Non-Parameterized Sentence Embeddings

by Nada Almarwani

B.Sc. in Computer Science, May 2007, Taibah University
M.Phil. in Computer Science, May 2013, The George Washington University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

August 31, 2020

Dissertation directed by

Mona T. Diab
Professor of Computer Science

The School of Engineering and Applied Science of The George Washington University certifies that Nada Almarwani has passed the Final Examination for the degree of Doctor of Philosophy as of Jun 26, 2020. This is the final and approved form of the dissertation.

Non-Parameterized Sentence Embeddings

Nada Almarwani

Dissertation Research Committee:

Mona T. Diab, Professor of Computer Science, Dissertation Director

Abdou Youssef, Professor of Computer Science, Committee Member

Robert Pless, Professor of Computer Science, Committee Member

Aylin Caliskan, Assistant Professor of Computer Science, Committee Member

Veselin Stoyanov, Research Scientist Manager, Facebook AI, Committee Member

© Copyright 2020 by Nada Almarwani
All rights reserved

Dedication

This dissertation is dedicated to my mother, Najah, my first teacher, who has always been the guiding light in my life; my father, Obaid, my greatest cheerleader, who always supports me to pursue what interests me; and my beloved grandmothers (Falha and Munirah) and in loving memory of my grandfathers (Faleh, Hamdan, and Awad). I am blessed with their endless love and encouragement.

Acknowledgments

The realization of this work was only possible due to the support and guidance of my professors, family, and friends, to whom I would like to express my gratitude.

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Mona Diab, for her continuous support and guidance during my PhD journey. I would also like to thank the rest of my dissertation committee — Professors Aylin Caliskan, Robert Pless, Veselin Stoyanov, and Abdou Youssef — for their insightful feedback and discussions. I would also like to express my deepest appreciation to my friend and collaborator Hanan Aldarmaki for pointing me in the right direction during this work and devoting her time to the discussion and interpretation of some of the results presented in this dissertation.

Also, I extend my sincere thanks to all those who have helped me during this journey including my sponsor (Taibah University), proofreader (Kyra Armstrong), colleagues, and anonymous reviewers. I would also like to thank my best friend Fatimah Alshamari, who always been there, in good and bad times.

Last but not least, the completion of this dissertation would not have been possible without the support of my parents and siblings. Thanks Mom, Dad, Adel, Abdullah, Bandar, Sara, Meaad, and Faisal for your love, encouragement, and tolerance.

Abstract

Non-Parameterized Sentence Embeddings

The field of Natural Language Processing (NLP) has progressed rapidly in recent years due to the evolution of deep neural models and their constituent, word embeddings. While word embeddings are often used as a primary component, the ultimate goal of any NLP system is to capture the underlying linguistic characteristics of word sequences, including phrases, sentences, or paragraphs, that are ideal for an end task. To generate these kinds of embeddings, most NLP models rely on a mathematical operation, such as averaging or other pooling mechanisms, over smaller units such as words, morphemes, or even characters. However, these representations tend to be task-specific and typically perform poorly when transferred to other tasks. Consequently, different models have been proposed to generate general-purpose sentence embeddings to be used in a pretraining kind of protocol. The most notable differences between these models are the efficiency and performance trade-offs. However, to date, most proposed embedding models suffer from indifference toward underlying syntactic and semantic characteristics of the text.

To this end, in this dissertation, we develop an efficient sentence embedding model that is capable of capturing both syntactic and semantic properties. First, we use Discrete Cosine Transform (DCT) to compress word sequences in an order-preserving manner. The lower-order DCT coefficients represent the overall feature patterns in sentences, which result in suitable embeddings for tasks that could benefit from syntactic features. Our results in semantic probing tasks demonstrate that DCT embeddings indeed preserve more syntactic information compared with most commonly used approach, vector averaging. With practically equivalent complexity, the DCT model yields better overall performance in downstream classification tasks that correlate with syntactic features. This illustrates the capacity of DCT to preserve word-order information.

We further validate the efficiency of our DCT embedding in multi- and cross-lingual

settings. Specifically, we investigate the generality of the representations across different languages that exhibit different linguistic properties as a language-independent model and a cross-lingual model. We empirically show that the performance of the DCT embeddings is comparable across different languages for all examined tasks. Moreover, in the cross-lingual setting, DCT embeddings resulted in superior performance in sentence translation retrieval compared to the other state-of-the-art models across all language pairs. These results reaffirmed the power of the structural properties encoded on the lower-order DCT coefficients which are used to generate the final fixed-length sentence

A major weakness of DCT, however, is loss of linguistic information, e.g. "man bitten by dog" embeddings rendered from the lower-order DCT coefficients are more similar to "man bites dog" embeddings than the semantically similar example "dog bites man". To address this deficiency, we propose to explicitly model linguistic information in the DCT framework using a block based representation protocol. The blocks reflect various levels of linguistic representation such as ngram chunk, syntactic dependencies and shallow semantic representations. Overall, our results show that augmenting the DCT encoding with Block-based representations improves performance relative to the vanilla baseline (sentence only encoding) for both probing and downstream classification tasks.

Table of Contents

Dedication . . . . . **iv**

Acknowledgments . . . . . **v**

Abstract . . . . . **vi**

List of Figures . . . . . **xi**

List of Tables . . . . . **xiii**

Preface . . . . . **xvii**

1. Introduction **1**

2. Literature Review **5**

1 Sentence Embedding Models 5

1.1 Unsupervised Sentence Representations 6

1.2 Supervised Sentence Representations 8

1.3 Contextualized Sentence Representation 10

1.4 Non-parameterized Sentence Representations 12

2 Discrete Cosine Transform 14

3. Sentence Embeddings Using Discrete Cosine Transform **17**

1 Background: Discrete Cosine Transform in Image Processing 17

2 Discrete Cosine Transform as a Sentence Encoder 18

2.1 A Note on Complexity 20

3 Evaluation Dataset 20

4 Experiment Setups 24

5 Evaluation Results 24

6 Comparison with State-of-The-Art Sentence Encoders 28

7 Comparison with Spectral Analysis Methods 30

8 Contextualized DCT Embeddings . . . 32
9 Analysis . . . 37
9.1 AVG vs. c[0] . . . 37
9.2 AVG vs. c[k] . . . 38
9.3 Sensitivity to Sequence Length . . . 41
9.4 Sensitivity to Linguistic phenomena . . . 42
10 Conclusion . . . 45
4. Universality of DCT Embeddings . . . 46
1 Mulit- and Cross-lingual Sentence Encoding . . . 46
2 Related Work . . . 48
3 Multi-lingual DCT Embeddings . . . 50
3.1 Evaluation Datasets and Experimental Setups . . . 51
3.2 Evaluation Results . . . 51
4 Cross-lingual Mapping based on DCT Encoding . . . 54
4.1 Approach . . . 54
4.1.1 Orthogonal Mapping . . . 55
4.2 Experiments . . . 56
4.3 Evaluation Datasets . . . 56
4.4 Results . . . 58
5 Conclusion . . . 62
5. Linguistically Guided Discrete Cosine Transform Sentence Encoding . . . 64
1 Approach . . . 65
1.1 General Architecture . . . 65
1.2 N-gram Guided Representation . . . 66
1.3 Syntactically Guided Representation . . . 68

2 Evaluation 72
2.1 Results 72
3 Detailed Analysis 79
3.1 Sensitivity to Sequence Length 80
3.2 Analysis of Pairwise Classification Tasks 80
4 Conclusion 82
6. Conclusions and Future Directions 84
1 Summary of Contributions 84
2 Limitations and Future Directions 87
Bibliography 90

List of Figures

Figure 3.1 An illustration of DCT embeddings. The size of the sentence to encode is 4×5, where 4 is the number of words and 5 is the number of word embedding dimensions. Each feature vector is transformed using DCT independently. In this example, the final representation is the concatenation of the first two coefficient from all transformed features. 18

Figure 3.2 Illustration of word vector averaging vs. DCT using the first 2 DCT coefficients. The word vectors are generated randomly from a standard normal distribution with $d = 10$. 19

Figure 3.3 Sentence embedding performance of BERT cls token, $c[0]$, $c[0:1]$, and $c[0:2]$ from different layers of BERT. Color value of 90 corresponds to the best performance on a given task. Numbers on the x-axis denote the BERT layer. 35

Figure 3.5 T-SNE visualization of different DCT coefficients and AVG on the TREC dataset. 39

Figure 3.6 Correlation between clustering purity and increasing number of coefficients on the TREC data. 41

Figure 3.7 Comparing number of words per sentence distribution in the TREC train and test sets. 41

Figure 3.8 TREC accuracy vs. sentence length for AVG and various DCT models. . 41

Figure 4.1 The results of the probing tasks relative to English results for each model. The white indicates a value of 1, demonstrating parity in performance. Red indicates better English performance while green indicates better X language performance. 53

Figure 5.1 The overall architecture for Block Based DCT Embeddings. 66

Figure 5.2 An example of constituent based division that is used in the constit-grams model, where (a) is the constituent parse tree of the given sentence generated from Stanford CoreNLP tool, and (b) is the corresponding chunks. In this example, the root, which comprised all words in the sentence will not be part of the constit-grams. 70

Figure 5.3 An example of dependency based division that is used in the depend-grams model, where (a) is the dependency parse tree of the given sentence generated from Stanford CoreNLP tool, and (b) is the corresponding chucks division. 71

Figure 5.4 An example of dependency based division that is used in the SPO-grams model, where (a) is the relation tree of the given sentence generated from Open IE in Stanford CoreNLP tool, and (b) is the corresponding chucks division. 71

Figure 5.5 Impact of Block-level models on the Probing and Downstream tasks relative to the BASELINE model reflecting the best results obtained using either $c[0]$ or $c[0:1]$. White indicates a value of 1, demonstrating parity in performance, while green and red indicate better and worse performance, respectively. 76

Figure 5.6 Impact of Block-level DCT coefficients $c[0:1]$ relative to $c[0]$ on the Probing and Downstream tasks. 78

List of Tables

Table 3.1 Probing Tasks Description and Samples 21

Table 3.2 Classification tasks. C is the number of classes and N is the number of samples (Conneau and Kiela, 2018). Note this table is reprinted from (Conneau and Kiela, 2018). 23

Table 3.3 Natural Language Inference and Semantic Similarity tasks. NLI labels are contradiction, neutral and entailment. STS labels are scores between 0 and 5. PD=paraphrase detection (Conneau and Kiela, 2018). Note this table is reprinted from (Conneau and Kiela, 2018). 23

Table 3.4 Probing tasks performance of vector averaging (AVG) and max pooling (MAX) vs. DCT embeddings with various K. Majority (baseline), Human (human-bound), and a linear classifier with sentence length as the sole feature (Length) as reported in (Conneau et al., 2018a), respectively. The σ denotes the standard deviations for the performance across different DCT coefficients. 25

Table 3.5 DCT embedding Performance in SentEval downstream tasks compared to vector averaging (AVG) and max pooling (MAX). The σ denotes the standard deviations for the performance across different DCT coefficients. . 26

Table 3.6 Evaluation of weighted embedding using SIF weighting schema on probing tasks. Both (weighted and unweighted models) trained using logistic regression only. 27

Table 3.7 Evaluation of weighted embedding using SIF weighting schema on downstream tasks. Both (weighted and unweighted models) trained using logistic regression only. 27

Table 3.8 DCT embedding Performance in SentEval Probing tasks compared to SOTA models including SkipThought (unsupervised model), Infersent (supervised model), and different random encoders (BOE, BOREP, RandLSTM, and ESN) as reported in (Radford et al.). **Bold** highlights the best overall results, while **gray** highlights the best results for DCT encoders. 28

Table 3.9 DCT embedding Performance in SentEval downstream tasks compared to SOTA models including SkipThought (unsupervised model), Infersent (supervised model), and different random encoders (BOE, BOREP, RandLSTM, and ESN) as reported in (Radford et al.). **Bold** highlights the best overall results, while **gray** highlights the best results for DCT encoders. . . 29

Table 3.10 Performance in text classification (20-NG, R-8) and sentiment (SST-5) tasks of various models as reported in (Kayal and Tsatsaronis, 2019), where DCT* refers to the implementation in (Kayal and Tsatsaronis, 2019). Our DCT embeddings are denoted as c[k] in the bottom row, with k=0:2 for the SST-5 and k=0:1 for 20-NG and R-8 datasets. **Bold** indicates the best result, and *italic* indicates second-best. 30

Table 3.11 Run time of the Non-Parameterized Models on encoding sentences per second in 20-NG test set. Results are obtained from 5 trials average. 31

Table 3.12 The results of the layer-wise BERT using the different pooling mechanisms: CLS, $c[0]$, $c[0:1]$, and $c[0:2]$ on the probing tasks. **Bold** highlights the layer-wise best result and **gray** highlights the best results across all layers. 33

Table 3.13 The results of the layer-wise BERT using the different pooling mechanisms: CLS, $c[0]$, $c[0:1]$, and $c[0:2]$ on the downstream tasks. **Bold** highlights the layer-wise best result and **gray** highlights the best results across all layers. 34

Table 3.14 Comparison of BERT cls and BERT-DCT models on the probing tasks. FT refer to the DCT model results based on FastText word embeddings.**Bold** denotes the best results across BERT embeddings, and **gray** highlights the best results across all models. 36

Table 3.15 Comparison of BERT cls and BERT-DCT models on the downstream tasks. FT refer to the DCT model results based on FastText word embeddings.**Bold** denotes the best results across BERT embeddings, and **gray** highlights the best results across all models. 36

Table 3.16 Nearest neighbors (based on the cosine similarity) to: Query 1 and Query 2. The label refer to the question categorizes as defined on the TREC data set, where DESC denotes description and NUM denotes number. The **green** highlights the level of noise in the retrieved results. While in "Query 1" the level of noise impacted the categorizes of the retrieved examples, in "Query 2" the level of noise impacted in the structural characteristic of the retrieved examples. 40

Table 3.17 Clustering purity with increasing k and DCT coefficients on TREC . . . 41

Table 3.18 Accuracy against sentence length on the TREC test set. The number of words in a sentence, in bins of size 5, where Bin-1 contains sentences with length [1-5], Bin-2 with [6-10], and Bin-3 with [11-]. 42

Table 3.19 Examples of sentence pairs in SICK with different linguistic phenomena including: Negation, Passivization, and Substitution. 42

Table 3.20 Confusion matrix of different models when negation is used. 43

Table 3.21 Confusion matrix of different models when passivization is used. 44

Table 3.22 Confusion matrix of different models when substitution is used. 44

Table 4.1 DCT embeddings Performance per language compared to AVG. **EN**=English, **ES**=Spanish, **FR**=French, **DE**=German, and **RU**=Russian 52

Table 4.2 Sentence translation retrieval accuracy. The arrows indicate the translation direction, with EN, ES, DE refer to English, Spanish, and German, respectively. 58

Table 4.3 Sentence translation retrieval accuracy. The arrows indicate the translation direction, with EN, DE, RU, CS, FI, TR, and LV stands for English, German, Russian, Czech, Finnish, Turkish, and Latvian, respectively 60

Table 4.4 Sentence translation retrieval accuracy using in-domain FastText and Contextualized mBERT embeddings. **Bold** denotes the best results for each row, and **gray** highlights the best overall result for each language direction. 61

Table 4.5 The average accuracy of various models across all language retrieval directions as reported in (Aldarmaki and Diab, 2019a) along with the different DCT-based models in this work, in which (word) refers to word-level mapping, (sent) refers to sentence-level mapping, and (dict) refers to the baseline (using a static dictionary for mapping). **Bold** shows the best overall result. 62

Table 5.1 Probing tasks performance of Block-based DCT models, which includes NGRAMS(2), NGRAMS(3), CONST, DEP, and SPO, with various K using logistic regression, vs. DCT embeddings with various K using logistic regression. **Bold** indicates the best result in each models types block, and **gray-highlight** indicates the best result among all models. 74

Table 5.2 Downstream tasks performance of sentence-level DCT embeddings vs. block-based DCT models, which includes NGRAMS(2), NGRAMS(3), CONST, DEP, and SPO, with various K using logistic regression. **Bold** indicates the best result in each models types block, and **gray-highlight** indicates the best result among all models 75

Table 5.3 Comparing Base Sentence encoders, $Sent.c[0]$, $Sent.c[0:1]$, $Sent.c[0:2]$, each concatenated with or without $DEP.c[0]$ on the TREC Task. Overall accuracy reflects the overall results on the entire test set, Bin-* indicates the results on the specific cluster of sentences of length 1-5, 6-10, 11+ tokens, respectively. 79

Table 5.4 List of modifiers from the Stanford typed dependencies used in the enhanced-grams model 81

Table 5.5 Examples for blocks produced by SPO-Enh model from the Stanford CoreNLP dependency parse tree. 81

Table 5.6 Results of the SPO-Enh model $SPO - Enh.c[0]$ vs. $SPO.c[0]$ on the SICK data sets: SICK-E and SICK-R. 82

Preface

This dissertation outlines the contribution of my PhD research to the field of NLP. I am the main author of all the chapters in this dissertation that were published or are intended to be published. The published paper is freely available online and has been reproduced in this dissertation with minor modification and extended analysis. The contributions to this work including original and cited content, as well as my collaborations with coauthors are outlined below.

While the content of the introduction in Chapter 1 was partially modified from (Almarwani et al., 2019), it has been largely extended in this dissertation. The literature review in Chapter 2, which provides comprehensive background for my research, is published in this dissertation for the first time. In Chapter 3, Section 1 to Section 5 and Section 7 are almost entirely reproduced from (Almarwani et al., 2019), in which Hanan Aldarmaki (coauthor) was involved in the development of the idea and contributed to the writing and analysis of the published paper. I added essential results in Section 6, Section 8, as well as various analytical findings in Section 9 to Chapter 3, which are original to this dissertation and have not been published yet. I developed and conducted all approaches, experiments, and analyses described in Chapter 4 and Chapter 5, which are original contributions and currently under review for possible publication in ACL-related conferences. It is worth noting that Dr. Mona Diab, the supervisor of this research and a Professor at the George Washington University, was heavily involved throughout the project and provided assistance with concept formation and editing.

Chapter 1: Introduction

Word embedding models are popular due to their ability to efficiently encode statistical, semantic, and syntactic properties of words, which entails capturing the variant relationships among words. For example, finding the capital of France, based on the relationship between Japan and Tokyo, can be identified through simple arithmetic operations between the given words vectors (Mikolov et al., 2013). The impressive results of such models led to more efforts in developing richer representations. While word embeddings have been rapidly advancing state-of-the-art performance in NLP systems, they are only a small piece in the process of understanding human languages.

Compositionality is a way of representing the full meaning of longer pieces of text (i.e., phrases, sentences, or paragraphs). As one of the most important aspects of understanding human language, the vast majority of NLP system pipelines require a way to model compositionality as a way to encode a sentence into a fixed-length vector (i.e., sentence embeddings). Unlike word embeddings, where individual word properties are encoded into a vector, sentence embeddings encode linguistic properties from all the words and their relations in a sentence into a fixed-length vector. Sentence embeddings can be used for various linguistic purposes such as predicting the relationship between sentence pairs and interpreting explicit or implicit values of a sentence. For example, sentiment classification, where the task is to determine whether a statement is negative or positive, is among the many tasks that operate at the sentence level and often desire a fixed-length sentence embeddings. Relying only on the representation of individual words in the statement may lead to a wrong interpretation of its sentiment.

Driven by the demand for a sentence embedding model that retains informative syntactic and semantic cues, a number of competing encoding models have emerged. On the one hand, the average of word embeddings is the simplest and fastest baseline to form sentence

embeddings as it relies only on pre-trained word embeddings. Thus, it is an efficient compositionality operation that leads to good performance. In fact, more complex models struggle to compete with averaging. An example of this is deep neural models, as illustrated across several classification tasks: topic categorization, semantic textual similarity, and sentiment classification (Aldarmaki and Diab, 2018). A major weakness of averaging, however, is the loss of syntactic structure information, e.g. “dog bites man" is deemed equivalent to “man bites dog".

On the other hand, sequential neural sentence encoders like Skip-thought (Kiros et al., 2015) and InferSent (Conneau et al., 2017a), can potentially encode rich semantic and syntactic features from the sentence structures. However, for practical applications, sequential models are rather cumbersome and inefficient and the gains in performance are typically mediocre compared with vector averaging (Aldarmaki and Diab, 2018). In addition, the more complex models typically lack generalization to out-of-domain data (Wieting et al., 2015).

FastSent encoder (Hill et al., 2016) is an unsupervised alternative approach with lower computational cost, but similar to vector averaging, it disregards word order. Tensor-based composition, another sentence encoder, can effectively capture word order, but current approaches rely on restricted grammatical constructs, such as transitive phrases, and cannot be easily extended to variable-length sequences of arbitrary structures (Milajevs et al., 2014). Therefore, despite its obvious disregard for structural properties, the efficiency and reasonable performance of vector averaging makes it more suitable for practical text classification.

Despite the potential capacity of deep neural models, (convolutional, recursive, or recurrent), in encoding semantic and syntactic features from sentences, these models are heavily parameterized and typically require a large computation capacity power. Thus, deep neural models are inefficient for practical applications. Most recently, a number of computationally-efficient, non-parameterized sentence embedding models have been

proposed to address the limitations of deep sentence encoders, such as the SIF, p-means, GEM, and EigenSent models (Rücklé et al., 2018; Yang et al., 2019b; Kayal and Tsatsaronis, 2019). The main objective of non-parameterized sentence embedding models, which mostly rely on pre-trained word embeddings, is to minimize the reliance on deep neural models while reducing the performance gap.

In this dissertation, we develop a non-parameterized model that makes use of Discrete Cosine Transform (DCT) as a simple and efficient way to model word order and structure in sentences while maintaining practical efficiency in Chapter 3. DCT is a widely-used technique in digital signal processing applications such as image compression (Watson, 1994) as well as speech recognition (Huang and Zhao, 2000). To the best of our knowledge, this is the first successful NLP application of DCT specifically for sentence embeddings.[1]

The main contributions of this dissertation can be summarized as follows:

- We develop an efficient parameter-free sentence embedding model based on Discrete Cosine Transformation (DCT). The developed model aims to capture sentence's structural information from the low-order DCT coefficients along with the semantic information preserved in the first DCT coefficient. Thus, efficiently compresses variable-length sentences into fixed-length vectors;

- We evaluate the developed model based on a range of various probing as well as downstream NLP benchmarks. We demonstrate that our model successfully retains sentence structural properties and generally performs well across different tasks;

- We also investigate the efficacy of using different linguistic priors to guide the DCT embeddings model to generate richer representation;

- We further explore the universality of our proposed DCT embeddings model across different languages in multi- and cross-lingual settings;

[1] In parallel with our work at time of publication, and in contrast to our implementation, DCT has been used in (Kayal and Tsatsaronis, 2019) for sentence embeddings. However, the DCT implementation in (Kayal and Tsatsaronis, 2019) performed relatively poorly compared to our implementation.

- We empirically demonstrate that our model is a language independent model that can be used for cross-lingual mapping.

This dissertation outlines the feasibility and applicability of our proposed DCT embeddings model across the larger genre of NLP sentence embeddings applications. In Chapter 2, we survey the related literature in sentence embedding models. In addition, we also provide a brief overview of discrete cosine transformation (DCT) in the signal processing domain. We also highlight how our contributions relate to the existing literature in both the NLP and signal processing domains, to the best of our knowledge. In Chapter 3, we present a detailed description of our DCT embedding model along with evaluation and analysis of the results. In Chapter 4, we investigate the applicability of DCT-based embedding across different languages in multi- and cross-lingual settings. We further demonstrate the particular effectiveness of DCT embeddings in sentence-level cross-lingual mapping in comparison to the commonly-used vector averaging model. In Chapter 5, we describe and evaluate our syntactic-guided DCT embedding that explicit encodes different levels of linguistic granularity. In Chapter 4 and Chapter 5, we provide a brief overview of related studies and how they are connected to our work. Finally, in Chapter 6, we summarize the main finding of this dissertation, research contributions, and the future research direction.

Chapter 2: Literature Review

1 Sentence Embedding Models

Modern NLP systems rely on word embeddings as input units to encode the statistical, semantic, and syntactic properties of words, ranging from standard context-independent embeddings such as word2vec and Glove, to contextualized embeddings such as ELMo and BERT (Mikolov et al., 2013; Pennington et al., 2014; Peters et al., 2018; Devlin et al., 2019). However, most applications operate at the phrase or sentence level. This specification compels NLP researchers to devise sentence representation models. These models can be grouped to two main categories based on their underlying training requirements: parameterized sentence encoders and non-parameterized sentence encoders.

The parametrized models typically rely on the optimization of a large number of parameters during the training process. This approach can be further classified based on the training objective used to build the model into unsupervised and supervised models. While the unsupervised models learn the sentence-level representation through a large unlabeled corpus, the supervised models learn the sentence-level representation using a task-specific objective such as sentence classification.

From another perspective, parametrized models can be divided based on the type of the learned representation into uncontextualized (a.k.a context-free) and contextualized models. Unlike the uncontextualized models, contextualized models are trained to capture the context in the final representation. Most of the earlier established works in the sentence embeddings domain are based on unsupervised and supervised context-free representations. Apart from these methods, there has been a substantial amount of research to generate sentence representations based on contextualized language models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). Recently, this line of research has received significant attention as it yields promising results in a wide variety of NLP tasks.

Although parametrized models have promising capacity to learn linguistic properties, they are inefficient for practical applications because they require a lot of computational power. Driven by the demands for fast and efficient alternatives to the heavily parameterized models, a number of non-parameterized models have been proposed. The non-parameterized models make use of pre-trained word embeddings with simple mathematical heuristic to render sentence representations. Additionally, the non-parameterized models require no further training, thus, the reliance on parameterized deep neural models diminish.

This section provides a background review of different sentence encoder models. Specifically, we delve into four main types of sentence embedding models: parameterized-context-free models including 1) unsupervised sentence representations and 2) supervised sentence representations, 3) contextualized sentence representations, and 4) non-parameterized sentence representations. In addition to the sentence embedding models, we subsequently give a brief overview the discrete cosine transformation (DCT) in the signal processing domain in Section 2, which form the foundation of the non-parameterized sentence encoder technique proposed in this dissertation.

1.1 Unsupervised Sentence Representations

Modeling words through distributed representations, in which words are encoded as continuous-dense vectors, are an efficient alternative to the traditional high-dimensional sparse bag-of-words approaches (Mikolov et al., 2013). Generally, a massive corpora is leveraged to generate the distributed continuous-dense word vectors using unsupervised models. These unsupervised methods, such as word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014), have shown that low-dimensional vectors encode useful semantic information from words. Hence, generating a sentence embedding extends this analog to learn similar representations for sentences or sequences of words from large unlabeled data. Such unsupervised models depend only on the order of the sentences in the unlabeled data. Hence, sentence order is used as a kind of weak supervision based on the assumption that

predicting the surroundings sentences of a given sentence lead to better sentence embeddings, which is an extension of the word-level distributional hypo dissertation (Harris, 1954).

Unsupervised models can be grouped into generative and discriminative models based on the training objective. In the generative setting, given a sentence S_i, the model is trained to predict the neighboring sentences $(S_i - 1, S_i + 1)$, which represent the previous and the next sentences, respectively. The first proposed model, Skip-Thought vectors, relies on a recurrent neural encoder-decoder architecture to predict the surrounding sentences (Kiros et al., 2015). While the recurrent encoding-decoding process in Skip-Thought preserves word order, it is computationally expensive during training and inference time. To remedy this inefficiency, Hill et al. (2016) developed FastSent with the same training objective to predict the surrounding sentences. Specifically, as reported in Hill et al. (2016), while training Skip-Thought took two weeks, training FastSent took only two hours. This efficiency in FastSent is achieved by simply representing the sentences as a Bag-of-Words (BoW), which sacrifices word-order for lower computational complexity. Thus, FastSent is more efficient, but better performance is achieved using Skip-Thought in some sentence classification tasks such as paraphrase identification and sentiment (Hill et al., 2016).

Comparably, Siamese C-BOW is also trained to predict neighboring sentences to learn sentence vectors (Kenter et al., 2016). However, it learns the Bag-of-Words (BoW) embeddings by utilizing a Siamese neural network to predict the adjacent sentences. The reported comparison in Kenter et al. (2016) shows that while Siamese C-BOW outperforms Skip-Thought on most of the evaluation tasks, there is no clear leader among Siamese C-BOW and FastSent, as each one of them outperforms the other on half of the evaluation sets.

Sent2vec is another unsupervised learning method for sentence embedding. However, it learns sentence representation by simultaneously training uni-grams along with n-grams that are presented in the sentence (Pagliardini et al., 2018). As reported in Pagliardini et al. (2018), while Skip-Thought outperforms Sent2vec, better performance is achieved

by Sent2vec compared to other unsupervised methods on downstream classification tasks. On the other hand, for similarity tasks, Sent2Vec achieved state-of-the-art performance compared to all the unsupervised models including Skip-Thought, FastSent, and the Siamese C-BOW. Moreover, the computational complexity of Sent2vec is only $O(1)$ during training and inference, thus, it is much more efficient compared to the other unsupervised methods (Pagliardini et al., 2018).

Alternatively, discriminative models leverage a classification task objective based on the problem of predicting the surrounding sentences. This is more efficient than the generative model because the decoder is replaced by a classifier in order to predict neighboring sentences from a set of candidates. For example, Quick-Thought (Logeswaran and Lee, 2018) is an unsupervised method with a classification objective that utilizes a classifier to learn a sentence vector. Quick-Thought, which builds upon Skip-Thought, uses a recurrent encoder and replaces the decoder component with a classifier. Quick-Thought is more efficient than Skip-Thought with an order of magnitude less training time. On the other hand, FastSent is more computationally efficient, with only two hours of training time compared to the one day required to train Quick-Thought. However, the Quick-Thought model captures better representations than FastSent (Logeswaran and Lee, 2018).

1.2 Supervised Sentence Representations

While the unsupervised sentence encoder captures the general linguistic information from large unlabeled training data, there are also supervised models that are based on specific training objective. These supervised sentence models generate embeddings that contain more domain-specific information, based on their training objective.

In 2017, Conneau et al. (2017a) proposed the first supervised sentence model, known as InferSent. InferSent adopted a Siamese bidirectional Long Short-Term Memory (LSTM) with a max-pooling architecture as the sentence encoder. Despite being trained to only detect inference between a pair of texts, the resulting sentence encoder performs well on other

tasks and significantly outperforms unsupervised encoders. The findings of Conneau et al. (2017a) have shifted researchers' attention from the unsupervised paradigm of sentence encoder toward developing supervised models.

Using the same training objective, a lightweight version of InferSent called InferLite addresses the run-time overhead of the recurrent layers in InferSent with comparable performance. In particular, InferLite eliminates recurrent layers and utilizes a lightweight encoder that relies on a collection of pre-trained word embeddings. A gated controller is used to decide how to weight each word representation and the final sentence embedding is obtained by max pooling (Kiros and Chan, 2018).

The Multi-Task Learning (MTL) framework, a family of neural models, combines multiple tasks with different learning objectives rather than a single task-specific objective. For example, the Universal Sentence Encoder (USE) combines varieties of unsupervised large corpora with supervised language inference data (Cer et al., 2018). Instead of relying on recurrent neural networks to produce sentence embeddings, USE has two models with two different encoders. The first encoder is trained with Transformer architecture, which is powerful but computationally expensive (Vaswani et al., 2017). The second encoder relies on the Deep Averaging Network (DAN) as an efficient alternative to the first model, with a minor drop in accuracy (Iyyer et al., 2015).

Similarly, to learn general purpose sentence vectors, the GenSen approach presented in (Subramanian et al., 2018) utilizes a MTL framework based on an encoder-decoder model. The GenSen framework leverages multiple weakly-related tasks that share a recurrent sentence encoder layer, which is used to generate the sentence embedding. Encapsulating multi-lingual neural machine translation, constituency parsing, Skip-Thought vectors, and natural language inference tasks, GenSen has shown competitive results compared to other task-specific models such as Infersent (Subramanian et al., 2018; Wang et al., 2018).

1.3 Contextualized Sentence Representation

Unlike the conventional language model designed to learn shallow word or sentence embeddings, contextualized language models are designed to learn word representations as a function of the entire input sentence (Peters et al., 2018; Devlin et al., 2019). On a diverse set of downstream NLP tasks, state-of-the-art results have been achieved using contextualized representations instead of the traditional context-free representations (Peters et al., 2018; Devlin et al., 2019). For example, Embeddings from Language Models (ELMo) presented in Peters et al. (2018), one of the earlier contextualized embedding models. ELMo utilizes a bi-directional LSTM to train language model in a large corpus. Typically, a BOW-ELMo style sentence embedding is generated by averaging the word-level representation from all internal layers of the bi-directional LSTM. BOW-ELMo style embeddings dramatically outperform the BOW representation extracted from traditional embeddings such as Word2vec, GloVe, and FastText (Peters et al., 2018; Si et al., 2019). Hence, for better representation, the pre-trained word embeddings can be replaced by the richer contextualized vectors of ELMo. However, ELMo is inefficient compared to the conventional pre-trained embeddings in terms of the required computational cost.

The encoder-decoder architecture for most of the language models, including ELMo, is based on complex recurrent or convolutional neural networks. Although these models are known for their competitive performance, the sequential computation remains a fundamental bottleneck in the model's efficiency (Wang et al., 2020). To address this computational complexity, Vaswani et al. (2017) proposed Transformer architecture, which utilizes a multi-headed attention mechanism as the main building block for the model's encoder-decoder. Unlike the sequential processing of recurrent-based models, the attention operations in the Transformer allow for parallel and independent processing of the input, which results in less training time and more efficiency. As a result, Transformer-based models, a family of current state-of-the-art pre-trained language models based on the Transformer architecture, have dominated a wide area of natural language processing tasks (Si et al., 2019; Yang et al.,

2019a; Dai et al., 2019). While the Transformer is a constituent part on these models, the Bi-directional Encoder Representations from Transformers (BERT) model proposed by Si et al. (2019) is the main breakthrough in the area of pre-trained contextualized representations. Typically, this framework includes two steps: first a deep neural model is trained on a large unlabeled text, then the pre-trained model is fine-tuned with all its parameters to create task-specific representation. Although there has been a tremendous amount of research inspired by BERT that builds on the Transformer model, such as Transformer-XL, GPT-2, RoBERTa, ALBERT, and XLNet, there are few studies that leverage pre-trained BERT specifically for sentence embeddings such as Sentence-BERT (SBERT) and SBERT-WK (Dai et al., 2019; Radford et al.; Liu et al., 2019b; Lan et al., 2019; Yang et al., 2019a; Reimers and Gurevych, 2019; Wang and Kuo, 2020).

The three common methods for extracting sentence embeddings from the pre-trained BERT are the use of a CLS token, SEP token, or the average of the word's embeddings from the last layer (Si et al., 2019; Wang and Kuo, 2020; May et al., 2019; Alberti et al., 2019). However, as shown by Reimers and Gurevych (2019), these methods are not optimal for learning a sentence representation for the textual similarity task, in which BOW-based GloVe embeddings significantly outperform BERT. To address this limitation, Reimers and Gurevych (2019) presented SBERT, a sentence model that fine-tunes a pre-trained BERT by utilizing the Siamese and triplet networks (Schroff et al., 2015). Thus, embeddings for a pair of sentences, computed using the MEAN-strategy,[1] are obtained independently by passing them through two networks that have tied weights. The SBERT then fine-tunes the model using the textual inference data with a 3-way classification objective for one epoch. This results represent an incremental improvement compared to the out-of-the-box BERT. Similarly, SBERT-WK utilizes a pre-trained BERT to generate a sentence embedding. In contrast to the previous model, SBERT-WK combined information from various BERT layers through a geometric analysis of the layer-wise space. More specifically, several

[1] Reimers and Gurevych (2019) experimented with different pooling strategies including CLS-token, MAX, and MEAN pooling and the best results were obtained using the Mean pooling.

hand-crafted measurements are used as post-processing step to determines word weights according to the subspace analysis (Wang and Kuo, 2020). As opposed to SBERT, SBERT-WK requires no further training, yet it achieved good performance compared to SBERT (Wang and Kuo, 2020).

Despite the success of contextualized-based models, their computational efficiency remains an ongoing challenge. Specifically, extracting sentence embeddings to be used as features for an end task is computationally expensive and generally does not work well without fine-tuning, which adds an additional overhead to the computational cost. Although there have been a number of studies that address the computational complexity of BERT, their solutions are still bounded by the availability of sufficient computation capacity power (eg. V100 GPU) (Wang et al., 2020). To address this challenge, non-parameterized sentence encoder focus on developing models that are computationally efficient and are also able to reduce the performance gap.

1.4 Non-parameterized Sentence Representations

The unsupervised and supervised approaches for sentence encoders that have been proposed are based on heavily-parameterized neural models. Hence, they are inefficient during training and during inference time. To the contrary, sentence embeddings which employ a simple averaging mechanism based solely on pre-trained word embeddings—the BoW method—have been proven to perform reasonably well without the overhead computations of sophisticated neural models (Arora et al., 2016). Given the superiority of results yielded by averaging, alternative parameter-free, fast, and high-quality sentence encoders that rely on pre-trained word embeddings and also require no further training are garnering more interest.

Arora et al. (2017) proposed a non-parameterized model, the Smooth Inverse Frequencies (SIF) model, which is a simple weighted sentence encoder. Conceptually, SIF weighting scheme is highly similar to the TF-IDF weighting scheme (commonly used for document

retrieval) (Jones, 1972). However, the SIF model does not adjust the weight of more frequent words, as sentences typically contain fewer repetitive words compared to documents. Specifically, the SIF encoder computes sentence embeddings as the weighted sum of pre-trained word embeddings. The weighted sum is then modified by removing the first principal component from the final sentence's representation. Ethayarajh (2018) improved the SIF model presented in Arora et al. (2017) by proposing uSIF, which was designed to overcome the confounding effect of vector length of the SIF on sentence embeddings. In uSIF, the probability of word generation is taken as inversely related to the angular distance between the word and sentence embeddings.

The power means (p-means) described in (Rücklé et al., 2018) is an additional non-parameterized method that leverages a concatenation of different pre-trained word embeddings as sentence representation, which uses the power means for mean computation. Recently, Geometric Embedding (GEM) has been proposed as another parameter-free sentence encoder based entirely on the geometric structure of word embeddings space (Yang et al., 2019b). The GEM sentence embeddings encapsulate information from both the sentence and corpus levels.

Aside from these approaches, Kayal and Tsatsaronis (2019) explored spectral methods of signal representation and summarization as mechanisms for generating sentence embeddings. Specifically, they propose EigenSent, which uses Higher-Order Dynamic Mode Decomposition (HODMD) (Le Clainche and Vega, 2017) to construct sentence embeddings. These embeddings summarize the transition dynamic properties of words in a sentence.

In this research, we build on this latter line of work, focusing on non-parameterized sentence embedding models. Methodologically, our work is most similar to EigenSent, as we borrow a mechanism from the signal processing domains and apply it to pre-trained word embeddings to generate sentence embeddings. More concretely, we propose the use of the discrete cosine transform as a way to compress information from different words in a sentence to render fixed-length sentence representation.

2 Discrete Cosine Transform

The Discrete Cosine Transform (DCT) was developed in 1974 by Ahmed et al. (1974). It is a well-known technique in the signal processing domain. In this section, we will shed light on and discuss some of the works that utilize DCT in the signal processing domain.

DCT is widely used for signal compression, a technique that plays a critical role in the transmission and storage of signal data (such as image, audio, and video) as digital information, where uncompressed data requires a sizable storage capacity and high transmission bandwidth (Telagarapu et al., 2011). It is one of the most popular and efficient technique for data compression, due to its fine properties. The main advantage of DCT is its ability to transform information from the space or time domain to the frequency domain, which is more compact, efficient, and suitable for further processing (Agostini et al., 2001). Besides these properties, another principle advantage of DCT is the separability of calculating multidimensional data. For image compression, which is an example of a two-dimensional signal, a 2D-DCT is computed by dividing the calculations into two separate 1D-DCTs. Also, in video, which can be seen as a 3D signal with a time dimension and two spatial dimensions, 3D-DCT extends the same principle and computes DCT along each dimension independently (Jain and Jain, 1981). For these reasons, DCT has been widely adopted for compression purposes (Shu et al., 2017).

Aside from signal compression, there have been a number of studies that utilize DCT as a feature extraction mechanism, where DCT coefficients are used to train an end task. For example, the lower DCT frequencies, which result from compressing an entire image by 2-dimensional DCT, can be used to train neural networks for facial expression recognition (Xiao et al., 1999; Yegui Xiao et al., 2006). Chen et al. (2005) also developed a facial recognition system based on features extracted from DCT, but unlike the Xiao et al. (1999) system, several low-frequency coefficients are discarded from the final feature space.

The time-series domain, such as speech recognition, gesture recognition, and Elec-

troencephalogram (ECG) data analysis, is another area that utilizes DCT to extract useful classification features (Batal and Hauskrecht, 2009). Time-series data tends to be large-dimensional data with strong and high correlation among features, which are the main challenges in the development of time-series classification models. To address these shortcomings, Batal and Hauskrecht (2009) employed DCT [2] to transform time-series data from the time domain into the frequency domain. Applying DCT to the feature's space reduces the correlation between features and reveals hidden time-series characteristics. As reported in (Batal and Hauskrecht, 2009), the classification features generated from the spectral domain improve the classification accuracy on a collection of time-series classification problems compared to the features from the time domain.

To classify Ictal and Interictal EEG signals, Parvez and Paul (2012) proposed an approach that leverages the high frequencies of DCT as features for the Least Squares SVM classifier, which has been shown to be useful when distinguishing between Ictal and Interictal signals. Furthermore, the Audio-Visual model presented in (Elmaghraby et al., 2019) is a deep learning model for speech recognition systems where the features from audio and visual speech signals are extracted using the Mel Frequency Cepstral Coefficient (MFCC) and DCT, respectively. As reported by the authors, combining the MFCC features for audio modality with the DCT features for visual signal improves the accuracy of the speech recognition system.

Word embeddings that are generated by neural models are known to exhibit linear behaviour, in which word analogies are interpreted by mathematically approximation using vectors addition or subtraction (Allen and Hospedales, 2019; Ethayarajh et al., 2019). Since there are correlations between adjacent words vectors in a give sentence, in principal this enables us to use 1D-DCT to decorrelate the word embeddings along the length of the sentence. The model presented in this dissertation compresses word sequences in an order-preserving manner, where the lower-order DCT coefficients represent the overall feature

[2] Batal and Hauskrecht (2009) also experimented with discrete wavelet transform (DWT) as an alternative approach to DCT.

patterns in a given sentence. This results in suitable and efficient embeddings for tasks that could benefit from syntactic features. Therefore, expanding on the idea of utilizing DCT to extract silent features that are useful in the signal processing domain, DCT is used in this dissertation as a feature extraction mechanism for textual data in the natural language processing domain.

Chapter 3: Sentence Embeddings Using Discrete Cosine Transform

1 Background: Discrete Cosine Transform in Image Processing

The Discrete Cosine Transform (DCT) has provided significant impact in the general domain of digital signal processing. In the area of image processing, DCT has been utilized for the purpose of feature selection in pattern recognition, such as face recognition applications (Podilchuk and Zhang, 1996). In addition, the strong energy compaction property of DCT (Cintra and Bayer, 2011) has made it one of the most widely adopted techniques in the signal compression domain, i.e., JPEG in images, MP3 in audio, and H.264 in videos (Shu et al., 2017). There are several variants of DCT, but DCT type II (Shao and Johnson, 2008) is the most commonly used one and it is the variant used in this study. The result of DCT type II is multiplied by an overall scale factor to make the resulting DCT matrix orthogonal (Shao and Johnson, 2008).

More formally, DCT is an invertible function that maps an input sequence of N real numbers to the coefficients of N orthogonal cosine basis functions. Given a vector of real numbers $\vec{v} = v_0, ..., v_{N-1}$, we calculate a sequence of DCT coefficients $c[0], .., c[k], .., c[N-1]$ for $1 \leq k < N$ as follows:

$$c[0] = \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} v_n, \tag{3.1}$$

and

$$c[k] = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} v_n \cos \frac{\pi}{N}(n + \frac{1}{2})k, \tag{3.2}$$

Note that $c[0]$ is the sum of the input sequence normalized by the square length, which is proportional to the average of the sequence. The N coefficients can be used to reconstruct the original sequence using the inverse transform.

In practice, DCT is used for compression by preserving only the coefficients with large

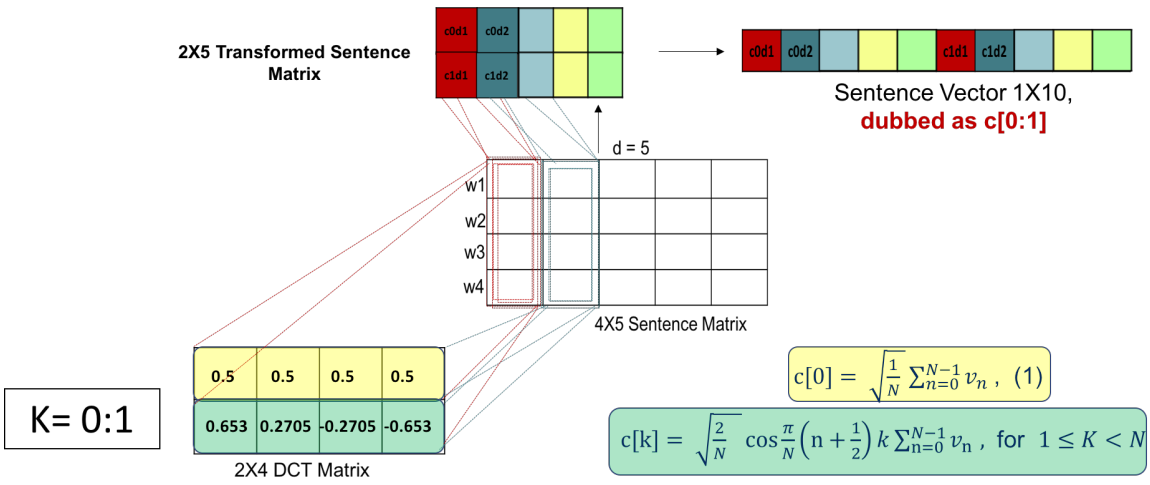


Figure 3.1: An illustration of DCT embeddings. The size of the sentence to encode is 4×5, where 4 is the number of words and 5 is the number of word embedding dimensions. Each feature vector is transformed using DCT independently. In this example, the final representation is the concatenation of the first two coefficient from all transformed features.

magnitudes. Lower-order coefficients represent lower signal frequencies, which correspond to the overall patterns in the sequence. Specifically, in DCT, the signal transforms as a sum of sinusoids of varying magnitudes and frequencies. The resulting transformation consists of one DC coefficient (i.e. $c[0]$) and a number of AC coefficients (i.e., $c[k]$) (Ahmed et al., 1974).[1]

2 Discrete Cosine Transform as a Sentence Encoder

For the purposes of this research, we apply DCT to the word vectors along the length of the sentence. Given a sentence of N words $w_1, ..., w_N$, we stack the sequence of d-dimensional word embeddings in an $N \times d$ matrix, then apply DCT along the columns. Specifically, each feature in the vector space is compressed independently, and the resultant DCT embeddings summarize the feature patterns along the word sequence. To get a fixed-length and consistent sentence vector, we extract and concatenate the first K DCT coefficients and and discard those of a higher order, which results in sentence vectors of

[1]DC and AC are terms from the electrical engineering domain, and they are used in the signal domain to describe the components of a signal. Specifically, DC refers to a direct current signal, and AC refers to an alternating current signal (Li et al., 2004)

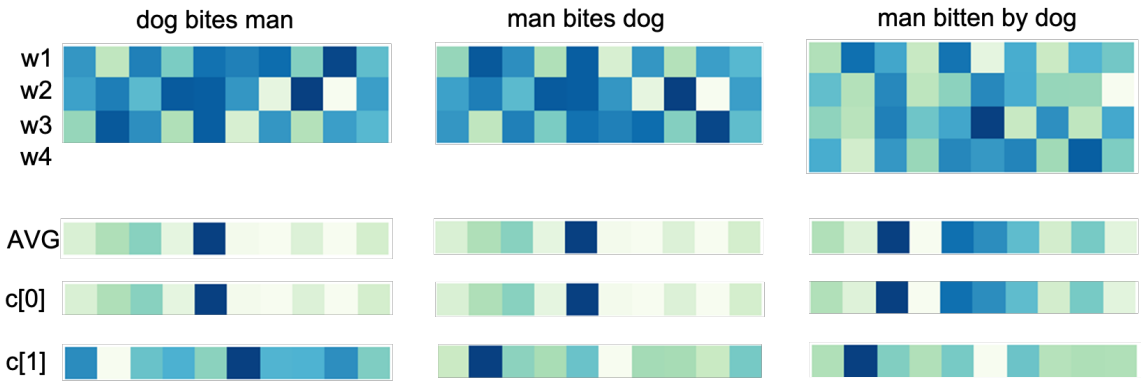


Figure 3.2: Illustration of word vector averaging vs. DCT using the first 2 DCT coefficients. The word vectors are generated randomly from a standard normal distribution with $d = 10$.

size Kd. For cases where $N < K$, we pad the sentence with $K - N$ zero vectors. An illustration of the DCT embeddings process is given in figure 3.1. In image compression, the magnitude of the coefficients tends to decrease with increasing k; however, this trend was not observed in text data, except that $c[0]$ tends to have larger absolute value than the remaining coefficients. Nonetheless, by retaining lower-order coefficients, a consistent representation of overall feature patterns in the word sequence was observed.

Figure 3.2 illustrates the properties of DCT embeddings compared to vector averaging (AVG). The first DCT coefficient $c[0]$ results in vectors that are independent from word order, since the lowest frequency represents the average energy in the sequence. In this sense, $c[0]$ is similar to AVG, where "dog bites man" and "man bites dog" have identical embeddings. The second-order coefficients, on the other hand, are sensitive to word order, which results in different representations for the above sentence pair. The counterexample "man bitten by dog" shows that $c[1]$ embeddings are most sensitive to the overall patterns—in this case, "man ... dog"—which results in an embedding more similar to "man bites dog" than the semantically similar "dog bites man." However, there are still some variations in the final embeddings due to the different word components ('bitten' vs. 'bite'), which can potentially be useful in downstream tasks. Since both DCT and AVG are non-parameterized, the downstream classifiers can incorporate a hidden layer to learn these subtle variations in

higher-order features, depending on the learning objective.

2.1 A Note on Complexity

The cosine terms in Equation 3.2 can be pre-calculated for efficiency. For a maximum sentence length $\hat{N}$ and a given K, the total number of terms is $(K-1)\hat{N}$ for each feature. Thus, the run-time complexity is equivalent to calculating K weighted averages, which is proportional to KN, where K should be set to a small constant relative to the expected length.[2] Also, it is important to note that the number of input parameters in downstream classification models will increase linearly with K. With parallel implementations, however, the difference in run-time complexity between AVG and DCT is practically negligible.

3 Evaluation Dataset

To evaluate the representations generated by the DCT models, sentence embeddings generated using the DCT encoding were used in different probing tasks as well as downstream classification tasks. This probing benchmark set was designed to study and analyze sentence embeddings generated by different encoders, where each task in this set focused on one simple linguistic features of sentences (Conneau et al., 2018a). The probing benchmark contains a set of 10 classification tasks that address a variety of linguistic properties including surface, syntactic, and semantic information (Conneau et al., 2018a). A summary of the datasets and an example of each task are given in Table 3.1.

Surface Information Tasks: The main goal of this set of tasks is to evaluate the capability of sentence embeddings in preserving surface-level information, such as sentence length and word content.

Sentence Length prediction **(SentLen)** is a classification task where the goal is to predict the number of words in a sentence (i.e. length of sentence). For the purposes of this task, sentences have been grouped into six bins by length with the following interval ranges:

[2] We experimented with $1 \leq K \leq 7$.

Task	**Description**	Example	Output
SentLen	The sentence length prediction among 6 length intervals	The Bible would be staying here .	0: (5-8 words)
WC	Predict which words form the 1000 target set appear in a given sentence	I couldn't even grasp being that old .	grasp
TreeDepth	Predict the depth of the syntactic tree of a given sentence	" That is what troubles me . "	8
TopConst	Sentences are classified in terms of their top constituents sequence, where the classes are given by the 19 most common top-constituent sequences in the given corpus	Did it all matter ?	VBD_NP_VP_.
BShift	If two words in a sentence have been inverted	This **new was** information	Inverted
Tense	The tense of the main verb in a sentence either past or present	It opens up a lot of colorful possibilities .	Present
SubjNum	The number of subject in a sentence either singular or plural	The yard was very peaceful .	Singular
ObjNum	The number of direct object in a sentence either singular or plural	Smooth paving stone covered the floors .	Plural
SOMO	Predict if the sentence has been modified, where random noun or verb in the sentence replaced by another noun or verb.	So go get the ibuprofen .	Changed
CoordInv	Detect if the clauses, in a sentence comprised of two coordinate clauses, are inverted or not	She poured tea for the two of them and she waved at the chair by the kitchen table .	Inverted

Table 3.1: Probing Tasks Description and Samples

0: (5-8), 1: (9-12), 2: (13-16), 3: (17-20), 4: (21-25), 5: (26-28). Thus, the problem of predicting sentence length has been cast as a six-way classification task.

For the Word Content Task **(WC)**, given a set of 1K target words set, the task of WC is to predict which word of the target set is contained in the sentence. Each sentence contains a single target word with a single occurrence. As there are 1K target words, the task is cast as a 1K-way classification.

Syntactic Information Tasks: This set of tasks evaluates the capability of sentence embeddings to preserve different syntactic information. The tasks are BiGram Shift, TreeDepth, and TopConst. Table 3.1 shows examples for this set of tasks. The Bigram shift **(BShift)**

task is a binary classification task with the goal of predicting whether the legal word orders of two adjacent tokens have been inverted or not. In Tree Depth prediction **(TreeDepth)**, the task is to predict the maximum depth of a sentence's syntactic tree, in which sentences are grouped by the longest path from root to any leaf. The Tree Depth values in this dataset range from 5 to 12, which makes the task an eight-way classification problem. The final task in this set is the Top Constituents **(TopConst)** task. Sentences in this task are classified in terms of their top constituents sequence below the (S) node. It is framed as a 20-way classification task, with 19 classes for the most frequent top constituent sequences and one class for all other sequence structures.

Semantic Information Tasks: In addition to the syntactic structure, this set of tasks relies on sentence embeddings to encode some of the semantic properties in a text. The **Tense** task, a binary classification task, checks for the tense of the main-clause verb in a sentence, whether it is in the present or past tense. Table 3.1 shows an example for the Tense task. In the binary classification Subject Number **(SubjNum)** task, the focus is on the subject of the main clause. Specifically, the task is to identify whether the subject is singular or plural. similarly to SubjNum, Object Number **(ObjNum)** is a binary classification identifying whether the main clause direct object is singular or plural. Examples for both: SubjNum and ObjNum tasks are shown in Table 3.1. In the semantic odd man out **(SOMO)** task, the goal of a binary classifier is to predict whether a sentence has been modified from the source corpus or not. Finally, the binary classification Coordination Inversion **(CoordInv)** task asks for distinguishing between intact original and modified sentence. Table 3.1 shows examples for the SOMO and CoordInv tasks.

The downstream set includes the following standard classification tasks: Sentiment analysis, relatedness and paraphrase detection, natural language inference, and question type classification. Table 3.2 and Table 3.3 show examples for the different tasks in this set as presented by Conneau and Kiela (2018).

Sentiment Analysis Tasks: This set includes the following sentiment classification

Name	N	Task	C	Examples	Label(s)
MR	11k	sentiment (movies)	2	Too slow for a younger crowd , too shallow for an older one.	neg
CR	4k	product reviews	2	We tried it out christmas night and it worked great .	pos
SUBJ	10k	subjectivity/objectivity	2	A movie that doesn't aim too high , but doesn't need to.	subj
MPQA	11k	opinion polarity	2	don't want; "would like to tell";	neg, pos
TREC	6k	question-type	6	What are the twin cities ?	LOC:city
SST-2	70k	sentiment (movies)	2	Audrey Tautou has a knack for picking roles that magnify her [..]	pos
SST-5	12k	sentiment (movies)	5	nothing about this movie works.	0

Table 3.2: Classification tasks. C is the number of classes and N is the number of samples (Conneau and Kiela, 2018). Note this table is reprinted from (Conneau and Kiela, 2018).

Name	N	Task	Output	Premise	Hypothesis	Label
SICK-E	10k	NLI	3	"A man is sitting on a chair and rubbing his eyes"	"There is no man sitting on a chair and rubbing his eyes"	contradiction
SICK-R	10k	STS	[0,5]	"A man is singing a song and playing the guitar"	"A man is opening a package that contains headphones"	1.6
STSB	7.1k	STS	[0,5]	"Liquid ammonia leak kills 15 in Shanghai"	"Liquid ammonia leak kills at least 15 in Shanghai"	4.6
MRPC	5.7k	PD	2	"The procedure is generally performed in the second or third trimester."	"The technique is used during the second and, occasionally, third trimester of pregnancy."	paraphrase

Table 3.3: Natural Language Inference and Semantic Similarity tasks. NLI labels are contradiction, neutral and entailment. STS labels are scores between 0 and 5. PD=paraphrase detection (Conneau and Kiela, 2018). Note this table is reprinted from (Conneau and Kiela, 2018).

datasets: Movie Reviews (MR) (Pang and Lee, 2004), Stanford Sentiment Analysis (SST2 and SST5) (Socher et al., 2013b), Customer Reviews (CR) (Hu and Liu, 2004), Multi-Perspective Question Answering (MPQA) (Wiebe et al., 2005), and Subjectivity/Objectivity (SUBJ) (Pang and Lee, 2004).

Semantic Relatedness and Paraphrase Tasks: This set includes three datasets. Two are semantic relatedness datasets: Sentences Involving Compositional Knowledge Semantic Relatedness (SICK-R) (Marelli et al., 2014) , and Semantic Textual Similarity Benchmark (STSB) (Cer et al., 2017). The third is a dataset for paraphrase detection in the Microsoft Research Paraphrase Corpus (MRPC) (Dolan et al., 2004).

Natural Language Inference Task: This includes the inference dataset from the Sentences Involving Compositional Knowledge Entailment (SICK-E) (Marelli et al., 2014).

4 Experiment Setups

The above evaluation datasets are part of the SentEval toolkit.[3] (Conneau and Kiela, 2018) For the word embeddings, in this study, we use FastText embeddings in size 300 (Mikolov et al., 2018) pre-trained on the Common-Crawl dataset. In addition, we experimented with a weighted version of the FastText embedding with the SIF weighting model (Arora et al., 2017). Briefly, each word vector v_w is assigned a weight equal to the inverse of its probability of occurrence as follows:

$$v_{sif} = \frac{\alpha}{\alpha + p_w} v_w, \tag{3.3}$$

where α is a smoothing constant and p_w is the word frequency in the training data. We generate DCT sentence vectors by concatenating the first K DCT coefficients, which we denote by $c[0:k]$. The performance is compared with the vector averaging of the same word embeddings, denoted by **AVG**, and the vector max pooling, denoted by **MAX**. For all tasks, we trained multi-layer perceptron (MLP) classifiers following the setup in SentEval. We tuned the following hyper-parameters on the validation sets: number of hidden states (in [0, 50, 100, 200, 512]) and dropout rate (in [0, 0.1, 0.2]). Note that the case with 0 hidden states corresponds to a Logistic Regression classifier.

5 Evaluation Results

We report the performance of the probing tasks in Table 3.4. In general, DCT yields better results compared to both averaging and max-pooling on all tasks, and larger K often yields improved performance in syntactic and semantic tasks. For the surface information tasks SentLen and WC, $c[0]$ significantly outperforms AVG and MAX, with 98.67% and 91.11% accuracy on the SentLen and WC, respectively. Distinctively, substantial gains of 34.55% are observed over AVG in the SentLen task, and 2.14% gains over MAX in

[3] https://github.com/facebookresearch/SentEval

		Surface		Syntactic			Semantic				
Model	Dim	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Majority	-	20.0	0.5	17.9	5.0	50.0	50.0	50.0	50.0	50.0	50.0
Human	-	100	100	84.0	84.0	98.0	85.0	88.0	86.5	81.2	85.0
Length	-	100	0.2	18.1	9.3	50.6	56.5	50.3	50.1	50.2	50.0
AVG	300	64.12	82.10	36.38	68.04	50.16	87.90	80.89	80.24	50.39	51.95
MAX	300	62.67	88.97	33.02	62.63	50.31	85.66	77.11	76.04	51.86	52.33
$c[0]$	300	**98.67**	**91.11**	38.6	70.54	50.42	88.25	80.88	80.56	**55.60**	55.00
$c[0:1]$	600	97.18	89.16	40.41	78.34	52.25	88.58	86.59	84.36	54.62	70.42
$c[0:2]$	900	95.84	86.77	43.01	80.41	54.84	88.87	88.06	86.26	53.07	71.87
$c[0:3]$	1200	94.63	84.96	**43.35**	81.01	57.29	88.88	88.36	86.51	53.79	**72.01**
$c[0:4]$	1500	93.25	83.24	43.26	81.49	60.31	**88.91**	**88.65**	87.15	52.77	71.91
$c[0:5]$	1800	92.29	81.84	42.75	**81.60**	62.01	88.82	88.44	87.98	52.38	70.96
$c[0:6]$	2100	91.56	79.83	43.05	81.41	**62.59**	88.87	**88.65**	**88.28**	52.07	70.63
$\boldsymbol{\sigma}$	-	**2.61**	**4.02**	1.83	**4.01**	**4.79**	0.24	**2.83**	**2.68**	**1.28**	**6.20**

Table 3.4: Probing tasks performance of vector averaging (AVG) and max pooling (MAX) vs. DCT embeddings with various K. Majority (baseline), Human (human-bound), and a linear classifier with sentence length as the sole feature (Length) as reported in (Conneau et al., 2018a), respectively. The $\boldsymbol{\sigma}$ denotes the standard deviations for the performance across different DCT coefficients.

the WC. This improvement is attributed to the non-linear scaling factor in DCT, where longer sentences are not discounted as much as in the averaging. However, the performance decreases with increasing K in $c[0:k]$, which reflects the trade-off between deep and surface linguistic properties, as discussed in (Conneau et al., 2018a). In particular, Conneau et al. (2018a) noted that deeper models, which tend to abstract away from surface level information, have more capacity to capture deeper linguistics features.

While increasing K has no positive effect on surface information tasks, syntactic and semantic tasks demonstrate performance gains with larger K. This trend is clearly observed in all syntactic tasks and four of the semantic tasks. In these tasks, DCT performs well above AVG and the performance improves with increasing K, with 6.97-13.56% and 1.01-19.68% gains, respectively. The only exception is SOMO, where increasing K actually results in lower performance, although all DCT results are still higher than AVG.

We further presented the standard deviations for the performance across different DCT coefficients in the last row in Table 3.4. As can be seen, generally, we noted variations on all tasks but both TopConst, and BShift from the syntactic tasks as well as CoordInv from

the semantic tasks appear to stand out with the highest performances variance. In fact, this results is consistent with our previous observation, in which increasing K has positive effect on both the syntactic and semantic tasks, and negatively affect the surface level tasks.

		Sentiment Analysis					**SUBJ**	**Relatedness&Paraphrase**			**Inference**	**TREC**
Model	Dim	MR	SST2	SST5	CR	MPQA		STSB	MRPC	SICK-R	SICK-E	
AVG	300	78.30	**84.13**	44.16	79.60	87.94	92.33	69.26	74.43	81.95	79.50	83.20
MAX	300	73.31	79.24	41.86	73.35	86.54	88.02	**71.57**	72.50	81.93	77.98	76.20
$c[0]$	300	**78.45**	83.53	44.57	79.81	**88.36**	**92.79**	71.11	72.93	82.61	78.91	84.80
$c[0:1]$	600	78.15	83.47	**46.06**	79.84	87.76	92.61	70.82	72.81	82.73	79.64	88.20
$c[0:2]$	900	78.02	82.98	45.16	79.68	87.62	92.50	70.36	72.87	**82.95**	79.76	**89.80**
$c[0:3]$	1200	77.81	83.80	45.79	79.66	87.54	92.40	69.79	73.57	82.93	**80.56**	88.20
$c[0:4]$	1500	77.72	83.75	44.03	**80.08**	87.40	92.61	69.31	72.35	82.53	79.72	**89.80**
$c[0:5]$	1800	77.42	82.43	43.30	78.60	87.21	92.19	68.90	73.91	82.36	79.89	88.80
$c[0:6]$	2100	77.47	82.81	42.99	78.78	87.06	92.15	68.17	**75.07**	81.86	79.76	86.40
$\boldsymbol{\sigma}$		0.37	0.52	**1.19**	0.57	0.42	0.23	**1.06**	0.91	0.38	0.48	**1.83**

Table 3.5: DCT embedding Performance in SentEval downstream tasks compared to vector averaging (AVG) and max pooling (MAX). The $\boldsymbol{\sigma}$ denotes the standard deviations for the performance across different DCT coefficients.

Conneau et al. (2018a) further discussed the correlations between the performance of probing tasks and the standard text downstream classification tasks. As shown in (Conneau et al., 2018a), most downstream tasks are only positively correlated with a small subset of the semantic or syntactic probing tasks, with the exception of TREC and some sentiment classification benchmarks. Furthermore, the performance of some tasks like SST and SICK-R, is actually negatively correlated with SubjNum, ObjNum, and BShift probing tasks. Furthermore, Conneau et al. (2018a) noted that all downstream tasks positively correlated with WC, in which the highest performance obtained with $c[0]$ model. Overall, as observed by Conneau et al. (2018a), simple averaging often outperforms more complex models in these tasks due to the pragmatic nature of the majority of downstream tasks. Our results in Table 3.5 are consistent with these observations, where we see improvements in most tasks. As shown in in the last row in Table 3.5, the standard deviations difference is not as significant as in the probing tasks, except in the TREC question classification task. Within the TREC task, increasing K leads to much better performance with up to 6.6% improvement compared to the AVG. We surmise that the ability to preserve word order

	Surface		Syntactic			Semantic				
Model	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Best-DCT	**83.47**	91.11	**36.49**	**78.58**	**52.07**	**87.91**	**86.27**	**84.09**	**51.06**	**66.91**
DCT-SIF	57.88	**93.35**	31.05	71.8	51.9	87.08	85.37	81.53	51.03	65.65

Table 3.6: Evaluation of weighted embedding using SIF weighting schema on probing tasks. Both (weighted and unweighted models) trained using logistic regression only.

	Sentiment Analysis					SUBJ	Relatedness&Paraphrase			Inference	TREC
Model	MR	SST2	SST5	CR	MPQA		STSB	MRPC	SICK-R	SICK-E	
Best-DCT	**78.15**	**82.98**	**44.84**	**80.29**	**87.98**	**92.59**	**82.95**	**71.11**	**80.19**	**80.19**	**90.00**
DCT-SIF	77.96	82.54	44.16	80.00	87.04	91.82	80.58	70.96	75.95	75.95	88.20

Table 3.7: Evaluation of weighted embedding using SIF weighting schema on downstream tasks. Both (weighted and unweighted models) trained using logistic regression only.

using DCT leads to improved performance in TREC. This finding aligns with the finding of Aldarmaki and Diab (2018), in which the authors discussed the relationship between model performance and its ability to preserve word order in the TREC task. Thus, encoding word order in the sentence embeddings provide useful clues for this task, which is the advantage of using DCT instead of AVG. Also note that while increasing K preserves more information, it leads to increasing the embeddings dimension, which in turn may negatively affect the generalization of the model by over-fitting. In our experiments, $1 \leq k \leq 2$ yielded the best trade-off.

Aside from the unweighted sentence embeddings results discussed above, weighted embeddings are used to determine if further improvement can be obtained over the unweighted DCT embeddings. We used the SIF weighting model introduced in (Arora et al., 2017) using Equation (3.3). Tables 3.6 and 3.7 show the results of the weighted schema on the probing and downstream benchmarks. Both the weighted and unweighted results are obtained using logistic regression only. While WC is the only task that acquires some improvement over the unweighted embedding, SentLen has shown a significant drop in accuracy. Also, the rest of tasks' accuracy only slightly dropped in the weighted embeddings compared to the unweighted embeddings model. Overall, we observed no additional improvements using weighted embedding compared to the unweighted model.

		Surface		Syntactic			Semantic				
Model	Dim	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
BOE	300	60.50	87.50	32.00	62.70	50.00	83.70	78.00	76.60	50.50	53.80
BOREP	4096	64.40	**97.10**	33.00	71.30	49.80	86.30	81.50	79.30	49.50	54.10
RandLSTM	4096	72.80	94.10	35.60	76.20	55.20	86.60	84.00	79.50	49.70	63.10
ESN	4096	78.80	92.40	36.90	76.20	62.90	86.60	82.30	79.70	49.70	60.30
Infersent-3	4096	**80.60**	93.50	37.10	78.20	57.30	86.80	84.80	80.50	53.00	65.80
SkipThought	4096	79.90	79.90	**39.50**	**82.10**	**69.40**	**90.20**	**86.20**	**83.40**	**54.50**	**68.90**
$c[0]$	300	72.60	**88.74**	32.92	63.33	49.47	83.62	78.37	76.30	49.79	53.55
$c[0:1]$	600	75.33	86.01	34.47	73.40	50.31	83.54	82.31	76.85	49.89	65.53
$c[0:2]$	900	76.74	82.60	34.93	75.86	50.93	83.83	**83.19**	**77.10**	**50.16**	66.15
$c[0:3]$	1200	**76.96**	79.90	**35.61**	**77.07**	**51.42**	**84.03**	83.14	77.09	49.72	**66.24**

Table 3.8: DCT embedding Performance in SentEval Probing tasks compared to SOTA models including SkipThought (unsupervised model), Infersent (supervised model), and different random encoders (BOE, BOREP, RandLSTM, and ESN) as reported in (Radford et al.). **Bold** highlights the best overall results, while **gray** highlights the best results for DCT encoders.

6 Comparison with State-of-The-Art Sentence Encoders

Recently, Radford et al. have explored various random sentence encoders leveraging random parameterizations without any training from pre-trained word embeddings. In particular, three architectures have been used to produce sentence embeddings from pre-trained word embeddings. The set of architectures include Bag Of Random Embedding Projections (BOREP), Random LSTMs (RandLSTM), and Echo State Networks (ESN) (Jaeger, 2001) and compare their encoders to bag-of-embeddings (BOE) and two well-studied sentence encoders with supervised and unsupervised objectives, the InferSent (Conneau et al., 2017a) and SkipThought (Kiros et al., 2015), respectively.

We use the same setups as those used in (Radford et al.). More specifically, GloVe with 300-dimensional embedding (Pennington et al., 2014) is used as pre-trained word embeddings and logistic regression classifier is used to train for an end-task with the default SentEval settings. Moreover, our DCT encoders have 300, 600, 900, and 1200 dimensions for embeddings size, while all other encoders in (Pennington et al., 2014) have 4096 dimensions, which are much higher in size than ours. Thus, it should be noted that it might not be an entirely fair comparison because projecting random embeddings to a higher-dimensional space leads to better performance of the sentence encoders, as shown in (Pennington et al.,

		Sentiment Analysis					SUBJ	Relatedness&Paraphrase			Inference	TREC
Model	Dim	MR	SST2	SST5	CR	MPQA		STSB	MRPC	SICK-R	SICK-E	
BOE	300	77.30	80.00	-	78.60	87.60	91.30	70.50	72.90	80.20	78.70	81.50
BOREP	4096	77.40	81.80	-	79.50	88.30	91.90	68.50	73.90	85.50	82.70	88.80
RandLSTM	4096	77.20	81.50	-	78.70	87.90	91.90	72.40	74.10	85.50	81.80	86.50
ESN	4096	78.10	83.00	-	80.00	88.50	92.60	74.40	73.40	86.10	83.10	87.90
Infersent-3	4096	**79.70**	**83.50**	-	**83.40**	88.90	92.60	**77.30**	**76.40**	**88.50**	**84.10**	**90.80**
SkipThought	4096	79.40	82.90	-	83.10	**89.30**	**93.70**	68.90	73.20	85.80	79.50	88.40
$c[0]$	300	**77.67**	**80.94**	**44.71**	**80.46**	**87.95**	91.72	63.01	72.35	80.87	78.24	83.80
$c[0:1]$	600	77.51	80.62	43.80	79.39	87.53	91.94	62.57	74.38	81.38	**78.40**	**88.20**
$c[0:2]$	900	77.19	80.51	43.94	79.77	87.41	**92.14**	62.91	74.09	**81.46**	77.37	87.80
$c[0:3]$	1200	76.98	79.85	42.94	79.00	87.17	92.10	**63.16**	**74.78**	81.10	77.76	87.80

Table 3.9: DCT embedding Performance in SentEval downstream tasks compared to SOTA models including SkipThought (unsupervised model), Infersent (supervised model), and different random encoders (BOE, BOREP, RandLSTM, and ESN) as reported in (Radford et al.). **Bold** highlights the best overall results, while **gray** highlights the best results for DCT encoders.

2014). Also, while all random sentence encoders in (Pennington et al., 2014) are used without any training, each model has different hyperparameters available for tuning in which a model selection has been done per-task over validation set performance. We further hypothesize that DCT might do a lot better if it had been used with better pre-trained word embeddings as shown in Section 5.

Table 3.8 shows the results on the probing benchmark tasks. We observe that DCT yields a comparable or better performance to all random sentence encoders. On the other hand, we observe that SkipThought outperforms DCT on syntactic and semantic tasks. On the surface level tasks, we observe that Infersent outperforms DCT models on the SentLen task and BOREP outperforms DCT models on the WC task.

Table 3.9 shows the results on the downstream benchmark tasks. Similar to the probing tasks, DCT yields a comparable or better performance to all random sentence encoders on the downstream tasks. We observe that Infersent outperforms all sentence encoders on most tasks, with the exception of MPQA and SUBJ tasks in which the best performances are obtained using Skiptought encoder.

Overall, although DCT encoders did not achieve the SOTA performances when compared to more complex encoders, their performances get very close to those models without any training or hyperparameters tuning and with more efficient computational speed on both

	20-NG			R-8			SST-5		
Model	P	R	F1	P	R	F1	P	R	F1
PCA	55.43	54.67	54.77	83.83	83.42	83.41	26.47	25.08	25.23
DCT*	61.07	59.16	59.78	90.41	90.78	90.38	30.11	30.09	29.53
Avg. vec.	68.72	68.19	68.25	96.34	96.30	96.27	27.88	26.44	24.81
p-means	*72.20*	71.65	**71.79**	96.69	96.67	96.65	33.77	33.41	33.26
ELMo	71.20	**71.79**	71.36	94.54	91.32	91.32	*42.35*	*41.51*	*41.54*
BERT	70.89	70.79	70.88	95.52	95.39	95.39	39.92	39.38	39.35
EigenSent	66.98	66.40	66.54	95.91	95.80	95.76	35.32	33.69	33.91
EigenSent⊕Avg	**72.24**	*71.62*	*71.78*	**97.18**	**97.13**	**97.14**	**42.77**	**41.67**	**41.81**
c[k]	*72.20*	71.58	71.73	*96.98*	*96.98*	*96.94*	37.67	34.47	34.54

Table 3.10: Performance in text classification (20-NG, R-8) and sentiment (SST-5) tasks of various models as reported in (Kayal and Tsatsaronis, 2019), where DCT* refers to the implementation in (Kayal and Tsatsaronis, 2019). Our DCT embeddings are denoted as c[k] in the bottom row, with k=0:2 for the SST-5 and k=0:1 for 20-NG and R-8 datasets. **Bold** indicates the best result, and *italic* indicates second-best.

benchmarks.

7 Comparison with Spectral Analysis Methods

Spectral analysis is frequently employed in signal processing to decompose a signal into separate frequency components, each revealing some information about the source signal to enable analysis and compression. Aside from our study, spectral methods have only been recently explored as a means of constructing sentence embeddings (Kayal and Tsatsaronis, 2019).[4] Kayal and Tsatsaronis (2019) propose EigenSent that utilizes Higher-Order Dynamic Mode Decomposition (HODMD) (Le Clainche and Vega, 2017) to construct sentence embedding. These embeddings summarize the transition dynamic properties in a sequence of words (i.e., sentence). In their work, they compare EigenSent with various sentence embedding models, including a variant from our implementation of the Discrete Cosine Transform (DCT*). In contrast to our implementation described in Section 2, DCT* is applied at the word level along the word embeddings dimension.

To ensure comparability, we use the same sentiment and text classification datasets as those used in Kayal and Tsatsaronis (2019). While the SST-5 dataset is the same sentiment

[4]independent from and in parallel with this work at time of publication.

	Embedding Size					
Model	**300**	**600**	**900**	**1200**	**1500**	**1800**
$c[k]$	0.73	0.73	0.73	0.73	0.73	0.73
p-means	5.02	10.38	16.13	24.47	32.66	40.89
PCA	54.00	52.58	49.54	50.37	47.06	52.66
EigenSent	77.53	117.08	156.33	196.35	356.08	825.54

Table 3.11: Run time of the Non-Parameterized Models on encoding sentences per second in 20-NG test set. Results are obtained from 5 trials average.

classification described in the Evaluation Dataset section, the 20 newsgroups (20-NG) and Reuters-8 (R-8)[5] are text classification datasets with 20 and 8 textual content classes, respectively. We also evaluate our model using the same pre-trained word embeddings, framework and approaches as described in their work. In particular, Word2vec word embeddings by (Mikolov et al., 2013) are used,[6] and for model training, simple linear-kernel support vector machine is used (Cortes and Vapnik, 1995). Table 3.10 shows the best results for the various models as reported in Kayal and Tsatsaronis (2019), in addition to the best performance of our model denoted as $c[k]$. Overall, the best results were achieved with $c[0:2]$ for the SST-5 and $c[0:1]$ for 20-NG and R-8.

Note that the DCT-based model, DCT*, described in Kayal and Tsatsaronis (2019) performed relatively poorly in all tasks, while our model achieved close to state-of-the-art performance in both the 20-NG and R-8 tasks. On both the 20-NG and R-8, our model outperformed EigenSent and generally performed better than or on par with p-means, ELMo, BERT, and EigenSent$\oplus$Avg. On the other hand, both EigenSent$\oplus$Avg and ELMo performed better than all other models on SST-5.

Training Speed. We also compare the speed of sentence encoders between the non-parametric models only on the 20-NG test set. For accurate comparison, we only compared the run-time our model (DCT) to non-deep neural models (PCA, p-means, and Eigensent). All models are run on a CPU (Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz). Table

[5]https://www.cs.umb.edu/ smimarog/textmining/datasets/
[6]https://drive.google.com/file/d/ 0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

3.11 shows the average run times obtained from 5 trials. It shows that, while the run-times increase as the number of dimensions increase on all models, the dimensionality increase has no effect on the run time of our model, $c[k]$. Also, DCT is by far the fastest model.

From the viewpoint of the computational cost, DCT is more efficient as the sentence embeddings are generated in one multiplication pass, while the eigend-based methods (PCA and Eigensent) generate sentence embedding by first obtaining eigenvalues and eigenvectors from the data using eigen decomposition technique. More specifically, DCT is data-independent transformation in which the transform matrix is known in advance and only defined by the input size. However, the PCA and Eigensent are data dependent techniques in which both are defined with respect to a given dataset. In particular, PCA rely on the covariance matrix to perform the eigen decomposition, where as the Eigensent rely on the Koopman operator derived from the multidimensional signal to perform the eigen decomposition (Kayal and Tsatsaronis, 2019).

8 Contextualized DCT Embeddings

Recently, BERT, a pre-trained contextualized word embedding, has achieved state-of-the-art results in a variety of NLP tasks (Devlin et al., 2019). In this section, we empirically compared the performance of our DCT based embeddings, which we extracted from the hidden state of the encoding layer of BERT on the time axis, and the CLS-BERT, which is the commonly-used representation corresponding to the [CLS] token in BERT. Furthermore, we examined the performance of the different pooling methods (the DCT and CLS) extracted from different BERT encoder layers.

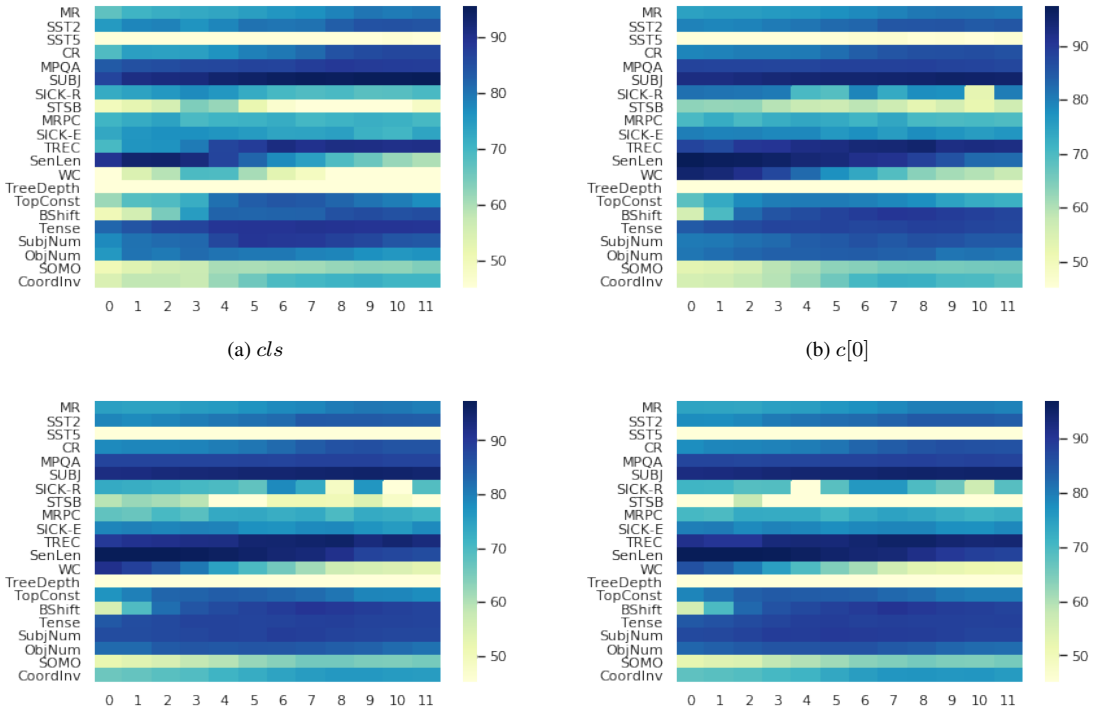
The layer-wise effect of the BERT encoder has been analyzed in different studies including the work by Ma et al. (2019), who performed a layer-wise analysis of BERT on the SentEval tasks. This is the most closely related study to ours. Their work solely focused on simpler pooling methods including: CLS, SEP, MEAN, and MAX sentence representations. However, our work is motivated by the promising results of the DCT-pooling compared to the

		Surface		Syntactic			Semantic				
BERT Layer	Model	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
0	CLS	89.50	15.60	34.00	61.50	50.30	81.80	77.50	75.3	50.50	54.60
	c[0]	**97.42**	**95.14**	35.78	68.47	55.96	84.83	80.78	81.10	**53.29**	55.76
	c[0:1]	97.36	91.68	37.3	77.24	55.40	**85.19**	87.03	82.66	52.32	66.61
	c[0:2]	96.98	85.85	**38.01**	**79.00**	**55.98**	85.00	**87.18**	**82.94**	52.69	**67.89**
1	CLS	93.20	54.40	35.70	68.10	55.80	84.30	80.50	80.20	53.60	58.10
	c[0]	96.77	**94.05**	37.58	73.44	**70.14**	86.45	80.99	82.07	54.01	56.39
	c[0:1]	**97.37**	88.80	39.07	79.85	69.73	**86.87**	87.12	82.79	53.83	67.40
	c[0:2]	96.99	83.60	**39.57**	**81.33**	69.89	85.77	**87.36**	**83.76**	**53.97**	**68.52**
2	CLS	93.34	59.19	36.01	68.69	64.33	86.42	81.25	79.35	**55.91**	58.02
	c[0]	96.45	**91.74**	39.69	78.44	82.50	87.30	81.99	83.14	55.56	58.30
	c[0:1]	**97.23**	85.36	41.00	83.34	82.21	**87.39**	87.55	**84.71**	55.35	68.59
	c[0:2]	97.00	80.25	**41.24**	**83.79**	**82.69**	86.98	**88.09**	84.60	54.56	**69.16**
3	CLS	91.70	69.20	34.90	71.7	74.60	86.70	81.60	81.10	57.00	57.20
	c[0]	95.58	**87.81**	41.03	80.40	**85.81**	88.29	82.75	84.04	**59.22**	59.67
	c[0:1]	**96.94**	80.86	43.03	83.61	85.55	**88.47**	87.93	85.31	58.40	69.43
	c[0:2]	96.61	74.89	**43.61**	**84.59**	85.69	87.67	**89.10**	**85.59**	58.48	**70.41**
4	CLS	86.70	69	35.40	80.60	82.10	88.80	86.30	80.20	60.20	61.60
	c[0]	94.34	**82.72**	42.48	80.54	87.12	88.19	84.34	84.01	**60.85**	62.26
	c[0:1]	**96.26**	74.92	43.64	**84.30**	**87.30**	88.38	88.40	85.24	59.69	**71.39**
	c[0:2]	95.80	69.58	**43.85**	84.24	87.05	**88.83**	**88.98**	**85.54**	59.97	71.28
5	CLS	81.90	60.40	35.00	83.00	82.00	**89.10**	88.80	79.70	60.00	65.70
	c[0]	93.68	**77.68**	42.55	80.46	**88.61**	88.48	85.07	84.39	**63.40**	66.50
	c[0:1]	95.41	70.16	**43.40**	83.87	88.37	88.34	88.34	85.64	62.45	72.91
	c[0:2]	**94.80**	64.31	43.19	**84.01**	88.29	88.95	89.53	**85.69**	61.50	**73.46**
6	CLS	77.30	53.00	34.80	**83.90**	82.20	**89.10**	88.40	79.70	60.60	69.20
	c[0]	91.80	**73.72**	42.31	79.83	89.45	87.85	86.36	84.60	**63.93**	70.77
	c[0:1]	**94.38**	66.01	42.53	83.38	**89.52**	88.45	89.26	85.47	63.87	74.55
	c[0:2]	94.10	61.10	**42.80**	83.70	89.40	88.30	**89.30**	**85.80**	63.50	**75.30**
7	CLS	74.10	48.90	33.90	83.60	82.40	**89.40**	88.20	78.50	60.80	70.00
	c[0]	91.20	**69.20**	41.50	78.80	90.50	88.20	85.30	84.00	65.30	72.60
	c[0:1]	**93.70**	61.50	41.70	82.50	**90.70**	88.80	89.00	**85.00**	**66.00**	75.70
	c[0:2]	**93.70**	56.50	**42.20**	**82.70**	90.60	88.80	**89.10**	84.80	64.30	**76.50**
8	CLS	68.90	43.50	32.60	82.00	84.80	**89.20**	87.20	78.00	61.90	70.80
	c[0]	88.70	**63.70**	40.00	77.90	**90.30**	89.00	86.40	83.80	65.50	72.40
	c[0:1]	91.80	56.90	40.60	81.30	90.20	89.00	88.50	**84.20**	**66.00**	76.50
	c[0:2]	**91.90**	53.40	**41.00**	**82.10**	89.80	88.50	**89.00**	84.00	65.70	**76.80**
9	CLS	65.91	40.57	31.42	80.28	85.47	**89.06**	86.39	76.36	62.64	71.56
	c[0]	86.05	**61.18**	37.91	74.81	**89.51**	88.64	86.11	81.32	**65.42**	70.29
	c[0:1]	88.50	55.80	39.60	79.30	89.20	88.50	88.10	83.7	64.90	76.30
	c[0:2]	**89.80**	52.70	**40.10**	**80.70**	89.00	88.50	**88.60**	**82.90**	64.70	**76.40**
10	CLS	61.90	39.00	30.20	79.00	84.80	**89.00**	84.20	75.40	62.40	70.30
	c[0]	82.71	**59.85**	36.37	72.97	**89.02**	88.13	85.10	81.67	**65.86**	70.02
	c[0:1]	87.90	54.90	38.00	79.10	88.60	88.40	87.8	**82.50**	64.70	75.90
	c[0:2]	**88.70**	51.70	**38.70**	**80.00**	88.80	88.40	**88.00**	82.20	64.30	**76.60**
11	CLS	59.90	45.00	30.70	76.80	85.40	**88.80**	83.70	76.30	63.90	70.40
	c[0]	82.50	**57.90**	35.30	72.10	88.20	88.00	85.00	81.20	**66.00**	68.50
	c[0:1]	87.11	53.30	**37.94**	78.15	88.22	88.35	87.78	81.81	65.51	75.78
	c[0:2]	**88.08**	51.26	37.06	**80.08**	**88.23**	88.54	**87.92**	**82.00**	65.33	**75.86**

Table 3.12: The results of the layer-wise BERT using the different pooling mechanisms: CLS, $c[0]$, $c[0:1]$, and $c[0:2]$ on the probing tasks. **Bold** highlights the layer-wise best result and **gray** highlights the best results across all layers.

		Sentiment Analysis					SUBJ	Relatedness&Paraphrase			Inference	TREC
BERT Layer	Model	MR	SST2	SST5	CR	MPQA		STSB	MRPC	SICK-R	SICK-E	
0	CLS	67.16	75.01	34.84	68.79	83.37	86.66	49.60	70.32	72.14	72.78	69.60
	c[0]	74.98	**79.63**	40.18	**79.40**	**88.46**	92.78	**63.30**	70.38	**82.06**	**80.09**	88.40
	c[0:1]	**75.18**	79.19	**40.59**	78.62	88.17	**92.84**	59.51	67.83	73.38	79.36	90.00
	c[0:2]	74.36	78.03	40.50	77.81	87.84	92.56	36.66	**70.43**	71.14	79.83	**91.40**
1	CLS	70.55	77.76	38.46	74.07	84.92	90.89	52.68	72.06	73.60	75.66	75.80
	c[0]	**75.69**	**79.13**	**42.90**	**79.61**	**88.36**	92.73	**62.65**	**72.99**	**81.56**	79.54	87.20
	c[0:1]	74.84	78.42	41.99	79.16	88.29	**92.94**	62.19	66.90	72.70	79.85	**91.40**
	c[0:2]	74.16	77.81	37.51	78.87	87.74	92.84	34.42	69.91	70.82	**80.17**	90.20
2	CLS	71.63	78.42	38.33	73.83	85.25	91.46	55.33	73.62	75.27	76.40	76.20
	c[0]	**75.64**	**79.46**	**43.17**	**80.32**	**88.24**	93.40	**62.90**	70.38	**81.35**	**79.62**	90.40
	c[0:1]	74.78	79.19	42.71	79.13	88.18	**93.49**	61.03	70.38	71.91	79.18	**91.80**
	c[0:2]	73.94	78.53	41.86	79.10	87.51	93.21	57.87	**73.22**	68.90	78.38	90.20
3	CLS	72.46	76.11	38.64	74.06	86.15	91.59	**64.05**	69.33	77.22	76.40	79.20
	c[0]	**76.22**	**80.45**	42.13	**80.24**	88.12	93.97	59.58	72.75	**80.80**	78.95	90.80
	c[0:1]	75.71	80.34	**42.40**	79.39	**88.29**	**93.95**	59.23	68.64	70.69	**79.89**	92.40
	c[0:2]	74.95	80.40	41.31	79.34	87.71	93.79	32.30	**72.87**	69.13	79.52	**93.20**
4	CLS	73.04	77.81	41.09	76.27	86.28	93.25	**61.89**	70.49	**75.56**	75.91	86.60
	c[0]	**76.60**	**81.27**	**44.89**	**80.83**	88.09	**94.42**	57.80	**73.97**	70.29	78.93	92.40
	c[0:1]	76.21	81.11	43.12	80.24	**88.27**	94.17	32.34	73.57	70.14	**79.70**	92.60
	c[0:2]	75.35	81.22	42.53	79.95	87.79	94.21	31.67	72.99	44.09	78.89	**93.60**
5	CLS	74.22	80.34	42.22	78.57	86.97	93.79	51.83	70.90	**72.10**	74.87	87.80
	c[0]	**77.72**	**82.76**	**45.48**	**82.26**	**88.77**	**94.78**	**57.16**	73.45	69.14	76.33	92.80
	c[0:1]	77.44	81.99	43.03	81.15	88.48	94.58	33.86	**73.74**	67.80	77.67	**94.60**
	c[0:2]	76.51	82.15	42.90	80.99	88.26	94.48	28.94	70.84	68.54	**77.78**	93.40
6	CLS	75.69	79.96	43.53	79.69	86.72	94.56	46.86	71.94	69.71	73.31	91.20
	c[0]	**78.68**	**83.14**	**46.29**	**84.22**	**88.71**	**94.85**	**59.09**	71.42	**79.27**	77.86	94.00
	c[0:1]	78.62	82.59	44.16	82.66	88.32	94.71	50.92	72.64	78.70	77.51	**94.80**
	c[0:2]	77.44	82.43	42.67	83.00	88.26	94.48	26.97	**73.39**	76.16	**78.57**	94.40
7	CLS	76.42	80.34	44.07	81.44	87.76	94.93	44.65	69.97	68.72	73.49	89.80
	c[0]	**79.77**	**85.50**	**46.47**	**85.55**	**88.75**	94.98	**56.99**	73.97	73.28	75.83	94.20
	c[0:1]	79.08	84.90	43.89	84.51	88.67	**95.07**	50.97	73.45	73.39	78.59	**95.20**
	c[0:2]	78.53	83.53	42.04	84.30	88.36	94.88	30.32	**74.55**	**76.13**	**79.01**	**95.20**
8	CLS	78.28	82.76	44.98	84.49	88.42	94.84	43.5	70.67	69.43	74.18	90.80
	c[0]	**80.94**	**85.67**	**45.48**	**86.53**	**88.95**	**95.28**	**53.10**	70.49	**78.02**	**78.20**	95.00
	c[0:1]	80.56	84.79	43.76	86.11	88.75	95.14	50.45	69.97	48.67	78.77	**95.60**
	c[0:2]	79.76	84.73	43.80	85.44	88.66	95.00	35.75	**73.74**	69.67	79.12	95.20
9	CLS	79.94	84.51	43.53	85.63	88.14	95.18	43.75	71.48	67.87	71.22	90.60
	c[0]	**81.25**	**85.94**	**45.84**	**86.71**	88.50	**95.59**	**55.87**	70.14	**77.66**	77.11	92.60
	c[0:1]	81.08	84.84	44.25	86.53	**88.60**	95.33	54.28	**73.28**	76.86	77.15	93.40
	c[0:2]	79.84	84.68	43.62	86.32	88.15	94.85	32.73	72.58	66.46	**77.67**	**94.20**
10	CLS	79.27	84.79	**45.61**	86.29	87.98	95.14	41.77	71.07	68.06	70.12	90.20
	c[0]	**81.17**	**84.84**	43.48	**86.87**	**88.37**	**95.44**	**52.19**	70.49	52.92	75.89	93.00
	c[0:1]	80.91	84.57	42.90	85.76	88.29	95.24	48.06	**72.81**	**69.49**	75.97	**94.60**
	c[0:2]	79.75	84.73	43.67	85.87	88.44	95.14	35.44	71.54	56.87	**77.43**	93.80
11	CLS	79.98	84.40	45.11	**86.40**	87.63	**95.49**	48.35	69.68	69.46	73.21	90.40
	c[0]	**80.82**	**85.34**	**46.11**	86.34	87.86	95.35	**56.48**	69.91	**80.22**	76.78	92.60
	c[0:1]	80.11	84.35	44.66	85.73	**88.31**	94.87	33.91	71.48	69.25	78.65	93.40
	c[0:2]	79.49	84.18	42.71	85.79	88.11	95.15	34.54	**72.29**	68.66	**78.71**	**94.00**

Table 3.13: The results of the layer-wise BERT using the different pooling mechanisms: CLS, $c[0]$, $c[0:1]$, and $c[0:2]$ on the downstream tasks. **Bold** highlights the layer-wise best result and **gray** highlights the best results across all layers.



(a) cls (b) $c[0]$

(c) $c[0:1]$ (d) $c[0:2]$

Figure 3.3: Sentence embedding performance of BERT cls token, $c[0]$, $c[0:1]$, and $c[0:2]$ from different layers of BERT. Color value of 90 corresponds to the best performance on a given task. Numbers on the x-axis denote the BERT layer.

AVG-pooling in the previous section, which utilized the context-free FastText embeddings. Thus, instead of relying on context-free word embeddings, our approach explores the contextualized hidden states of BERT to extract sentence representations using DCT-pooling.

We followed the same evaluation suite, "the SentEval," used in the previous section. However, for the classifier, we relied only on a logistic regression layer added on top of the sentence representations. Figure 3.3 shows the results of the layer-wise BERT using the different pooling mechanisms: CLS, $c[0]$, $c[0:1]$, and $c[0:2]$; the complete numerical results are shown in Table 3.12 and Table 3.13. For the downstream tasks, regardless of pooling methods used, higher BERT layers generally performed better than lower layers. On the contrary, for the probing tasks, lower and middle layers achieved the best performances. This observation aligns with the finding of Ma et al. (2019), in which the author discussed the effect of a layer-wise BERT encoder on the performance of a given model. Aside

	Surface		Syntactic			Semantic				
Model	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
$c[0]$ (FT)	**83.47**	**91.11**	33.42	64.50	50.42	**86.80**	79.89	79.61	49.59	51.87
$c[0:1]$ (FT)	82.15	89.16	34.44	74.40	49.82	86.66	**84.47**	80.88	49.44	66.19
$c[0:2]$ (FT)	81.55	86.77	**35.61**	**77.08**	**52.07**	86.46	84.31	**81.50**	**50.21**	**66.80**
$c[0]$	**97.42**	**95.14**	42.48	80.40	90.50	88.20	86.36	84.60	65.50	72.40
$c[0:1]$	97.36	91.68	43.64	83.61	**90.70**	88.80	89.26	85.47	**66.00**	76.50
$c[0:2]$	96.98	85.85	**43.85**	**84.59**	90.60	88.80	**89.30**	**85.80**	65.70	**76.80**
CLS	89.50	15.60	35.40	71.70	82.40	**89.40**	88.40	79.70	61.90	70.80

Table 3.14: Comparison of BERT cls and BERT-DCT models on the probing tasks. FT refer to the DCT model results based on FastText word embeddings.**Bold** denotes the best results across BERT embeddings, and **gray** highlights the best results across all models.

	Sentiment Analysis					SUBJ	Relatedness&Paraphrase			Inference	TREC
Model	MR	SST2	SST5	CR	MPQA		STSB	MRPC	SICK-R	SICK-E	
$c[0]$ (FT)	**78.15**	**82.98**	**44.84**	79.60	**87.98**	92.37	**71.11**	72.41	82.61	79.54	84.80
$c[0:1]$ (FT)	77.86	81.99	44.21	**80.29**	87.88	92.36	70.82	**73.80**	82.73	**79.68**	88.20
$c[0:2]$ (FT)	78.02	82.48	43.03	79.81	87.45	**92.59**	70.36	73.68	**82.95**	79.60	**90.00**
$c[0]$	**81.25**	**85.94**	**46.47**	**86.87**	**88.95**	**95.59**	59.58	73.97	**82.06**	79.54	95.00
$c[0:1]$	81.08	84.84	43.89	85.76	88.75	95.33	59.23	73.45	73.38	79.85	**95.60**
$c[0:2]$	79.84	84.68	42.04	85.87	88.66	94.85	32.30	**74.55**	71.14	**80.17**	95.20
CLS	79.94	84.51	44.07	86.29	88.42	95.18	**64.05**	69.97	72.14	75.66	90.80

Table 3.15: Comparison of BERT cls and BERT-DCT models on the downstream tasks. FT refer to the DCT model results based on FastText word embeddings.**Bold** denotes the best results across BERT embeddings, and **gray** highlights the best results across all models.

from the layer-wise performance, we compared the performance of the CLS model and the DCT-based models. Table 3.14 shows the best BERT performances across different layers for each task category within the probing. On all task categories, the $c[0]$ model outperforms the CLS token. Further improvements are seen for the syntactic and semantic categories, which belong to the probing benchmark, with larger number of coefficients (K), in which $c[0:2]$ model achieved the highest performance gains. We further compare the performance of BERT word embeddings to the results of Fasttext word embeddings, in which we observed significant performance gains.

On the other hand, the results for the downstream categories on Table 3.15, for most categories there is no significant improvement with larger K values, except for $Inference$ and $TREC$ datasets. While the performance gains on the $Inference$ task is only about 0.8% with $c[0:2]$ model compared to the $c[0]$ model, the performance gains on the $TREC$

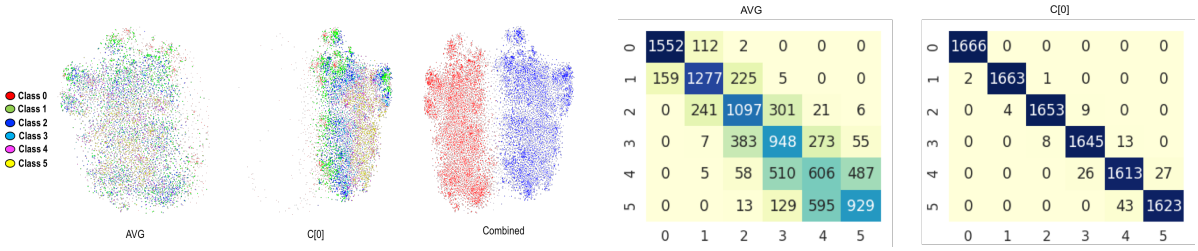
task is 1.37% with $c[0:2]$ model compared to the $c[0]$ model. Similar to the probing tasks, we observed significant performance gains using BERT word embeddings compared to FastText embeddings on most of the downstream tasks.

Overall, in our experiments, we noted significant changes in the performances using BERT word embeddings without fine-tuning instead of the static pre-trained FastText embeddings utilized in Section 5. These improvements are attributed to contextual information provided by BERT word embeddings.

9 Analysis

In this section, we qualitatively identify the differences between the AVG sentence embeddings and our method, DCT sentence embeddings.

9.1 AVG vs. c[0]



(a) The joint t-SNE embeddings of the AVG and $c[0]$ from the Length task, and both together (AVG in red, $c[0]$ in blue.)

(b) Conf. Matrix: left is the AVG result, and right is the $c[0]$ result.

The AVG and $c[0]$ both compute the mean of the word embeddings, but the element-wise sum in the AVG is divided by the sentence length, while in $c[0]$ it is divided by the square root of the sentence length. Dividing by the square root of the sentence length has been used in sentence embeddings literature, in particular the USE model uses this method to compute the final fixed length representation, which mitigates the impact of the sentence length variations on the sentence embeddings (Cer et al., 2018).

The performance gain of dividing by the square root of the sentence length is clearly shown in the **Length** task from the probing benchmark. This task measures how well a given

representation of a sentence encodes its length. The task is framed as a six-way classification corresponding to the length intervals as described in Section 3. In this task, $c[0]$ outperforms AVG by a significant margin. Therefore, we performed a qualitative analysis of the sentence representation from both models on the test split of the Length dataset. The joint t-SNE (Liu et al., 2019a)[7] is used to draw a comparison on how the AVG embedded sentences relative to the $c[0]$ in Figure 3.4a. The t-SNE plot of $c[0]$ shows that the clusters are more compact with a very clear structure and separation. It can be observed that most partial overlaps are between clusters belonging to neighboring classes. For example, the green cluster corresponds to class '1', in which the sentence lengths are between (9-12). This cluster overlaps with the red cluster, which corresponds to class '0', and the blue cluster, which corresponds to class '2', in which the sentence lengths are between (5-8) and (13-16) respectively. In contrast, the t-SNE of the AVG shows more scattered clusters, including fewer clear points with multiple classes overlapping. This clearly shows the benefit of dividing by the square root instead of the sentence length. The joint embeddings plot shows that the AVG and $c[0]$ are mapped to different locations.

To further investigate these observations, we draw the confusion matrix of both models in Figure 3.4b. The confusion matrix of the AVG shows that there is a lot of confusion between classes, especially with longer sentences. However, in $c[0]$ many of the examples are predicted correctly with higher accuracy.

9.2 AVG vs. c[k]

We performed an additional qualitative analysis on the TREC dataset, focusing on the sentence representation from different models. In particular, we used a dimensionality reduction technique, K-means clustering purity, and a nearest neighbor analysis.

Figure 3.5 shows the t-SNE plots of the AVG as well as DCT coefficients $c[0]$ to $c[0:5]$. As previously noted, the AVG and the first DCT coefficient $c[0]$ both represent the average of

[7] https://github.com/GWUvision/Embedding_visualization

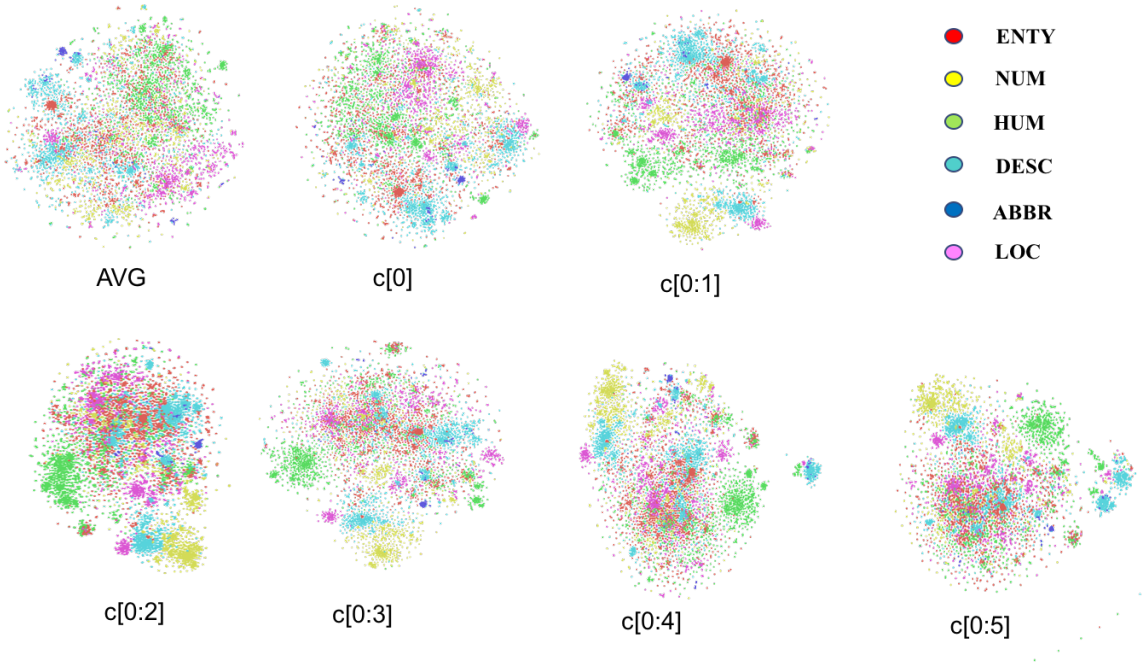


Figure 3.5: T-SNE visualization of different DCT coefficients and AVG on the TREC dataset.

the word embeddings in the sentence. Hence, similar clustering is seen in their t-SNE plots. By increasing the DCT coefficient numbers, the clustering appears to be more accurate at grouping sentences according to their labels.

Table 3.16 shows examples of nearest neighbors based on the cosine distance between sentences' embeddings using different DCT coefficients and AVG. It should be noted that when adding more coefficients, the retrieved results are more similar in their structures. The first query "Query 1" in Table 3.16, shows how adding more coefficients help in retrieving more examples that belong to the same question types as shown in the retrieved results of $c[0:1]$ to $c[0:3]$ when compared to AVG and $c[0]$ results. Furthermore, the structure similarity can be clearly seen in the second example in Table 3.16. While all the results retrieved from all models were from the same class "NUM", AVG and $c[0]$ did not account for the structural similarity, in which the top retrieved example syntactically different then the given query. On the other hand, the results from $c[0:2]$ model are more similar in the exact structure and word orders of the sentences.

Furthermore, to check how well the clustering based on DCT embeddings has performed,

Model	Sample	Label
Query 1:	**How did serfdom develop in and then leave Russia ?**	DESC
AVG	How can people from Ireland come to the U.S. , work , and then return to Ireland ?	DESC
	How many colonies did Germany get to keep after World War I ?	NUM
	How many dollars a day did Arthur Frommer say you could get by on in Europe in 1968 ?	NUM
$c[0]$	When did communist control end in Hungary ?	NUM
	How much did Alaska cost when bought from Russia ?	NUM
	How far is Yaroslavl from Moscow ?	NUM
$c[0:1]$	How much did Alaska cost when bought from Russia ?	NUM
	How did the tradition of best man start ?	DESC
	How did names come about ?	DESC
$c[0:2]$	How much did Alaska cost when bought from Russia ?	NUM
	How did the months of the year get there name ?	DESC
	How do I start a bank ?	DESC
$c[0:3]$	How do I do this ?	DESC
	How did the months of the year get there name ?	DESC
	How did the tradition of best man start ?	DESC
Query 2:	**How old is Stevie Wonder ?**	NUM
AVG and $c[0]$	How old was Stevie Wonder when he signed with Motown Records ?	NUM
	How old is Benny Carter ?	NUM
	How old is Bernadette Peters ?	NUM
$c[0:1]$	How old is Benny Carter ?	NUM
	How old is Bernadette Peters ?	NUM
	How old is the universe ?	NUM

Table 3.16: Nearest neighbors (based on the cosine similarity) to: Query 1 and Query 2. The label refer to the question categorizes as defined on the TREC data set, where DESC denotes description and NUM denotes number. The **green** highlights the level of noise in the retrieved results. While in "Query 1" the level of noise impacted the categorizes of the retrieved examples, in "Query 2" the level of noise impacted in the structural characteristic of the retrieved examples.

we calculate the clustering purity using K-means clustering with different K values ranging from 6 to 50, where 6 is equal to the number of classes in the TREC dataset. In order to calculate clustering purity, we used the following formula (Manning et al.):

$$P(C, L) = \frac{1}{N} \sum_{k} \max_{j} |c_k \cap l_j|, \tag{3.4}$$

where N is the total number of examples, $C = \{c_1, ..., c_k\}$ is the clusters set, and $L = l_1, ..., l_k$ is the labels set. As shown in Table 3.17, c[0:2] clustering with k=6 resulted in a higher purity score than other models. It can be noted that larger k clustering resulted in better purity in all models. However, the rates of increase in AVG and $c[0]$ are lower than those of models $c[0:1]$ and $c[0:2]$, this is clearly shown in Figure 3.6.

	K=6	K=10	K=20	K=30	K=40	K=50
AVG	0.409	0.477	0.505	0.526	0.553	0.552
$c[0]$	0.386	0.479	0.509	0.538	0.538	0.560
$c[0:1]$	0.468	0.479	0.568	0.586	0.599	0.614
$c[0:2]$	0.487	0.524	0.564	0.599	0.599	0.624

Table 3.17: Clustering purity with increasing k and DCT coefficients on TREC

Figure 3.6: Correlation between clustering purity and increasing number of coefficients on the TREC data.

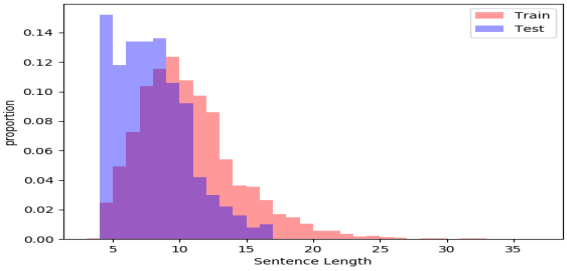


Figure 3.7: Comparing number of words per sentence distribution in the TREC train and test sets.

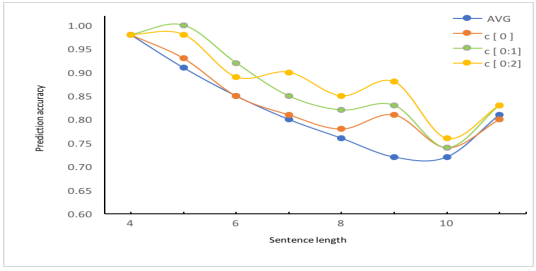


Figure 3.8: TREC accuracy vs. sentence length for AVG and various DCT models.

9.3 Sensitivity to Sequence Length

Since natural language datasets consist of sentences with different lengths, we performed additional analysis to study the impact of sentence length on the models' performance. To make these comparisons, we filtered the sentences in the TREC test data based on their length and divided them into subsets with varying lengths. Specifically, for the test data in the TREC task, we split them into 3 subsets with the following intervals: Bin-1: [1-5], Bin-2:[6-10], and Bin-3:[11-]. Table 3.18 shows the models' performance based on sentence length. While all models, as expected, perform reasonably well on short sentences, model performance dropped on longer sentences. As shown in Figure 3.8, there is almost a linear correlation between sentence length and the accuracy degradation. Nevertheless, AVG and $c[0]$ seem to degrade faster than other models.

	Test	Bin-1	Bin-2	Bin-3	Weighted-Acc
AVG	83.20	95.55	78.03	81.66	85.08
$c[0]$	84.80	96.29	80.65	80.00	85.64
$c[0:1]$	88.40	99.25	84.59	83.33	89.05
$c[0:2]$	89.60	98.51	86.88	83.33	89.57

Table 3.18: Accuracy against sentence length on the TREC test set. The number of words in a sentence, in bins of size 5, where Bin-1 contains sentences with length [1-5], Bin-2 with [6-10], and Bin-3 with [11-].

Premise	Hypothesis	label
Negation		
A woman is cutting an onion	A woman is **not** cutting meat	Entailment
The American footballer is **not** wearing a red and white strip	The American footballer is wearing a red and white strip	No-Entailment
Passivization		
A big wave is being ridden by a surfer across dark green water.	The surfer is riding a big wave.	Entailment
An old man with a grey beard is riding a bike.	The bicycle is being ridden by a man wearing a beret on a recreation trail.	No-Entailment
Substitution		
A **man** is riding a dirt bike down a dirt hill.	A **person** is riding a dirt bike down a dirt hill.	Entailment
A man on inline skates is **skating** at a skate park.	A man on inline skates is **resting** at a skate park.	No-Entailment

Table 3.19: Examples of sentence pairs in SICK with different linguistic phenomena including: Negation, Passivization, and Substitution.

9.4 Sensitivity to Linguistic phenomena

In this section, we test the sensitivity of AVG and DCT to three different linguistic phenomena that imply syntactic and semantic variations on the SICK dataset for entailment (SICK-E). Entailment is a task designed to capture the inference relationship between a pair of sentences. The SICK-E are labelled for 3-way classification using the following relationships: Entailment, in which the sentences in the pair entail each other; Contradiction, when the sentences in the pair contradict one another; and Neutral, when there is no clear relationship between the sentences in the pair. In this analysis, for simplicity, we treated the relationships as a two-way classification: Entailment and No-Entailment, in which Neutral and Contradiction are both mapped into the No-Entailment class.

	AVG		$c[0]$		$c[0:1]$		$c[0:2]$		$c[0:3]$	
Predict / **True**	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment
Entailment	0%	0.84%	0%	0.84%	0%	0.84%	0%	0.84%	0%	0.84%
No-Entailment	12.32%	86.83%	12.60%	86.55	12.60%	86.55	12.04%	87.11%	**9.24%**	**89.91%**

Table 3.20: Confusion matrix of different models when negation is used.

Generally, there are different linguistic phenomena involved in semantic tasks such as entailment. In this analysis, we limited our focus to three phenomena relevant to the entailment: **Negation** (if one text has a negation mark such as "not"), **Passivization** (if one text is in active voice and the second is in passive voice), and **Substitution** (if the pair differs by one word; one word is substituted by another). While negation and passivization involve some kind of syntactic and structural modifications in a sentence, substitution only involves semantic changes. Table 3.19 shows examples of sentence pairs from the SICK-E dataset with negation, passivization and substitution.

First, we assess the models' capability of identifying the correct relation between a pair of sentences when one sentence contains a negation word. From the SICK-E test-split, a total of 357 pairs have been collected in which an explicit negation mark, indicated by the negation relation in the syntactic dependency,[8] contained in one of the text in a pair. We examined the differences between AVG, $c[0]$, $c[0:1]$, $c[0:2]$, and $c[0:3]$. Table 3.20 shows the confusion matrix of different models in predicting the correct class with sentence that contain negation marks. It can be seen that while AVG and $c[0]$ ignores negation because they do not encode word order, higher order DCT models predict the correct class for more examples because they do encode word order. In particular, $c[0:3]$ is the best model at predicting No-Entailment with a negation mark in the sentence, where negation words strongly indicate a contradiction.

We assess the models' capability at identifying the correct relationship between a pair when one text is expressed using active voice and the other is expressed using passive voice. In textual entailment, sentences in a pair entail each other if both convey the same meaning

[8]SpaCy has been used to extract the syntactic dependency relations (Honnibal and Montani, 2017)

	AVG		$c[0]$		$c[0:1]$		$c[0:2]$		$c[0:3]$	
Predict / **True**	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment
Entailment	12.96%	22.22%	16.66%	18.51%	**18.51%**	**16.66%**	14.81%	20.37%	14.81%	20.37%
No-Entailment	5.55%	59.25%	7.40%	57.40%	**7.40%**	**57.40%**	9.25%	55.55%	7.40%	57.40%

Table 3.21: Confusion matrix of different models when passivization is used.

	AVG		$c[0]$		$c[0:1]$		$c[0:2]$		$c[0:3]$	
Predict / **True**	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment	Entailment	No-Entailment
Entailment	**42.26%**	**10.19%**	41.61%	10.85%	40.13%	12.33%	38.48%	13.98%	38.81%	13.65%
No-Entailment	25.16%	22.36%	21.87%	25.65%	17.10%	30.42%	17.59%	29.93%	**14.80%**	**32.73%**

Table 3.22: Confusion matrix of different models when substitution is used.

even when they are expressed differently. In order to evaluate this phenomena, a total of 54 pairs are extracted from the SICK-E test-set using syntactic dependency,[8] which meet the requirements of passivization. Table 3.21 shows the confusion matrix results of this subset. It can be seen that DCT models perform better than the AVG on identifying the entailment relationship. However, increasing the number of DCT coefficients has no effect on improving model performance. For example, $c[0:1]$ is marginally better than $c[0]$, however, the number of true positive dropped when adding more coefficients in $c[0:2]$ and $c[0:3]$. This observation aligned with our initial assumption about the drawbacks related to passivization, which emphasized the challenges of passivization in a linear model such as DCT.

We investigate the models' capability at capturing the correct relationship between a pair of sentences when they differ by one word. For example, synonyms and antonyms are common phenomena often seen in textual inference. In particular, the pair entail each other if the words are synonyms and contradict each other if the words are antonyms. To extract examples that differ by only one word, we calculate the Levenshtein edit-distance between a pair of sentences.[9] In this case, while the sentences in the pair varies semantically, the length, order, and structure information are exactly the same. Interestingly, while AVG performs

[9]We use NLTK.

better than all other DCT models on the Entailment cases, the $c[0:3]$ model performs better than all other models in the No-Entailment cases.

This finding, which is consistent with the findings in the previous sections, validate the effectiveness of the DCT based models on encoding syntactic information in a given input better than the standard AVG model.

10 Conclusion

In this chapter we introduced a novel method to compute sentence embeddings using the Discrete Cosine Transform (DCT). The proposed mechanism efficiently compresses variable-length sentences into fixed-length vectors in a manner that preserves some of the structural characteristics of the original sentences. By applying DCT on each feature along the word embedding sequence, we efficiently encode the overall feature patterns as reflected in the low-order DCT coefficients. We showed that these DCT embeddings reflect average semantic features, similar to the vector averaging but with a more suitable normalization, in addition to syntactic features like word order. Experiments using the SentEval suite showed that DCT embeddings outperform the commonly-used vector averaging on most tasks, particularly tasks that correlate with sentence structure and word order. We verified qualitatively that DCT based models encode subtle syntactic variations in the input better than the standard model such as AVG. Without compromising practical efficiency relative to averaging, DCT provides a suitable mechanism to represent both the average of the features and their overall syntactic patterns. This makes DCT a suitable alternative mechanism to the popular bag of words approach by exploiting a word embedding space to generate a general-purpose sentence representation, particularly in tasks that require syntactic information.

Chapter 4: Universality of DCT Embeddings

In the previous chapter, we proposed Discrete Cosine Transform (DCT) as an alternative that captures the underlying syntactic characteristics of a given text without compromising practical efficiency compared to averaging. However, as with most other sentence encoders, the DCT sentence encoder in Chapter 3 was only tested in English. To this end, in this chapter, we investigate the efficiency of a DCT encoder across different languages that exhibit different linguistic properties in both multi-lingual and cross-lingual settings.

1 Mulit- and Cross-lingual Sentence Encoding

Earlier work in the realm of NLP-distributed representations was centered around the idea of learning word representations, in which a word's statistical, syntactic, and semantic information is encoded in real-valued vectors. As a result, several approaches have been developed to compute different types of word embeddings. For example, earlier dense word embedding vector representations such as Word2Vec reflect context-free embeddings while more recent proposed word embeddings such as ELMo and BERT allow for the computation of context-rich embeddings (Mikolov et al., 2013; Peters et al., 2018; Devlin et al., 2019). However, for tasks that involve sentence-level understanding, individual word representations are not suitable for learning adequate information about longer sequences. Therefore, a number of sentence encoding representations have been developed to accommodate this need; some of these models are discussed in (Hill et al., 2016; Logeswaran and Lee, 2018; Conneau et al., 2017b), yet most of these representations have focused on English only.

To generate sentence representations in different languages, the most obvious solution is to train monolingual sentence encoders for each language. However, training a heavily parameterized monolingual sentence encoder for every language is inefficient and computationally expensive. On the other hand, utilizing a non-parameterized model with ready-to-use

word embeddings is another efficient approach to generate sentence representations. For example, averaging word embeddings is one of the most common and efficient approaches to obtain sentence encoders (Arora et al., 2017).

In addition to the averaging method, a number of non-parameterized models have been proposed to derive sentence representations from pre-trained word embeddings (Rücklé et al., 2018; Yang et al., 2019b; Kayal and Tsatsaronis, 2019). However, most of these models, including averaging, disregard structural information, which is an important aspect of any given language. Recently, we proposed a structural-sensitive sentence encoder, which utilizes Discrete Cosine Transformation (DCT) as an efficient alternative to averaging. We showed that this approach is versatile and scalable because it relies only on word embeddings, which can be easily obtained from large unlabeled data. Hence, in principle, this approach can be adapted to different languages. Furthermore, having an efficient, ready-to-use language-independent sentence encoder can enable knowledge transfer between different languages in cross-lingual settings, empowering the development of efficient and performant NLP models for low-resource languages.

Generally, finding a multi- and cross-lingual sentence encoder that generalizes across different languages and is computationally efficient is a challenging task. Thus, a natural question would be: how well does the proposed DCT encoder work for multi- and cross-lingual scenarios, i.e. across different languages that exhibit different linguistic properties? In this chapter, we empirically investigate the generality of DCT representations across languages as both a single language model and a cross-lingual model in order to assess the effectiveness of DCT across different languages.

To this end, we extend the application of DCT to additional languages in a multi-lingual setting, where a fixed-length representation is generated for each language. These representations are capable of learning cross-lingual vectors by mapping them onto a shared vector space at the sentence-level. Thus, we further present a non-parameterized sentence-level-based approach for cross-lingual mapping without any additional training parameters.

In this context, the DCT encoding is used to generate the sentence representations, which are then used in a sentence level alignment process.

2 Related Work

Sentence Encoding: Supervised and unsupervised approaches for sentence encoding, such as InferSent (Conneau et al., 2017a) and Skip-Thought (Kiros et al., 2015), respectively, have been proposed based on heavily-parameterized neural models. One of their drawbacks is that they are inefficient both at training and inference times. Especially in contrast to sentence encoders which employ simple averaging based solely on pre-trained word embeddings— a Bag-of-Words (BoW) method, averaging has been proven to perform reasonably well without the overhead computations of sophisticated neural models (Arora et al., 2016). Given the superior results yielded by averaging, alternative parameter-free, fast, and high-quality sentence encoders that rely on pre-trained word embeddings which crucially do not require further training are garnering more interest. A thorough overview of the different sentence encoders is given in Chapter 2.

Most of the proposed methods suffer from indifference toward underlying syntactic characteristics of a given text. Recently, we used DCT as an efficient alternative to the commonly-used vector averaging model, in which the lower-order DCT coefficients preserve some of the structural characteristics of the original sentences. As shown in Chapter 3, DCT is a suitable alternative mechanism to the popular BoW approach as it exploits a word embedding space to generate a general-purpose sentence representation, particularly in tasks that require syntactic information. In this chapter, we extend the applicability of DCT sentence encoders to new languages thereby generating multi-lingual representations without additional training.

Sentence-level Cross-lingual Mapping: Aside from the multi-lingual representations, increased attention has been given to cross-lingual learning frameworks in recent years. A number of approaches have been proposed as a way to learn cross-lingual representations

that enable better NLP applications across languages, especially in low-resource scenarios. The vast majority of cross-lingual mapping relies on word-level alignment; a thorough overview can be found in (Ruder et al., 2019). Similar to word-level alignment, sentence-level alignment has proven to be effective at generating textual representations that perform well across multiple languages (Aldarmaki and Diab, 2016, 2019a; Conneau et al., 2018b; Aldarmaki and Diab, 2019b; Fu et al., 2020).

Assuming a supervised framework, Aldarmaki and Diab (2016) utilized co-occurrence counts leveraging matrix factorization to simultaneously learn word and sentence representations. As discussed in (Aldarmaki, 2019), this method has a major practical limitation due to the model's inefficiency during training and inference times. More recently, notable improvements have been observed using neural sentence embedding proposed in (Conneau et al., 2018b; Aldarmaki and Diab, 2019b). Both approaches utilized the same supervised inference-based classifier, ($InferSent$) detailed in (Conneau et al., 2017b), with parallel natural language inference (NLI) data to align a multilingual sentence encoder.

In an unsupervised setting, Aldarmaki and Diab (2019b) uses unsupervised Sequential Denoising Auto-Encoder (SDAE) from (Hill et al., 2016) to learn cross-lingual alignments between languages. Aldarmaki and Diab (2019a) incorporates contextualized word embeddings, which are dynamically generated from ELMo (Peters et al., 2018) to form a contextualized sentence embedding for sentence-level cross-lingual mappings. Also, Fu et al. (2020) proposes an Adversarial Bi-directional Sentence Embedding Mapping (ABSent) framework to learn cross-Lingual sentence representations from parallel data. Specifically, the ABSent framework utilizes a GAN-based technique (Goodfellow et al., 2014) to learn sentence-level transformations between two languages.

Leveraging a multilingual training objective is one of the most successful recent approach for cross-lingual encoders (Artetxe and Schwenk, 2019; Conneau and Lample, 2019). For example, LASER framework uses parallel corpora to train a BiLSTM encoder that generates a 1,024-dimensional language-agnostic sentence representation (Artetxe and Schwenk, 2019).

Similarly, Conneau and Lample (2019) uses a Transformer architecture (Vaswani et al., 2017) with 1024 hidden units to learn text representation from two unsupervised multilingual training objectives: Causal Language Modeling and Masked Language Modeling.

Although the neural sentence embedding frameworks in (Aldarmaki and Diab, 2019a; Conneau et al., 2018b; Aldarmaki and Diab, 2019b; Fu et al., 2020; Artetxe and Schwenk, 2019; Conneau and Lample, 2019) result in reasonable cross-lingual alignment and performance, they suffer from several drawbacks. Typically, these models are not practical for real-time applications due to their underlying complexity and cumbersome computational requirements. For example, the LASER framework leverages a significant amount of parallel data (more than 200 million parallel sentences) to train an encoder to produce cross-lingual sentence embeddings (Artetxe and Schwenk, 2019) . Thus, in this chapter, we investigate the effectiveness of DCT sentence embeddings as an alternative in cross-lingual mapping using the sentence-level alignment framework introduced in (Aldarmaki and Diab, 2019a).

3 Multi-lingual DCT Embeddings

The non-parameterized DCT mechanism is utilized to generate sentence embeddings. Sentence embeddings are generated by concatenating different orders of DCT coefficients from pre-trained word embeddings without additional training parameters. This allows the model to be easily extended to additional languages with existing monolingual word embeddings. Given the promising results of the DCT model discussed in Chapter 3, when modeling sentences in English, in this section, we used the same model to learn a separate encoder for each language.

In this section, we evaluate the performance of the DCT embeddings on different languages, namely English, Spanish, French, German, and Russian. For all languages, we rely on a pre-trained word embedding to be used as a source for the DCT sentence encoder.

3.1 Evaluation Datasets and Experimental Setups

To evaluate DCT embeddings across different languages, we used the probing benchmark provided by Ravishankar et al. (2019), which includes a set of multi-lingual probing datasets. The benchmark covers five languages: English,[1] French, German, Spanish and Russian, derived from Wikipedia. The task set is comprised of 9 Probing tasks. Ravishankar et al. (2019) used the datasets to evaluate different sentence encoders trained by mapping sentence representations to English. Unlike Ravishankar et al. (2019), we used the datasets to evaluate DCT embeddings for each language independently. In addition to the DCT embeddings, we used vector averaging to extract sentence representations from the pre-trained embeddings as baselines.

For model evaluations, we utilized the SentEval framework introduced in (Conneau and Kiela, 2018). In all experiments, we used a single-layer MLP on top of DCT sentence embeddings with the following parameters: kfold=10, batch_size=128, nhid=50, optim=adam, tenacity=5, epoch_size=4. For the word embeddings, we relied on the publicly available pre-trained FastText embeddings introduced in (Grave et al., 2018).[2] and detailed descriptions of the framework and the Probing tasks contained within the benchmarks are given in Chapter 3.

3.2 Evaluation Results

Table Table 4.1 shows the complete numerical results. However, for better visual understanding, the probing results of different languages relative to English are shown a heat-map in Figure 4.1. Overall, French (FR) seems to be the closest one to English (EN) followed by Spanish (ES) then German (DE) and then finally Russian (RU) across the various DCT coefficients. Higher coefficients reflect majority better performance across most tasks for FR, ES and DE. We see the most variation with worse results than English on

[1]The English data in (Ravishankar et al., 2019) is drawn from a different domain than the one in (Conneau et al., 2018a)

[2]The pre-trained word embeddings are available at: https://fasttext.cc

	Language	AVG	c[0]	c[0:1]	c[0:2]	c[0:3]	c[0:4]
SentLen	EN	56.28	**89.03**	88.91	88.95	88.7	88.08
	ES	59.92	89.59	90.00	89.8	89.73	**90.05**
	FR	57.9	**93.72**	93.44	93.14	92.82	92.38
	DE	53.41	**88.81**	88.36	88.16	87.54	87.69
	RU	54.42	**89.66**	89.12	89.18	88.26	88.04
WC	EN	26.97	**66.69**	64.55	62.49	60.39	59.08
	ES	25.4	**64.80**	62.18	60.62	58.76	57.64
	FR	27.14	**68.60**	66.13	64.71	62.8	61.04
	DE	29.33	**64.99**	64.52	63.93	63.12	61.54
	RU	36.33	**67.50**	65.58	64.69	62.69	61.32
Bshift	EN	54.78	54.98	54.58	54.86	54.81	**55.58**
	ES	54.7	54.52	54.53	54.21	54.71	**55.77**
	FR	54.69	54.7	54.68	54.91	55.53	**56.50**
	DE	54.23	54.22	54.35	54.43	54.6	**56.46**
	RU	56.48	56.8	56.81	56.28	57.4	**58.51**
TreeDepth	EN	41.34	45.18	48.64	49.84	49.44	**50.47**
	ES	42.9	48.53	52.29	53.34	**53.87**	53.54
	FR	41.06	47.68	50.05	51.65	**52.27**	52.15
	DE	37.06	41.97	45.14	47.33	**47.55**	47.36
	RU	35.27	39.21	40.76	**41.02**	40.65	40.51
Tense	EN	86.49	89.23	91.83	92.17	**92.26**	92.21
	ES	94.52	95.97	**96.68**	96.67	96.62	96.53
	FR	91.96	94.06	95.7	95.96	**96.12**	95.99
	DE	94.13	94.71	95.82	**96.44**	96.28	95.92
	RU	86.07	86.39	90.28	**90.4**	90.16	90.38
CoordInv	EN	73.47	74.22	84.56	**87.20**	87.03	87.19
	ES	67.08	68.13	81.61	84.15	85.17	**85.77**
	FR	71.06	71.12	85.97	88.03	89.21	**89.61**
	DE	74.25	74.33	89.99	92.52	93.45	**94.09**
	RU	60.33	60.77	79.95	83.13	84.03	**84.34**
SubjNum	EN	76.46	77.41	80.49	81.68	81.76	**82.31**
	ES	86.4	86.68	89.34	90.42	90.12	**90.84**
	FR	88.48	88.62	91.05	92.23	92.72	**92.76**
	DE	75.94	75.78	78.79	78.9	79.25	**79.28**
	RU	70.47	70.44	72.31	72.81	73.12	**73.13**
ObjNum	EN	68.44	69.71	71.78	73.24	73.98	**74.93**
	ES	78.31	79.23	82.21	83.96	85.2	**85.7**
	FR	77.47	78.5	83.74	85.82	86.92	**88.1**
	DE	68.38	68.74	69.88	70.41	71.14	**71.90**
	RU	63.9	63.79	65.33	65.32	**65.54**	65.11
SOMO	EN	50.12	50.91	**51.72**	51.71	51.36	50.42
	ES	51.7	51.98	51.34	49.62	50.71	**53.07**
	FR	**50.7**	48.85	48.87	49.44	49.56	49.36
	DE	**50.57**	50.47	49.99	49.99	49.99	49.99
	RU	52.49	52.91	52.86	52.8	53.07	**53.13**

Table 4.1: DCT embeddings Performance per language compared to AVG. **EN**=English, **ES**=Spanish, **FR**=French, **DE**=German, and **RU**=Russian

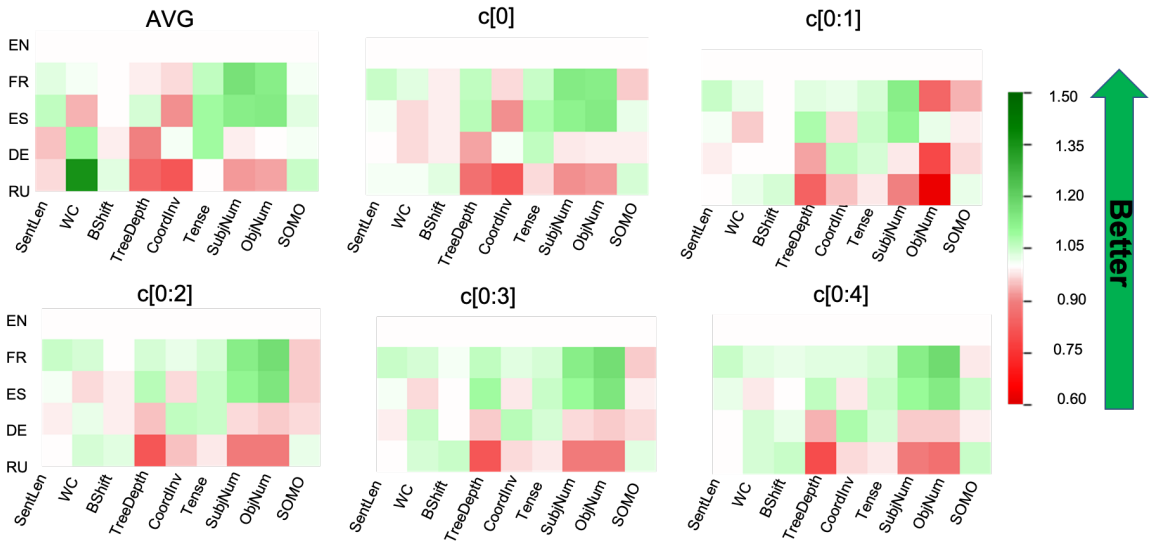


Figure 4.1: The results of the probing tasks relative to English results for each model. The white indicates a value of 1, demonstrating parity in performance. Red indicates better English performance while green indicates better X language performance.

the syntactic tasks of TreeDepth, CoordInv, Tense, SibjNum and ObjNum for RU. SOMO stands out for RU where it outperforms EN. The variation in Russian might be due to the nature of RU being much more complex and morphologically rich language with flexible word order (Toldova et al., 2015).

In term of the performance gain of the number of DCT coefficients, we observed consistence performance gain across different languages that is similar to the English performance trend as shown in Chapter 3. Specifically, for the surface level tasks, among the DCT models the $c[0]$ model significantly outperforms the AVG with an increase of $\sim$30 percentage points in all languages. The surface level tasks (SentLen and WC) show the most notable variance in the performance, in which the highest results are obtained using the $c[0]$ model. However, the performance decreases in all languages when k is increased. On the other hand, for all languages, we observed a positive effect on the model's performance with larger K in both the syntactic and semantic tasks.

Guided by the consistent results obtained using DCT embeddings across the different languages tested in this section, we explore sentence representations learned from DCT

models as a way for generating cross-lingual mappings. It could be beneficial to integrate the syntactic information preserved in the lower-order DCT coefficients to guide the cross-lingual alignment between syntactically sensitive languages. Thus, in the next section, we discuss a sentence-level cross-lingual alignment that utilizes DCT embeddings as an efficient alternative to the FastText averaging representations proposed by (Aldarmaki and Diab, 2019a).

4 Cross-lingual Mapping based on DCT Encoding

4.1 Approach

Aldarmaki and Diab (2019a) proposed sentence-level transformation approaches to learn context-aware representations for cross-lingual mapping. While the word-level cross-lingual transformations utilize an aligned dictionary of word embeddings to learn the mapping, the sentence-level transformations utilize a large dictionary of parallel sentence embeddings. Since sentences provide contexts that are useful for interpretation and disambiguating the individual word's specific meaning, sentence-level mapping yields a better cross-lingual representation compared to word-level mapping as shown in (Aldarmaki and Diab, 2019a).

Generally, any sentence embeddings model can be used to generate monolingual sentence representations for sentence-level cross-lingual transformations. Aldarmaki and Diab (2019a) shows that a simple model like sentence averaging can be used to learn the transformations between two languages. However, vectors constructed by averaging word embeddings fail to capture structural information such as word order, which may result in poor cross-lingual alignment. Therefore, guided by the results shown in Chapter 3 and Section 3.2 in this chapter, we further utilize DCT to construct sentence representations for the sentence-level cross-lingual modeling.

Similar to Aldarmaki and Diab (2019a), we rely on large parallel data to learn orthogonal mappings, in which two independent monolingual sentence representations are aligned in a shared embedding space, in which similar cross-lingual sentence embeddings end up being

close to one another.

4.1.1 Orthogonal Mapping

The objective of the orthogonal mapping is to find a transformation matrix based on a set of source- and target-sentence pairs along with their matrix representations X and Z. More specifically, given a set of sentence pairs $\{x_i^s, z_i^t\}_{i=1}^n$, in which $x_i \in R^{d1}$ is the source sentence-level embeddings of the sentence i, and $z_i \in R^{d2}$ is the sentence-level representation of the associated target language translation, orthogonal transformation matrix W is optimized to approximate Z via WX. Practically, the alignment matrix W is found by minimizing the following distances:

$$W^{s \to t} = \underset{\hat{W}}{argmin} ||\hat{W}X - Z||. \tag{4.1}$$

Typically, enforcing the orthogonality constraint on W, such that $\hat{W}\hat{W}^T = I$ where I denote the identity matrix, preserves distances between inter-lingual sentences in the source vectors space. Thus, under this constraint the singular value decomposition (SVD) can be used to find the exact solution for W (Schönemann, 1966). Formally,

$$W^{s \to t} = UV^T, \tag{4.2}$$

in which the SVD of the matrix YX^T is:

$$YX^T = U\Sigma V^T. \tag{4.3}$$

At test time, the resulting transformation matrix W is used to map any given sentence vector to the other language's space (Aldarmaki and Diab, 2019a).

4.2 Experiments

For model evaluation, we used the same cross-lingual evaluation framework introduced by Aldarmaki and Diab (2019a). Intuitively, as discussed in (Aldarmaki and Diab, 2019a), sentences tend to be clustered with their translations when their vectors exist in a well-aligned cross-lingual space. Thus, in this framework, cross-lingual mapping approaches are evaluated using sentence translation retrieval by calculating the accuracy of correct sentence retrieval (Aldarmaki and Diab, 2019a). Formally, the cosine similarity is used to find the nearest neighbor for a given source sentence from the target side of the parallel corpus.

Unlike contextualized sentence embedding, which utilizes ELMo (Aldarmaki and Diab, 2019a), we rely on static word embeddings to drive the sentence-level representations. Specifically, for each language, we utilized pre-trained FastText word embeddings that are trained on the common-crawl dataset (Grave et al., 2018). Subsequently, we follow the DCT approach as described in Chapter 3 to calculate the sentence embeddings using various DCT coefficients. As a baseline, we used the average of the word embeddings. For all sentence embedding models, we applied the smooth inverse frequency to assign weights to the word embeddings, which resulted in better representations for textual similarity tasks as shown in (Arora et al., 2017; Aldarmaki and Diab, 2019a).[3] Also, in our experiments, all sentences in a given source language were aligned to English.

4.3 Evaluation Datasets

To demonstrate the efficacy of cross-lingual mapping using the sentence-level representation generated by DCT models, similarly to Aldarmaki and Diab (2019a), we used the WMT'13 data set that includes EN, ES and DE languages (Bojar et al., 2013). We further used five language pairs from the WMT'17 translation task to evaluate the effectiveness of DCT-based embeddings. Specifically, we used a sample of 1 million parallel sentences from

[3] In our experiments, we found that weighted averaging and weighted DCT embeddings yield better results than unweighted representations

WMT'13 common-crawl data; this subset is the same one used in (Aldarmaki and Diab, 2019a). To assess efficacy of the DCT models for the cross-lingual mapping, we reported the performances of the sentence translation retrieval task within the WMT'13 test set, which includes EN, ES, and DE as test languages (Bojar et al., 2013).[4] Specifically, we first used the 1M parallel sentences for the alignment between source languages (ES and DE) to a target language (EN) independently. We evaluated the translation retrieval performance in all language directions, from source languages to English: ES-EN and DE-EN, as well as between the sources languages: ES-DE. Similarly, we conduct a series of experiments on 5 different language pairs from the WMT'17 translation task, which includes DE, Latvian (LV), Finnish (FI), Czech (CS), and Russian (RU), each of which is associated with an English translation (Zhang et al., 2018).[5] For each language pair, we sampled 1M parallel sentences from their training corpus for the cross-lingual alignment between each source language and EN. Also, we used the test set available for each language pair to evaluate the translation retrieval performances. A detailed description of these languages in the WMT'17 dataset is given below:

- **DE-EN:** To align the German-English language pair, we sampled 1M parallel sentences from the training corpus, which consists of 5.85M sentence pairs (Zhang et al., 2018). We evaluated the translation retrieval on a test set resulted from combining the 2014 and 2015 newstest datasets from WMT English-German translation task.

- **FI-EN:** For the Finnish-English language pair, we sampled 1M parallel sentences from the training corpus, which consists of 2.63M sentence pairs (Zhang et al., 2018). We concatenated the newstest of 2015 and 2016 as a test set to evaluate the translation retrieval task.

- **LV-EN:** For the Latvian-English language pair, we sampled 1M parallel sentences

[4]We used the same preprocessing and data splits as detailed in (Aldarmaki and Diab, 2019a).

[5]The preprocessed version of the WMT'17 dataset was used. For more information refer to (Zhang et al., 2018).

Language pair	AVG	$c[0]$	$c[0:1]$	$c[0:2]$	$c[0:3]$
From source languages to EN:					
ES→EN	65.67	64.87	71.26	**71.80**	70.13
DE→EN	51.80	50.30	57.23	**58.13**	56.57
From EN to source languages:					
EN→ES	69.97	69.50	73.73	**73.87**	71.73
EN→DE	67.50	66.23	**69.27**	68.70	65.83
Between source languages (Zero-Shot performance):					
DE→ES	43.80	42.20	49.50	**51.20**	51.17
ES→DE	57.67	56.46	**60.53**	59.83	57.87

Table 4.2: Sentence translation retrieval accuracy. The arrows indicate the translation direction, with EN, ES, DE refer to English, Spanish, and German, respectively.

from the training corpus, which consists of 4.46M sentence pairs (Zhang et al., 2018). To evaluate the translation retrieval task, we used the newstest2017 as the test set for the Latvian-English pair.

- **RU-EN:** To align the Russian-English language pair, we sampled 1M parallel sentences from the training corpus, which consists of 25M sentence pairs (Zhang et al., 2018). We evaluated the translation retrieval performance on a test set resulting from the concatenation of newstest2014, newstest2015 and newstest2016 as the test set.

- **CS-EN:** Similarly, to align the Czech-English language pair, we sampled 1M parallel sentences from the training corpus, which consists of 52M sentence pairs (Zhang et al., 2018). The translation retrieval was evaluated using the concatenation of newstest2014, newstest2015 and newstest2016 as the test set.

4.4 Results

The results of the sentence translation retrieval from WMT'13 data set are shown in Table 4.2. This data includes English, Spanish, and German in all language directions. Overall, DCT embedding models yielded the highest retrieval accuracy in all translation directions. For the $EN \rightarrow$ directions, the embeddings generated by the $c[0:2]$ model significantly outperform the AVG and $c[0]$ models, and is slightly better than the $c[0:1]$

model. Notably, with the $c[0:2]$ sentence-level-mapping model, a substantial gain of 5.59% is observed over the AVG model in the $\rightarrow ES$ direction, and 6.33% gains over the AVG model in the $\rightarrow DE$ direction. However, the performance dropped for both languages when the $c[0:3]$ model was used.

In the opposite translation direction from EN to source languages (ES and DE), DCT-based models outperformed the AVG model. In particular, for both languages, we observed accuracy gains of about 1.5 to 3.81 percentage points using more coefficient in DCT-based models compared to the AVG and $c[0]$ models. While the $c[0:2]$ model provided the highest accuracy for the $EN \rightarrow ES$ translation direction, the $c[0:1]$ model shows the highest accuracy for $EN \rightarrow DE$ translation direction. However, the accuracy dropped for both languages when more DCT coefficients were added.

A similar trend is observed in the zero-shot translation retrieval between the two source languages (ES and DE), in which DCT-based models outperform the AVG and $c[0]$ models. Interestingly, these observations may indicate that DCT-based models ($c[0:k]$), where $1 < k < 3$ complement the semantic meaning of the sentence provided by the AVG and $c[0]$ models, with additional structural information that results in better cross-lingual mapping in all directions.

In addition to the languages in the WMT'13 data set, we conducted another set of experiments based on five language pairs from the WMT'17 in Table 4.3. This set included DE,[6] LV, FI, CS, and RU datasets, each of which are associated with English translation. For all language pairs, the cross-lingual alignment are performed from source language to English.

For all language pairs, DCT-based embedding models outperformed AVG and $c[0]$ models in the sentence translation retrieval task. In the direction $\rightarrow EN$, while the $c[0:2]$ model achieved the highest accuracy for DE, RU, and FI languages, the $c[0:3]$ model achieved the highest accuracy for cs and lv languages. Specifically, the $c[0:2]$ model

[6] A different set from the one in the WMT'13

Language pair	AVG	$c[0]$	$c[0:1]$	$c[0:2]$	$c[0:3]$
From source languages to EN:					
DE→EN	42.41	46.90	56.17	**58.05**	57.20
RU→EN	45.22	52.75	61.91	**64.35**	63.33
CS→EN	41.87	42.50	52.89	54.99	**55.05**
FI→EN	40.46	42.00	47.57	**47.80**	46.16
LV→EN	21.26	40.13	51.42	56.37	**60.16**
From EN to target languages:					
EN→DE	55.73	59.04	**62.71**	62.10	60.51
EN→RU	38.09	44.29	54.73	59.51	**60.94**
EN→CS	39.73	40.40	50.99	54.00	**54.12**
EN→FI	39.34	42.52	51.67	**52.59**	51.74
EN→LV	15.83	33.55	47.08	53.22	**55.72**

Table 4.3: Sentence translation retrieval accuracy. The arrows indicate the translation direction, with EN, DE, RU, CS, FI, TR, and LV stands for English, German, Russian, Czech, Finnish, Turkish, and Latvian, respectively

yielded increases of about 7-30% in the direction from source languages (DE, RU, and FI) to English compared to the AVG model. Also, while the $c[0:3]$ model yielded an increase of about 13% gains over the baseline for CS, it provides the most notable increases of about 38% for LV. For the opposite direction $EN \rightarrow Source$, the DCT-based embeddings model also outperformed AVG and $c[0]$ models. In particular, for $EN \rightarrow DE$ data set, the $c[0:1]$ model outperformed AVG by about 6%. The $c[0:3]$ model shows the highest accuracy for $\rightarrow RU$, $\rightarrow CS$, and $\rightarrow LV$, with gains of about 22%, 14%, and 39%, respectively. For the $\rightarrow FI$ direction, the $c[0:2]$ model shows a gain of about 13% compared to the AVG model.

To ensure comparability to state-of-the-art results, we further utilized in-domain FastText embeddings as those used in (Aldarmaki and Diab, 2019a) as well as contextualized-based word embeddings. More specifically, Aldarmaki and Diab (2019a) leverages different methods to evaluate the cross-lingual translation retrieval. In their work, they compared the performance of word-based and sentence-level cross-lingual mapping using both context-free and contextualized word embeddings as shown in Table 4.5. Thus, we used FastText (Bojanowski et al., 2017) to generate word embeddings following the same setups detailed

Language pair	Embeddings	AVG	$c[0]$	$c[0:1]$	$c[0:2]$	$c[0:3]$
From source languages to en:						
ES→EN	FastText	82.97	82.40	**84.50**	83.97	82.90
	BERT	92.10	92.00	**93.23**	93.13	92.20
DE→EN	FastText	79.33	78.73	**81.87**	80.20	77.93
	BERT	89.76	89.66	**91.83**	91.20	90.57
From EN to source languages :						
EN→ES	FastText	82.33	82.07	**85.47**	84.60	83.17
	BERT	93.63	93.66	**94.10**	94.00	92.80
EN→DE	FastText	74.73	74.50	**79.10**	78.70	76.90
	BERT	91.30	91.43	**91.90**	91.53	90.30
Between source languages (Zero-Shot performance):						
DE→ES	FastText	73.27	72.20	**77.43**	75.96	74.60
	BERT	87.80	87.57	90.23	**90.36**	88.96
ES→DE	FastText	68.90	68.07	**73.97**	73.10	72.43
	BERT	87.70	87.70	**89.67**	89.50	88.53

Table 4.4: Sentence translation retrieval accuracy using in-domain FastText and Contextualized mBERT embeddings. **Bold** denotes the best results for each row, and **gray** highlights the best overall result for each language direction.

in (Aldarmaki and Diab, 2019a), in which the 1 Billion Word benchmark (Chelba et al., 2014) for English, and equivalent subsets of about 400 million tokens from WMT'13 (Bojar et al., 2013) news crawl data are used to train FastText using the default parameters. For the contextualized-based embeddings, we utilize a multilingual BERT (mBERT) introduced in (Devlin et al., 2019) as contextual word embeddings, in which representations from the last BERT layer are taken as the word embeddings.

Table 4.4 shows the results using in-domain FastText and mBERT word embeddings across different models. Using in-domain word embeddings yields stronger results compared to the pre-trained embeddings we used in the previous experiments. On the other hand, we observed additional improvements using mBERT as word embeddings on all models. Furthermore, increasing K has positive effect on both embeddings, in which $c[0:1]$ demonstrate performance gains compared to other models in all language directions. This trend is clearly observed in the zero-shot performance between the source languages.

Furthermore, Table 4.5 shows the average accuracy of each model as reported in (Aldar-

Model	Average Accuracy
FastText (dict)	69.04
ELMo (word)	82.23
FastText (word)	74.00
FastText AVG (sent)	76.92
ELMo AVG (sent)	84.03
FastText $c[0]$ (sent)	76.33
FastText $c[0:1]$ (sent)	**80.39**
FastText $c[0:2]$ (sent)	79.42
FastText $c[0:3]$ (sent)	77.99
mBERT AVG (sent)	90.38
mBERT $c[0]$ (sent)	90.34
mBERT $c[0:1]$ (sent)	**91.83**
mBERT $c[0:2]$ (sent)	91.62
mBERT $c[0:3]$ (sent)	90.56

Table 4.5: The average accuracy of various models across all language retrieval directions as reported in (Aldarmaki and Diab, 2019a) along with the different DCT-based models in this work, in which (word) refers to word-level mapping, (sent) refers to sentence-level mapping, and (dict) refers to the baseline (using a static dictionary for mapping). **Bold** shows the best overall result.

maki and Diab, 2019a) along with the average accuracy of the DCT-based models in this work. Overall, we observe substantial improvements when using the mBERT $c[0:1]$ based-model, in which a new state-of-the-art result is obtained with 7.8% gains over averaged ELMo in (Aldarmaki and Diab, 2019a). Moreover, sentence-level and Word-level ELMo performed better than sentence-level FastText model. Nevertheless, utilizing DCT-based model with context-free embeddings performed relatively to contextualized embeddings. Specifically, we note particular improvements with FastText when using $c[0:1]$ model, in which close to the AVG ELMo model performance is achieved.

5 Conclusion

In this chapter, we extended the application of DCT-based sentence embeddings to multi- and cross-lingual settings. Experimental results across different languages showed that (similar to English) using DCT embeddings outperform the vector averaging. These results mostly showed in tasks with high demands to sentence structure and word order information.

Hence, DCT embedding is a robust and ready-to-use alternative to the commonly-used vector averaging across different languages. Moreover, DCT embedding can be used to generate multi-lingual sentence representations without additional training.

Having an efficient ready-to-use language-independent sentence encoder can enable knowledge transfer between different languages in the cross-lingual paradigm. Therefore, we further presented a non-parameterized sentence-level based approach for cross-lingual mapping without any additional training parameters. In this context, the DCT embedding is used to generate the sentence representations, which are then used in the alignment process. In this chapter, we have shown that incorporating structural information encoded in the lower-order DCT coefficients yields significant performance gains compared to the AVG model in sentence translation retrieval.

Generally, the DCT-based models performed particularly well for all languages tested in this chapter. However, while increasing K leads to better performances for some language pairs in the cross-lingual setting, we observed the sharpest increases using K=2 in the $c[0:1]$ model for all languages in all directions. Also, increasing K resulted in embeddings dimension growth, which in turn may negatively effect the generalization of the model by over-fitting. Therefore, in our experiments, $1 \leqslant K \leqslant 2$ yielded the best trade-off.

Through this research, we have shown that our proposed model, the DCT embedding, can be easily extended to cover additional languages, as it has achieved comparable performances across different languages in multi- and cross-lingual settings. In the next chapter, we will explore several approaches for enhancing the DCT models based on a number of linguistic resources that incorporate deeper linguistic information about textual data.

Chapter 5: Linguistically Guided Discrete Cosine Transform Sentence Encoding

Despite the competitive results produced by the DCT embeddings model detailed in Chapter 3, a weakness of this method is the processing of a sentence as one complete chunk. Currently the DCT model treats sentences of varying length in the same manner, leveraging the same number of coefficients. For example, in the TREC dataset, the minimum sentence length is ~ 4 tokens while the maximum is ~ 26 tokens. Accordingly, the optimal number of DCT coefficients in which useful information about the sentence is retained varies. Adding more coefficients increases the complexity and decreases the generality of the model. Thus, in order to alleviate the variation of the required number of coefficients and maintain a less complex and more generalized model, we explore adding a chunk-level representation component to the model. In this setting, a sentence is divided into several chunks, each of which is compressed independently using DCT. The final chunk-level representation is generated using the average aggregation of all the compressed representations of the chunks.

This method is inspired by the implementation of DCT in JPEG compression, where an image is divided into $N \times N$ blocks, each of which is transformed independently (Raid et al., 2014). The quality of the image compression is dependent on the size of the blocks; where the maximum compression is obtained by dividing an image into regions with highly correlated adjacent pixels (Deng et al., 2019). This means that only a few coefficients are required to achieve optimal energy compaction. Thus, we were motivated to adapt the same framework to NLP, namely by devising and exploring a block-based DCT model, which generates sentence embeddings according to the local characteristic and importance of a chunk in a given sentence.

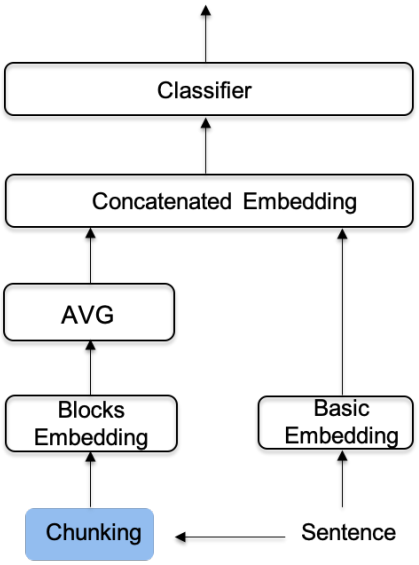
In a given image, neighboring pixels correlate more with each other, thus, dividing an image into blocks is a trivial task, but it is not as trivial to divide sentences. This is because the syntactic and semantic characteristics of a sentence complicate the process of

dividing it into blocks. Therefore, in this model different linguistic priors with different levels of granularity are used to divide a sentence into meaningful chunks. To this end, we considered n-gram-guided method vs. syntactic-guided methods to break sentences into smaller blocks. While the n-grams method is based on dividing the sentence into chunks relying on a predefined size, the syntactic-guided methods split the sentence into more meaningful segments based on the sentence linguistic information, its syntactic structure.

1 Approach

1.1 General Architecture

The presented approach primarily relies on a combination of two main components. The first component is the DCT embeddings as described in Chapter 3. This component is used to obtain the sentence vector, in which DCT is applied to the embedding dimensions of a sentence. The second component, hereafter referred to as a block-based DCT, relies on the division of a sentence into smaller blocks. More specifically, a sentence is first divided into blocks based on different chunking protocols. To model the relation between words in a given block, DCT is applied to each block independently. Subsequently, the individual vectors of the blocks are then aggregated into a fixed-length embedding using averaging. In other words, for a given sentence with n words divided into M blocks, $U = [u_1; u_2, ...; u_M]$ is the set of generated blocks, where $u_i = [w_{i1}, w_{i2}, ..., w_{ij}]; 1 < j <= n$. For each block, DCT is independently applied to the word embedding dimensions. To get a fixed-length and consistent block vector representations, we concatenated the first K DCT coefficients. Then, we stacked the sequence of the Kd-dimensional blocks' embeddings in an $M \times Kd$ matrix and applied the average operation to the columns to generate the final fixed-length representation at the block level. The representations from both components, at the sentence-level and the block-level, are then concatenated and passed to a classifier as the final representation. The overall architecture of the model is shown in Figure 5.1. The different chunking mechanisms considered in this work are detailed in the following subsections.



Chunking	
Method	**Chunks**
BASELINE	(The girl saw the man with the telescope)
NGRAMS(2)	(The girl), (girl saw), (saw the), (the man), (man with), (with the), (the telescope)
NGRAMS(3)	(The girl saw), (girl saw the), (saw the man), (man with the), (with the telescope)
CONST	(The girl), (saw the man with the telescope), (the man with the telescope), (the man), (with the telescope), (the telescope)
DEP	(The girl), (girl saw), (saw man), (the man), (man telescope), (with telescope), (the telescope)
SPO	(girl saw man), (girl saw with telescope)

Figure 5.1: The overall architecture for Block Based DCT Embeddings.

1.2 N-gram Guided Representation

N-gram chunks have been among the simplest approaches used in language modeling (LM), which is one of the oldest problems in NLP. For LM, n-grams are used to model the co-occurrence and correlations among previous n consecutive words in order to estimate the probability distribution of a given word (Katz, 1987; Kneser and Ney, 1995). Moreover, in classical machine-learning models, the bag-of-n-grams has been used to generate features to solve different NLP problems, in which a text is represented as collections of n-grams (Pang et al., 2002; Zhang and Lee, 2003).

Although most recent NLP studies have been dominated by neural models that focused on mapping words (uni-grams) independently, n-grams play an important role in obtaining semantic information for consecutive words that are not captured by uni-gram based models (Li et al., 2016, 2017a; Pagliardini et al., 2018). To capture semantic information about a

given text beyond its individual word embeddings, there have been a number of studies that explored the integration of n-gram information into various neural models. For example, the Ngram2vec model in (Zhao et al., 2017) shows that the performance of word analogy and word similarity tasks is significantly improved by incorporating n-grams into several baseline embedding models such as word2vec, GloVe, and PPMI. Gupta et al. (2019) observed a boost in the quality of word embeddings when they are trained jointly with different n-gram embeddings. A similar observation is seen using the Neural-BoN model proposed in (Li et al., 2017a), in which a text vector is represented using its n-gram vectors.

Typically, bag-of-n-grams embeddings are generated by combining individual word vectors using different heuristics such as summation, averaging, or multiplication. For example, in the DisC model (Arora et al., 2018), which leveraged bag-of-n-grams information as well as word embedding to construct a text vector, the element-wise multiplication of the words vectors is used to obtain the representation of all possible n-grams extracted from a given text. To construct the final text embeddings in the DisC model, the average of all n-grams and the average of all uni-grams are concatenated together. The heuristics used to encode the compositional information of n consecutive words, however, fail to capture the order of the words that make up the n-gram embeddings. Thus, utilizing DCT operation to compress the n-gram embeddings into a fixed-length encoding would be a better alternative to capture word order information. The DCT operation when applied to textual vectors captures both the semantic information of the n-gram as well as the word order information as shown in (Almarwani et al., 2019). The DisC model is the most similar approach to our proposed n-gram representation, but there are a number of differences between this model and our work. The first difference is the operation used to generate the n-gram embeddings. In the DisC model the element-wise multiplication while in our work, a DCT operation is employed. Also, while the final representation in the DisC model comes from sentence level, bi-grams, **and** tri-grams, in our proposed model, the sentence level encoding along with either the bi-grams **or** tri-grams form the final representation.

N-grams model (NGRAM) In our framework, to exploit the n-grams method along with DCT embedding, a sentence is divided into overlapping blocks using a predefined window size to form chunks of n consecutive words. Each chunk of n-gram is treated as an independent sequence and mapped to a unique encoding using DCT. More formally, given a sentence of M words, where $w_1, ..., w_M \in R^{M \times d}$, an n-gram is defined as follows: $u_i = [w_i, ..., w_{i+n-1}] \in R^{n \times d}; 0 \leq i \leq n - M$. For each u_i, we calculated DCT as described in Section 1.1. For example, as shown in Figure 5.1, the overlapping bi-gram blocks of the sentence "The girl saw the man with the telescope" are ["The girl", "girl saw", "saw the", "the man", "man with", "with the", "the telescope"]. Given the extracted bi-grams along with the original sentence, the final representation is formed using the sentence-level DCT embeddings concatenated with the average of all bi-gram DCT embeddings. By the same token, we extract and represent tri-grams. The n-grams-based block models rely on word adjacency in the raw text, in which a uniform window size is used to extract the overlapping multi-word representations.

1.3 Syntactically Guided Representation

The n-grams-based model depends on the sequential information in the raw text, which does not capture the underlying linguistic phenomena of a given text. Therefore, to address the shortcoming of the n-gram-based model, we explore the impact of using linguistically informed blocks representing syntactic and semantic information.

More specifically, while traditional n-grams rely on a predefined window size when forming a block of neighboring words, syntactic parses can yield more meaningful blocks as well as capture long distance relations that would be missed by ngrams.

Syntactic parsing is widely recognized for learning compositional representation , which boosts model performance for end tasks such as text classification, semantic parsing system, and classification of temporal relations between events, etc. (Nastase et al., 2006; Monroe and Wang; Choubey and Huang, 2017; Levy and Goldberg, 2014; Angeli et al., 2015; Parikh

et al.; Bowman et al., 2016; Toutanova et al., 2016). Several alternatives have been proposed to integrate syntactic parsing knowledge into different NLP models.

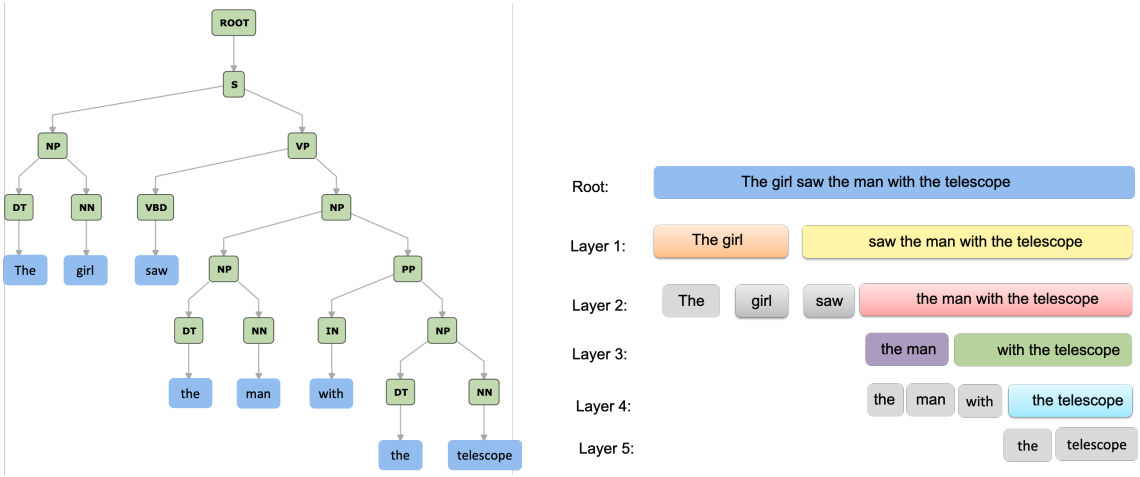
For example, in the current Deep learning frameworks, Recursive Neural Networks approaches incorporate tree-based neural models to encode complex compositional meanings of textual data without any hand-crafted features. These approaches yielded promising results in a number of tasks such as sentiment classification, semantic relatedness, named entity recognition, and rumor detection (Socher et al., 2010, 2013a,b, 2011; Tai et al., 2015; Li et al., 2017b; Bowman et al., 2016; Ma et al., 2018).

Moreover, incorporating syntactically-oriented tasks within a multi-task learning strategy is another way to learn the structure of language (Hashimoto et al., 2017; Kiperwasser and Ballesteros, 2018). Hashimoto et al. (2017) shows that including a dependency parsing task, which identifies syntactic relations between words in a given sentence, in a hierarchical multi-task architecture improves the results of relatedness and entailment tasks. Similarly, Zhou et al. (2019) presented a Linguistic Informed Multi-Task BERT (LIMIT-BERT) that incorporates linguistic knowledge from multiple tasks including constituent and dependency parsing within a multi-task learning protocol.

Aside from these approaches, integrating syntactic information in the form of syntactic n-grams (a special type of n-grams) as an alternative to the traditional n-grams model has been explored (Posadas-Durán et al., 2017; Shin et al., 2016). The traditional n-gram method detailed previously benefits from the surface level information of a given sentence and ignores all syntactic information. However, using a syntactic parser to construct n-gram chunks, which relies on the path between words in a given parse tree, enriches the n-gram with linguistic knowledge.

In our syntactic representation, we build on this line of work. More specifically, we use both a constituency and a dependency parser as two sources of information to incorporate linguistic knowledge into our block-based DCT embeddings. Specifically, in this paper, we extend the n-grams approach using three linguistic perspectives: a. The phrase structure as in

base syntactic phrases providing noun and verb phrases (CONST); b. Bigram Long-distance dependencies provided by a dependency parser (DEP); And, c. Trigrams derived from a dependency parser reflecting the functional components of a sentence as in triplets of Subject, Predicate, and Object (SPO).

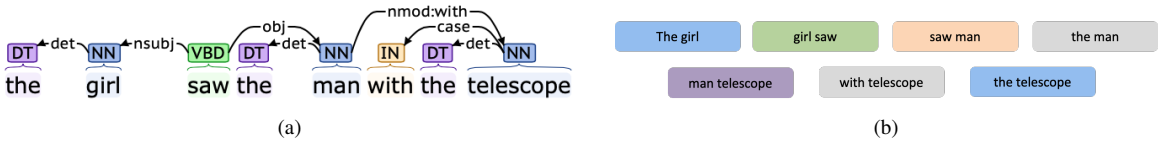


(a) (b)

Figure 5.2: An example of constituent based division that is used in the constit-grams model, where (a) is the constituent parse tree of the given sentence generated from Stanford CoreNLP tool, and (b) is the corresponding chunks. In this example, the root, which comprised all words in the sentence will not be part of the constit-grams.

CONST comprising base syntactic chunks such as NPs, VPs, AdjPs, etc. as derived from a constituency parser as shown in Figure 5.2a. In this case, the chunks are composed of neighboring words derived from a phrase-structure grammar as per a constituency parser. This results in n-grams of overlapping chunks with varying sizes, as shown in Figure 5.2b. Hence the main difference with NGRAM is that the constituent phrases are variable in length while crucially still being composed of adjacent words similar to the ngram blocks.

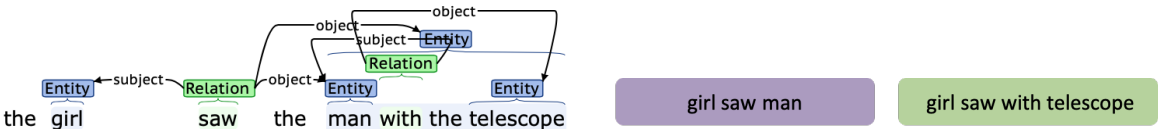
DEP Furthermore, to address the second limitation of word adjacency imposed by NGRAM and CONST, we explore long distance dependency representations as derived from a dependency parser as shown in Figure 5.3a. In this case the chunks comprise words that are related by a syntactic dependency relationship, hence no adjacency constraint on the words. Accord-



(a) (b)

Figure 5.3: An example of dependency based division that is used in the depend-grams model, where (a) is the dependency parse tree of the given sentence generated from Stanford CoreNLP tool, and (b) is the corresponding chucks division.

ingly, we extract dependency relations between pairs of words within a sentence reflecting all the pairwise relations among the words. we extract all possible governor-dependent pairs that are related through specifiable dependency links such as subject, direct object, indirect object, etc. The dependency grams extracted are all bigrams as shown in Figure 5.3b.



(a) (b)

Figure 5.4: An example of dependency based division that is used in the SPO-grams model, where (a) is the relation tree of the given sentence generated from Open IE in Stanford CoreNLP tool, and (b) is the corresponding chucks division.

SPO We extend DEP to a linguistically guided trigram framework by extracting triplets of Subject/Predicate/Objects from the dependency parses, with the intent of capturing the underlying shallow semantics of the sentence. The subject-predicate-object chunk refer to three main linguistic information: the subject (the who or what of the sentence), the predict (the verb in the sentence), and the object (the noun or concept that is part of the action). We use an off-the-shelf tool [1] to obtain the set of subject-predicate-object triples (SPO-grams) from the dependency relation of a given sentence, which results in trigram as shown in Figure 5.4b.

[1]Relation extraction plays an important role in NLP domain, hence, a number of algorithms and tools has been developed to extract these relations from the sentence

2 Evaluation

The SentEval toolkit (Conneau and Kiela, 2018) is used for model evaluation. SentEval includes standard benchmarks, which covers a breadth of Probing as well as Downstream tasks. The Probing benchmark has been designed to measure the capability of sentence embeddings to probe specific linguistic phenomena at the sentence-level, which includes surface, syntactic, and semantic levels of linguistic information (Conneau et al., 2018a). The downstream benchmark is utilized to measure model generalization in a wide area of different domains with mixed linguistic phenomena. The downstream benchmark spans various standard NLP tasks including text classification, textual inference, semantic similarity, and paraphrase identification. Detailed descriptions of the framework and the sets of tasks (the Probing and downstream tasks) contained within the benchmarks are given in Chapter 3 and in the original paper (Conneau and Kiela, 2018).

For each classification task, sentence embeddings are used as features for a logistic regression classifier. To train the classifier, this experiment utilizes the default settings of SentEval. All experiments are trained using 4 epochs with a batch size of 64 and the Adam optimizer (Kingma and Ba, 2014) with an optimization learning rate of 0.001. To generate the syntactic level representation, we used Stanford CoreNLP tools, which provides full syntactic analyses, including both constituent and dependency representations (Manning et al., 2014). Specifically, for each data set from the evaluation benchmarks, we pre-process the sentences and extract the associated constituent and dependency parse trees. Furthermore, for the SPO-grams model, we used the Stanford Open Information Extraction (Open IE) (Angeli et al., 2015) to extract subject-predicate-object (SPO) triples.

2.1 Results

The sentence DCT embeddings are concatenated with the average of the block DCT embeddings for all the chunks in the sentence. In this study, for the block-level component,

we only compared the representations from $c[0]$ and $c[0:1]$. As shown in Chapter 3, the **scaled** average of the word embeddings is rendered using $c[0]$ preserving the local semantic information and word order is captured using $c[0:1]$ and $c[0:2]$ preserving the local semantic and structural information. However, we restrict our experimentation to $c[0]$ and $c[0:1]$ since they yielded the best results in their study. We assess the efficacy of the following chunking methods on the probing and downstream tasks:

- **BASELINE:** This is the vanilla-based DCT model that is trained without aggregating any block-based representation in the final model;
- **NGRAMS(2) and NGRAMS(3):** In this model, we concatenate the sentence representation from the DCT-based model using the average of all bi-grams (for NGRAMS(2)) or tri-gram (for NGRAMS(3)) representations generated using DCT over each n-gram in a sentence;
- **CONST:** In this model, we use a syntactic parser to extract all possible phrases from the sentence and concatenate the sentence representation with the representation of the phrases using the same processing as the NGRAMS;
- **DEP:** Similar to CONST and NGRAMS but the blocks are derived from a dependency parser reflecting bigrams of governor-dependent relations;
- **SPO:** Similar to DEP but here we have the triples constituting the blocks as obtained by the Open IE tool.

Figure 5.5 shows the best results obtained using the various Block-based embeddings where the blocks DCT coefficient is set to $c[0]$ or $c[0:1]$ on the Probing and Downstream tasks relative to the BASELINE (basic sentence DCT encoding) as a heat-map.The complete numerical results for probing and downstream tasks are presented in Table 5.1 and Table 5.2, respectively. We observe general performance gains with the integration of additional information rendered from the Block-based component compared to the BASELINE models

	Surface		Syntactic			Semantic				
Model	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
BASELINE										
$Sent.c[0]$	**83.47**	**91.11**	33.42	64.50	50.42	**86.80**	79.89	79.61	49.59	51.87
$Sent.c[0:1]$	82.15	89.16	34.44	74.40	49.82	86.66	**84.47**	80.88	49.44	66.19
$Sent.c[0:2]$	81.55	86.77	**35.61**	**77.08**	**52.07**	86.46	84.31	**81.50**	**50.21**	**66.80**
NGRAMS(2)										
$Sent.c[0] \oplus NGRAMS(2)[0]$	95.10	90.87	35.06	69.40	49.49	87.06	80.07	80.01	49.52	55.56
$Sent.c[0] \oplus NGRAMS(2)[0:1]$	**96.29**	**90.91**	35.35	73.80	49.74	**87.07**	81.04	80.12	49.49	57.08
$Sent.c[0:1] \oplus NGRAMS(2)[0]$	95.65	89.28	36.03	75.49	49.59	86.31	84.68	81.40	50.25	65.99
$Sent.c[0:1] \oplus NGRAMS(2)[0:1]$	95.13	89.30	36.29	76.28	50.01	86.49	84.76	81.59	50.14	66.35
$Sent.c[0:2] \oplus NGRAMS(2)[0]$	94.26	87.06	36.82	77.60	52.00	86.91	84.75	81.64	**50.88**	66.83
$Sent.c[0:2] \oplus NGRAMS(2)[0:1]$	93.95	87.05	**37.06**	**77.91**	**52.22**	86.94	**84.89**	**81.80**	50.24	**66.96**
NGRAMS(3)										
$Sent.c[0] \oplus NGRAMS(3)[0]$	95.65	90.71	34.59	72.19	52.53	86.59	80.21	80.74	49.58	56.7
$Sent.c[0] \oplus NGRAMS(3)[0:1]$	**96.17**	**90.78**	34.99	76.46	53.95	86.84	83.83	81.15	49.91	61.47
$Sent.c[0:1] \oplus NGRAMS(3)[0]$	96.05	89.07	35.51	76.00	52.47	86.61	84.59	81.48	49.96	66.56
$Sent.c[0:1] \oplus NGRAMS(3)[0:1]$	95.86	89.00	35.91	77.45	53.58	86.47	84.81	82.96	49.93	66.60
$Sent.c[0:2] \oplus NGRAMS(3)[0]$	95.32	86.93	36.64	77.74	52.25	**86.95**	84.54	81.78	50.21	66.66
$Sent.c[0:2] \oplus NGRAMS(3)[0:1]$	95.05	86.89	**36.98**	**78.21**	**54.05**	86.93	**85.19**	**83.17**	**50.49**	**66.93**
CONST										
$Sent.c[0] \oplus CONST[0]$	83.28	90.38	36.63	74.47	56.77	87.38	82.98	80.02	49.24	53.09
$Sent.c[0] \oplus CONST[0:1]$	83.13	**90.39**	38.02	75.64	61.53	87.79	83.61	80.48	48.98	52.33
$Sent.c[0:1] \oplus CONST[0]$	82.96	89.01	37.74	77.90	56.44	87.48	84.97	81.26	49.63	66.23
$Sent.c[0:1] \oplus CONST[0:1]$	83.79	88.98	38.80	78.51	61.55	87.79	85.93	81.65	49.80	66.48
$Sent.c[0:2] \oplus CONST[0]$	83.37	86.54	37.76	79.77	57.18	87.34	85.26	81.80	**50.19**	67.01
$Sent.c[0:2] \oplus CONST[0:1]$	**84.01**	86.90	**39.38**	**80.19**	**61.65**	**87.87**	**86.17**	**82.21**	50.05	**67.12**
DEP										
$Sent.c[0] \oplus DEP[0]$	**93.34**	**90.80**	34.85	68.17	56.61	87.56	80.08	80.32	49.37	54.72
$Sent.c[0] \oplus DEP[0:1]$	92.34	90.97	35.92	69.43	57.99	88.06	80.56	80.44	50.07	56.95
$Sent.c[0:1] \oplus DEP[0]$	92.57	89.10	36.12	75.03	56.29	86.85	84.69	81.48	49.47	66.43
$Sent.c[0:1] \oplus DEP[0:1]$	92.83	89.09	36.44	75.93	57.57	87.81	84.81	82.06	49.63	66.41
$Sent.c[0:2] \oplus DEP[0]$	90.57	86.59	36.84	77.69	56.66	87.39	**85.09**	82.17	**50.41**	**66.96**
$Sent.c[0:2] \oplus DEP[0:1]$	91.16	87.08	**38.02**	**78.23**	**58.36**	**88.09**	84.95	**82.35**	49.95	66.77
SPO										
$Sent.c[0] \oplus SPO[0]$	**83.24**	**87.49**	34.70	67.85	52.58	88.31	82.69	81.39	49.44	52.92
$Sent.c[0] \oplus SPO[0:1]$	81.75	85.48	34.36	70.39	53.56	88.22	86.41	82.01	**50.65**	57.53
$Sent.c[0:1] \oplus SPO[0]$	81.35	85.55	34.85	76.05	53.26	88.24	86.54	82.54	49.68	66.08
$Sent.c[0:1] \oplus SPO[0:1]$	81.09	83.83	35.37	76.01	53.78	88.19	87.35	**84.10**	49.91	65.91
$Sent.c[0:2] \oplus SPO[0]$	81.53	83.06	**36.63**	78.76	53.52	**88.40**	86.82	82.27	49.40	66.17
$Sent.c[0:2] \oplus SPO[0:1]$	81.63	81.7	36.35	**78.97**	**54.34**	88.35	**87.62**	84.03	50.06	**66.63**

Table 5.1: Probing tasks performance of Block-based DCT models, which includes NGRAMS(2), NGRAMS(3), CONST, DEP, and SPO, with various K using logistic regression, vs. DCT embeddings with various K using logistic regression. **Bold** indicates the best result in each models types block, and **gray-highlight** indicates the best result among all models.

Model	Sentiment Analysis					SUBJ	Relatedness&Paraphrase			Inference	TREC
	MR	SST2	SST5	CR	MPQA		STSB	MRPC	SICK-R	SICK-E	
BASELINE											
$Sent.c[0]$	**78.15**	**82.98**	**44.84**	79.6	**87.98**	92.37	**71.11**	72.41	82.61	79.54	84.80
$Sent.c[0:1]$	77.86	81.99	44.21	**80.29**	87.88	92.36	70.82	**73.80**	82.73	**79.68**	88.20
$Sent.c[0:2]$	78.02	82.48	43.03	79.81	87.45	**92.59**	70.36	73.68	**82.95**	79.6	**90.00**
NGRAMS(2)											
$Sent.c[0] \oplus NGRAMS(2)[0]$	78.12	83.2	45.97	**80.76**	88.04	92.13	71.14	72.87	82.88	79.36	86.00
$Sent.c[0] \oplus NGRAMS(2)[0:1]$	**78.31**	**83.31**	**46.52**	80.6	**88.10**	92.29	**71.27**	74.09	83.06	79.78	86.20
$Sent.c[0:1] \oplus NGRAMS(2)[0]$	78.28	82.92	46.15	80.52	88.08	92.37	70.79	**74.67**	83	79.7	89.60
$Sent.c[0:1] \oplus NGRAMS(2)[0:1]$	77.94	83.03	46.11	80.52	88.02	92.36	70.79	74.09	83.02	79.58	90.40
$Sent.c[0:2] \oplus NGRAMS(2)[0]$	78.25	82.54	45.61	80.07	88.03	**92.53**	70.43	72.99	83.09	79.58	90.40
$Sent.c[0:2] \oplus NGRAMS(2)[0:1]$	78.12	81.82	45.38	80.04	87.95	92.38	70.49	73.22	**83.11**	**80.37**	**90.80**
NGRAMS(3)											
$Sent.c[0] \oplus NGRAMS(3)[0]$	**78.25**	82.48	**46.11**	80.23	88.02	92.01	71.25	73.39	82.88	80.09	86.00
$Sent.c[0] \oplus NGRAMS(3)[0:1]$	78.01	82.48	45.11	80.31	87.9	92.35	**71.34**	72.58	**83.17**	78.53	87.80
$Sent.c[0:1] \oplus NGRAMS(3)[0]$	78.02	82.37	45.97	**80.92**	87.89	92.09	70.69	**74.32**	82.99	**80.21**	89.40
$Sent.c[0:1] \oplus NGRAMS(3)[0:1]$	77.99	**82.65**	45.66	80.47	**88.12**	**92.36**	70.74	73.39	83.03	79.89	89.20
$Sent.c[0:2] \oplus NGRAMS(3)[0]$	78.19	81.88	45.52	80.28	87.73	92.24	70.35	72.81	83.05	79.74	**90.60**
$Sent.c[0:2] \oplus NGRAMS(3)[0:1]$	78.14	82.04	45.02	80.15	87.97	92.18	70.33	72.35	83.12	80.03	**90.60**
CONST											
$Sent.c[0] \oplus CONST[0]$	**78.27**	82.92	45.16	80.04	**88.25**	92.34	71.11	73.04	83.06	79.18	84.00
$Sent.c[0] \oplus CONST[0:1]$	78.24	**83.09**	45.11	**80.52**	88.21	92.02	**71.27**	72.64	**83.61**	79.42	84.80
$Sent.c[0:1] \oplus CONST[0]$	77.88	82.59	45.43	80.15	87.75	92.45	70.73	74.03	82.95	79.34	88.60
$Sent.c[0:1] \oplus CONST[0:1]$	77.8	83.03	45.57	80.2	87.89	92.5	70.82	73.68	83.23	79.34	89.40
$Sent.c[0:2] \oplus CONST[0]$	78.18	82.43	**45.70**	79.75	87.69	**92.67**	70.7	72.58	82.99	79.58	**90.40**
$Sent.c[0:2] \oplus CONST[0:1]$	78.09	82.7	45.38	79.43	87.66	92.63	70.81	**74.09**	83.23	**79.68**	89.80
DEP											
$Sent.c[0] \oplus DEP[0]$	78.3	**84.02**	44.43	**81.14**	87.98	92.56	**71.21**	73.86	82.99	79.66	87.20
$Sent.c[0] \oplus DEP[0:1]$	**78.40**	83.42	44.62	80.69	**88.43**	92.71	70.78	72.7	83.31	80.19	87.20
$Sent.c[0:1] \oplus DEP[0]$	77.62	81.88	**46.29**	80.32	88.03	92.3	70.55	74.32	83.13	79.95	90.20
$Sent.c[0:1] \oplus DEP[0:1]$	77.95	82.7	46.2	80.38	87.9	92.51	70.25	**74.43**	83.46	80.6	90.40
$Sent.c[0:2] \oplus DEP[0]$	77.88	82.26	44.21	80.08	87.87	92.7	70.2	73.8	83.33	79.85	**91.60**
$Sent.c[0:2] \oplus DEP[0:1]$	77.99	82.26	44.43	80.56	87.85	**92.90**	69.88	73.28	**83.59**	**80.92**	91.40
SPO											
$Sent.c[0] \oplus SPO[0]$	**78.39**	**82.98**	44.93	**80.70**	**88.01**	92.4	**70.64**	73.86	82.9	79.54	84.6
$Sent.c[0] \oplus SPO[0:1]$	77.92	82.26	**45.25**	79.82	87.84	92.03	69.54	**75.19**	82.99	**80.37**	86.4
$Sent.c[0:1] \oplus SPO[0]$	77.85	82.26	44.89	80.03	87.81	92.59	70.47	74.14	83.28	79.89	88.6
$Sent.c[0:1] \oplus SPO[0:1]$	77.91	81.88	43.21	79.55	87.95	92.41	69.59	74.55	83.18	80.33	89.8
$Sent.c[0:2] \oplus SPO[0]$	78.02	82.7	44.8	80.11	87.88	**92.60**	69.81	74.26	**83.60**	79.76	**90.00**
$Sent.c[0:2] \oplus SPO[0:1]$	78.32	81.93	43.48	79.42	87.81	92.36	69.07	72.64	83.5	80.01	**90.00**

Table 5.2: Downstream tasks performance of sentence-level DCT embeddings vs. block-based DCT models, which includes NGRAMS(2), NGRAMS(3), CONST, DEP, and SPO, with various K using logistic regression. **Bold** indicates the best result in each models types block, and **gray-highlight** indicates the best result among all models

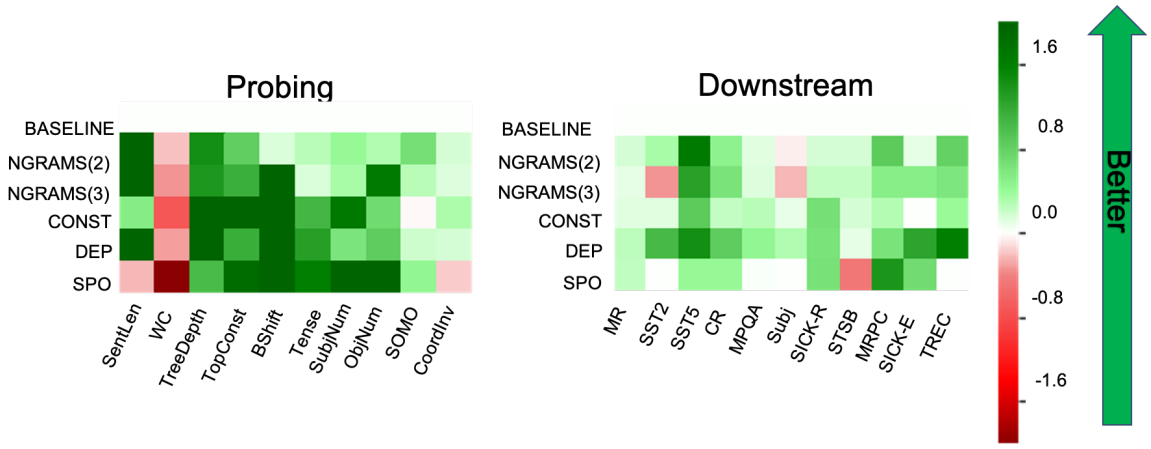


Figure 5.5: Impact of Block-level models on the Probing and Downstream tasks relative to the BASELINE model reflecting the best results obtained using either $c[0]$ or $c[0:1]$. White indicates a value of 1, demonstrating parity in performance, while green and red indicate better and worse performance, respectively.

with the exception of the WC task as illustrated by the prevalence of the red(ish) blocks across models.

Probing Tasks For surface level tasks, we observe performance gains by adding block based DCT information for SentLen except for the SPO protocol, with NGRAMS, DEP being the superior protocols. However, for WC, we note a degradation in performance. This may be because WC is a surface level task in which the BASELINE sentence embeddings of $c[0]$ are sufficient to capture word content information.

For syntactic level tasks (TreeDepth, TopConst, and BShift), all the protocols improve performance significantly over BASELINE with superior performance for CONST for all three tasks. BShift yielded best performance for all NGRAMS(3), CONST, DEP, SPO. For the TreeDepth task, we note that CONST and DEP yield the best performance followed by NGRAMS then SPO. TopConst garners the best performance using CONST and DEP. For morphosyntactic tasks (Tense, SubjNum, and ObjNum), we see the same trends as in the Syntactic ones, all significantly outperforming BASELINE, with SPO yielding the best relative performance. We surmise that the subject, predict, and object features conveyed and encoded in SPO provide useful clues for these three morpho-syntactic tasks as SPO models

morphological agreement. For more semantic Probing tasks (SOMO and CoordInv), we note a less impact of the block based methods on the task. NGRAMS(2) and SPO yield the best results for SOMO while CONST leads to a degradation in performance relative to BASELINE. For CoordInv, CONST yields the best results among all the Block based protocols, with NGRAMS and DEP showing marginal improvement over BASELINE, while SPO yields worse results relative to BASELINE.

Downstream Tasks As shown in Figure 5.5, among the proposed block types, the DEP model yields the best/top accuracy on 8 downstream tasks, namely MR, SST2, CR, MPQA, Subj, SICK-R, SICK-E, and TREC. For SST5 NGRAMS(2) yields the best performance, and for STSB, NGRAMS(3) yields the best performance and for MRPC, SPO yields the highest performance. For the sentiment tasks (MR, SST2, SST5, CR, MPQA), all the protocols improve performance over BASELINE with superior performance for DEP for 3 out of the 4 tasks. However, for the SST5, we noticed a different pattern, where using the NGRAMS(2) model leads to a slight improvement over DEP model. Similarly to the performance trends of the sentiment tasks, we observed that the SUBJ and SICK-E tasks benefit from the features captured by the DEP component, which yields improvement over the BASELINE. We noted a similar performance gain on the TREC task, in which the DEP model yields the highest improvement over the BASELINE. A different trend is observed among the relatedness and paraphrasing tasks that including SICK-R, STSB, and MRPC. In particular, for the SICK-R task the best result is obtained using CONST model compared to the BASELINE. For the STSB task, we observed that the performance increased only slightly using the NGRAMS(3) model. For the MRPC task, the SPO model yields better results compared to the BASELINE.

In general, the block-based models yield better results than BASELINE across tasks for both evaluation benchmarks. This observation indicates that the combination of sentence-level information and block-level representations provides complementary information to

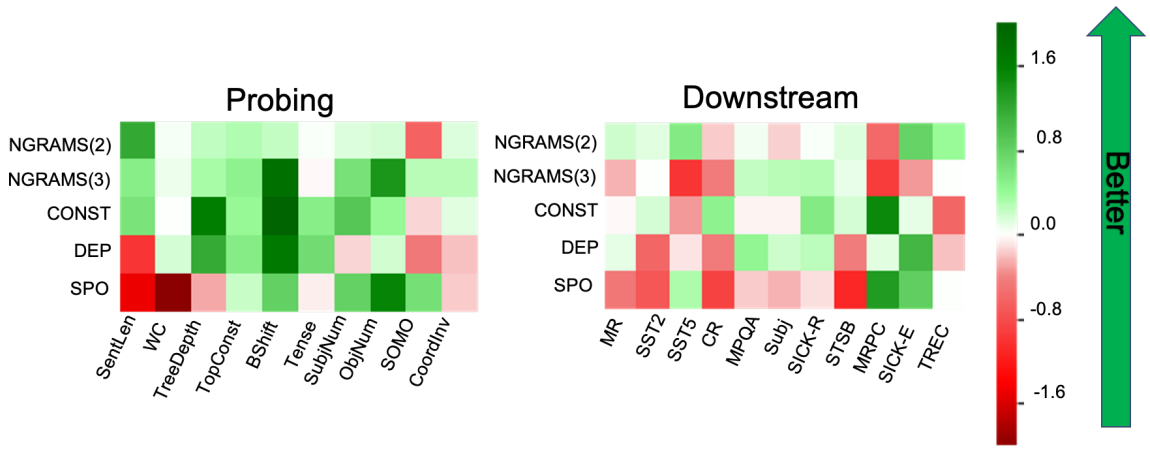


Figure 5.6: Impact of Block-level DCT coefficients $c[0:1]$ relative to $c[0]$ on the Probing and Downstream tasks.

the end-task classifier.

DCT coefficients at the Block-level The effect of the number of DCT coefficients at the block-level on the model performance within Probing and Downstream tasks is shown in Figure 5.6 as a heat-map. The complete numerical results for both probing and downstream tasks are shown in Table 5.1 and Table 5.2, respectively. Generally, $c[0:1]$ encoding of DEP, CONST, and NGRAMS(3) all outperform $c[0]$ for syntactic and morphosyntactic Probing tasks. $c[0]$ outperforms $c[0:1]$ for SentLen and CoordInv for DEP and SPO. The patterns are flipped for SOMO and WC. In SOMO, $c[0]$ outperforms $c[0:1]$ using DEP while the opposite is true for SPO model. For WC, $c[0:1]$ outperforms $c[0]$ for DEP while $c[0]$ yields better results for SPO.

For Downstream Tasks, the difference between $c[0]$ and $c[0:1]$ is more mixed. For sentiment tasks, we observe that $c[0:1]$ encoding is slightly better than $c[0]$ encoding in some models. This pattern are shown in DEP and NGRAMS(2) models for the MR task, NGRAMS(2) and CONST models for for the SST2 task, NGRAMS(2) and SPO models for the SST5 task, CONST model for the CR task, and in both NGRAMS(3) and DEP models for the MPQA task. However, the patterns are flipped using other models. Similarly, while we observed slight improvement using $c[0:1]$ of NGRAMS(3) and DEP for the SUBJ, $c[0]$

encoding outperforms c[0:1] encoding on other block-based models. For the SICK-R and SSTB relatedness tasks, $c[0:1]$ yields some improvement over $c[0]$ of all models except when using SPO model for the SICK-R and both DEP and SPO models for the SSTB task. On the other hand, for the MRPC relatedness task, while the $c[0:1]$ encoding of CONST, DEP, and SPO all outperform the $c[0]$ encoding, the $c[0]$ encoding yields better results for NGRAMS(2) and NGRAMS(3). On the entailment task (SICK-E), $c[0:1]$ encoding of NGRAMS(2), CONST, DEP, and SPO all outperform $c[0]$ while the opposite is true using NGRAMS(3). Finally, for TREC task, while $c[0]$ and $c[0:1]$ encoding yield comparable performances for NGRAMS(3) and SPO models, $c[0:1]$ encoding outperforms $c[0]$ of NGRAMS(2) and the opposite is true for CONST and DEP models.

Overall, augmenting the final representation with additional information using different level of granularity introduced by the different chunking methods has shown performance gains over the baseline model. Furthermore, local structural information rendered at the block level using $c[0:1]$ has shown to be useful for capturing the syntactic information of the chunks in a sentence.

3 Detailed Analysis

Model	**Overall Acc.**	**Bin-1**	**Bin-2**	**Bin-3**	**Weighted-Acc.**
$Sent.c[0]$	84.80	96.29	80.65	80.00	85.64
$Sent.c[0] \oplus DEP.c[0]$	87.20	98.51	83.60	80.00	87.37
$Sent.c[0:1]$	88.20	**99.25**	84.59	83.33	89.05
$Sent.c[0:1] \oplus DEP.c[0]$	90.20	**99.25**	87.86	81.66	89.59
$Sent.c[0:2]$	90.00	98.51	86.88	83.33	89.57
$Sent.c[0:2] \oplus DEP.c[0]$	**91.60**	**99.25**	**89.18**	**86.66**	**91.69**

Table 5.3: Comparing Base Sentence encoders, $Sent.c[0]$, $Sent.c[0:1]$, $Sent.c[0:2]$, each concatenated with or without $DEP.c[0]$ on the TREC Task. Overall accuracy reflects the overall results on the entire test set, Bin-* indicates the results on the specific cluster of sentences of length 1-5, 6-10, 11+ tokens, respectively.

3.1 Sensitivity to Sequence Length

We compare BASELINE and DEP based on their performance sensitivity to sentence length in the TREC, a question-classification task, since DEP at coefficient $c[0]$ rendered the best performance among all the chunk based strategies. Moreover, in (Almarwani et al., 2019), they note that higher coefficients render better results for the TREC task, accordingly we measure the impact of adding DEP at $c[0]$ to Sentence DCT encoding with coefficients $c[0:1]$ ($Sent.c[0:1]$) and $c[0:2]$ ($Sent.c[0:2]$). For this comparison, the sentences in the TREC collection are clustered based on their length into three subsets as follows: Bin-1[1-5] tokens, Bin-2[6-10] tokens, and Bin-3[11-] tokens. Table 5.3 shows performance based on sentence length. Adding more coefficients to the sentence encoding improved the results: $Sent.c[0:2]$ outperforms $Sent.c[0:1]$ which in turn outperforms $Sent.c[0]$. But also adding block DCT information to each of the base sentence encoding improves each of them, respectively, going from the BASELINE $Sent.c[0]$ Of 84.80% accuracy to 91.60% for $Sent.c[0:2] \oplus DEP[0]$. We also note that the performance on shorter sentences Bin-1 is higher than on Bin-2 followed by Bin-3 across all conditions. However, we observe the biggest gains for concatenating $DEP[0]$ for Bin-2 and Bin-3, sentences longer than 6 tokens, by relatively large absolute margins of 2.3% and 3.33%, respectively.

3.2 Analysis of Pairwise Classification Tasks

The experimental results demonstrate that in general the Block-based channel provides some useful information for both Probing and Downstream tasks. Having explicit linguistic knowledge represented is useful. For example as per Figure 5.1, the linguistically informed block strategies of CONST, DEP, SPO outperform NGRAMS for SICK-R and are comparable or outperforming NGRAMS for SICK-E. However, the performance improvements are not significant on the Downstream tasks. This may be because different tasks require different types of linguistic knowledge. For instance, negation is a critical piece of information for pairwise classification tasks such as textual entailment and relatedness. Given two text

List of Modifiers		
neg: negation modifier	**advmod:** adverb modifier	**amod:** adjectival modifier
num: numeric modifier	**prep:** prepositional modifier	**quantmod:** quantifier phrase modifier

Table 5.4: List of modifiers from the Stanford typed dependencies used in the enhanced-grams model

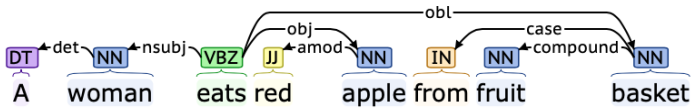
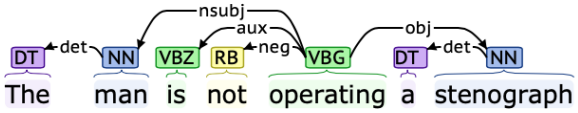
Sentence Dependency Tree	SPO-Enh
	woman eats **red** apple
	man **not** operating stenograph

Table 5.5: Examples for blocks produced by SPO-Enh model from the Stanford CoreNLP dependency parse tree.

fragments, the entailment task is defined as predicting if a pair of text entail or contradict each other. For contradiction identification, checking for explicit negation is a simple but strong heuristic to use as illustrated in Lai and Hockenmaier (2014). This information is captured by NGRAMS, CONST and DEP but not captured by the SPO representation. More specifically, the rules used to extract a subject-predicate-object triple in the SPO model ignore negation words. As a result, the sentence pair: "A women is cutting an onion", and "A woman is not cutting an onion", which negate each other, and the sentences pair: "A women is cutting an onion on the table", and "A woman is cutting an onion", which entail each other, will have the same subject-predicate-object triple: "women-cutting-onion". Similarly, modifiers such as adjectives, play an important role in the entailment classification. To verify the utility of these relations on rendering better representations, we augment the SPO representations with additional pertinent information extracted from the dependency parse. We define a set of rules to extend the chunks by including: negation, which indicates the existence of words like “not” and “no”, adjectival modifiers, which modify the meaning of the nominal, adverbial modifiers, prepositional modifiers, numeric modifiers, and quantifier

Embedding	SICK-E	SICK-R
$Sent.c[0] \oplus DEP.c[0]$	79.66	82.99
$Sent.c[0] \oplus SPO.c[0]$	79.54	82.90
$Sent.c[0] \oplus SPO - Enh.c[0]$	80.62	84.01
$Sent.c[0:1] \oplus DEP.c[0]$	79.95	83.13
$Sent.c[0:1] \oplus SPO.c[0]$	79.89	83.28
$Sent.c[0:1] \oplus SPO - Enh.c[0]$	81.17	84.14
$Sent.c[0:2] \oplus DEP.c[0]$	79.85	83.33
$Sent.c[0:2] \oplus SPO.c[0]$	79.76	83.60
$Sent.c[0:2] \oplus SPO - Enh.c[0]$	**81.27**	**84.27**

Table 5.6: Results of the SPO-Enh model $SPO - Enh.c[0]$ vs. $SPO.c[0]$ on the SICK data sets: SICK-E and SICK-R.

phrase modifiers. We will refer to this chunking method as SPO-Enh. Table 5.4 shows the list of dependency relations we considered in the enhanced-grams. Figure 5.5 shows examples for dependency relations and the resulted chunks from the SPO-Enh component.

We evaluate the SPO-Enh model on the SICK data set, which includes the SICK-E (entailment task) and SICK-R (relatedness task). Table 5.6 shows the results for the two SICK tasks using different DCT coefficients on sentence similar to the strategy in Table 5.3 and block levels. We observe that increasing the number of coefficients for sentence encoding does not improve the results for SICK-E, in contrast for SICK-R where we see a modest improvement from $Sent.c[0]$ to $Sent.c[0:2]$. Adding the Enhanced blocks improves the performance across the board for both SICK-E and SICK-R for every condition ranging from 0.7% absolute to 1.5%. Indicating the importance of these modifiers in the representation. We further observed a similar improvement using Enhanced blocks compared to the DEP based model for both SICK-E and SICK-R tasks with 1.42% and 0.94% improvement gains, respectively.

4 Conclusion

In this chapter, we introduce a novel method for explicitly encoding linguistic knowledge in DCT sentence encoding. We present a unified framework for encoding variable sized

blocks into the model corresponding to different levels of linguistic variability from simple ngrams to syntactic phrases to long distance dependencies. We show that combining block information with basic sentence level encoding improved results on both Probing and Downstream tasks. Moreover, adding explicitly informed syntactic blocks showed gains over simple ngrams for syntactically oriented tasks in the Probing set of tasks. However the difference between ngrams and syntactically guided blocks is less pronounced.

Chapter 6: Conclusions and Future Directions

In this chapter we summarize our contributions and findings from this dissertation, and identify the limitations and directions for future research.

1 Summary of Contributions

Efficient sentence encoder that preserves both the structural and semantic properties of textual data can improve the performance of real-time downstream applications. To date, the complexity of optimizing sentence representations, which are derived from state-of-the-art, heavily-parameterized models, creates a potential bottleneck for real-time applications. Yet, the long-standing Vector Averaging model achieves comparable performance without the computational overhead. Although vector averaging is fast enough to be used in real-time applications, it results in representations that are independent from structural properties. Disregarding the structural characteristics of textual data is problematic for syntax-dependent tasks, in which word order is vital when distinguishing between two events. For example, the meaning of "man bites dog" versus "dog bites man" is indicated entirely by word order, but vector averaging makes no clear distinction as it results in an identical embedding for both. In this dissertation, we addressed the problem of developing efficient, semantic preserving, and structural preserving sentence embeddings. Our main research contributions and insights from this dissertation can be summarized as follows:

- **Developing a non-parameterized sentence encoder:** We developed a sentence encoder that utilizes Discrete Cosine Transform (DCT) to address both computational efficiency and sensitivity to structural information. DCT is a well known technique in the signal processing domain that transfers data from the spatial domain to the frequency domain. In the image processing area, DCT is often used to compress a given image in a manner that preserves the original image quality and other important

characteristics with a higher compression ratio, where fewer DCT coefficients are used to represent the image. In the case of sentence embeddings, the objective is to model the relationship between words in the sentence, in which words are represented as real-valued vectors. Thus, one-dimensional DCT is applied along every word embedding dimension independently. This provides compact sentence representations.

- **Empirically evaluating the developed sentence encoder:** Using sentence embeddings based on DCT coefficients yielded the best performance across various probing and downstream tasks compared to the commonly-used Vector Averaging. For example, based on our analysis of the textual inference task, lower-order DCT coefficients play an important role in capturing the correct relationship (entailment or contradiction) between pairs of texts due to the embeddings' sensitivity to word order. Our analysis reaffirmed our initial observation that lower-order DCT coefficients preserve structural information such as word orders more effectively in the final fixed-length representations. In terms of model efficiency, we demonstrated that, compared to other non-parameterized sentence embeddings, DCT embeddings achieve the highest computational efficiency with low computational cost for the training and inference times. Furthermore, the model does not add any computational complexity to the averaging model, as it only depends on the availability of pre-trained word embeddings. This observation further highlights the importance of DCT embeddings as a suitable alternative to the Vector Averaging model.

- **Empirically evaluating the universality of the DCT encoder across languages:** We also extend the applicability of our DCT encoder to various languages in multi- and cross-lingual settings. In multi-lingual settings, we showed that the DCT embedding's performance is almost equivalent across different languages for all examined tasks. These results indicate that DCT embeddings exhibit similar properties across different languages, where lower-order DCT coefficients encode the syntactic characteristic

in the final fixed-length embedding. As highlighted by these results indicate DCT embeddings can be used to devise efficient language-independent sentence encoders.

- **Empirically evaluating the efficacy of DCT embeddings for cross-lingual mapping:** Our model provides efficient and compact sentence embeddings that can be used for sentence-level, cross-lingual mapping. We utilized our DCT models to extract sentence-level representations for cross-lingual alignment. As expected, DCT embeddings resulted in superior performances in sentence translation retrieval compared to the baseline model, also known as vectors averaging, across all language pairs. These results reaffirmed the power of the structural properties encoded in the lower-order DCT coefficients. However, for the best trade-off between model performance and computational complexity, it might be suitable to consider using K=2 in the $c[0:1]$ model, which combines the semantic and structural properties of the text in the final representations.

- **Developing a linguistically-guided sentence encoder:** We explored the efficacy of enriching sentence representations with linguistic information extracted from different resources. More specifically, we developed a block-based model that combines sentence-level and chunk-level representations to draw the final fixed-length sentence embeddings. In the sentence-level component, DCT is applied to compress a given sentence as one complete chunk; however, in the chunk-level component, DCT is applied to compress chunks of words as smaller, independent blocks within a sentence, which is then averaged to provide the final representation. At the chunk-level, we explored different aspects of extracting meaningful linguistically-informed chunks from a given text. We specifically utilized two main techniques: the n-gram technique, which relies on word adjacency, and the syntactic parser technique, which relies on either phrase-structure grammar extracted from the constituency parser, or dependency grammar extracted from a dependency parser. Based on our

analysis, incorporating linguistic information yields consistent performance gains across different tasks. However, despite the benefits of the linguistic cues integrated in the block-based model, relying on external resources to drive the representations adds to the computational complexity of the model. In fact, the block-based model failed to yield a reasonable trade-off between the model's performance and efficiency. Therefore, the linguistic-free, DCT encoder is a more attractive option, especially for practical real-time applications.

2 Limitations and Future Directions

Overall, the DCT technique is extremely computationally efficient and can be used as an alternative to the Vector Averaging as initiated by the results of this dissertation. However, there remains some very interesting directions to explore. The following are some of the limitations and interesting lines of research to explore in the future:

- **Handling Static, Pre-trained Embeddings:** We generally observed improvements in all sentence-level-based tasks when using DCT models compared to the averaging model. However, we identified some limitations that potentially make it inferior to other state-of-the-art sentence encoders. Since the sentence embeddings,in this dissertation, are calculated from pre-trained word embeddings, the quality of the final sentence representation is constrained by the quality of the pre-trained word embeddings. To alleviate this limitation, it could be beneficial to fine-tune word embeddings for each task, which would enable task-specific representations.

- **Using Neural Networks:** an interesting idea to explore in the future is the parameterizing of DCT coefficients in a way that is similar to the parameterization of the convolutional neural network filters, in which weights are learned during training for the different convolutional filters. For example, more recently in the image processing domain, Ulicny et al. (2020) utilize DCT by replacing the conventional convolutional

layers with DCT filters to produce partially or fully harmonic versions of new or existing CNN architectures. The authors demonstrated the benefits of their model in image classification such as segmentation and edge detection applications. Thus, we identify this as an interesting area to expand our work in the future.

- **Exploring other Linguistic Priors:** Since adding explicitly informed syntactic blocks showed gains over simple n-grams for syntactically-oriented tasks in the probing tasks, it might be beneficial to investigate other methods for incorporating semantic information. For example, unlike syntactic dependency, semantic dependency parsing captures the semantic relationships between words in a sentence using graph-structured representations.

- **Using Other Compression Mechanisms:** It would be interesting to explore other compression mechanisms given the promising results of using DCT to exploit the signal correlations, which are exhibited in the underlying word embeddings between words in a sentence. For example, in the image processing domain, Discrete Wavelet Transformation (DWT) maintains better picture quality with higher compression ratios when compared to the DCT technique (Bhawna, 2010). Since both techniques, DCT and DWT, lead to similar image compression representations, it might be feasible to extend the application of DWT to textual data as well.

List of Publications

- **Related Publications:**

  - **Almarwani, Nada**, Hanan Aldarmaki, and Mona Diab. "Efficient Sentence Embedding using Discrete Cosine Transform." Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019. [https://www.aclweb.org/anthology/D19-1380.pdf]

- **Unrelated Publications:**

  - **Almarwani, Nada**, and Mona Diab. "Arabic textual entailment with word embeddings." Proceedings of the Third Arabic Natural Language Processing Workshop. 2017. [https://www.aclweb.org/anthology/W17-1322.pdf]

  - **Almarwani, Nada**, and Mona Diab. "GW_QA at SemEval2017 Task 3: Question Answer Reranking on Arabic Fora." Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval2017). 2017. [https://www.aclweb.org/anthology/S17-2056.pdf]

Bibliography

Luciano Volcan Agostini, Ivan Saraiva Silva, and Sergio Bampi. 2001. Pipelined fast 2d dct architecture for jpeg image compression. In *Symposium on Integrated Circuits and Systems Design*, pages 226–231. IEEE.

Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. 1974. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93.

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*.

Hanan Aldarmaki. 2019. *Cross-Lingual Alignment of Word & Sentence Embeddings*. Ph.D. thesis, The George Washington University.

Hanan Aldarmaki and Mona Diab. 2016. Gwu nlp at semeval-2016 shared task 1: Matrix factorization for crosslingual sts. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 663–667.

Hanan Aldarmaki and Mona Diab. 2018. Evaluation of unsupervised compositional representations. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2666–2677.

Hanan Aldarmaki and Mona Diab. 2019a. Context-aware cross-lingual mapping. 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 3906–3911.

Hanan Aldarmaki and Mona Diab. 2019b. Scalable cross-lingual transfer of neural sentence embeddings. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (* SEM 2019)*, pages 51–60.

Carl Allen and Timothy Hospedales. 2019. Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning*, pages 223–231.

Nada Almarwani, Hanan Aldarmaki, and Mona Diab. 2019. Efficient sentence embedding using discrete cosine transform. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3663–3669.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. 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 344–354.

Sanjeev Arora, Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and lstms. In *6th International Conference on Learning Representations, ICLR 2018*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Iyad Batal and Milos Hauskrecht. 2009. A supervised time series feature extraction technique using dct and dwt. In *2009 International Conference on Machine Learning and Applications*, pages 735–739. IEEE.

Gauatm Bhawna. 2010. *Image compression using discrete cosine transform and discrete wavelet transform*. Ph.D. thesis.

Piotr Bojanowski, Édouard Grave, Armand Joulin, and Tomáš Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Samuel Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Lyn Untalan Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the*

International Speech Communication Association, Singapore, September 14-18, 2014, pages 2635–2639. ISCA.

Yuehui Chen, Shuyan Jiang, and Ajith Abraham. 2005. Face recognition using dct and hybrid flexible neural tree. In *2005 International Conference on Neural Networks and Brain*, volume 3, pages 1459–1463. IEEE.

Prafulla Kumar Choubey and Ruihong Huang. 2017. A sequential model for classifying temporal relations between intra-sentence events. *arXiv preprint arXiv:1707.07343*.

Renato J Cintra and Fábio M Bayer. 2011. A dct approximation for image compression. *IEEE Signal Processing Letters*, 18(10):579–582.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017a. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017b. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018a. What you can cram into a single $ &!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, pages 7059–7069.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018b. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Chengzhi Deng, Hongbo Zhang, and Yulin Wang. 2019. Effect of different block sizes on the quality of jpeg-like compressed image. In *Proceedings of the 2019 3rd International Conference on Artificial Intelligence and Virtual Reality*, pages 70–73.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics.

Eslam Eid Elmaghraby, Amr Refaat Gody, and Mohamed Hashem Farouk. 2019. Speech recognition using historian multimodal approach. *The Egyptian Journal of Language Engineering*, 6(2):44–58.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. *ACL 2018*, page 91.

Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. 2019. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262.

Zuohui Fu, Yikun Xian, Shijie Geng, Yingqiang Ge, Yuting Wang, Xin Dong, Guang Wang, and Gerard de Melo. 2020. Absent: Cross-lingual sentence representation mapping with bidirectional gans. In *AAAI*, pages 7756–7763.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Prakhar Gupta, Matteo Pagliardini, and Martin Jaggi. 2019. Better word embeddings by disentangling contextual n-gram information. In *Proceedings of NAACL-HLT*, pages 933–939.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL-HLT*, pages 1367–1377.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Jun Huang and Yunxin Zhao. 2000. A dct-based fast signal subspace technique for robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, 8(6):747–751.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. 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)*, volume 1, pages 1681–1691.

Herbert Jaeger. 2001. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13.

Jaswant Jain and Anil Jain. 1981. Displacement measurement and its application in inter-frame image coding. *IEEE Transactions on communications*, 29(12):1799–1808.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Subhradeep Kayal and George Tsatsaronis. 2019. Eigensent: Spectral sentence embeddings using higher-order dynamic mode decomposition. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4536–4546.

Tom Kenter, Alexey Borisov, and Maarten De Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Jamie Kiros and William Chan. 2018. Inferlite: Simple universal sentence representations from natural language inference data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4868–4874.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184. IEEE.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *SemEval@ COLING*, pages 329–334.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Soledad Le Clainche and José M Vega. 2017. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308.

Bofang Li, Tao Liu, Zhe Zhao, Puwei Wang, and Xiaoyong Du. 2017a. Neural bag-of-ngrams. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Bofang Li, Zhe Zhao, Tao Liu, Puwei Wang, and Xiaoyong Du. 2016. Weighted neural bag-of-n-grams model: New baselines for text classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1591–1600.

Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou, and Wei-Yun Ma. 2017b. Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2664–2669.

Ze-Nian Li, Mark S Drew, and Jiangchuan Liu. 2004. *Fundamentals of multimedia*. Springer.

Xiaotong Liu, Hong Xuan, Zeyu Zhang, Abby Stylianou, and Robert Pless. 2019a. Visualizing how embeddings generalize. *arXiv preprint arXiv:1909.07464*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. Association for Computational Linguistics.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Universal text representation from bert: An empirical study. *arXiv preprint arXiv:1910.07973*.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schutze. Introduction to information retrieval? cambridge university press 2008. *Ch*, 20:405–416.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 1–8.

Chandler May, Alex Wang, Shikha Bordia, Samuel Bowman, and Rachel Rudinger. 2019. On measuring social biases in sentence encoders. 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 622–628.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan. European Language Resource Association.

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

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Will Monroe and Yushi Wang. Dependency parsing features for semantic parsing.

Vivi Nastase, Jelber Sayyad Shirabad, and Maria Fernanda Caropreso. 2006. Using dependency relations for text classification. In *Proceedings of the 19th Canadian conference on artificial intelligence*, pages 12–25. Citeseer.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of NAACL-HLT*, pages 528–540.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Ankur P Parikh, Hoifung Poon, and Kristina Toutanova. Grounded semantic parsing for complex knowledge extraction.

Mohammad Zavid Parvez and Manoranjan Paul. 2012. Features extraction and classification for ictal and interictal eeg signals using emd and dct. In *2012 15th International Conference on Computer and Information Technology (ICCIT)*, pages 132–137. IEEE.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Christine Podilchuk and Xiaoyu Zhang. 1996. Face recognition using dct-based feature vectors. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 4, pages 2144–2147. IEEE.

Juan-Pablo Posadas-Durán, Grigori Sidorov, Helena Gómez-Adorno, Ildar Batyrshin, Elibeth Mirasol-Mélendez, Gabriela Posadas-Durán, and Liliana Chanona-Hernández. 2017. Algorithm for extraction of subtrees of a sentence dependency parse tree. *Acta Polytechnica Hungarica*, 14(3):79–98.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

AM Raid, WM Khedr, MA El-dosuky, and Wesam Ahmed. 2014. Jpeg image compression using discrete cosine transform-a survey. *International Journal of Computer Science and Engineering Survey*, 5(2):39.

Vinit Ravishankar, Lilja Øvrelid, and Erik Velldal. 2019. Probing multilingual sentence representations with xprobe. *ACL 2019*, page 156.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3973–3983.

Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. 2018. Concatenated power mean word embeddings as universal cross-lingual sentence representations. *arXiv preprint arXiv:1803.01400*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*, 65:569–631.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Xuancheng Shao and Steven G Johnson. 2008. Type-ii/iii dct/dst algorithms with reduced number of arithmetic operations. *Signal Processing*, 88(6):1553–1564.

HaeYong Shin, Byung-Gul Ryu, Woo-Jong Ryu, GeunJae Lee, and SangKeun Lee. 2016. Bringing bag-of-phrases to odp-based text classification. In *2016 International Conference on Big Data and Smart Computing (BigComp)*, pages 485–488. IEEE.

Xiao Shu, Xiaolin Wu, and Bolin Liu. 2017. A study on quantization effects of dct based compression. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3500–3504. IEEE.

Yuqi Si, Jingqi Wang, Hua Xu, and Kirk Roberts. 2019. Enhancing clinical concept extraction with contextual embeddings. *Journal of the American Medical Informatics Association*, 26(11):1297–1304.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 deep learning and unsupervised feature learning workshop*, volume 2010, pages 1–9.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

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

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. 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 1556–1566.

Prabhakar Telagarapu, V Jagan Naveen, A Lakshmi Prasanthi, and G Vijaya Santhi. 2011. Image compression using dct and wavelet transformations. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 4(3):61–74.

S Toldova, O Lyashevskaya, A Bonch-Osmolovskaya, and M Ionov. 2015. Evaluation for morphologically rich language: Russian nlp. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, page 300. The Steering Committee of The World Congress in Computer Science, Computer ....

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1434–1444.

Matej Ulicny, Vladimir A Krylov, and Rozenn Dahyot. 2020. Harmonic convolutional networks based on discrete cosine transform. *arXiv preprint arXiv:2001.06570*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Bin Wang and C-C Jay Kuo. 2020. Sbert-wk: A sentence embedding method by dissecting bert-based word models. *arXiv preprint arXiv:2002.06652*.

Yuxuan Wang, Yutai Hou, Wanxiang Che, and Ting Liu. 2020. From static to dynamic word representations: a survey. *International Journal of Machine Learning and Cybernetics*, pages 1–20.

Andrew B Watson. 1994. Image compression using the discrete cosine transform. *Mathematica journal*, 4(1):81.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

Yegui Xiao, NP Chandrasiri, Yoshiaki Tadokoro, and Masaomi Oda. 1999. Recognition of facial expressions using 2d dct and neural network. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 82(7):1–11.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019a. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Ziyi Yang, Chenguang Zhu, and Weizhu Chen. 2019b. Parameter-free sentence embedding via orthogonal basis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 638–648.

Yegui Xiao, L. Ma, and K. Khorasani. 2006. A new facial expression recognition technique using 2-d dct and neural networks based decision tree. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 2421–2428.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32.

Zhe Zhao, Tao Liu, Shen Li, Bofang Li, and Xiaoyong Du. 2017. Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 244–253.

Junru Zhou, Zhuosheng Zhang, and Hai Zhao. 2019. Limit-bert: Linguistic informed multi-task bert. *arXiv preprint arXiv:1910.14296*.