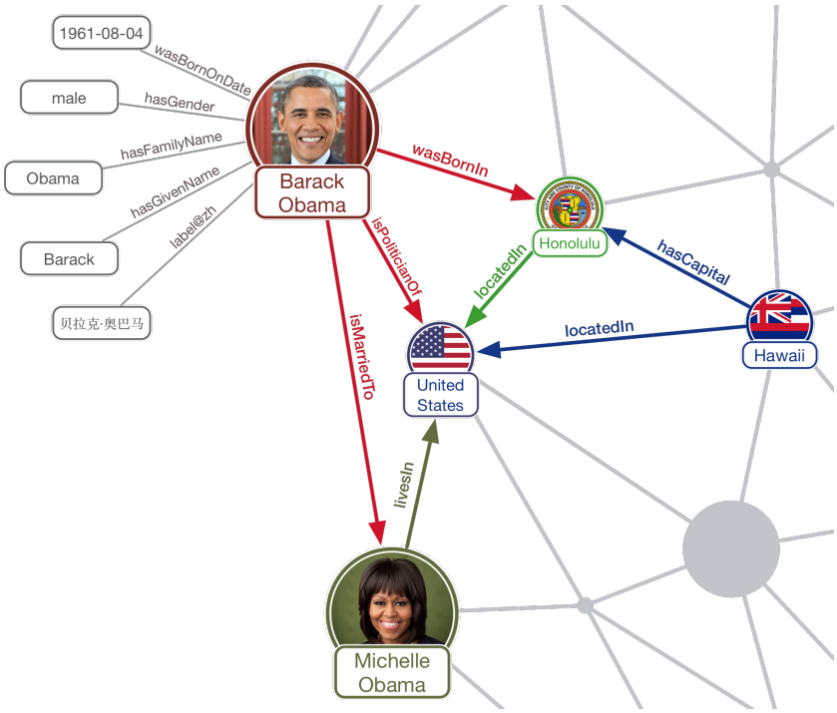


FIGURE 2.1: Example of Knowledge Graph from [Zhou *et al.* 2019] showing entity nodes in circles, with labels directly underneath. Directed edges represent relationships between entities. Each entity may have one or more properties associated with it, such as gender or given name if the entity is a person.

(Barack Obama, **wasBornOnDate**, 1961-08-04)
(Barack Obama, **hasGender**, male)
(Barack Obama, **hasFamilyName**, Obama)
(Barack Obama, **hasGivenName**, Barack)
(Barack Obama, **wasBornIn**, Honolulu)
(Barack Obama, **isPoliticianOf**, United States)
(Barack Obama, **isMarriedTo**, Michelle Obama)
(Michelle Obama, **livesIn**, United States)
(Honolulu, **locatedIn**, United States)
(Hawaii, **hasCapital**, Honolulu)
(Hawaii, **locatedIn**, United States)

FIGURE 2.2: Facts of Knowledge Graph in Figure 2.1 written in triple notation.

2.2 Embedding Techniques

For computers to work with text data efficiently, the text must first be converted into numbers. The representation of this text by a group of real-valued numbers, also known as a vector, is referred to as an embedding. There have been several approaches proposed to produce embeddings. From basic techniques such as One Hot Encoding to Transformer-based contextual embeddings. Details of these approaches are outlined in the following subsections.

2.2.1 One Hot Encoding

One of the earlier techniques, One Hot Encoding assigns a vector to each distinct item in a corpus [Berry *et al.* 1998]. Hence, if given a document of words, each distinct word can be represented by a different vector. A simple approach to achieve this representation is to assign an index to every unique word. Vectors filled with 0's will be created with a dimension equal to the vocabulary of words. A value of 1 will be inserted whenever a word belongs to its index. An example of this approach is shown in Table 2.1. Each word in the sentence will then have the following resulting vectors as illustrated in Figure 2.3.

s	hot	in	**road**	**shimmered**	**the**	**wide**	**sun**
the	0	0	0	0	1	0	0
wide	0	0	0	0	0	1	0
road	0	0	1	0	0	0	0
shimmered	0	0	0	1	0	0	0
in	0	1	0	0	0	0	0
the	0	0	0	0	1	0	0
hot	1	0	0	0	0	0	0
sun	0	0	0	0	0	0	1

TABLE 2.1: Table of one hot encoding vectors for every word in sentence s (along the first column). Unique, case-insensitive words of the sentence "The wide road shimmered in the hot sun" is shown as column headers, ordered in alphabetical order. For each word $w \in s$, a (1) is placed whenever w matches a column header and (0) elsewhere.

2.2.2 Term Frequency-Inverse Document Frequency (TF-IDF)

A drawback of One Hot Encoding is that the vectors of the words do not convey any significant information. In a sentence, certain words are more significant than others and it would be beneficial to represent that distinction. Instead of assigning a value of 1 as done in One Hot Encoding, we can assign a different number, or weight, to indicate the importance of a word in the current sentence. This weight can be attained using Term Frequency-Inverse Document Frequency (TF-IDF) [Ramos *et al.* 2003].

TF-IDF is a statistical measure to determine the relevance of word w in a document d with respect to a text corpus D. The frequency of occurrences of a word is

```
[[0, 0, 0, 0, 1, 0, 0], the
 [0, 0, 0, 0, 0, 1, 0], wide
 [0, 0, 1, 0, 0, 0, 0], road
 [0, 0, 0, 1, 0, 0, 0], shimmered
 [0, 1, 0, 0, 0, 0, 0], in
 [0, 0, 0, 0, 1, 0, 0], the
 [1, 0, 0, 0, 0, 0, 0], hot
 [0, 0, 0, 0, 0, 0, 1]] sun
```

FIGURE 2.3: One hot encoding vectors for sentence "The wide road shimmered in the hot sun". These vectors can be found in the columns of Table 2.1

used to determine its importance. Words such as "of", "as", "and", "the", "in" occur frequently throughout the corpus D and are considered insignificant. Conversely, words that occur infrequently throughout corpus D but are unique to document d indicate high relevance to d. Therefore, the insignificant words are assigned a lower weighting than rare words [Berger *et al.* 2000; Salton and Buckley 1988]. An example of the calculation is shown in Table 2.2, where weight of a word per document w_d is:

$$w_d = f_{w,d} \times \log\left(\frac{|D|}{f_{w,D}}\right), \tag{2.1}$$

where $f_{w,d}$ equal to the number of occurrences of w in d and $f_{w,D}$ equal to the number of documents containing w.

The TF-IDF columns in Table 2.2 can be used as vectors to represent each document $d \in D$.

words	TF(w, A)	TF(w, B)	IDF(w, D)	TFIDF(A)	TFIDF(B)
blue	0	1/7	$\ln(2/1) = 0.69$	0	0.099
fish	0	1/7	$\ln(2/1) = 0.69$	0	0.099
hot	1/8	0	$\ln(2/1) = 0.69$	0.086	0
in	1/8	1/7	$\ln(2/2) = 0.00$	0	0
road	1/8	0	$\ln(2/1) = 0.69$	0.086	0
shimmered	1/8	0	$\ln(2/1) = 0.69$	0.086	0
sun	1/8	0	$\ln(2/1) = 0.69$	0.086	0
swam	0	1/7	$\ln(2/1) = 0.69$	0	0.099
tank	0	1/7	$\ln(2/1) = 0.69$	0	0.099
the	2/8	1/7	$\ln(2/2) = 0.00$	0	0
two	0	1/7	$\ln(2/1) = 0.69$	0	0.099
wide	1/8	0	$\ln(2/1) = 0.69$	0.086	0

TABLE 2.2: TF-IDF calculations for corpus $D = [A, B]$, with A = "The wide road shimmered in the hot sun" and B = "Two blue fish swam in the tank".

2.2.3 Word2Vec

While TF-IDF is suitable for simple applications in information retrieval or keyword extraction, two major deficiencies exist. That being, the resultant vectors are sparse, i.e. vectors contain a large number of zeros that occupy an unnecessary amount of space. Secondly, they convey no meaning of the word they represent, i.e. similar words such as "dog" and "puppy" can have completely dissimilar vector representations. To overcome these shortcomings, models which can "learn" a word embedding from a corpus have been proposed.

Progress in machine learning techniques has allowed for more complex models to be developed, such as Neural Probabilistic Language Model (NPLM) by Bengio *et al.* [2003]. NPLM learns a probability function for word sequences in a language using a neural network. That is, the probability of the next word given the previous words:

$$\hat{P}(w_1^T) = \prod_{t=1}^{T} \hat{P}(w_t|w_1^{t-1}) \,, \tag{2.2}$$

where w_t is the t-th word, and sub-sequence $w_i^j = (w_i, w_{i+1}, \ldots, w_{j-1}, w_j)$.

Then for a Neural Model, having the training set being a sequence of words $w_1 \ldots w_T$ for $w_t \in V$, where V is a vocabulary of a large but finite set. The objective of the neural network is to learn a model

$$f(w_t, \ldots, w_{t-n+1}) = \hat{P}(w_t|w_i^{t-1}) \,,$$

where n represents the number of consecutive words to combine, or *n-grams*. Each element i of V is mapped to a real vector $C(i) \in R^m$, where C is a mapping function and m is the number of features. The resulting mapped elements represent *distributed feature vectors* which are learnt **simultaneously** with the probability function. The initialisation of the vectors is either done according to a particular distribution (Normal distribution), using prior knowledge of semantic features, or randomly initialised. Vectors can be of fixed size m, which is significantly smaller than the size of the vocabulary. This approach alleviates data sparsity issues present in the One Hot Encoding and TF-IDF methods by producing dense vectors with a multitude of features.

The final function f of the neural architecture as shown in Figure 2.4 then becomes:

$$f(i, w_{t-1}, \ldots, w_{t-n+1}) = g(i, C(w_{t-1}), \ldots, C(w_{t-n+1})) \,, \tag{2.3}$$

where g is the neural network with parameters ω. Overall parameter set is $\theta = (C, \omega)$.

Training maximises log-likelihood as shown in Equation 2.4

$$L = \frac{1}{T} \sum \log f(w_t, w_{t-1}, \ldots, w_{t-n+1}; \theta) + R(\theta) \,, \tag{2.4}$$

where $R(\theta)$ is a regularization parameter.

Since the word feature vectors are learnt over a corpus, words such as "dog" and "puppy" will have similar feature vectors because the probability function is a smooth function over these feature values. A small change in a feature will induce a small change in the probability. The production of feature vectors can be extended to not only words but to combinations of words or sentences.

Extending on NLPM's approach to learning distributed representations of words, Mikolov *et al.* [2013a] proposed word2vec, a novel technique to learn high-quality

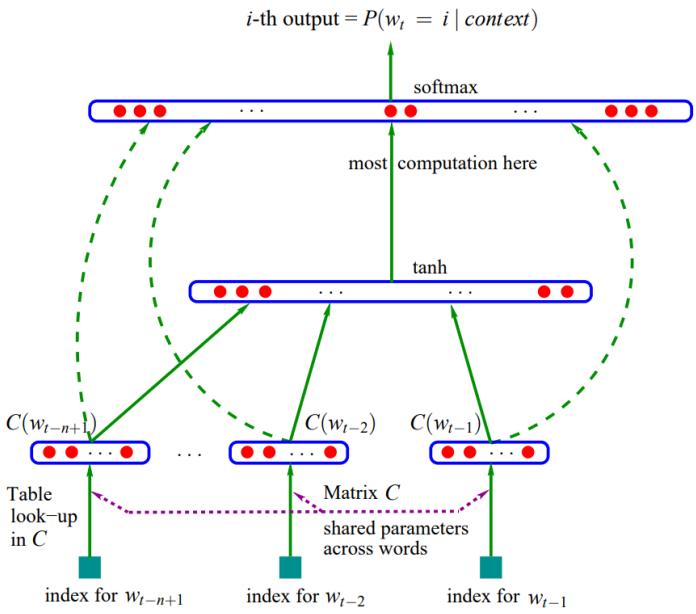


FIGURE 2.4: Bengio *et al.* [2003] neural architecture f depicted by Equation 2.3, where g is the neural network.

word embeddings from large corpora (billions of words). A pivotal contribution of word2vec was that the embeddings produced after initial training could be successfully applied to a range of NLP applications. Since these embeddings are pre-trained, their development is a form of Transfer Learning, which offers significant improvements over training from scratch such as improvements in generalisation accuracy [Turian *et al.* 2010].

Two variants of word2vec exist - Continuous Bag of Words (CBOW) and Skip-gram. In CBOW, the neural network tries to predict the current word given the context words that surround it. Skip-gram performs the opposite by predicting the closely related context words given the current word. These architectures are shown in Figure 2.5.

Figure 2.6 illustrates an example text and the skip-grams that are desired to be attained. Window size m defines the context size, meaning the model will look at positions $t - m$ to $t + m$ as the context, with the centre word at position t. Assume, given centre word w_t, context words are generated independently from each other, then, probability formula for the co-occurrence of the context words in the example text with $w_t = \textit{shimmered}$ and window size $m = 2$ is shown in Figure 2.7. The likelihood is generalised by Equation 2.5.

$$L(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} | w_t; \theta)\,, \tag{2.5}$$

where θ represents the parameters of the model and w_t is a word at position t.

In summary, the dataset for training the model is attained from a large corpus of text (such as Wikipedia). Using a sliding window of maximum size $2m + 1$, training samples are generated to train the model in a self-supervised manner. The training methodology has allowed word2vec to produce high-quality embeddings by reducing model complexity and increasing the volume of training data. It has also been found that performing certain simple algebraic operations such as

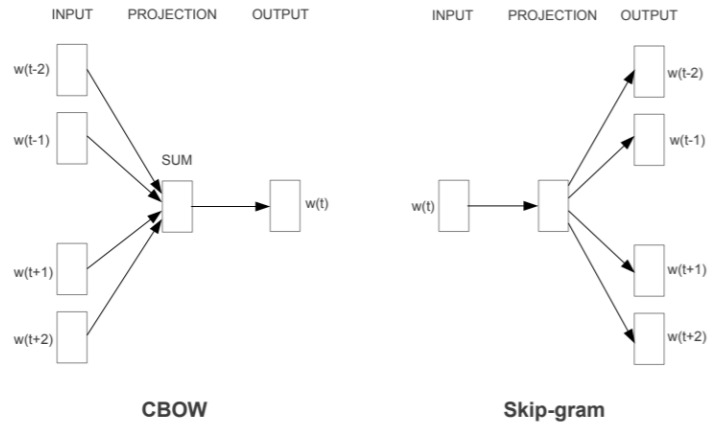


FIGURE 2.5: CBOW and Skip-gram models [Mikolov *et al.* 2013a]. CBOW predicts the current word based on the context. Skip-gram predicts the context words given the current word.

VECTOR("KING") − VECTOR("MAN") ≈ VECTOR("QUEEN") is linearly consistent with regularities among words [Mikolov *et al.* 2013b].

In recent years, there has been a transition from the way NLP models are built. While models have become more complex and incorporate deep learning, most have moved away from simply using pre-trained word embeddings as the input layer of a model. Instead, the focus has been placed on initialising an entire model with pre-trained weights. With the ever-expanding resources of free text [Merity *et al.* 2016; Raffel *et al.* 2020a; El-Khair 2016], pre-training entire models have become more viable and practical due to the easily accessible digitised literature and world knowledge. Pre-trained Language Models have now exhibited state-of-the-art results in a range of NLP tasks. One of the most influential models is BERT.

2.2.4 BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers (BERT) is a language representation model introduced by researchers at Google in 2018 [Devlin *et al.* 2019]. Like most language models, it is pre-trained on a huge corpus of unlabelled text (BooksCorpus [Zhu *et al.* 2015] and Wikipedia) on different pre-training tasks. After pre-training, the model can be further fine-tuned for a broad range of NLP downstream tasks such as Question Answering, Sequence Classification, and Sentiment Analysis. BERT utilises the Transformer architecture [Vaswani *et al.* 2017] which was initially designed for Machine Translation.

Transformer

A machine translation problem can be represented as a statistical model:

$$P(x|e) = \prod_{i=1}^{m} P(x_i|x_{1:i-1}, e)\,, \tag{2.6}$$

which estimates the probability P of a target sentence $\mathbf{x}$ with m words, being translated to a source sentence e, where x_i represents target word at position i [Kalchbrenner and Blunsom 2013]. Neural Machine Translation fits a neural model to maximise the conditional probability of sentence pairs using a parallel training corpus [Bojar

Window Size	Text	Skip-grams
2	[The **wide** road shimmered] in the hot sun.	wide, the wide, road wide, shimmered
	The [wide road **shimmered** in the] hot sun.	shimmered, wide shimmered, road shimmered, in shimmered, the
	The wide road shimmered in [the hot **sun**].	sun, the sun, hot
3	[The **wide** road shimmered in] the hot sun.	wide, the wide, road wide, shimmered wide, in
	[The wide road **shimmered** in the hot] sun.	shimmered, the shimmered, wide shimmered, road shimmered, in shimmered, the shimmered, hot
	The wide road shimmered [in the hot **sun**].	sun, in sun, the sun, hot

FIGURE 2.6: Skip-grams and window sizes of example text: The wide road shimmered in the hot sun [Har 1969; Abadi *et al.* 2015].

et al. 2014; Alotaibi 2017], i.e. a collection of texts which is translated into one or more different languages.

Since translation involves converting a sequence of words into another sequence of words, sequence-to-sequence models such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) have successfully been applied in solving the Machine Translation problem [Sutskever *et al.* 2014; Cho *et al.* 2014].

The translation process has been subdivided into two separate components: an encoding phase and a decoding phase. The encoding phase takes an input sequence of vectors $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ (representing words in the sentence) and generates vector c from a series of hidden states h_t using RNN f:

$$h_t = f(x_t, h_{t-1}), \tag{2.7}$$

$$c = q(h_1, \ldots, h_m), \tag{2.8}$$

with q being non-linear functions, which in general is simply $q(h_1, \ldots, h_m) = h_m$.

The decoder is trained to predict the target word $x_{\hat{i}}$, given context vector c and the previously predicted words $x_1, \ldots, x_{\hat{i}-1}$:

$$P(x) = \prod_{t=i}^{m} P(x_i | x_{1:i-1}, c), \tag{2.9}$$

where $P(x_i | x_{1:i-1}, c)$ is being predicated by RNN g with hidden state s_i as

$$P(x_i | x_{1:i-1}, c) = g(x_{i-1}, s_i, c). \tag{2.10}$$

Instead of incorporating all contextual information into a single context vector c, a pivotal "attention" mechanism was proposed [Bahdanau *et al.* 2015; Luong *et al.* 2015]. The intention behind the attention mechanism is to distribute the contextual

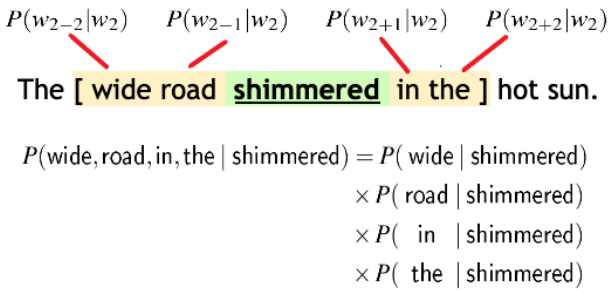


FIGURE 2.7: Probability formula for the co-occurrence of all the words in the context (wide, road, in, the) of the example text, given the centre word w_2 = shimmered and window size $m = 2$

information throughout the sequence of hidden states instead of enforcing all the information into a single fixed-sized vector c. Non-linear function g is now described by:

$$g(x_{i-1}, s_i, c_i)\,, \tag{2.11}$$

with RNN hidden state s_i computed as:

$$s_i = f(s_{i-1}, x_{i-1}, c_i)\,. \tag{2.12}$$

A distinct context vector c_i is generated for each target word x_i. Generation is done similarly as before - by using the sequence of hidden states or annotations h_i. Each hidden stater h_i contains information about the entire sentence, with a stronger focus on the words surrounding the i-th word. Each context vector c_i is calculated as a weighted sum of h_i:

$$c_i = \sum_{j=1}^{m} \alpha_{ij} h_j\,. \tag{2.13}$$

Weight a_{ij} is computed according to:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{m} \exp e_{ik}}\,, \tag{2.14}$$

where

$$e_{ij} = a(s_{i-1}, h_j)\,, \tag{2.15}$$

is an alignment model which scores how well the inputs around position j and outputs around position i match. The alignment model a is parameterised as a feed-forward neural model which is jointly trained with the entire transition model. Intuitively, this mechanism allows the decoder to decide on which parts of a sentence to pay attention to.

The concept of attention forms the foundation of the Transformer model. Being initially designed for the task of Machine Translation, the Transformer also utilises an encoder-decoder structure. However, unlike the traditional sequence-to-sequence models, Transformers attempts to exploit the power of parallelisation to be more computationally efficient. Instead of the additive attention function in Equation 2.14, the attention function by Vaswani *et al.* [2017] is multiplicative. The weights are produced through matrix multiplication, which is the typical operation Graphical Processing Units (GPUs) are optimal for.

The overall Transformer architecture is shown in Figure 2.8. It comprises of $N =$

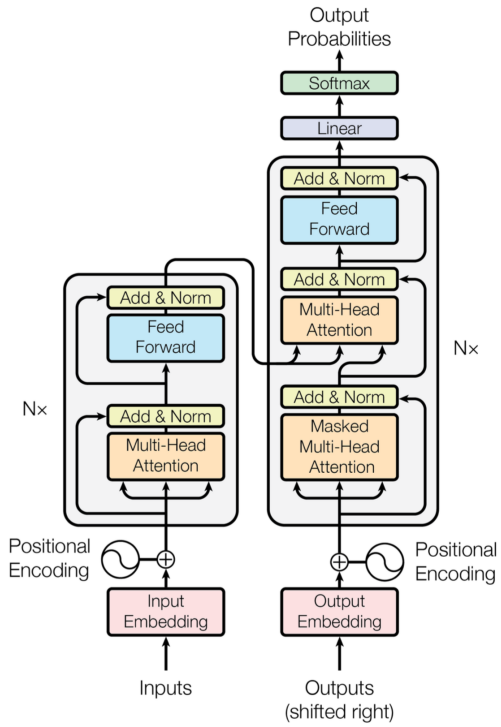


FIGURE 2.8: Transformer architecture from [Vaswani *et al.* 2017]

6 encoder layers stacked together, as shown in the left component, and similarly, the decoder on the right comprises $N = 6$ identical layers. As done in other Neural Machine Translation models, it transforms one sequence into another sequence. The input sequence is firstly segmented into tokens (which can be thought of as words in most cases [Gage 1994; Schuster and Nakajima 2012]). Each of these tokens will have a learnt embedding of size d_{model} associated with it. These input embeddings are fed into the encoder and are added with a positional encoding. Equation 2.16 and 2.17 respectively show the sinusoidal and cosine functions used by Vaswani *et al.* [2017] to produce the positional encodings.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}}), \tag{2.16}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}}), \tag{2.17}$$

where pos is the position and i is the dimension. That is, for every token in position pos, generate a position vector of dimension d_{model}.

Positional encoding is necessary because the processing of tokens is now done in parallel, instead of sequentially. Positional information has been lost and it's required to inject back this positional information by means of a positional encoding.

The resultant embeddings are fed into an attention block which performs a scaled dot product:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.18}$$

where Q, K, V are matrices corresponding to Queries, Keys, and Values. Queries represent the target vectors, while Keys and Values represent the source vectors.

Q, K, V are attained by performing matrix multiplication between the matrix of input embeddings $\mathbf{X}$ (from either target or source) and weight matrix $\mathbf{W}$. For example, Value matrix is attained by $V = \mathbf{X} \cdot \mathbf{W^V}$ where $\mathbf{W^V}$ is of dimension $d_v \times d_v$.

The Query and Key matrices are produced in the same manner but with their corresponding weight matrices of dimension $d_k \times d_k$. The $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$ corresponds to the "weight" component of the attention mechanism. Similarly, the α component in the attention mechanism in Equation 2.13 corresponds to the "weight" as well.

The final dot product between the weight component and Value matrix V will be passed through a fully connected feed-forward neural network. The output of the network will be passed on to the next encoder block where the process repeats. The scaling by $\frac{1}{\sqrt{d_k}}$ is done for numeric stability for the gradient computation.

If the target and source sequence which is fed into the attention block is the same, the type of attention is said to be self-attention [Cheng *et al.* 2016; Lin *et al.* 2017]. Using self-attention, we can focus on important parts of a sentence. However, for most sentences, there are likely multiple important aspects of a sentence that requires attention. Thus, instead of saturating a single attention function with all the information, we employ multiple attention *heads* to spread the load. This type of attention is Multi-Head Attention and is simply a concatenation of multiple attention heads h:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_\text{h})W^O\,, \tag{2.19}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)\,. \tag{2.20}$$

After receiving the output from the last layer of encoders, the process of transforming the contextualised embeddings into a different sequence using the decoder component is done. However, this study is only concerned with the encoding process to produce contextualised embeddings. It is through the use of multiple Transformer encoder blocks that BERT can produce contextualised embeddings.

BERT Architecture

$\text{BERT}_{\text{BASE}}$, the base version of BERT, stacks $L = 12$ Transformer encoder layers together. The number of self-attention heads A is 12 and hidden size H of the feed-forward layer is 768. These parameters indicate that each input token will be of dimension H and passed through L layers of transformer encoder blocks. The multi-head attention block in each encoder will consist of A heads.

BERT is pre-trained on BooksCorpus (800M words) [Zhu *et al.* 2015] and English Wikipedia (2,500M words). The objectives used during pre-training are Masked-Language-Modelling (MLM) and Next-Sentence-Prediction (NSP). In MLM, a percentage of the input tokens are masked and the model is tasked to predict those masked tokens. Given two sentences A and B, NSP is concerned with correctly predicting whether B is the next sentence of A. The pre-trained BERT model can be fine-tuned for a wide range of downstream tasks such as Question Answering, Natural Language Inference, Sentiment Analysis etc. [Devlin *et al.* 2019].

Based on the WordPiece sub-word tokenization algorithm [Schuster and Nakajima 2012], a vocabulary for BERT is constructed. The use of the WordPiece algorithm indicates that BERT works on a sub-word level and therefore segments text into sub-words before retrieving their relevant embedding. For example, the word "playing" would be split into two separate tokens "play" and "##ing". The leading "##" is an indication that the characters that follow form part of the previous token, i.e. they join to "play". BERT's input representation is illustrated in Figure 2.9.

Tokens [CLS] and [SEP] are special tokens that must also be included in the input. The final hidden state of the special classification token [CLS] is used as an aggregate sentence representation for classification tasks. [SEP] indicates the end of

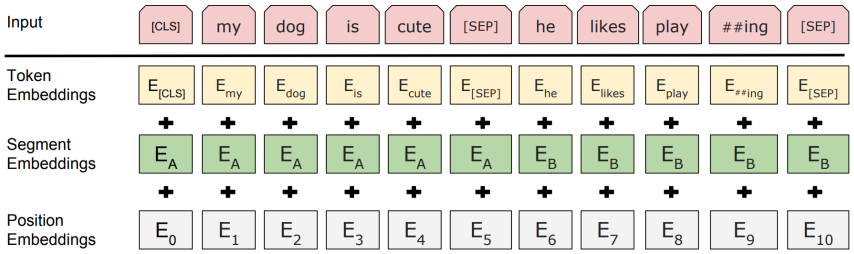


FIGURE 2.9: BERT input representation [Devlin *et al.* 2019]

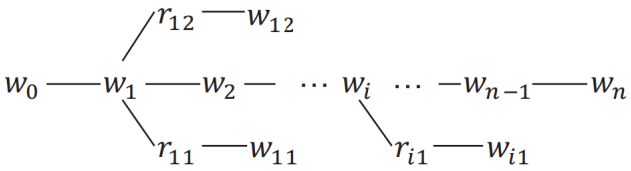


FIGURE 2.10: Sentence Tree structure [Liu *et al.* 2020]

the sentence. Since BERT permits a text pair input, for tasks such as Question Answering, a segment embedding is required to indicate to which sentence each token is associated with. Lastly, since blocks of Transformers are used in BERT, a positional embedding is required to indicate token order. The final input representation is the sum of the token, segment, and positional embeddings.

2.3 K-BERT

K-BERT is a knowledge-enabled language representation model based on BERT [Liu *et al.* 2020]. It infuses triples from a Knowledge Graph into sentences for additional domain knowledge. The knowledge-enhanced sentences are then fed into a modified BERT model for improved performance on downstream tasks, as a result of the additional knowledge.

The major change in the K-BERT architecture compared to the original BERT is the manipulation of the input. This change must be accommodated by the rest of the model pipeline and necessitates additional internal changes within BERT. All such changes are discussed in the following subsections.

2.3.1 Knowledge Layer

The purpose of the Knowledge Layer is to infuse sentences with triples $(w_i, r_i, w_k) \in K$, where w_i is an entity, r_i is relationship/property and K is a Knowledge Graph. Output of the Knowledge Layer is a knowledge-rich sentence tree. The structure is illustrated in Figure 2.10.

2.3.2 Embedding Layer

Conversion of the sentence tree to an embedding is almost identical to BERT. However, positional information is handled differently due to the non-sequential structure of the sentence tree. Branches must first be unrolled into a sequential form. However, while performing the unrolling, it is important to ensure that the structure and meaning of the original sentence remain intact. When knowledge is inserted for

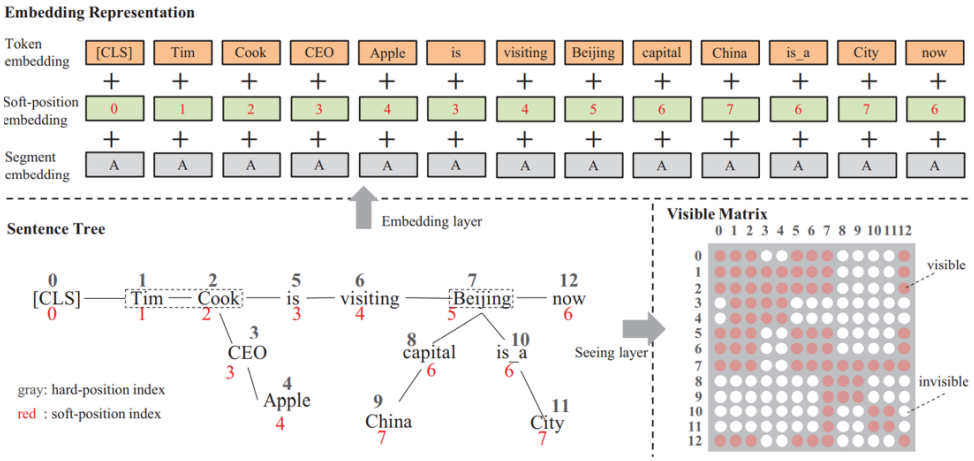


FIGURE 2.11: The processing of converting a sentence tree to an embedding representation with a visible matrix [Liu *et al.* 2020].

a particular token, it should not affect the representations of the tokens surrounding it. Conversely, the representation of the [CLS] token should acknowledge the semantic changes induced by the injected knowledge on the original tokens. These acknowledgements extend to any attention score calculation in the Transformer architecture.

To accommodate the aforementioned requirements, a visible matrix M is introduced. The visible matrix contains binary values indicating which tokens may "see" each other. Hence, for the sentence tree in Figure 2.11 [Apple] modifies [Cook], but should not affect [Beijing]. Also, [CLS] can only see tokens from the original sentence. We can see how the visible matrix indicates the visibility for every token in the input.

Lastly, the self-attention mechanism in the Transformer must be adapted to only attend to regions specified by M. An extension of self-attention called mask-self-attention is then given by:

$$Q^{i+1},\ K^{i+1},\ V^{i+1} = h^i W_q,\ h^i W_k,\ h^i W_v\,, \tag{2.21}$$

$$S^{i+1} = softmax\left(\frac{Q^{i+1}{K^{i+1}}^T + M}{\sqrt{d_k}}\right), \tag{2.22}$$

$$h^{i+1} = S^{i+1} V^{i+1}\,, \tag{2.23}$$

where, W_q, W_k, W_v are trainable model parameters and h_i is the hidden state of the i-th mask-self-attention blocks.

The rest of the fine-tuning process remains the same as BERT.

Chapter 3

Related Work

3.1 Introduction

Measuring the degree of similarity between pairs of text is an open research problem in the field of Natural Language Processing (NLP). Since natural language is so versatile, it is difficult to define rule-based methods for measuring how semantically equivalent pairs of text are [Marsi and Krahmer 2013; Marneffe and Manning 2008; Schuster and Manning 2016]. For example, although the sentences "The boat that passed by the river bank piqued his interest" and "The money she had deposited in the bank had earned interest" contain similar words such as "bank" and "interest", they convey very different meanings. Similarly, the following two sentences, "Sarah does not eat meat, only seafood" and "Sarah is a pescatarian" contain different words, but convey the same meaning.

Techniques have been proposed over the years to solve semantic textual similarity and several related NLP problems. Ranging from simple edge-counting approaches in Section 3.3 to complex Deep Neural Network-based methods with billions of parameters in Section 3.6. Lastly, hybrid methods exploiting the advantages of two or more types of techniques are mentioned in Section 3.7.

3.2 String-based Methods

The simplest techniques of similarity matching consist of techniques such as Longest Common Substring, Levenshtein, and Jaro distance [Weiner 1973; Levenshtein 1965; Jaro 1989]. Working on a character-level, these techniques attempt to align strings or find the longest contiguous sequence of characters that match. Some incorporate transpositions, insertions, and deletions into their approach. Although useful for tasks such as spelling correction or searching, it is not entirely suitable for *semantic* similarity, which requires looking beyond the character-level and uncovering the meaning behind the text itself.

3.3 Knowledge-based Methods

Knowledge-based semantic similarity utilises information attained from one or more knowledge sources to measure the similarity between two terms. These knowledge sources are typically ontologies, lexical databases, dictionaries, thesauri, knowledge bases, etc [Chandrasekaran and Mago 2021]. Knowledge-based methods exploit the fact that the content within these knowledge sources is semantically linked and ambiguity free.

3.3.1 Edge-based Methods

A taxonomically linked ontology of words can be thought of as a graph. Edge-based methods exploit these linkages by simply counting the edges between two terms as a measure of how similar they are. The similarity between the terms is therefore inversely proportional to the path length [Rada *et al.* 1989]. Wu and Palmer [1994] additionally consider the relative depth in the taxonomy as an important parameter. Since greater depth will further specialisation, terms of the same depth further up the taxonomy (more generic) should be given a lower similarity score than those of the same depth at a lower level of the taxonomy. The similarity is then calculated using the number of is-a links between terms and their Least Common Subsumer (LCS) and the path from the LCS to the root node.

3.3.2 Feature-based Methods

The edge-based methods depend on the structure of the taxonomy, which is not entirely uniform. Differences exist in the formalisation of the ontology. Certain words may be more explicitly stated than others. Therefore, techniques that consider other features of the ontology have been developed. These are known as feature-based methods and are calculated as a function of the properties of words such as gloss (meaning of a word), synsets, and related concepts. The degree of overlap between the features of two terms corresponds to how similar the two terms are.

Rodriguez and Egenhofer [2003] compute similarity as a weighted sum of three separate similarity assessments using the information from WordNet [Miller 1998]. That is, i) the similarity between the set of synonyms, ii) the set of features (relations such as composition and hyponymy, attributes, class, parts of class, functions), and iii) the set of neighbouring concepts within entity classes. Using weights based on ontology characteristics, they achieved higher correlation coefficients than simple edge-based methods. Highlighting the importance of feature matching for similarity within an ontology.

The feature-based approach by Sánchez *et al.* [2012] uses the taxonomical categorisation of concepts as features to calculate the amount of dissimilarity between the concepts. The smaller the dissimilarity, the more the concepts are considered similar. They approach the problem of disambiguating polysemic words by simply retrieving all related concepts to a word, computing similarities for each possible pair, and selecting as the final result, the pair with the maximum similarity value attained.

As with most ontology-based similarity measures, performance is dependent on the level of detail, completeness, factual accuracy, and multiple inheritance capability of the taxonomical knowledge source.

3.4 Information-based Methods

To complement the edge-based methods with additional information, Resnik [1995] explored the notion of Information Content (IC) of a term. The additional IC resulted in a probability associated with each term computed from their occurrences in a given corpus, $IC(a) = -\log P(a)$, leading to infrequent terms considered to have more information content. The similarity between terms is then dependent on the information that is shared between them. Combining IC with LCS, the similarity is computed as $sim_{a,b} = IC(LCS(a,b))$. An important note of using the IC methods is the dependence on the underlying corpus, i.e. any change to the corpus would

require recomputation of the probabilities which can be time-consuming for large corpora.

The workflow followed by Majumder *et al.* [2016] combine various techniques such as POS tagging, String Matching, utilising WordNet (taxonomy and relationships), and Information Content. Although achieving a 50% performance improvement over exclusively using WordNet taxonomy relationships, conditions in the pipeline are complex and arbitrary. Additionally, the performance of these methods is heavily reliant on the quality of the underlying knowledge source. Not all knowledge sources will contain the same generalisations/specialisations, making switching to a different knowledge source or combining information from multiple knowledge sources an arduous process, if possible at all.

3.5 Corpus-based Methods

Corpus-based methods use information from large corpora to compute the similarity between terms. These techniques do not consider the actual meaning of words but are based on the idea that words with similar meanings occur together frequently in a large corpus - which is most commonly known as the distributional hypothesis [Harris 1954].

Latent Semantic Analysis (LSA) [Landauer 1997] is a popular method proposed in the '90s to simulate the meaning of words and documents with respect to a corpus of natural text. LSA utilises techniques such as bag-of-words and Singular Value Decomposition (SVD) to produce vectors of words or documents. The similarity between these vectors can be computed using algebraic methods such as cosine similarity and dot product. The intuition behind LSA is that a document vector can be expressed as an average of the word meanings that it contains. Reciprocally, the meaning of a word can be expressed as its average effect on the meaning of all documents in which it occurs. A word-document matrix is constructed, where cells contain the frequency of the occurrence of the word in the corresponding document. SVD then decomposes and reduces the dimensionality of the matrix. Enhancements to LSA include weighting terms by their importance using TF-IDF. Another extension would be to substitute the latent features extracted from the corpus itself with features extracted from an existing knowledge base such as Wikipedia (Explicit Semantic Analysis [Gabrilovich and Markovitch 2007]).

In an effort to construct semantic spaces without the need for human judges, Lund and Burgess [1996] propose Hyperspace Analogue to Language (HAL). The aim of their model is to construct a lexical co-occurrence matrix from a large corpus of natural text (160 Million words from Usenet newsgroup, February 1995) and produce vectors which contain semantic information. The co-occurrence matrix is constructed by sliding a "window" over the words and calculating a co-occurrence strength value, which is inversely proportional to the number of other words separating a pair of words. This co-occurrence strength value represents how "strong" or "weak" the co-occurrence between a pair of words is. For instance, words that occur next to each other in the window will receive a maximum co-occurrence value.

Experiments conducted by Lund and Burgess [1996] verify that semantic information is captured in the vectors, hence, words that are similar do have vectors that are close to each other in vector space (computed with Euclidean or city-block metric). Since these vector representations are essentially measures of context, words that can be substituted for each other tend to have similar vectors because the words that surround them are generally the same or similar.

The dimension size of the vectors from HAL grows linearly with the vocabulary of the corpus. Although dimensionality reduction techniques can reduce vector dimensionality to a reasonable size, there is an accuracy trade-off. However, more recent approaches such as the Skip-gram and CBOW model of Word2Vec [Mikolov *et al.* 2013a] produce fixed-sized dimensional vectors.

Continuing with the notion of sliding a "window", these models learn distributed representations of words from a large corpus of text (6B Google News corpus). Learning occurs while simultaneously being trained through a shallow neural network to predict either, the current word within a context window, or, the context words that surround it. The pre-trained embeddings produced from Word2vec can be applied successfully in downstream tasks or as part of an NLP pipeline [Sinoara *et al.* 2019; Nguyen *et al.* 2019; Tien *et al.* 2019].

A number of other pre-trained word embedding algorithms have been proposed. GloVe uses a word co-occurrence matrix to incorporate global and local information when producing word embeddings. FastText [Bojanowski *et al.* 2016] attempts to accommodate morphological languages and represent rare words better by incorporating character n-grams into the skip-gram model. Word vectors are represented by the sum of their character n-gram vectors.

Table 3.1 shows a comparison of word embedding algorithms: Word2vec, GloVe, and FastText. Performance is comparable between the three algorithms and is dependent on the dataset used and the problem domain. For instance, Li *et al.* [2018] shows GloVe outperforming both Word2vec and FastText when classifying crisis tweets with an SVM classification model. While Dharma and Soewito [2022] and Iqbal *et al.* [2021] show FastText being the most performant for text classification and sentence similarity respectively.

Model Dataset	**SVM Classification 2CTweets [Li *et al.* 2018]**	**CNN Classification UCI KDD Archive [Dharma and Soewito 2022]**	**Sentence similarity SemSD [Iqbal *et al.* 2021]**
Word2vec	89.0	92.5	76.3
GloVe	90.4	95.8	70.9
FastText	84.0	97.2	77.3

TABLE 3.1: Average Accuracy from literature for 2CTweets Dataset and UCI KDD Archive. Pearson correlation $r \times 100$ between model cosine similarity and scores by human judges are shown for SemSD.

One issue persistent among word embedding algorithms is the inability to model polysemy and homonymy. The final word representation for a polysemic word such as "bank" will essentially have an averaged embedding of its constituent senses (in proportion to their frequency of occurrence in the corpus). For example, the representation of "bank" may be a weighted average of a *financial* bank and *geographical* bank, where the weight relates to the frequency of the word sense in the corpus. Research to incorporate Word Sense Disambiguation (WSD) into the word embedding pipeline has been explored [Du *et al.* 2019; Ansell *et al.* 2021; Pilevar *et al.* 2017], some are categorised under Deep Neural Network-based and Hybrid Methods in Section 3.6 and 3.7 respectively.

3.6 Deep Neural Network-based Methods

DNN-based methods have become more viable as computational power becomes cheaper and more accessible to attain for the average person. Two primary ways that DNNs may be used for semantic textual similarity are i) using pre-trained word embeddings as an input, and ii) using embeddings produced by the DNN architecture itself.

Typical architectures used for approach i) are Convolutional Neural Networks (CNNs), RNNs, and LSTMs. Tien *et al.* [2019] propose a DNN method to capture linguistic properties from several different pre-trained word embeddings (Word2vec, FastText, GloVe) and encode them into a novel embedding used to compare the similarity between sentences. In their model, CNNs are used to learn multi-aspects from different word embeddings using the max-pooling scheme. Since word order information is not maintained with CNNs, LSTMs are required to retain this information. Their M-MaxLSTM-CNN model achieves strong performance due to the multi-level similarity approach taken. The similarity is compared on a word-word, word-sentence, and sentence-sentence level. An appealing factor of M-MaxLSTM-CNN is that no hand-crafted features are required.

The primary approach for ii) is through Language Modelling. Given a large corpus of text, Language Modelling deals with the task of predicting the next token given the previously seen context. Peters *et al.* [2018] produce deep contextual word representations through forward and backward Language Modelling (predicting the previous token given the future context). They use LSTMs to capture context-dependent aspects of word meaning and CNNs to produce the initial character embedding input. Through a concatenation and weighted sum process of the hidden states and initial embeddings, a contextualised embedding is produced. Each produced embedding contains contextual information since it is a function of the entire input sentence. These embeddings have been termed ELMo embeddings which model syntax, semantics and polysemy.

The ELMo embeddings are computed using biLMs (bidirectional Language Models) with character convolutions. A biLM combines both a forward and backward Language Model. A backward Language Model processes the sequence in reverse, predicting the previous tokens given the future context. More formally, given a sequence of N tokens, $(t_1, t_2, \ldots, t_N)$, the probability of token t_k given its future context is attained with $\prod_{k=1}^{N} p(t_k|t_{k+1}, t_{k+2}, \ldots, t_N)$. The forward Language Model computes the probability of a token t_k given its history $(t_1, \ldots, t_{k-1})$ in a similar manner.

It is argued that a biLM formed through a combination of two unidirectional language models is sub-optimal [Devlin *et al.* 2019]. Preference has therefore been given to architectures, such as the Transformer, which can consider and incorporate context in both directions using the Attention mechanism. An additional benefit is the longer sequence modelling and parallelisation capabilities - which are major deficiencies of the LSTM architecture [Vaswani *et al.* 2017].

Instead of LSTMs, the GPT model by Radford *et al.* [2018] uses a Transformer Decoder architecture for Language Understanding and Modelling. Representations produced are also contextual due to the attention mechanism within the Transformer. GPT is unidirectional since it uses a decoder block which solely uses previous tokens to predict the next token.

In contrast to GPT, BERT uses the Transformer Encoder architecture and pre-trains deep representations from text in a bidirectional manner. Results outperform GPT and therefore highlight the importance of bi-directionality for Language Modelling [Devlin *et al.* 2019].

The success of Transformers has encouraged many others to exploit them in their models. Universal Sentence Encoder (USE) [Cer *et al.* 2018] is specifically designed to generate sentence embeddings to be used across NLP tasks. The encoders employed by USE are Deep Averaging Networks (DANs) [Iyyer *et al.* 2015] and Transformer Encoders. While DAN is computationally efficient, the Transformer Encoder has stronger performance in comparison. Fixed-length sentence embeddings are generated from context-aware word representations through an element-wise sum and scaling. Being intended for universal sentence embedding, encoders are trained with conversational and free text data from web sources such as Wikipedia, forums, question-answer sites, news pages, etc. The model is also encouraged to be general-purpose by using multi-task learning. These properties make USE ideal for the task of semantic similarity.

Since BERT had produced state-of-the-art (SOTA) results in several NLP tasks, much research has been focused on improving the model performance or efficiency. Potential improvements can be achieved through modifying hyperparameters, masking strategy, learning objectives, vocabulary, or enhancement of the transformer architecture [Liu *et al.* 2019; Zaheer *et al.* 2020]. Techniques to improve BERT in task-specific areas such as question answering, document classification, sentiment analysis, word sense disambiguation (WSD), and sentence similarity are also being researched [Du *et al.* 2019; Sun *et al.* 2019a; Adhikari *et al.* 2019].

A novel method by Ansell *et al.* [2021], termed PolyLM, models word *senses* instead of individual words. Modelling word *senses* has been shown to be beneficial in WSD or Word Sense Induction (WSI) tasks. PolyLM is an unsupervised Language Model and outperforms BERT-Large in most WSI tasks, despite it using a fraction of the parameters. However, because the approach is unsupervised, the number of word senses per word must be explicitly stated. The word sense embedding space can therefore contain frivolous sense embeddings. Using a lexical resource that contains a list of different senses could be used to minimise the number of these frivolous embeddings.

Sentence-BERT (SBERT) [Reimers *et al.* 2019] updates BERT weights such that the sentence embeddings are more semantically meaningful. The updated weights are attained by using siamese networks and triplet networks [Schroff *et al.* 2015] to fine-tune BERT. SBERT is computationally more efficient because it relies on cosine similarity instead of complex regression functions as most state-of-the-art methods learn.

The performance of BERT has been shown to improve when the model size increases. The improvement is evident when inspecting the performance of other large Transformer-based models such as T5-11B by Raffel *et al.* [2020b] which contains 11 billion parameters and outperforms its smaller variants, including BERT-Large which contains 340 million parameters. Authors of T5-11B further confirm that performance can be improved if larger models are built.

More recent work by Brown *et al.* [2020] shows evidence of larger models performing better with their model GPT-3. Containing a massive 175 billion parameters, GPT-3 shows strong performance on many NLP tasks. In some cases, under one-shot and few-shot settings, GPT-3 matches the performance of SOTA fine-tuned systems. Despite the improved performance, the authors point out that feeding enormous text corpora to the model is not the ultimate answer because there is a limit to the word/sentence prediction pre-training objective, which we may be running into soon. Hence, improving performance in this regard is not future-proof.

Additionally, it becomes computationally expensive and time-consuming to train

the model on such massive corpora. While large corporations can utilise high-performance computing to achieve more reasonable training times, not everyone has that level of computational power available. Cloud computing does address these needs but can come at a substantial price [Al-Roomi *et al.* 2013; Aldossary *et al.* 2019]. Furthermore, it is becoming imperative to keep our models "green" to reduce environmental impact. Therefore, techniques which maximise performance while maintaining model size or training complexity are advantageous to the research community and industry professionals.

The research team at Google realised this need for compact models for environments with restricted computational resources. They produced smaller, more compact BERT models which are most effective in the context of knowledge distillation [Turc *et al.* 2019]. However, performance degradation is noticeable as the model size becomes smaller.

The ranking framework proposed by Guo *et al.* [2020] called DeText is BERT-based and aims to be an effective and efficient ranking solution. It employs representation-based methods that allow for document embedding pre-computing, expediting the similarity computation process. Since the framework is intended for industry use cases, such optimisations are necessary. Their version of BERT, termed LiBERT, achieves better performance than BERT. A huge contributing factor to the improved performance is that LiBERT is pre-trained on domain-specific data instead of general domain data like BERT. They also mention that combining deep learning features with hand-crafted traditional features can help achieve improved performance.

Table 3.2 gives an overview and comparison of some of the DNN models mentioned in this section. Details of the architecture and training data are also provided.

The experiments conducted by Petroni *et al.* [2019] showed that factual relational knowledge does exist in the weights of a pre-trained BERT without any fine-tuning. However, language models cannot handle knowledge-intensive tasks such as common-sense reasoning or tasks which require domain-specific knowledge. Research to inject knowledge into these models has emerged [Nguyen *et al.* 2019; Lu *et al.* 2020; Yu *et al.* 2022; Liu *et al.* 2020]. Injecting knowledge is most commonly done with the aid of a knowledge base. Existing knowledge bases contain either human-defined information or automatically extracted facts from Web text. These techniques are discussed in Section 3.7.

3.7 Hybrid Methods

Each type of method described in the previous sections has intrinsic advantages and disadvantages. For example, knowledge-based methods typically rely on hand-crafted features. Also, their success is dependent on the quality of the knowledge base used. Manual intervention is required for any further updates or improvements. While on the other side, DNN or corpus-based methods are typically autonomous once the algorithm has been developed. However, large computational resources are required to create successful and more powerful models. There is also the issue of poor interpretability from DNN methods - which knowledge-based methods do not necessarily face [Dong *et al.* 2017]. To exploit the advantages and overcome the disadvantages, techniques which combine multiple types of methods have been proposed.

Sinoara *et al.* [2019] combine knowledge-base and corpus-based methods to construct embedded representations of documents for text classification tasks such as e-mail classification, spam filtering, article organisation, etc. For their knowledge

DNN Model performance comparison on STS-B dataset	
Tien *et al.* [2019]	
Architecture: 1600 Convolutional filters, LSTM of dimension 1600, Fully connected layer of dimension 250	
Data: Five pre-trained word embeddings of dimension 300. 1. Word2vec - trained on Google News dataset (100 billion tokens). 2. FastText - trained on Wikipedia. 3. GloVe - trained on Common Crawl (840 billion tokens). 4. SL999 - trained on paraphrase dataset PPDB. Tuned on SimLex-999 dataset [Hill *et al.* 2015]. 5. Beroni - 400-dimensional embedding trained on 2.8 billion tokens constructed from ukWac, English Wikipedia and the British National Corpus.	
Model	Pearson correlation $r \times 100$
M-MaxLSTM-CNN	82.5
Reimers *et al.* [2019]	
Architecture: Fine-tunes BERT in a siamese/triplet network architecture	
Data: Pre-trained on Wikipedia (via BERT), SNLI [Bowman *et al.* 2015] and Multi-Genre NLI [Williams *et al.* 2018] dataset	
Model	Pearson correlation $r \times 100$
$\text{SBERT}_{\text{BASE}}$	84.7
$\text{SBERT}_{\text{LARGE}}$	84.5
Radford *et al.* [2018]	
Architecture: Unidirectional 12-layer decoder Transformer, 768 dimension, 12 attention heads	
Data: Pre-trained on BooksCorpus dataset	
Model	Pearson correlation $r \times 100$
GPT	82.0
Cer *et al.* [2018]	
Architecture: TRANSFORMER - uses Transformer encoder. DAN - uses Deep Averaging Network. Outputs 512-dimensional vector as sentence embedding	
Data: Wikipedia, web news, web question and answer pages, discussion forums. Supervised training on SNLI corpus [Bowman *et al.* 2015]	
Model	Pearson correlation $r \times 100$
USE_{DAN}	71.9
$\text{USE}_{\text{TRANSFORMER}}$	78.2
Devlin *et al.* [2019]	
Architecture: Multi-layer bidirectional Transformer encoder. BASE - 12 layers, 768 hidden size, 12 attention heads, 110M parameters. LARGE - 24 layers 1024 hidden size, 16 attention heads, 340M parameters	
Data: Pre-trained on BooksCorpus [Zhu *et al.* 2015] and English Wikipedia (2,500M words)	
Model	Pearson correlation $r \times 100$
$\text{BERT}_{\text{BASE}}$	85.8
$\text{BERT}_{\text{LARGE}}$	86.5

TABLE 3.2: Pearson test scores $\times 100$ reported for STS-B dataset across various DNN models in literature.

Table 3.2 - *Continued from previous page*

Raffel *et al.* [2020b]	
Architecture: Multi-layer encoder-decoder Transformer. Large - 24 layers, 1024 hidden size, 16 attention heads, 770M parameters. 11B - 24 layers, 1024 hidden size, 128 attention heads, 11B parameters.	
Data: Pre-trained on The Colossal Clean Crawled Corpus scraped from Common Crawl	
Model	Pearson correlation $r \times 100$
T5-Large	89.9
T5-11B	93.1

source, they consider BabelNet - a large-scale multilingual encyclopedic dictionary containing a network of synsets and relations among them. The utilisation of BabelNet allows their approach to disambiguate word senses, thereby creating more accurate vector representations of documents. NASARI vector representations and Word2vec embeddings are the primary embedding models used for word senses and words respectively. Synsets in BabelNet are represented by NASARI vectors, which leverage knowledge from Wikipedia and pre-trained Word2vec embeddings. Document vectors are represented as the average of the synset vectors contained within the document. When a synset vector does not exist, the Word2vec embedding for the word is used instead. An advantage of their approach is that neighbouring word senses are more meaningful and have greater interpretability, however, Word2vec did perform better in certain settings (based on Lee's dataset [Lee *et al.* 2005]). Improved interpretability is possible since NASARI vectors correspond to synsets in BabelNet and this knowledge can be easily inspected.

To *semantify* input text, Pilevar *et al.* [2017] integrates sense-level knowledge into a CNN text classifier by disambiguating input text using information from a WordNet semantic network. The aim of integrating the semantic information is to disambiguate text prior to being fed into a system. They highlight that simple disambiguation of the input can bring about a performance gain for longer texts. Since their disambiguation algorithm relies on constructing a graph from the input text, short texts do not provide sufficient connections to function effectively, thereby producing lower performance.

Knowledge-based methods for semantic similarity rely on the information in the underlying knowledge sources such as ontologies/ lexical databases to disambiguate synonyms and word senses. Nguyen *et al.* [2019] was able to achieve performance that slightly underperformed BERT. They exploit the similarity of word contexts built by word embeddings and semantic relatedness between concepts based on external knowledge sources such as WordNet and Wiktionary to more accurately assess the similarity between two short texts. Their approach is however supervised and requires feature engineering. They suggest an area for further exploration is combining unsupervised models with external knowledge sources.

Lu *et al.* [2020] combine the capability of BERT with Vocabulary Graph Convolutional Network (VGCN). VGCN convolves directly on a vocabulary graph to induce embeddings based on node properties and their neighbourhoods. Their experiments show that a combination of the local information from BERT and global information from VGCN outperforms BERT and VCGN alone. More formally, given an input sentence, relevant parts of the vocabulary graph are selected and transformed into an embedding. BERT is then fed both the input sentence embedding and graph

embedding. These two embeddings interact with each other via the self-attention mechanism in BERT. The resultant model, VGCN-BERT, produces better results on the majority of evaluation datasets compared to a model (Vanilla-VGCN-BERT [Lu *et al.* 2020]) which does not allow interactions between the embeddings.

The vocabulary graph by Lu *et al.* [2020] is constructed using normalised pointwise mutual information (NPMI) and word co-occurrences with documents. When building the vocabulary graph, an NPMI threshold of 0.2 is set to filter out non-meaningful words. The authors believe utilising knowledge from other lexical resources such as WordNet, or combinations of such, should be explored as it contains useful connections that NPMI cannot cover. Although VGCN-BERT does outperform BERT, since the combined embeddings are obtained in separate ways, their vector-space is inconsistent [Sun *et al.* 2017; Liu *et al.* 2020].

Yu *et al.* [2022] proposed a contextualised language model, CoCoLM, enhanced by complex common-sense knowledge from the ASER knowledge graph. ASER is a large-scale eventuality knowledge graph including states, actions, events, and their relations. Their approach aims to incorporate discourse knowledge into the pre-trained language models. Instead of whole-word masking, CoCoLM employs Whole Eventuality Masking.

Given an input sentence "I feel sleepy because I drink a cup of coffee", BERT would typically be tasked to predict a single masked word such as "coffee". When using Whole Eventuality Masking, we instead mask the entire eventuality "I drink a cup of coffee". Experiments on common-sense and reasoning tasks show the effectiveness of CoCoLM compared to BERT. One notable obstacle introduced by masking longer spans of text is that it limits the context that the Masked Language Model can learn.

Another attempt at achieving common-sense or domain knowledge is shown in K-BERT [Liu *et al.* 2020] where they utilise existing knowledge graphs (CN-DBpedia, HowNet, MedicalKG) to inject knowledge into sentences, making the sentence knowledge-rich. The injection is done through a lookup mechanism which matches the entity e from the input sentence to its corresponding entity in the knowledge graph. For an entity e, the first n triplets of knowledge are injected.

K-BERT achieved performance improvements over BERT in some open-domain NLP tasks. The improvements were more significant in domain-specific tasks, due to the utilisation of MedicalKG, which contains domain-specific knowledge. Moreover, the K-BERT model attempts to overcome challenges such as Heterogeneous Embedding Space (HES) [Lu *et al.* 2020] and Knowledge Noise (KN) [Yu *et al.* 2022]. While K-BERT outperforms BERT on particular Chinese NLP tasks, it is unknown how well the performance would translate to other tasks while utilising English knowledge graphs.

3.8 Conclusion

This chapter outlined various types of methods for the task of semantic similarity. The types of methods outlined range from simple string-based methods theorised in the 1960s [Weiner 1973; Levenshtein 1965; Jaro 1989] to modern DNN-based methods whose practical relevance is becoming more prevalent with each significant breakthrough [Vaswani *et al.* 2017; Devlin *et al.* 2019; Brown *et al.* 2020].

From inspecting Table 3.2 it is evident that Transformer-based models have performance advantages over other architectures such as LSTM, CNN, and DANs. The prevalence of high-performance computing and the ability for language models to

be pre-trained makes Transformers viable for a wide range of NLP applications. Additionally, we see the larger models perform better than their smaller variants. Despite the general trend to achieve improved performance in this manner, there will always be a need for smaller, more efficient models [Guo *et al.* 2020; Turc *et al.* 2019; Yang *et al.* 2017]. This is especially true when these models are used practically or are required to be cognisant of environmental impacts. For this reason, the focus of this research is to attain performance improvements while also prioritising model efficiency.

The promising hybrid approach by Liu *et al.* [2020] will be replicated in the English domain and improved on. The main improvements include modifying the K_{Query} mechanism to consider sentence context when matching entities from the knowledge graph, instead of using the first n entities as done by Liu *et al.* [2020]. Lastly, the injection of knowledge is studied closely. In particular, an objective of this research is to understand the effects of knowledge incorporation, as well as the type of knowledge and context under which it is most beneficial. The remaining chapters will outline details of the English knowledge graph, the proposed method, the experimental setting, results, and discussion thereof.

Chapter 4

Wikidata Knowledge Graph

4.1 Introduction

The work by Liu *et al.* [2020] employed three Chinese Knowledge Graphs (CN-DBpedia, HowNet, MedicalKG) to serve as their knowledge source. These are stored in a triple format, i.e. (subject, predicate, object) or (word, contain, sememes). Of the prominent Knowledge Graphs that exist in the English domain, integration of YAGO, ConceptNet, Microsoft Concept Graph (MCG), and Wikidata was attempted. The Knowledge Graph that worked well for the purposes of this study was Wikidata.

4.2 Data Structure

Being a sister project to Wikipedia, the information in Wikidata has primarily been crowd-sourced as Wikipedia has been. However, instead of free-flowing information, Wikidata conforms to RDF and is therefore handled in a structured format. At a high level, Wikidata contains information and statements about data items [wik 2016b]. An item represents *things* in human knowledge such as objects, concepts, or topics. For example, a data item can range from prominent figures such as "Elon Musk" to simple words such as "bank". Statements are typically properties such as gender, name, instance of, etc. Each distinct data item and property is assigned a unique identifier (e.g. Q42, P21). An example data item "Douglas Adams" is shown in Figure 4.1.

4.3 Motivation

An advantage of Wikidata over the previously mentioned Knowledge Graphs is that data items are inherently unambiguous [wik 2016b]. Consider the polysemic term "race". Wikidata contains distinct items Q3254959 and Q22938576 whose labels match "race" (among other items). These data items are shown in Figure 4.2. Ideally, one would assume that a sentence about a swimming race would be more similar to the Q22938576 item since it is a subclass of a sporting event. In the following chapters, we will explore how to exploit this fact for knowledge injection.

4.4 Data Retrieval

While Wikidata dumps are made freely available, due to the amount of information the Knowledge Graph contains, an uncompressed dump is well over 100 GB [wik 2022]. To ensure computational feasibility, the size is reduced to 50 GB by limiting

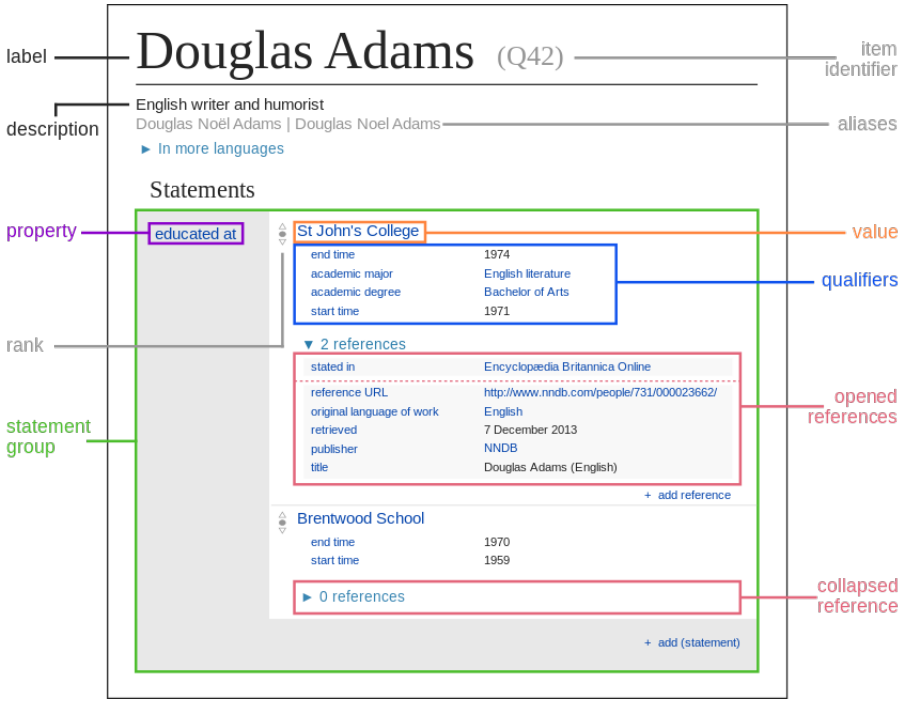
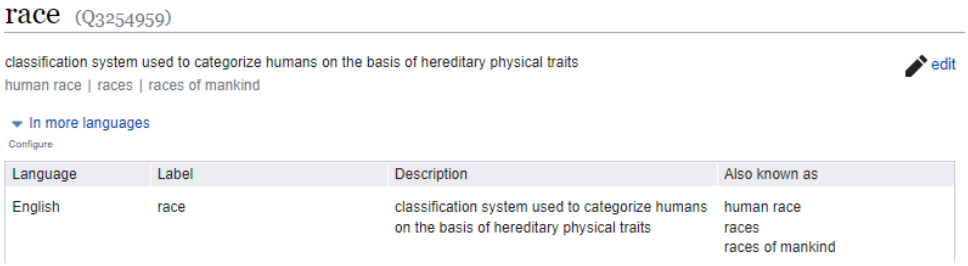
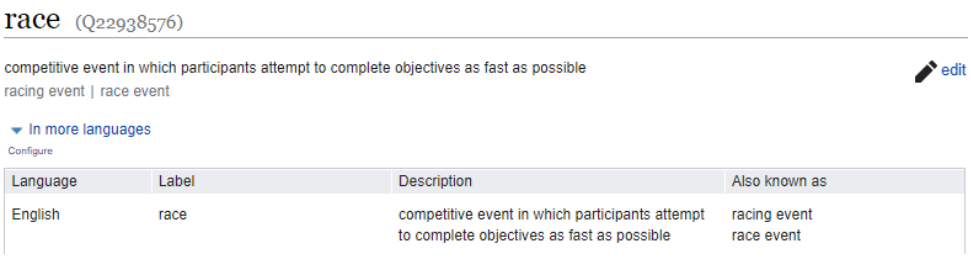


FIGURE 4.1: Wikidata information on English writer Douglas Adams [wik 2016a]



(A) Data item **Q3254959** referring to the race classification system



(B) Data item **Q22938576** referring to the competitive sporting event

FIGURE 4.2: Two instances of data item "race" **Q3254959** and **Q22938576**

the domain. For data items that belong to the domain, the Wikidata Query Service[1] is utilised to query data items that match the specified domain requirements. The SPARQL query used to limit the domain is shown in Appendix A. Thereafter, API requests with the attached item identifiers are used to attain the corresponding information and statements [2].

4.4.1 Data Processing

The aforementioned domain is restricted to humans, businesses/organisations, sports, cities/countries. Only important and common information from data items is retained. These are item identifiers, labels, descriptions, aliases, and properties: instance of and subclass of. This restricted information attempts to maximise the degree to which the data items are described while minimising storage usage. Non-English information is also filtered out. After data processing Knowledge Graph size is now 500 MB.

4.4.2 Data Storage

Although the Knowledge Graph can be stored in triple format, to reduce storage usage and query time, a relational database PostgreSQL was used instead.

[1] https://query.wikidata.org/
[2] https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q42

Chapter 5

Methodology

This chapter will outline the contributions made in this study. Firstly, an overview of the architecture is presented. Thereafter, details of the Wikidata Knowledge Graph integration, two architectural changes in the Knowledge Layer, and two optimisations are explained.

5.1 Architecture Overview

Figure 5.1 shows the architecture with the new contextualised injection mechanism. Input sentences first go through a NER component. Details of the extracted named entities are then queried from the Knowledge Graph and fed into a knowledge injection component to produce a sentence tree. Visible matrix and token embeddings are constructed from the sentence tree. These serve as the primary input to the Mask Transformer Encoder, which is described in Chapter 2.

5.2 English Knowledge Graph

To test the application of K-BERT in the English domain, Wikidata is used as the Knowledge Graph in this work. Since Wikidata contains almost one hundred million data items, utilising the *entire* Knowledge Graph is computationally infeasible. Therefore, we focus exclusively on data items belonging to particular domains (as described in Chapter 4).

Despite reducing the size of the Knowledge Graph significantly, it is still magnitudes larger than the Knowledge Graphs utilised by Liu *et al.* [2020]. Therefore, it cannot reasonably fit into memory. Storing the Knowledge Graph data into a database such as PostgreSQL facilitates successful integration with the model, albeit, in a non-optimal manner.

To overcome this non-optimal approach, the size of the Knowledge Graph is further reduced by caching the parts of it that are relevant to the dataset at hand. This caching process is shown in the Knowledge Layer component illustrated in Figure 5.1, where entities from input sentences are queried from the Knowledge Graph and the results are stored in the cache. Since querying is a deterministic process, caching can be done as a preprocessing step prior to training. The cache provides a significant speedup over querying from a database since it can be stored in memory.

5.3 Term-based Sentence Tree

Since Chinese is logographic, K-BERT was initially designed to work on a character-level. Model inputs such as the sentence tree and visible matrix must therefore be

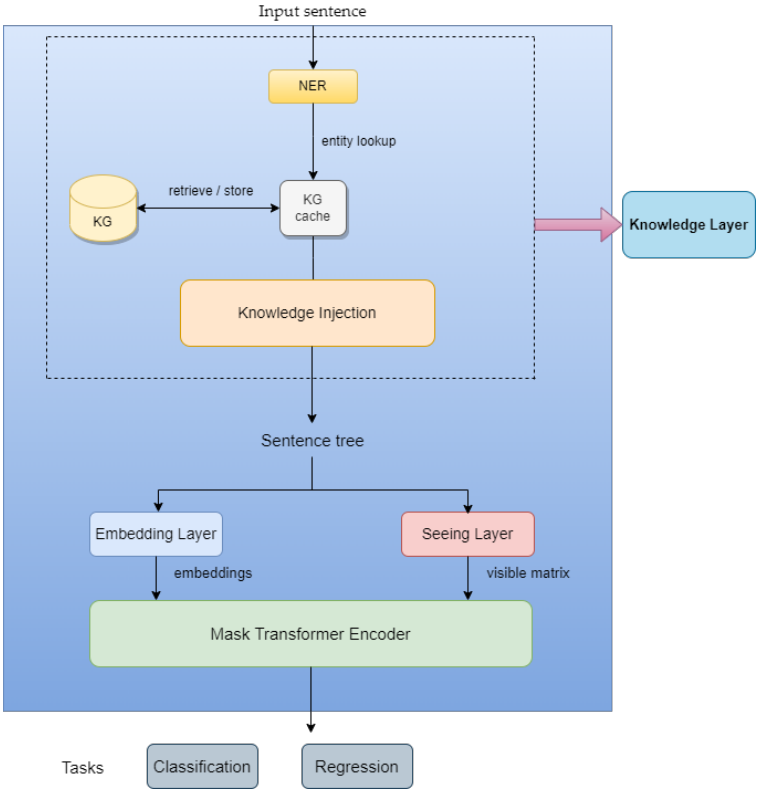


FIGURE 5.1: K-BERT architecture

adapted to accommodate the alphabetic English language on a word-level. However, a further extension to the multi-word-level or *term*-level is also necessary as there are cases where entities may span across two or more words in a sentence (e.g. name of a person or place). Knowledge injection is therefore done per **group** of tokens, instead of a single token.

More formally, given an input sentence $s = [w_0, w_1, \ldots, w_n]$ comprised of tokens w_i in vocabulary V, enclose contiguous related tokens w_o to w_q together in a group $\{\}$. Groups are not required to consist of multiple tokens. We can therefore have a singular group w_j, where w_j has no other contiguously related tokens. Knowledge is injected per group of related tokens to produce a sentence tree t:

$$t = \Big[\{w_0\}, \{w_1\}, \ldots, \{w_o, \ldots, w_q\}\big[(r_{op0}, w_{op0}), \ldots, (r_{opk}, w_{opk})\big], \ldots, \{w_n\}\Big], \quad (5.1)$$

where $w \in V$ is the set of entities in the Knowledge Graph K, $r \in V$ is the set of relations/properties in the Knowledge Graph, and k represents the number of triples inserted. Related tokens are enclosed by braces and their knowledge, if any, is inserted directly afterwards.

Construction of the visible matrix and positional encoding can be done similarly as before, however, an additional "unrolling" step is required to associate positions per token. As a last step, the visible matrix ensures token groups and their knowledge all "see" each other by appointing a value of 1 in their cell. This optimisation

alleviates duplicating the same knowledge for each token in the group, thereby minimising sequence lengths provided to the model.

5.4 Contextualised Knowledge Injection

This section highlights the approach taken to inject knowledge into sentences. For an input sentence s, perform named entity recognition to extract entities such as names of people, places, sports teams, etc. This NER is accomplished with a small pre-trained model from spaCy[1] based on a transition-based parser [Lample *et al.* 2016; Honnibal 2013] and an "embed, encode, attend, predict" framework [Honnibal 2016]. Thereafter, the extracted entities are queried from the Knowledge Graph to retrieve a list of triples. This processing is done by K_{Query}:

$$E = K_{\text{Query}}(s, \text{K}) \,, \tag{5.2}$$

where collection of triples is $E = [(w_i, r_{i0}, w_{i0}), \ldots, (w_i, r_{ik}, w_{ik})]$. The function, K_{Inject}, then injects E into the correct position and generates the sentence tree t as:

$$t = K_{\text{Inject}}(s, E) \,. \tag{5.3}$$

However, before injection occurs, additional processing to find the most relevant triples is done. Hence, instead of injecting the first n entities as done by Liu *et al.* [2020], a pre-trained Transformer model is used to determine which retrieved entities from E are most relevant to the current sentence s. This injection is done by concatenating each entity and its related properties into a single text sequence seq_i. The pre-trained Transformer model T then generates contextualised embeddings emb_i for each sequence seq_i using:

$$emb_i = T(seq_i) \,. \tag{5.4}$$

Contextualised embedding emb_s is generated for sentence s in the same manner. The similarity between embeddings is computed using cosine similarity metric $\|\cdot, \cdot\|_{\cos}$. The entity corresponding to the most similar embedding is selected to be injected using:

$$max_pos = \arg\max_{i \in I} \left(\|emb_i, emb_s\|_{\cos} \right) \,, \tag{5.5}$$

where I is the set of distinct entities.

To limit the amount of noise injected, a threshold parameter is introduced such that information is only injected if $\|emb_i, emb_s\|_{\cos} >$ threshold.

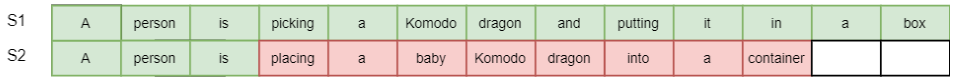
5.5 Corrections & Optimisations

5.5.1 Sequence Truncation

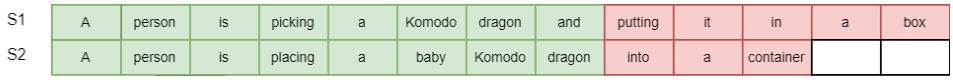
Truncation has been modified to be fairer when K-BERT is fed in sentence pairs. Given a *max_length*, the number of tokens in sentences S1 and S2 is each limited to size $max_length/2$. Any unoccupied positions are assigned as leftover and given to the remaining sentence that requires it. This differs from the initial truncation method of K-BERT, where the sequence is limited to the first *max_length* tokens. Figure 5.2 illustrates the difference between the two truncation methods.

[1] https://spacy.io/

The modified truncation method reduces biases that may arise when S2 is primarily being truncated, while S1 is not. For example, consider S1="A person is picking a Komodo dragon and putting it in a box." and S2="A person is placing a baby Komodo dragon into a container.". Intuitively, for the task of sentence similarity, aggressive truncation of S2 will cause degraded performance since information lost is unequal between sentences, i.e. "container" is truncated in S2 while "box" remains in S1. The new truncation method attempts to address this issue.



(A) Previous truncation method



(B) Truncating sentences equally

FIGURE 5.2: Comparison of truncation methods where $max_length = 16$. The words highlighted in red are truncated, while the green words are kept.

5.5.2 Memory Usage

Dimension of the visible matrix is $max_length \times max_length$. Implying that memory usage grows exponentially as sequence length increases, i.e. $O(n^2)$. Such growth can become infeasible for large datasets containing upward of 100,000 samples such as ag_news_subset [Zhang *et al.* 2015]. Since the visible matrix is symmetrical, memory usage is optimised by intermediately storing non-duplicate entries from the visible matrix into a vector of size $N(N+1)/2$. When the matrix is required to be fed into K-BERT, the vector is converted back into a symmetrical matrix - done per batch. Although this does require additional computation, the trade-off is worthwhile - especially if memory capacity is a bottleneck.

Lastly, since the visible matrix strictly contains binary values ("visible", "invisible"), a final memory optimisation explicitly restricts values to one byte each. With the described optimisations, the visible matrix memory consumption is reduced by a factor of four *at minimum*. Space complexity can therefore be represented by $O(n^2/r)$, where $4 \leq r < 8$.

Chapter 6

Experiments

Evaluation datasets used in this work focus on the task of semantic similarity. In the general domain, the Semantic Textual Similarity Benchmark (STS-B) dataset is used [Cer *et al.* 2017]. STS-B contains pairs of sentences which are manually annotated by human judges according to how similar each sentence pair is. An average score between 0 to 5 is assigned for each pair, where larger scores imply higher similarity. Examples are illustrated in Table 6.1.

Sentence 1	Sentence 2	Score
A young girl is sitting on Santa's lap.	A little girl is sitting on Santa's lap	4.8
A women sitting at a table drinking with a basketball picture in the background.	A woman in a sari drinks something while sitting at a table.	2.6
Norway marks anniversary of massacre	Norway Marks Anniversary of Breivik's Massacre	3.8
A man is cutting a potato.	A woman is cutting a tomato.	1.3

TABLE 6.1: Examples from STS-B dataset

An equivalent domain-specific dataset to STS-B is scarce in the NLP field. This scarcity is due to the reliance on experts in each respective domain to manually annotate text, which can be an expensive and time-consuming process. Hence, we consider a text classification task to evaluate domain-specific performance. The ag_news_subset dataset [Zhang *et al.* 2015] requires classifying paragraphs according to a topic (World, Sci/Tech, Sports, Business). This task is a good assessment of model performance as it requires domain-specific knowledge in the aforementioned fields to correctly classify each paragraph. Table 6.2 illustrates a few examples from ag_news_subset.

Title	Description	Label
Wood's Suspension Upheld (Reuters)	Reuters - Major League Baseball \Monday announced a decision on the appeal filed by Chicago Cubs \pitcher Kerry Wood regarding a suspension stemming from an \incident earlier this season.	1 (Sports)
Bush reform may have blue states seeing red	President Bush #39;s quot;revenue-neutral quot; tax reform needs losers to balance its winners, and people claiming the federal deduction for state and local taxes may be in administration planners #39; sights, news reports say.	2 (Business)
'Halt science decline in schools'	Britain will run out of leading scientists unless science education is improved, says Professor Colin Pillinger.	3 (Sci/Tech)

TABLE 6.2: Examples from ag_news_subset dataset

6.1 Baselines

K-BERT modified for the English language will first be compared against BERT. This comparison will highlight the influence of knowledge incorporation into the fine-tuning process. Models which are more suitable for sentence similarity will also be compared. These are SBERT [Reimers *et al.* 2019] and USE [Cer *et al.* 2018]. Since they have been specifically designed for the task of sentence similarity and sentence embedding, it is important to consider their performance as well. Details of the exact pre-trained models used during experiments can be found in Appendix A.

6.2 Fine-tuning

All models and baselines are fine-tuned for each task. The hyperparameters used for fine-tuning can be found in Table 6.4. Final evaluation on the test set is done thereafter. Although all models are based on the Transformer architecture, additional architectural changes are built upon that for each model. Consequentially, minor differences exist in the classification models used during the fine-tuning process. In general, however, models are used to produce embeddings which are then fed into a classification layer for the final prediction.

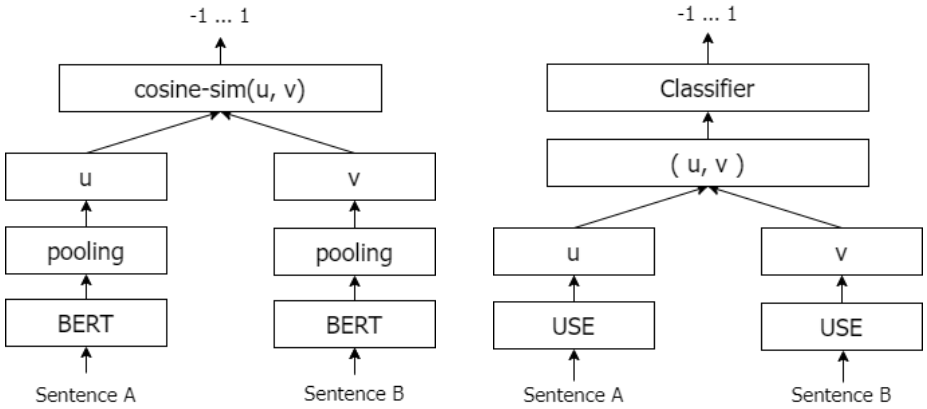
The fine-tuning process for K-BERT is the same as for BERT. Further explanations regarding the fine-tuning process of BERT will therefore apply to K-BERT as well.

BERT has been designed to work for various downstream classification and regression tasks on both single and pair-input problems. During training, the [CLS] token is used as the aggregate embedding to base classification/regression on. BERT is therefore influenced to encode pertinent information in this token representation. The pooled [CLS] representation is then passed into a classification head for the final output.

SBERT works similarly to BERT. However, instead of using the [CLS] token representation, SBERT uses a mean pooling across all input tokens. In contrast, USE is intended for simply generating meaningful sentence embeddings. Pairs of sentences are not accommodated. However, a siamese network of two USE models can be used to produce embeddings for pairs of sentences. An amalgamated representation of these sentence embeddings can be used in place of the pooled representation produced by BERT. The minor architectural differences required for fine-tuning are explained in Section 6.2.1 and Section 6.2.2.

6.2.1 STS-B

Fine-tuning for STS-B is treated as a regression problem by normalising similarity scores between 0 and 1. SBERT produces this score by computing the cosine similarity between sentence embeddings u and v. The architecture of the USE classifier has been designed to complement SBERT. The only change is the replacement of the cosine similarity block with a classification block that accepts a concatenated u and v to produce the final prediction. These regression architectures are illustrated in Figure 6.1.

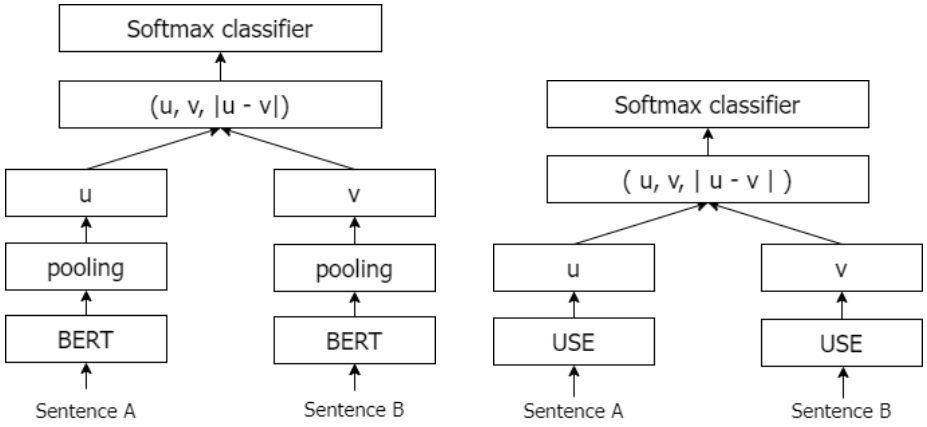


(A) SBERT (B) USE

FIGURE 6.1: SBERT [Reimers *et al.* 2019] and USE regression architectures

6.2.2 Ag_news_subset

USE and SBERT architectures are modified slightly to accommodate classification. A softmax classifier is applied at the final layer to retrieve final predictions. Additionally, embeddings u and v are now concatenated with $|u - v|$ to represent the aggregate embedding. These changes are shown in Figure 6.2.



(A) SBERT (B) USE

FIGURE 6.2: SBERT [Reimers *et al.* 2019] and USE classification architectures

6.3 Experiment Outline

During experimentation, hyperparameters and additional configurations were kept consistent across models and runs wherever possible. Hyperparameters were tuned based on the validation set performance. Specific seeds were used for all runs to ensure result reproducibility. Fine-tuning was done for ten epochs. The best-performing model on the validation set was saved and used to evaluate the test set. Batch sizes 16 and 32 were used for STS-B and ag_news_subset respectively. Being cognisant of the catastrophic forgetting problem [McCloskey and Cohen 1989; Sun *et al.* 2019b],

only modest learning rates were considered as hyperparameters (2e-5 to 5e-5). Best performing learning rates and thresholds, based on the validation set, were (4e-5, 0.5) and (5e-5, 0.6) for STS-B and ag_news_subset respectively.

While best-performing hyperparameters and configurations are preferred, balancing computational requirements and performance was carefully considered during the experimentation phase. It was, therefore, necessary to limit sequence length to 128 for ag_news_subset and 256 for STS-B. Additionally, the pre-trained $\text{BERT}_{\text{BASE}}$ was used instead of the better performing, yet computationally demanding, $\text{BERT}_{\text{LARGE}}$.

Adam optimiser with weight decay and scheduler parameters remained unchanged from the work by Liu *et al.* [2020] and Devlin *et al.* [2019]. That is Adam with L2 weight decay of 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and linear decay of learning rate.

STS-B has predefined train, validation, and test splits. This work uses these exact splits but shuffles the training set before every run. The same is true for ag_news_subset. However, since it does not have a predefined validation set, the last 8% of the training set served as the validation set throughout all experiments.

The number of samples per split is shown in Table 6.3.

Dataset	**Train**	**Validation**	**Test**
STS-B	5,749	1,500	1,379
ag_news_subset	110,400	9,600	7,600

TABLE 6.3: Evaluation Datasets

Parameter	**Models**			
	USE	**SBERT**	**BERT**	**K-BERT**
Learning rate	2e-5, 3e-5, 4e-5, 5e-5			
Batch size	8, 16, 32			
Epochs	10			
Max Sequence length	–	128, 256	128, 256	128, 256
Pooling	–	Mean	First	First
Dropout	–	0.1	0.1	0.1
Optimiser	Adam	AdamW	AdamW	AdamW
Weight decay	–	0.01	0.01	0.01
Warmup	–	0.1	0.1	0.1

TABLE 6.4: Hyperparameters used for fine-tuning each model. Final values used for STS-B and ag_news_subset are highlighted in blue and orange respectively.

6.3.1 Experiments

Exploring the benefits and drawbacks of knowledge incorporation into BERT is one of the core purposes of this work. For this reason, an extended comparison between the two models has been done. Results are averaged over ten runs to limit biases and highlight statistical significance. Each of these runs uses consecutive seeds (8 to 17) to ensure the reproducibility of the results.

We aim to attain insight into i) what type of knowledge is most beneficial and ii) when knowledge incorporation harms performance.

The following types of experiments are performed to gain such insight:

1. Knowledge Ablation: As described in Chapter 4, the type of knowledge injected can be categorised into aliases, categorical, and descriptive information. To identify what type of information is most beneficial, I exclude each type from being injected into K-BERT and rerun experiments.

2. Knowledge-Gating: Since K-BERT requires sequences to be truncated passed a maximum sequence length, injecting additional information can lead to important information from the original sentence being truncated. To see the extent to which this is true, I perform Knowledge-Gating. That is, to only inject information into sentences when sequence lengths are below the maximum sequence length - thereby avoiding truncation.

3. Manual Knowledge Injection: To identify deficiencies in the automated similarity-based approach presented in K_{Inject} for relevant knowledge fusion, a manual selection of knowledge is instead performed. The manual approach is expected to contain minimum noise since humans are capable of matching relevant knowledge to sentence context to a much higher degree. Manual Knowledge Injection is done for all dataset splits as described in detail below.

Manual Knowledge Injection

More specifically, the systematic approach for performing Manual Knowledge Injection requires manual matching of extracted entities E to the context of sentence s. When sentence s lacks context, description information is typically selected to be injected. In situations where descriptive information would introduce noise or be inappropriate, I default to categorical or alias information. While most of this process is subjective, the primary objective of only injecting beneficial information, for the problem at hand, is maintained throughout. For example, the occurrence of an entity "Manchester United" in ag_news_subset will likely be associated with external knowledge "football club" - **if** the terms do not already exist in the original sentence s.

6.4 Research Limitations

Best-performing models often require expensive hardware and longer training or fine-tuning phases. As mentioned briefly in Section 6.3, the selection of model size and hyperparameters was limited by the availability of computational resources and time constraints. The lower sequence length limitation imposed for the ag_news_subset dataset may have affected the performance to a certain degree.

Despite Semantic Similarity being the task of focus in this research, no appropriate domain-specific dataset exists to measure such performance. A domain-specific classification dataset was used as consolation. Justification for this decision is mentioned at the beginning of Chapter 6.

Chapter 7

Results

7.1 Baseline Comparison

Table 7.1 shows mean results taken over ten runs for USE, SBERT, BERT, & K-BERT on STS-B and ag_news_subset datasets. Spearman and Pearson correlation coefficients are reported for the STS-B dataset. Classification accuracy is reported for ag_news_subset. The baseline results here closely resemble published results and benchmarks [Reimers *et al.* 2019].

Using all knowledge categories, K-BERT outperforms baselines USE and SBERT. However, the average performance is lower than BERT on the STS-B dataset. When injecting knowledge manually, indicated by K-BERT$_{\text{MANUAL}}$, K-BERT performs $\approx$ 0.7% better than the best baseline (BERT) on knowledge-driven tasks such as the ag_news_subset classification problem. While the performance of K-BERT compared to baseline models USE & SBERT is superior, the reduction in performance compared to the base BERT on STS-B suggests the injected knowledge has introduced noise. An extended comparison between BERT and K-BERT, along with other knowledge ablations, is provided in the following sections.

Model	**STS-B**		**AG NEWS SUBSET**
	Spearman	**Pearson**	**Accuracy**
USE [Cer *et al.* 2018]	–	81.40 / 78.20	–
SBERT [Reimers *et al.* 2019]	–	– / 84.67	–
BERT [Devlin *et al.* 2019]	–	– / 85.80	–
USE	82.59 / 78.02	82.89 / 79.45	92.89 / 92.70
SBERT	87.56 / 85.15	87.41 / 84.18	94.24 / 94.28
BERT	89.67 / 85.73	90.00 / 86.77	94.55 / 94.60
K-BERT	89.58 / 85.66	89.94 / 86.70	94.55 / 94.65
K-BERT$_{\text{MANUAL}}$	89.63 / 85.65	89.97 / 86.68	95.16 / 95.24

TABLE 7.1: Mean validation / test scores on best performing model recorded over ten runs with sequential seeds on evaluation datasets. K-BERT$_{\text{MANUAL}}$ refers to injecting knowledge manually in K-BERT

7.2 Comparison between BERT and K-BERT

Table 7.2 shows the Knowledge Ablation results on the STS-B dataset. Total Mean Squared Error (MSE) loss, Spearman correlation, and Pearson correlation is reported. The correlations are between the predicted and actual scores. Combinations of knowledge categories from set {ALIAS, CAT, DESC} were removed from K-BERT, and the

results are presented here. Abbreviated knowledge categories consecutively correspond to {aliases, categorical information, descriptive information}. Additionally, a model with minimal noise, K-BERT$_{\text{MANUAL}}$, indicates that knowledge selection is done manually by humans instead of the similarity-based approach of K-BERT.

For ag_news_subset, classification accuracies are reported for BERT and all K-BERT ablation variations in Table 7.3. Compared to STS-B, sequence length limitations is a more stringent factor for ag_news_subset due to its much longer paragraphs of text. This factor has been further exacerbated by the 128 sequence length limitation imposed by the experiment configuration. Knowledge-Gating experiments have been performed to investigate the effect of retaining maximal information from the original sentences. The results of the Knowledge-Gating experiments are shown in column Knowledge-Gating=on.

Model	**MSE Loss**	**Spearman**	**Pearson**
BERT	1.662 / 2.014 $\pm$0.054	89.67 / 85.73 $\pm$0.4930	90.00 / 86.77 $\pm$0.3660
K-BERT	1.675 / 2.028 $\pm$0.063	89.58 / 85.66 $\pm$0.5633	89.94 / 86.70 $\pm$0.4301
After Removing			
ALIAS	1.655 / 2.025 $\pm$0.055	89.65 / 85.71 $\pm$0.4659	90.00 / 86.70 $\pm$0.3795
CAT	1.655 / 2.018 $\pm$0.059	89.63 / 85.64 $\pm$0.5364	89.98 / 86.71 $\pm$0.4167
DESC	1.667 / 2.019 $\pm$0.056	89.60 / 85.76 $\pm$0.4885	89.95 / 86.77 $\pm$0.4123
CAT + DESC	1.667 / 2.009 $\pm$0.068	89.54 / 85.80 $\pm$0.5740	89.89 / 86.79 $\pm$0.4621
ALIAS + DESC	1.664 / 2.018 $\pm$0.059	89.59 / 85.80 $\pm$0.4352	89.96 / 86.80 $\pm$0.3504
ALIAS + CAT	1.653 / 2.009 $\pm$0.058	89.59 / 85.77 $\pm$0.5017	89.95 / 86.78 $\pm$0.3920
K-BERT$_{\text{MANUAL}}$	1.657 / 2.024 $\pm$0.059	89.63 / 85.65 $\pm$0.5714	89.97 / 86.68 $\pm$0.4294

TABLE 7.2: Knowledge Ablation results on STS-B dataset for mean (μ-validation / μ-test $\pm\sigma$) scores taken across ten runs.

Model	**Knowledge-Gating**	
	Off	**On**
BERT	94.55 / 94.60 $\pm$0.1947	
K-BERT	94.55 / 94.65 $\pm$0.2805	94.58 / 94.69 $\pm$0.1269
After Removing		
ALIAS	94.60 / 94.60 $\pm$0.2272	94.52 / 94.67 $\pm$0.1138
CAT	94.53 / 94.61 $\pm$0.1144	94.53 / 94.67 $\pm$0.1474
DESC	94.54 / 94.63 $\pm$0.1409	94.62 / 94.63 $\pm$0.2274
CAT + DESC	94.61 / 94.59 $\pm$0.2071	94.55 / 94.60 $\pm$0.2070
ALIAS + DESC	94.53 / 94.59 $\pm$0.1378	94.61 / 94.67 $\pm$0.0989
ALIAS + CAT	94.57 / 94.63 $\pm$0.1585	94.53 / 94.54 $\pm$0.1113
K-BERT$_{\text{MANUAL}}$	95.16 / 95.24 $\pm$0.1361	95.15 / 95.23 $\pm$0.1824

TABLE 7.3: Knowledge Ablation and Knowledge-Gating results on ag_news_subset dataset. Mean (μ-validation / μ-test $\pm\sigma$) accuracies taken across ten runs are reported.

Chapter 8

Analysis

To identify the significance of the results, one-tail Student t-tests are conducted between test results from BERT and K-BERT. Results were recorded after repeating the same experiment ten times using different seeds. These are the same results used to produce the averages reported in Chapter 7. Independent sample t-tests are performed between results from BERT and various K-BERT knowledge ablations. This section begins with an analysis of the results of the STS-B dataset. Analysis of ag_news_subset results is done thereafter.

8.1 STS-B Analysis

Inspecting the K-BERT model results in Table 7.2, we see an overall reduction in performance after injecting knowledge. Implying that the knowledge is introducing noise. It is expected that reducing this noise will improve performance. The Knowledge Ablation and Manual Knowledge Injection experiments described in Chapter 6 will achieve this.

After performing Knowledge Ablation, overall performance improves to a marginal degree compared to the standard K-BERT model. Descriptive information tends to be the most beneficial *type* of knowledge to inject, with categorical information also providing some benefit. These insights can be identified by inspecting the changes in metrics when descriptive and categorical information is removed from the model. The ALIAS + CAT variant produced the lowest MSE loss, while on average, ALIAS + DESC had the highest correlation coefficients.

However, overall results for this dataset indicate that noise exists in all knowledge types. Moreover, a manual injection of knowledge could not improve performance at all. Similar to other research [Zhang *et al.* 2021] which injects knowledge into pre-trained language models for the GLUE benchmark dataset [Wang *et al.* 2018], it can be concluded that the inclusion of knowledge for this problem is not beneficial. This conclusion is further substantiated by insignificant t-test results in Table 8.1.

8.1.1 STS-B Sentence Pair Ablation Study

The STS-B dataset contains pairs of text which may be similar. Naturally, it is possible for both sentences to contain the same words and therefore have the same information injected into them - possibly biasing sentences to have a higher similarity by the nature of the method. An ablation study was performed to investigate if such a bias does exist. Two additional experiments were conducted. The first exclusively injects information for sentence S1. And the second exclusively injects information for sentence S2. The results of the said experiment were compared to K-BERT, which has knowledge injected into both sentences (S1 + S2). As shown in Table 8.2, which

Model	Metric					
	MSE Loss		Spearman		Pearson	
	t-statistic	p-value	t-statistic	p-value	t-statistic	p-value
K-BERT	-0.5016	0.6890	0.2545	0.5990	0.3596	0.6383
After Removing						
ALIAS	-0.4561	0.6731	0.0761	0.5299	0.3750	0.6440
CAT	-0.1535	0.5602	0.3348	0.6292	0.3008	0.6165
DESC	-0.2065	0.5806	-0.1569	0.4385	0.0185	0.5073
CAT + DESC	0.1547	0.4394	-0.2858	0.3891	-0.1043	0.4590
ALIAS + DESC	-0.1526	0.5598	-0.3317	0.3720	-0.1759	0.4312
ALIAS + CAT	0.1915	0.4251	-0.1911	0.4253	-0.0772	0.4697
K-BERT$_{\text{MANUAL}}$	-0.3911	0.6498	0.3009	0.6165	0.4562	0.6732

TABLE 8.1: Student t-test results between metrics of BERT & K-BERT Models evaluated on STS-B test dataset

contains the average test results of the best-performing model across ten runs, no significant bias exists. Additional evidence can be seen in Figure 8.1, where the mean MSE loss of K-BERT is generally higher across the ten runs instead of lower - which would have indicated a bias.

Metric	S1 + S2	S1	S2
MSE Loss	2.028	2.008	2.008
Spearman	85.66	85.83	85.78
Pearson	86.70	86.82	86.78

TABLE 8.2: Comparison of test results when exclusively injecting knowledge for sequences S1, S2 and both S1 + S2.

8.2 AG News Subset Analysis

In contrast to STS-B, the ag_news_subset dataset appears to benefit more from the inclusion of knowledge. K-BERT has an improved average test accuracy compared to BERT, as shown in Table 7.3. Removing noise through Knowledge Ablation does not benefit the test performance compared to K-BERT. However, these test accuracies are generally improved when Knowledge-Gating=on, that is, when truncation is avoided. As we have seen with STS-B, all knowledge types introduce noise. Descriptive and categorical information still appears to be the most beneficial for this classification problem, with aliases providing a more significant benefit than in STS-B.

Although the reported average accuracies indicate gradual performance improvements, the t-test results in Table 8.3 conclude that the difference is statistically insignificant. Further noise removal through K-BERT$_{\text{MANUAL}}$, however, produces the best results compared to every K-BERT variation as well as the original BERT model. When Knowledge-Gating=on, performance is negatively affected to a very marginal degree. This highlights that, in the absence of noise, additional knowledge provides contextual information that can be more beneficial for this problem than the already

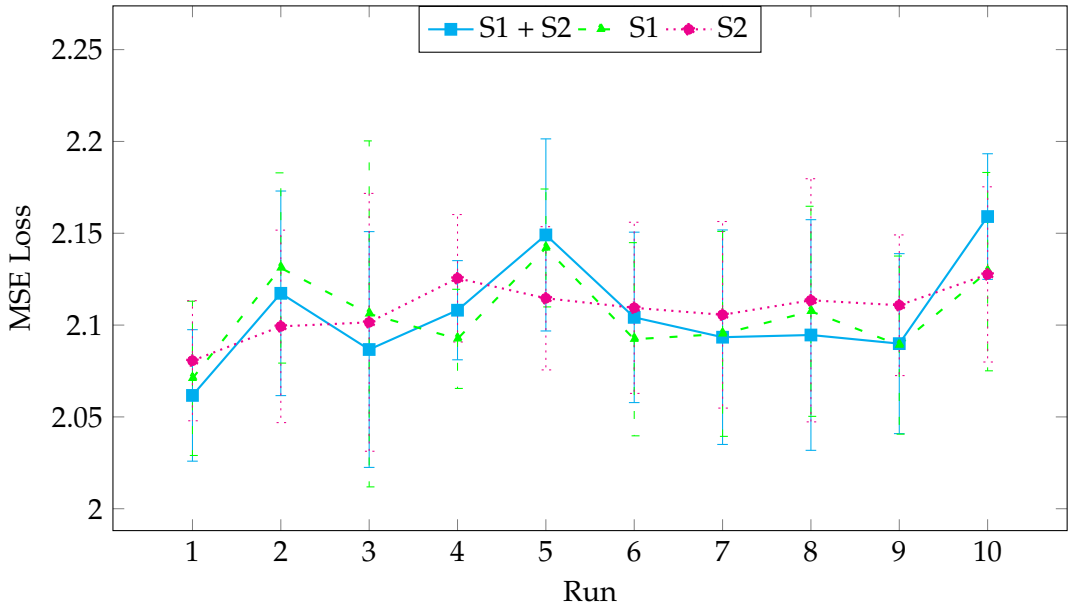


FIGURE 8.1: Mean MSE test loss with standard deviation error bars on best performing model per run. Mean is taken across ten trails.

existing information in the original sentences. Since the extracted entities for this problem such as *Intel, Microsoft, Apple, Toyota, Yankees* occur relatively frequently within the dataset, the associated additional knowledge enforces computed embeddings to be aggregated into more well-defined clusters. With a t-statistic=-8.0708 and p-value < 0.0001, K-BERT$_{\text{MANUAL}}$ has shown a statistically significant benefit with the inclusion of knowledge.

Model	**Knowledge-Gating**			
	Off		**On**	
	t-statistic	**p-value**	**t-statistic**	**p-value**
K-BERT	-0.3866	0.3518	-1.1488	0.1328
After Removing				
ALIAS	0.0100	0.5039	-0.9179	0.1854
CAT	-0.0531	0.4791	-0.8354	0.2072
DESC	-0.3869	0.3517	-0.2806	0.3911
CAT + DESC	0.1372	0.5538	0.0739	0.5290
ALIAS + DESC	0.1635	0.5640	-0.9617	0.1745
ALIAS + CAT	-0.3465	0.3665	0.8429	0.7948
K-BERT$_{\text{MANUAL}}$	-8.0708	< 0.0001	-7.0396	< 0.0001

TABLE 8.3: Student t-test results between test accuracies of BERT & K-BERT Models evaluated on ag_news_subset dataset

8.3 Discussion

In conclusion, the incorporation of knowledge from the Wikidata Knowledge Graph into the fine-tuning stages of the Language Modelling process does possess potential benefits. When inspecting Table 7.3, K-BERT with Knowledge-Gating=on produces better results and has more significant t-test scores than its corresponding Knowledge-Gating=off ablations (including but not limited to ALIAS, CAT, ALIAS + DESC). This improvement indicates that the knowledge within those categories does benefit the model. It is therefore highly probable that filtering out more noise will lead to more significant results.

Results from the Knowledge-Gating experiments highlight that existing information within the input sentence is important. Another interpretation of the improved results when Knowledge-Gating=on is that, since knowledge injection is done more sparingly, noise is reduced and performance improves. In terms of what type of knowledge is most beneficial - categorical and descriptive information tend to produce better results across both datasets.

The results reported in Chapter 7 corroborate the findings of other researchers [Zhang *et al.* 2019, 2021]. Two conclusions can be drawn. The first conclusion is that the benefits of knowledge injection are task-dependent. An open-domain semantic similarity task does not benefit much from knowledge injection as the nature of the problem is not knowledge-driven. Instead, STS-B requires an improved linguistic understanding of the current text. The ag_news_subset dataset does benefit from knowledge injection as it contains examples which require additional domain-specific context. Secondly, knowledge injection may introduce noise which can negatively affect performance. The aforementioned conclusions are both expected and substantiated by the results of this research.

Applying the model practically would first require an appropriate knowledge-driven task and corresponding source of knowledge. For example, Medical Q/A [Zeng *et al.* 2020] with medical knowledge source [Santos *et al.* 2020]. Since the Wikidata knowledge graph in this research was integrated in a loosely coupled manner, it can be easily replaced with any knowledge source stored in triple format. The success of the application would depend on the quality of the information in the knowledge source and the performance of the K_{Query} mechanism.

Although the cosine similarity-based contextual knowledge injection mechanism did not have success in the experiments conducted, it can be argued that it performed as expected due to its simplicity. Possible techniques to improve the mechanism in future work include:

1. Implementing a filtering step for disambiguating or removing noise.
2. Improve the embedding quality of the knowledge graph entity by either:
    - Utilising additional features or entity attributes.
    - Replacing cosine-similarity with a machine learning model.

The work done in this research shows evidence that K-BERT can be successfully adapted to multiple languages. Results from experiments highlight the significance and shortcomings of knowledge injection in the Language Modelling process. As a contribution to the growing evidence of knowledge being beneficial, the results here also indicate certain types of knowledge are more appropriate than others. Despite advocating for noise removal techniques, achieving such, especially in an autonomous manner, is non-trivial. The decision to use knowledge-driven approaches should therefore be dependent on the required effort versus the potential gain.

One benefit of knowledge-driven approaches is the greater interpretability it provides. Similar to work which can provide explicit sources to substantiate reasoning [Guu *et al.* 2020], K-BERT is capable of making the injected knowledge discernible. Multi-modal representation learning learns a unified representation of a text-image input [Chen *et al.* 2021]. K-BERT has the potential to also be multi-modal since the Wikidata Knowledge Graph entities can be associated with an image.

Chapter 9

Conclusion

This study investigated the incorporation of knowledge into the fine-tuning stages of Language Models for the task of semantic similarity. Since language is so versatile, it is difficult to define rule-based methods to measure the semantic differences between texts. Techniques that consider information content, using external information from knowledge sources, and neural-based corpus methods have attempted to resolve the issue with some success. More complex Transformer-based architectures have produced state-of-the-art results for a range of NLP tasks. However, the Language Models that utilise these architectures are general-purpose and often have high computational requirements. The K-BERT model by Liu *et al.* [2020] attempts to address these issues by incorporating information from a Knowledge Graph into the fine-tuning stage of BERT. Their approach injects domain knowledge into sentences for additional global context, thereby improving model performance while retaining model size.

This work extends K-BERT by adapting it for the English language and testing its effectiveness for semantic similarity tasks. The main contributions of this study are:

- K-BERT successfully integrated with Wikidata Knowledge Graph.
- An adaptation of the sentence tree and visible matrix to work on a term-level instead of a character-level. Memory optimisations for visible matrix, permitting larger datasets to be used.
- Modification of K_{Query} to inject contextually relevant information. This is based on the similarity between contextualised embeddings of the original sentence and concepts from Knowledge Graph, where contextualised embeddings are computed using a pre-trained transformer model.
- Evidence on when knowledge injection may be beneficial, and what type of knowledge is suitable.

During experiments, we analyse the effect of knowledge incorporation and compare performance to the original BERT model. Knowledge ablation studies conducted indicate that additional knowledge is beneficial, but can also introduce noise. Avoidance of truncating information from the original sentences offered more significant performance improvements. The preferred approach for this problem is a modest injection of information when space permits. Given these insights, it is recommended that knowledge injection be done sparingly, and with reliable, high-quality information.

Performance of K-BERT tends to become inconsistent with more noise, we notice this by inspecting the high p-values of Student t-tests for both ag_news_subset and STS-B. The Manual Knowledge Injection experiments conducted showed that

when this noise is minimised, K-BERT's performance on knowledge-driven tasks improves. Approaches to reduce noise such as Selective Injection and Spectral Regularization by Zhang *et al.* [2021] has also produced similar improvements. However, current automated approaches provide marginal benefits, which can often be seen as unrewarding given the non-trivial nature of the problem. Nevertheless, this work shows that external knowledge can be beneficial, especially when the context is limited. While not ideal for semantic similarity, other downstream tasks which benefit from additional domain information such as question answering could exploit K-BERT to greater fidelity.

Despite prioritising K-BERT's modification to accommodate the English language, the changes made to the sentence tree and visible matrix also accommodate other alphabetic languages. Adapting K-BERT for other writing systems remains to be explored.

Possible improvements of the contextualised method in K_{Query} could involve using a machine learning model or classifier to learn to decide on the entity to inject. Having an objective which identifies entities that complements the original sentence, instead of simply being very similar, should limit biases and offer performance improvements.

In this work, the technique for knowledge injection is similarity-based. Alternative approaches which prioritise disambiguation may perform better. This could involve an additional step of first identifying if ambiguity exists within the text, and if so, only thereafter incorporate contextually relevant information for disambiguation. This intuition proved to be successful for Pilevar *et al.* [2017] and may also apply to K-BERT.

Since the similarity-based contextualised approach proposed here was intended to work on general text, possible sources of information can easily be extended to more free-flowing text. Or, conversely, more focused feature-based approaches could use additional Knowledge Graph properties/relationships - which was left unexplored here. Since Wikidata is a multilingual knowledge source, applying K-BERT to other languages is also possible, given the appropriate pre-trained Language Model.

Appendix A

Appendix A

```
SELECT
  ?item ?itemLabel ?label
WHERE
{
  { ?item wdt:P31*/wdt:P279* wd:Q349 . } # sport
  UNION
  { ?item wdt:P31*/wdt:P279* wd:Q12973014 .} # sports team
  UNION
  { ?item wdt:P31*/wdt:P279* wd:Q5 .} # human
  UNION
  { ?item wdt:P31*/wdt:P279* wd:Q515 .} # city
  UNION
  { ?item wdt:P31*/wdt:P279* wd:Q6256 .} # country
  UNION
  { ?item wdt:P31*/wdt:P279* wd:Q891723 .} # public company
  UNION
  { ?item wdt:P31*/wdt:P279* wd:Q4830453 .} # business
  UNION
  { ?item wdt:P31*/wdt:P279* wd:Q25351891 .} # business
  UNION                                        # services
  { ?item wdt:P31*/wdt:P279* wd:Q806750 .} # banking
  UNION                                      # services
  { ?item wdt:P31*/wdt:P279* wd:Q43229 .} # organization
}
```

LISTING A.1: SPARQL query

Architecture	Pre-trained model name/path
BERT & K-BERT	bert-base-uncased
SBERT	bert-base-nli-mean-tokens
USE	https://tfhub.dev/google/universal-sentence-encoder-large/5
Similarity Transformer	nli-distilroberta-base-v2

TABLE A.1: Pre-trained models used

Bibliography

[Har 1969] Ieee recommended practice for speech quality measurements. *IEEE No 297-1969*, pages 1–24, 1969. doi: 10.1109/IEEESTD.1969.7405210.

[wik 2016a] 2016a. URL https://www.wikidata.org/wiki/Wikidata:Introduction.

[wik 2016b] 2016b. URL https://www.wikidata.org/wiki/Help:Items.

[wik 2022] 2022. URL https://www.wikidata.org/wiki/Wikidata:Main_Page.

[Abadi *et al.* 2015] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/tutorials/text/word2vec. Software available from tensorflow.org.

[Adhikari *et al.* 2019] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*, 2019.

[Al-Roomi *et al.* 2013] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013.

[Aldossary *et al.* 2019] Mohammad Aldossary, Karim Djemame, Ibrahim Alzamil, Alexandros Kostopoulos, Antonis Dimakis, and Eleni Agiatzidou. Energy-aware cost prediction and pricing of virtual machines in cloud computing environments. *Future Generation Computer Systems*, 93:442–459, 2019. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2018.10.027. URL https://www.sciencedirect.com/science/article/pii/S0167739X18310288.

[Alotaibi 2017] Hind Alotaibi. Arabic-english parallel corpus: A new resource for translation training and language teaching. *Arab World English Journal*, 8:319–337, 09 2017. doi: 10.24093/awej/vol8no3.21.

[Ansell *et al.* 2021] Alan Ansell, Felipe Bravo-Marquez, and Bernhard Pfahringer. Polylm: Learning about polysemy through language modeling. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 563–574, 2021.

[Auer *et al.* 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon

Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76298-0.

[Bahdanau *et al.* 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[Bengio *et al.* 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003. ISSN 1532-4435.

[Berger *et al.* 2000] Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. Bridging the lexical chasm: Statistical approaches to answer-finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '00, page 192–199. Association for Computing Machinery, 2000. ISBN 1581132263. doi: 10.1145/345508.345576. URL https://doi.org/10.1145/345508.345576.

[Berry *et al.* 1998] Kenneth J. Berry, Jr. Paul W. Mielke, and Hariharan K. Iyer. Factorial designs and dummy coding. *Perceptual and Motor Skills*, 87(3):919–927, 1998. doi: 10.2466/pms.1998.87.3.919. URL https://doi.org/10.2466/pms.1998.87.3.919.

[Bobadilla *et al.* 2013] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[Bojanowski *et al.* 2016] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 07 2016. doi: 10.1162/tacl_a_00051.

[Bojar *et al.* 2014] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3302. URL https://aclanthology.org/W14-3302.

[Bowman *et al.* 2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075.

[Brown *et al.* 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances*

in Neural Information Processing Systems, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[Carlson *et al.* 2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom Michael Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[Cer *et al.* 2017] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. 08 2017. doi: 10.18653/v1/S17-2001.

[Cer *et al.* 2018] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2029. URL https://aclanthology.org/D18-2029.

[Chandrasekaran and Mago 2021] Dhivya Chandrasekaran and Vijay Mago. Evolution of semantic similarity—a survey. *ACM Comput. Surv.*, 54(2), February 2021. ISSN 0360-0300. doi: 10.1145/3440755. URL https://doi.org/10.1145/3440755.

[Chen *et al.* 2021] Kezhen Chen, Qiuyuan Huang, Yonatan Bisk, Daniel McDuff, and Jianfeng Gao. Kb-vlp: Knowledge based vision and language pretraining. In *Proceedings of the 38 th International Conference on Machine Learning, PMLR 139, 2021. ICML, workshop, 2021*, July 2021. URL https://www.microsoft.com/en-us/research/publication/kb-vlp-knowledge-based-vision-and-language-pretraining-2/.

[Cheng *et al.* 2016] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1053. URL https://aclanthology.org/D16-1053.

[Cho *et al.* 2014] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179.

[Devlin *et al.* 2019] 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.

[Dharma and Soewito 2022] EDDY MUNTINA Dharma and BENFANO Soewito. The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification. *Journal of Theoretical and Applied Information Technology*, 100(2):31, 2022.

[Dong *et al.* 2017] Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. Improving interpretability of deep neural networks with semantic information. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 975–983, 2017.

[Du *et al.* 2019] Jiaju Du, Fanchao Qi, and Maosong Sun. Using bert for word sense disambiguation, 2019.

[El-Khair 2016] Ibrahim Abu El-Khair. 1.5 billion words arabic corpus. *ArXiv*, abs/1611.04033, 2016.

[Etzioni *et al.* 2004] Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW '04*, 2004.

[Frost 1986] Richard A. Frost. Introduction to knowledge base systems. 1986.

[Gabrilovich and Markovitch 2007] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. volume Vol. 6, 01 2007.

[Gage 1994] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788.

[Guo *et al.* 2020] Weiwei Guo, Xiaowei Liu, Sida Wang, Huiji Gao, Ananth Sankar, Zimeng Yang, Qi Guo, Liang Zhang, Bo Long, Bee-Chung Chen, *et al.* Detext: A deep text ranking framework with bert. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2509–2516, 2020.

[Guu *et al.* 2020] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[Harris 1954] Zellig S. Harris. Distributional structure. WORD, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL https://doi.org/10.1080/00437956.1954.11659520.

[Hill *et al.* 2015] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[Honnibal 2013] Matthew Honnibal. Parsing english in 500 lines of python, 2013. URL https://explosion.ai/blog/parsing-english-in-python.

[Honnibal 2016] Matthew Honnibal. Embed, encode, attend, predict: The new deep learning formula for state-of-the-art nlp models, 2016. URL https://explosion.ai/blog/deep-learning-formula-nlp.

[Iqbal *et al.* 2021] MD. Asif Iqbal, Omar Sharif, Mohammed Moshiul Hoque, and Iqbal H. Sarker. Word embedding based textual semantic similarity measure in bengali. *Procedia Computer Science*, 193:92–101, 2021. ISSN 1877-0509. doi: https:

//doi.org/10.1016/j.procs.2021.10.010. URL https://www.sciencedirect.com/science/article/pii/S1877050921020512. 10th International Young Scientists Conference in Computational Science, YSC2021, 28 June – 2 July, 2021.

[Iyyer *et al.* 2015] 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 Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1162. URL https://aclanthology.org/P15-1162.

[Jaro 1989] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989. ISSN 01621459. URL http://www.jstor.org/stable/2289924.

[Ji *et al.* 2022] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, February 2022. ISSN 2162-237X. doi: 10.1109/TNNLS.2021.3070843. Publisher Copyright: IEEE Copyright: Copyright 2021 Elsevier B.V., All rights reserved.

[Kalchbrenner and Blunsom 2013] Nal Kalchbrenner and P. Blunsom. Recurrent continuous translation models. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 3:1700–1709, 01 2013.

[Lample *et al.* 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL https://aclanthology.org/N16-1030.

[Landauer 1997] Thomas K. K Landauer. A solution to plato's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review.*, 104(2), 1997. ISSN 0033-295X.

[Lee *et al.* 2005] Michael D Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. In *Proceedings of the annual meeting of the cognitive science society*, volume 27, 2005.

[Levenshtein 1965] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710, 1965.

[Li *et al.* 2018] Hongmin Li, Xukun Li, Doina Caragea, and Cornelia Caragea. Comparison of word embeddings and sentence encodings as generalized representations for crisis tweet classification tasks. *Proceedings of ISCRAM Asia Pacific*, 2018.

[Lin *et al.* 2017] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=BJC_jUqxe.

[Liu *et al.* 2020] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-BERT: enabling language representation with knowledge graph. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2901–2908. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/5681.

[Liu *et al.* 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL http://arxiv.org/abs/1907.11692. cite arxiv:1907.11692.

[Lu *et al.* 2020] Zhibin Lu, Pan Du, and Jian-Yun Nie. Vgcn-bert: Augmenting bert with graph embedding for text classification. In Joemon M. Jose, Emine Yilmaz, João Magalhães, Pablo Castells, Nicola Ferro, Mário J. Silva, and Flávio Martins, editors, *Advances in Information Retrieval*, pages 369–382, Cham, 2020. Springer International Publishing. ISBN 978-3-030-45439-5.

[Lund and Burgess 1996] Kevin Lund and Curt Burgess. Producing high-dimensional semantic space from lexical co-occurence. *Behavior Research Methods Instruments & Computers*, 28:203–208, 06 1996. doi: 10.3758/BF03204766.

[Luong *et al.* 2015] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

[Majumder *et al.* 2016] Goutam Majumder, Dr. Partha Pakray, Alexander Gelbukh, and David Pinto. Semantic textual similarity methods, tools, and applications: A survey. *Computacion y Sistemas*, 20:647–665, 12 2016. doi: 10.13053/CyS-20-4-2506.

[Marneffe and Manning 2008] Marie-Catherine Marneffe and Christopher Manning. The stanford typed dependencies representation. *COLING Workshop on Cross-framework and Cross-domain Parser Evaluation*, 01 2008. doi: 10.3115/1608858.1608859.

[Marsi and Krahmer 2013] Erwin Marsi and Emiel Krahmer. *Automatic Tree Matching for Analysing Semantic Similarity in Comparable Text*, pages 129–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-30910-6. doi: 10.1007/978-3-642-30910-6_8. URL https://doi.org/10.1007/978-3-642-30910-6_8.

[McCloskey and Cohen 1989] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

[Merity *et al.* 2016] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[Mikolov *et al.* 2013a] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and

Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781.

[Mikolov *et al.* 2013b] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013b. Association for Computational Linguistics. URL https://aclanthology.org/N13-1090.

[Miller 1998] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998. URL https://doi.org/10.7551/mitpress/7287.001.0001.

[Navigli and Ponzetto 2012] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2012.07.001. URL https://www.sciencedirect.com/science/article/pii/S0004370212000793.

[Nguyen *et al.* 2019] Hien T. Nguyen, Phuc H. Duong, and Erik Cambria. Learning short-text semantic similarity with word embeddings and external knowledge sources. *Knowledge-Based Systems*, 182:104842, 2019. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.07.013. URL https://www.sciencedirect.com/science/article/pii/S095070511930317X.

[Noy *et al.* 2019] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, 62 (8):36–43, 2019. URL https://cacm.acm.org/magazines/2019/8/238342-industry-scale-knowledge-graphs/fulltext.

[Ormerod *et al.* 2021] Mark Ormerod, Jesús Martínez del Rincón, and Barry Devereux. Predicting semantic similarity between clinical sentence pairs using transformer models: Evaluation and representational analysis. *JMIR Med Inform*, 9(5):e23099, 05 2021. ISSN 2291-9694. doi: 10.2196/23099. URL https://doi.org/10.2196/23099.

[Pellissier Tanon *et al.* 2020] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. Yago 4: A reason-able knowledge base. In Andreas Harth, Sabrina Kirrane, Axel-Cyrille Ngonga Ngomo, Heiko Paulheim, Anisa Rula, Anna Lisa Gentile, Peter Haase, and Michael Cochez, editors, *The Semantic Web*, pages 583–596, Cham, 2020. Springer International Publishing. ISBN 978-3-030-49461-2.

[Peters *et al.* 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[Petroni *et al.* 2019] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, AH Miller, and Sebastian Riedel. Language models as knowledge bases? 01 2019. doi: 10.18653/v1/D19-1250.

[Pilevar *et al.* 2017] Mohammad Taher Pilevar, José Camacho-Collados, Roberto Navigli, and Nigel Collier. Towards a seamless integration of word senses into downstream nlp applications. pages 1857–1869, 10 2017. doi: 10.18653/v1/P17-1170.

[Prud'hommeaux and Seaborne 2008] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf, 2008. URL https://www.w3.org/TR/rdf-sparql-query/.

[Rada *et al.* 1989] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, 1989. doi: 10.1109/21.24528.

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

[Raffel *et al.* 2020a] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020a. URL http://jmlr.org/papers/v21/20-074.html.

[Raffel *et al.* 2020b] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020b.

[Ramos *et al.* 2003] Juan Ramos *et al.* Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

[Reimers *et al.* 2019] Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019.

[Resnik 1995] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, page 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603638.

[Rodriguez and Egenhofer 2003] M.A. Rodriguez and M.J. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456, 2003. doi: 10.1109/TKDE.2003.1185844.

[Salton and Buckley 1988] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing: Management*, 24(5): 513–523, January 1988. doi: 10.1016/0306-4573(88)90021-0. URL https://doi.org/10.1016%2F0306-4573%2888%2990021-0.

[Santos *et al.* 2020] Alberto Santos, Ana Colaço, Annelaura Nielsen, Niu Lili, Philipp Geyer, Fabian Coscia, Nicolai Wewer Albrechtsen, Filip Mundt, Lars Jensen, and Matthias Mann. Clinical knowledge graph integrates proteomics data into clinical decision-making. 05 2020. doi: 10.1101/2020.05.09.084897.

[Schroff *et al.* 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. doi: 10.1109/cvpr.2015.7298682. URL http://dx.doi.org/10.1109/CVPR.2015.7298682.

[Schuster and Nakajima 2012] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.

[Schuster and Manning 2016] Sebastian Schuster and Christopher D. Manning. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378, Portorož, Slovenia, 05 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1376.

[Sinha *et al.* 2015] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *WWW - World Wide Web Consortium (W3C)*, May 2015. URL https://www.microsoft.com/en-us/research/publication/an-overview-of-microsoft-academic-service-mas-and-applications-2/.

[Sinoara *et al.* 2019] Roberta A. Sinoara, Jose Camacho-Collados, Rafael G. Rossi, Roberto Navigli, and Solange O. Rezende. Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems*, 163:955–971, 2019. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2018.10.026. URL https://www.sciencedirect.com/science/article/pii/S0950705118305124.

[Song *et al.* 2020] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[Sun *et al.* 2019a] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. 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 380–385, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1035. URL https://www.aclweb.org/anthology/N19-1035.

[Sun *et al.* 2019b] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In Maosong Sun, Xuanjing Huang, Heng Ji, Zhiyuan Liu, and Yang Liu, editors, *Chinese Computational Linguistics*, pages 194–206, Cham, 2019b. Springer International Publishing. ISBN 978-3-030-32381-3.

[Sun *et al.* 2017] Zequn Sun, Wei Hu, and Chengkai Li. Cross-lingual entity alignment via joint attribute-preserving embedding. In Claudia d'Amato, Miriam Fernandez, Valentina Tamma, Freddy Lecue, Philippe Cudré-Mauroux, Juan Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web – ISWC 2017*, pages 628–644, Cham, 2017. Springer International Publishing. ISBN 978-3-319-68288-4.

[Sutskever *et al.* 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[Swick and Lassila 1999] Ralph R. Swick and Ora Lassila. Resource description framework (rdf) model and syntax specification, 1999. URL https://www.w3.org/TR/PR-rdf-syntax/Overview.html.

[Sánchez *et al.* 2012] David Sánchez, Montserrat Batet, David Isern, and Aida Valls. Ontology-based semantic similarity: A new feature-based approach. *Expert Systems with Applications*, 39:7718–7728, 03 2012. doi: 10.1016/j.eswa.2012.01.082.

[Tien *et al.* 2019] Nguyen Huy Tien, Nguyen Minh Le, Yamasaki Tomohiro, and Izuha Tatsuya. Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity. *Information Processing & Management*, 56(6):102090, 2019. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2019.102090. URL https://www.sciencedirect.com/science/article/pii/S0306457319301335.

[Turc *et al.* 2019] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models, 2019.

[Turian *et al.* 2010] Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-1040.

[Union 2021] International Telecommunication Union. *Measuring digital development: Facts and Figures 2021*. 2021. ISBN 978-92-61-35401-5. URL https://www.itu.int/en/mediacentre/Pages/PR-2021-11-29-FactsFigures.aspx.

[Vaswani *et al.* 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010. Curran Associates Inc., 2017. ISBN 9781510860964.

[Vrandečić and Krötzsch 2014] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, September 2014. ISSN 0001-0782. doi: 10.1145/2629489. URL https://doi.org/10.1145/2629489.

[Wang *et al.* 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018. URL http://arxiv.org/abs/1804.07461.

[Weiner 1973] Peter Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 1–11, 1973. doi: 10.1109/SWAT.1973.13.

[Williams *et al.* 2018] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

[Wu and Palmer 1994] 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. Association for Computational Linguistics, 1994. doi: 10.3115/981732.981751. URL https://doi.org/10.3115/981732.981751.

[Yan *et al.* 2016] Jihong Yan, Chengyu Wang, Wenliang Cheng, Ming Gao, and Aoying Zhou. A retrospective of knowledge graphs. *Frontiers of Computer Science*, 12, 09 2016. doi: 10.1007/s11704-016-5228-9.

[Yang *et al.* 2017] Tien-Ju Yang, Yu-Hsin Chen, Joel Emer, and Vivienne Sze. A method to estimate the energy consumption of deep neural networks. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 1916–1920, 2017. doi: 10.1109/ACSSC.2017.8335698.

[Yu *et al.* 2022] Changlong Yu, Hongming Zhang, Yangqiu Song, and Wilfred Ng. CoCoLM: Complex commonsense enhanced language model with discourse relations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1175–1187, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.93. URL https://aclanthology.org/2022.findings-acl.93.

[Zaheer *et al.* 2020] Manzil Zaheer, Guru Prashanth Guruganesh, Avi Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Minh Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Mahmoud El Houssieny Ahmed. Big bird: Transformers for longer sequences. 2020. URL https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

[Zeng *et al.* 2020] Guangtao Zeng, Wenmian Yang, Zeqian Ju, Yue Yang, Sicheng Wang, Ruisi Zhang, Meng Zhou, Jiaqi Zeng, Xiangyu Dong, Ruoyu Zhang, Hongchao Fang, Penghui Zhu, Shu Chen, and Pengtao Xie. MedDialog: Large-scale medical dialogue datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9241–9250, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.743. URL https://aclanthology.org/2020.emnlp-main.743.

[Zhang *et al.* 2021] Ningyu Zhang, Shumin Deng, Xu Cheng, Xi Chen, Yichi Zhang, Wei Zhang, and Huajun Chen. Drop redundant, shrink irrelevant: Selective knowledge injection for language pretraining. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4007–4014. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/552. URL https://doi.org/10.24963/ijcai.2021/552.

[Zhang *et al.* 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA, 2015. MIT Press.

[Zhang *et al.* 2019] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: enhanced language representation with informative entities. *CoRR*, abs/1905.07129, 2019. URL http://arxiv.org/abs/1905.07129.

[Zhou *et al.* 2019] Zhixuan Zhou, Guan Huankang, Meghana Bhat, and Justin Hsu. Fake news detection via nlp is vulnerable to adversarial attacks. pages 794–800, 01 2019. doi: 10.5220/0007566307940800.

[Zhu *et al.* 2015] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, Los Alamitos, CA, USA, 12 2015. IEEE Computer Society. doi: 10.1109/ICCV.2015.11. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.11.