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.

---

[*]Both authors contributed equally.

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:[1]

$$c^0 = \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} v_n, \qquad (1)$$

and

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

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

---

[1]There are several variants of DCT. We use DCT type II (Shao and Johnson, 2008) in our implementation
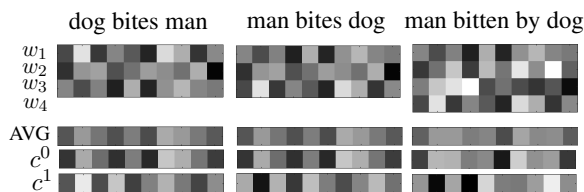

Figure 1: Illustration of word vector averaging vs. DCT using the first 2 DCT coefficients. The word vectors are generated randomly from a standard normal distribution with $d = 10$.

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-

3673

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 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. 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

---

Task	Description
SentLen	Length prediction
WC	Word Content analysis
TreeDepth	Tree depth prediction
TopConst	Top Constituents prediction
BShift	Word order analysis
Tense	Verb tense prediction
SubjNum	Subject number prediction
ObjNum	Object number prediction
SOMO	Semantic odd man out
CoordInv	Coordination Inversion

Table 1: Probing Tasks

DCT sentence vectors by concatenating the first K DCT coefficients, which we denote by $\mathbf{c^{0:K}}$. 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

---

	Surface		Syntactic			Semantic				
Model	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	64.12	82.1	36.38	68.04	50.16	87.9	80.89	80.24	50.39	51.95
MAX	62.67	88.97	33.02	62.63	50.31	85.66	77.11	76.04	51.86	52.33
c^0	**98.67**	**91.11**	38.6	70.54	50.42	88.25	80.88	80.56	**55.6**	55
$c^{0:1}$	97.18	89.16	40.41	78.34	52.25	88.58	86.59	84.36	54.62	70.42
$c^{0:2}$	95.84	86.77	43.01	80.41	54.84	88.87	88.06	86.26	53.07	71.87
$c^{0:3}$	94.63	84.96	**43.35**	81.01	57.29	88.88	88.36	86.51	53.79	**72.01**
$c^{0:4}$	93.25	83.24	43.26	81.49	60.31	**88.91**	**88.65**	87.15	52.77	71.91
$c^{0:5}$	92.29	81.84	42.75	**81.60**	62.01	88.82	88.44	87.98	52.38	70.96
$c^{0:6}$	91.56	79.83	43.05	81.41	**62.59**	88.87	**88.65**	**88.28**	52.07	70.63

Table 2: 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 sole feature (Length) as reported in (Conneau et al., 2018), respectively.

	Sentiment Analysis					SUBJ	Relatedness/Paraphrase			Inference	TREC
Model	MR	SST2	SST5	CR	MPQA		SICK-R	STSB	MRPC	SICK-E	
AVG	78.3	**84.13**	44.16	79.6	87.94	92.33	81.95	69.26	74.43	79.5	83.2
MAX	73.31	79.24	41.86	73.35	86.54	88.02	81.93	**71.57**	72.5	77.98	76.2
c^0	**78.45**	83.53	44.57	79.81	**88.36**	**92.79**	82.61	71.11	72.93	78.91	84.8
$c^{0:1}$	78.15	83.47	**46.06**	79.84	87.76	92.61	82.73	70.82	72.81	79.64	88.2
$c^{0:2}$	78.02	82.98	45.16	79.68	87.62	92.5	**82.95**	70.36	72.87	79.76	**89.8**
$c^{0:3}$	77.81	83.8	45.79	79.66	87.54	92.4	82.93	69.79	73.57	**80.56**	88.2
$c^{0:4}$	77.72	83.75	44.03	**80.08**	87.4	92.61	82.53	69.31	72.35	79.72	**89.8**
$c^{0:5}$	77.42	82.43	43.3	78.6	87.21	92.19	82.36	68.9	73.91	79.89	88.8
$c^{0:6}$	77.47	82.81	42.99	78.78	87.06	92.15	81.86	68.17	**75.07**	79.76	86.4

Table 3: DCT embedding Performance in SentEval downstream tasks compared to vector averaging (AVG) and max pooling (MAX).

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

	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 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.

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).[5].

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.[6]

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. 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.

---

[5] independent from and in parallel with this work

[6]The best results were achieved with k=3 for SST-5 and k=2 for 20-NG and R-8.

References

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

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

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.

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. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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.

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

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.

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.

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.

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

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.

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)*.

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.

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.

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.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. 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.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207. ACM.

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.