Efficient Sentence Embedding using Discrete Cosine Transform

Nada Almarwani,∗1,2 Hanan Aldarmaki,∗3 Mona Diab 1,4
1 Dept. of Computer Science, The George Washington University
2 Dept. of Computer Science, Taibah University
3 Computer Science & Software Engineering Dept., UAEU
4 AWS, Amazon AI
nadaoh@gwu.edu, h-aldarmaki@uaeu.ac.ae, diabmona@amazon.com

Abstract
Vector averaging remains one of the most popular sentence embedding methods in spite of its obvious disregard for syntactic structure. While more complex sequential or convolutional networks potentially yield superior classification performance, the improvements in classification accuracy are typically mediocre compared to the simple vector averaging. As an efficient alternative, we propose the use of 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 results 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 vector averaging. With practically equivalent complexity, the model yields better overall performance in downstream classification tasks that correlate with syntactic features, which illustrates the capacity of DCT to preserve word order information.

1 Introduction
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 (Mikolov et al., 2013) and Glove (Pennington et al., 2014) to contextualized embeddings such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). However, most applications operate at the phrase or sentence level. Hence, the word embeddings are averaged to yield sentence embeddings. Averaging is an efficient compositional operation that leads to good performance. In fact, averaging is difficult to beat by more complex compositional models as illustrated across several classification tasks: topic categorization, semantic textual similarity, and sentiment classification (Aldarmaki and Diab, 2018). Encoding sentences into fixed-length vectors that capture various full sentence linguistic properties leading to performance gains across different classification tasks remains a challenge. Given the complexity of most models that attempt to encode sentence structure, such as convolutional, recursive, or recurrent networks, the trade-off between efficiency and performance tips the balance in favor of simpler models like vector averaging. Sequential neural sentence encoders, like Skip-thought (Kiros et al., 2015) and InferSent (Conneau et al., 2017), can potentially encode rich semantic and syntactic features from 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 don’t generalize well to out-of-domain data (Wieting et al., 2015). FastSent (Hill et al., 2016) is an unsupervised alternative approach of lower computational cost, but similar to vector averaging, it disregards word order. Tensor-based composition 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 them more suitable for practical text classification.

In this work, we propose to use the Discrete Cosine Transform (DCT) as a simple and efficient way to model word order and structure in sentences while maintaining practical efficiency.
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), but to our knowledge, this is the first successful application of DCT for NLP applications, and in particular sentence embedding. We use DCT to summarize the general feature patterns in word sequences and compress them into fixed-length vectors. Experiments in probing tasks demonstrate that our DCT embeddings preserve more syntactic and semantic features compared with vector averaging. Furthermore, the results indicate that DCT performance in downstream applications is correlated with these features.

2 Approach

2.1 Discrete Cosine Transform

Discrete Cosine Transform (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 \( c^0, ..., c^{N-1} \) as follows:

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

and

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

for \( 1 \leq k < N \). 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 exactly using the inverse transform. In practice, DCT is used for compression by preserving only the coefficients with large magnitudes. Lower-order coefficients represent lower signal frequencies which correspond to the overall patterns in the sequence (Ahmed et al., 1974).

2.2 DCT Sentence Embeddings

We apply DCT on 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 rows. In other words, 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 discard higher-order coefficients, which results in sentence vectors of size \( Kd \). For cases where \( N < K \), we pad the sentence with \( K - N \) zero vectors. In image compression, the magnitude of the coefficients tends to decrease with increasing \( k \), but we didn’t observe this trend in text data except that \( c^0 \) tends to have larger absolute value than the remaining coefficients. Nonetheless, by retaining lower-order coefficients we get a consistent representation of overall feature patterns in the word sequence.

Figure 1 illustrates the properties of DCT embeddings compared to vector averaging (AVG). Notice that the first DCT coefficients, \( c^0 \), result in vectors that are independent of 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 from the different word components (‘bitten’ vs. ‘bite’), which can potentially be useful in downstream tasks. Since both DCT and AVG are un-
parameterized, the downstream classifiers can incorporate a hidden layer to learn these subtle variations in higher-order features depending on the learning objective.

2.3 A Note on Complexity

The cosine terms in Equation 2 can be precalculated 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. 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.\(^2\) Note also 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 Experiments and Results

3.1 Evaluation Framework

We use the SentEval toolkit\(^3\) (Conneau and Kiela, 2018) to evaluate the sentence representations on different probing as well as downstream classification tasks. The probing benchmark set was designed to analyze the quality of sentence embeddings. It contains a set of 10 classification tasks, summarized in Table 1, that address varieties of linguistic properties including surface, syntactic, and semantic information (Conneau et al., 2018). The downstream set, on the other hand, includes the following standard classification tasks: binary and fine-grained sentiment classification (MR, SST2, SST5) (Pang and Lee, 2004; Socher et al., 2013), product reviews (CR) (Hu and Liu, 2004), opinion polarity (MPQA) (Wiebe et al., 2005), question type classification (TREC) (Voorhees and Tice, 2000), natural language inference (SICK-E) (Marelli et al., 2014), semantic relatedness (SICK-R, STSB) (Marelli et al., 2014; Cer et al., 2017), paraphrase detection (MRPC) (Dolan et al., 2004), and subjectivity/objectivity (SUBJ) (Pang and Lee, 2004).

3.2 Experimental setup

For the word embeddings, we use pre-trained FastText embeddings of size 300 (Mikolov et al., 2018) trained on Common-Crawl. We generate DCT sentence vectors by concatenating the first $K$ DCT coefficients, which we denote by $c^{0:K}$. We compare the performance against: vector averaging of the same word embeddings, denoted by AVG, and vector max pooling, denoted by MAX.\(^4\)

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.

3.3 Result & Discussion

We report the performance in probing tasks in Table 2. In general, DCT yields better performance compared to averaging on all tasks, and larger $K$ often yields improved performance in syntactic and semantic tasks. For the surface information tasks, SentLen and Word content (WC), $c^{0}$ significantly outperforms AVG. This is attributed to the non-linear scaling factor in DCT, where longer sentences are not discounted as much as in averaging. The performance decreased with increasing $K$ in $c^{0:K}$, which reflects the trade-off between deep and surface linguistic properties, as discussed in Conneau et al. (2018).

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 three of the semantic tasks, where DCT performs well above AVG and the performance improves with increasing $K$. The only exception

---

\(^2\)We experimented with $1 \leq K \leq 7$.

\(^3\)https://github.com/facebookresearch/SentEval

\(^4\)To compare with other sentence embedding models, refer to the results in Conneau and Kiela (2018) and Conneau et al. (2018)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SentLen</td>
<td>Length prediction</td>
</tr>
<tr>
<td>WC</td>
<td>Word Content analysis</td>
</tr>
<tr>
<td>TreeDepth</td>
<td>Tree depth prediction</td>
</tr>
<tr>
<td>TopConst</td>
<td>Top Constituents prediction</td>
</tr>
<tr>
<td>BShift</td>
<td>Word order analysis</td>
</tr>
<tr>
<td>Tense</td>
<td>Verb tense prediction</td>
</tr>
<tr>
<td>SubjNum</td>
<td>Subject number prediction</td>
</tr>
<tr>
<td>ObjNum</td>
<td>Object number prediction</td>
</tr>
<tr>
<td>SOMO</td>
<td>Semantic odd man out</td>
</tr>
<tr>
<td>CoordInv</td>
<td>Coordination Inversion</td>
</tr>
</tbody>
</table>

Table 1: Probing Tasks
is SOMO, where increasing $K$ actually results in lower performance, although all DCT results are still higher than AVG by about 1% to 34%.

The correlation between the performance in probing tasks and the standard text classification tasks is discussed in Conneau et al. (2018), where they show that most tasks are only positively correlated with a small subset of semantic or syntactic features, with the exception of TREC and some sentiment classification benchmarks. Furthermore, some tasks like SST and SICK-R are actually negatively correlated with the performance in some probing tasks like SubjNum, ObjNum, and BShift. This explains why simple averaging often outperforms more complex models in these tasks. Our results in Table 3 are consistent with these observations, where we see improvements in most tasks, but the difference is not as significant as the probing tasks, except in TREC question classification where increasing $K$ leads to much better performance. As discussed in Aldarmaki and Diab (2018), the ability to preserve word order leads to improved performance in TREC, which is exactly the advantage of using DCT instead of AVG. Note also that increasing $K$, while preserves more information, leads to increasing the number of model parameters, which in turn may negatively affect the generalization of the model by overfitting. In our experiments, $1 \leq k \leq 2$ yielded the best trade-off.

### 3.4 Comparison w. Related 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. To the best of our knowledge, spectral methods have only been recently exploited to construct sentence embedding (Kayal and Tsatsaronis, 2019).\footnote{Independent from and in parallel with this work.}

Kayal and Tsatsaronis propose EigenSent that utilized Higher-Order Dynamic Mode Decomposition (HODMD) (Le Clainche and Vega, 2017) to construct sentence embedding. These embeddings summarize the dynamic properties of the sentence. In their work, they compare EigenSent with various sentence embedding models, including a different implementation of the Discrete Cosine Transform (DCT*). In contrast to our implementation described in section 2.2, DCT* is applied at the word level along the word embedding dimension.

For fair comparison, we use the same sentiment and text classification datasets, the SST-5, 20 newsgroups (20-NG) and Reuters-8 (R-8), as those used in Kayal and Tsatsaronis (2019). We also evaluate using the same pre-trained word embedding, framework and approaches as described in their work. Table 4 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 $c^k$.\footnote{The best results were achieved with $k=3$ for SST-5 and $k=2$ for 20-NG and R-8.}

<table>
<thead>
<tr>
<th>Model</th>
<th>20-NG</th>
<th>R-8</th>
<th>SST-5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>PCA</td>
<td>55.43</td>
<td>54.67</td>
<td>54.77</td>
</tr>
<tr>
<td>DCT*</td>
<td>61.07</td>
<td>59.16</td>
<td>59.78</td>
</tr>
<tr>
<td>Avg. vec.</td>
<td>68.72</td>
<td>68.19</td>
<td>68.25</td>
</tr>
<tr>
<td>p-means</td>
<td>72.20</td>
<td><strong>71.65</strong></td>
<td><strong>71.79</strong></td>
</tr>
<tr>
<td>ELMo</td>
<td>71.20</td>
<td>71.79</td>
<td>71.36</td>
</tr>
<tr>
<td>BERT</td>
<td>70.89</td>
<td>70.79</td>
<td>70.88</td>
</tr>
<tr>
<td>EigenSent</td>
<td>66.98</td>
<td>66.40</td>
<td>66.54</td>
</tr>
<tr>
<td>EigenSent⊕Avg</td>
<td><strong>72.24</strong></td>
<td><strong>71.62</strong></td>
<td><strong>71.78</strong></td>
</tr>
<tr>
<td>$c^k$</td>
<td>72.20</td>
<td>71.58</td>
<td>71.73</td>
</tr>
</tbody>
</table>

Table 4: 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. **Bold** indicates the best result, and *italic* indicates second-best.

Our model achieved close to state-of-the-art performance in both the 20-NG and R-8 tasks. Our model outperformed EignSent on all tasks and generally performed better than or on par with p-means, ELMo, BERT, and EigenSent⊕Avg on both the 20-NG and R-8. On the other hand, both EigenSent⊕Avg and ELMo performed better than all other models on SST-5.

4 Conclusion

We proposed using the Discrete Cosine Transform (DCT) as a mechanism to efficiently compress 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, as in 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. 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.
References


Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke


