EigenSent: Spectral sentence embeddings using higher-order Dynamic Mode Decomposition

Subhradeep Kayal **George Tsatsaronis**
Elsevier B.V.
Amsterdam, NL.
subhradeep.kayal@gmail.com

Abstract

Distributed representation of words, or *word embeddings*, have motivated methods for calculating semantic representations of word sequences such as phrases, sentences and paragraphs. Most of the existing methods to do so either use algorithms to learn such representations, or improve on calculating weighted averages of the word vectors. In this work, we experiment with spectral methods of signal representation and summarization as mechanisms for constructing such word-sequence embeddings in an unsupervised fashion. In particular, we explore an algorithm rooted in fluid-dynamics, known as *higher-order Dynamic Mode Decomposition*, which is designed to capture the eigenfrequencies, and hence the fundamental transition dynamics, of periodic and quasi-periodic systems. It is empirically observed that this approach, which we call *EigenSent*, can summarize transitions in a sequence of words and generate an embedding that can represent well the sequence itself. To the best of the authors' knowledge, this is the first application of a spectral decomposition and signal summarization technique on text, to create sentence embeddings. We test the efficacy of this algorithm in creating sentence embeddings on three public datasets, where it performs appreciably well. Moreover it is also shown that, due to the positive combination of their complementary properties, concatenating the embeddings generated by *EigenSent* with simple word vector averaging achieves state-of-the-art results.

1 Introduction

1.1 Relevant concepts

Word embeddings are dense vectors that capture the semantic and contextual information of a word, and are ubiquitous in natural language processing tasks across many domains (Camacho-Collados and Pilehvar, 2018). Several different algorithms and models for constructing these embeddings have been proposed and evaluated in literature (Perone et al., 2018).

A natural next step is to extend the notion of word embeddings to the level of a sentence (or paragraph, or document). Such representations are known as *sentence embeddings*, often interchangeably used with the terms *paragraph embeddings* or *document embeddings*, and should, ideally, capture the meaning of a sentence (Le and Mikolov, 2014).

More recently, the concept of *universal sentence embeddings* has gained traction, as they leverage models trained on large text corpuses in a way which is task-agnostic. These pre-trained models can then be used in a wide array of downstream tasks, often performing better in those tasks when little training data is available (Subramanian et al., 2018).

1.2 A brief review of literature

Word embedding methods vary from complex neural language models (Bengio et al., 2003) and semi-supervised approaches (Turian et al., 2010), to simpler and faster methods such as *Word2Vec* (Mikolov et al., 2013), *GloVe* (Pennington et al., 2014), *ELMo* (Peters et al., 2018) and *BERT* (Devlin et al., 2019), that can be trained on much larger volumes of data.

In order to learn high-quality word embeddings, a method must capture the contextually-relevant semantic meaning of a word. This is often done by training a language model on a dataset; an example is the method known as *Embedding from Language Models* or *ELMo* (Peters et al., 2018), which uses representations from the internal layers of a bi-directional LSTM that is trained with a language model objective. Very recently, Devlin et al. (2019) introduced another generalizable language model, named as *BERT* or *bi-directional Encoder*

Representations from Transformers, consisting of layers of *transformers* (Vaswani et al., 2017) with bi-directional self-attention, which delivered state-of-the-art results in numerous benchmarks.

Similar to word embeddings, there has been a substantial amount of research on constructing sentence embeddings, in recent years. Several self-supervised approaches have been proposed, such as the extension of the Word2Vec model to include sentences and learn their representations (Le and Mikolov, 2014), and encoder-decoder approaches that try to reconstruct the surrounding sentences of an encoded passage (Kiros et al., 2015). Recently, bi-directional LSTM models were trained in a strongly supervised fashion on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) by Conneau et al. (2017). This method, known as *InferSent*, has produced state-of-the-art results, as a universal sentence encoder, on various other downstream tasks.

Aside from these state-of-the-art approaches which require some sort of model training, a common approach to embed sentences is to simply compute the dimension-wise arithmetic mean of the embeddings of the words in a particular sentence. Further improvements were made, using weighted averages of the word embeddings and modifying them using singular-value decomposition, by Arora et al. (2017) and, recently, power-mean embeddings (Rücklé et al., 2018). These methods have narrowed the performance deficit to other complex sentence embedding methods such as *InferSent*.

1.3 Hypothesis and contribution of this work

As a complement to most of the aforementioned work, in this paper, we aim at utilizing spectral methods in order to construct sentence embeddings. Spectral analysis is widely used in signal processing to decompose a signal into its component frequencies, thereby revealing the important dynamics that make up the signal and summarizing the transitions in it. The hypothesis of this paper is: *if we could use similar techniques on sentences, which are also composed of meaningful transitions (between words), as we do with signals, then it should be possible to capture the important transitional dynamics that make up the respective sentences*. The first step towards our goal is to represent a sentence as a *signal*, which has some meaningful transitional properties that we can capture. In order to do this, we rely on the word embeddings.

A key observation which motivates the use of word embeddings to represent a sentence as a signal, is the fascinating property of word vectors to approximately obey the laws of algebra, as they seem to capture word relationships and analogies. The original paper by Mikolov et al. (2013) presented an example wherein *vector("King") - vector("Man") + vector("Woman")* results in a vector that is most similar to the representation for the word *Queen*. Following this observation, we posit that using spectral techniques, it should be possible to capture the dynamic properties of a sentence by treating it as a multi-dimensional signal over time, where the vector representation of each word in the sentence is a single point in the signal.

The major innovation introduced in this paper is the use of the *higher-order Dynamic Mode Decomposition (HODMD)* (Le Clainche and Vega, 2017) algorithm to exploit the temporal dynamics in a sequence of word vectors, in order to construct sentence embeddings. HODMD is an efficient extension of the basic *Dynamic Mode Decomposition (DMD)* algorithm, which has been widely used in fluid dynamics in order to capture the fundamental frequencies of complex fluid flows (Schmid, 2010). We compare the generated sentence embeddings using the said method against state-of-the-art methods such as *BERT* (Devlin et al., 2019), ELMo (Peters et al., 2018) and p-means (Rücklé et al., 2018), followed by comparisons with *Discrete Cosine Transform (DCT)* (Ahmed et al., 1974), which is a spectral method that has been used widely for data compression, and *Principal Components Analysis (PCA)* (Hotelling, 1933), by extensive experiments on three public datasets. We also show that by concatenating the embeddings generated by HODMD, which captures the *dynamics* of a sequence, with a method such as word vector averaging, which grasps the notion of *scale*, we can further improve the resultant embeddings for downstream tasks.

1.4 Paper structure

Having introduced the key concepts and motivators of this work in Section 1, we proceed by describing the higher-order Dynamic Mode Decomposition algorithm and the other relevant benchmark methods in Section 2. Section 3 outlines the datasets and the software implementations that have been used in this paper, followed by the experimental procedures being stated in Section 4.1 and the results being analyzed in Section 4.2. Finally, in Section 5 we summarize the important conclusions of this paper.

2 Methodology

This section introduces the relevant concepts pertaining to higher-order Dynamic Mode Decomposition, the algorithm that has been used in this work to construct sentence embeddings, as well as competing methods that are being tested against.

2.1 Higher-order Dynamic Mode Decomposition and motivation for use

2.1.1 Preliminaries

Let S_1^N be a sentence composed of words $w_1, ..., w_N$, where w_i is the i^{th} word in the sentence. A word, w_i, can be replaced by a pre-trained word embedding, such as the one provided by Mikolov et al. (2013)[1], or can be *learned* for the experiment dataset itself, using any of the methods mentioned in Section 1.2. Then the sentence can be written as a multidimensional signal as follows:

$$S_1^N = [w_1, w_2, ..., w_N] = \begin{bmatrix} w_1^1 & w_2^1 & \dots & w_N^1 \\ \vdots & \vdots & \ddots & \\ w_1^m & w_2^m & & w_N^m \end{bmatrix} \tag{1}$$

where, m is the dimension of the word vector and N is the number of words in a sentence.

In order to apply standard DMD (Schmid, 2010) to such a *signal*, the *first-order Koopman assumption* is employed, which can be written in multiple different forms:

$$\begin{gathered} w_k \approx A.w_{k-1} \\ S_1^N \approx [w_1, A.w_1, A^2.w_1, ..., A^{N-1}.w_1] \end{gathered} \tag{2}$$

where $k = 2, ..., N$ and A is a $m \times m$ square matrix.

Thus, the assumption is that each sentence has words in it which lie in a constant subspace generated by A (Brunton et al., 2016), or that the words in a sentence transition from one another smoothly, transformed by the constant operator A. This assumption can be seen as an extension of the observation that word vectors seem to approximately obey the laws of simple algebra. The operator A then captures the overall transition dynamics of the sentence and summarizing A would lead to the construction of the desired sentence embedding.

The first-order Koopman assumption, although a good starting point, constrains a snapshot of a system, i.e., a word in a sentence in our case, to transition solely from the previous one. To further relax this constraint in an attempt to make our assumption more realistic, we look towards the work of Le Clainche and Vega (2017), who propose a *higher-order Koopman assumption*:

$$w_k \approx A_1.w_{k-d} + A_2.w_{k-d+1} + ... + A_d.w_{k-1} \tag{3}$$

where $k = 2, ..., N-d+1$ and d can be understood as the *order* parameter.

This may also be written in a form similar to equation 2:

$$\widetilde{w}_k \approx \widetilde{A}.\widetilde{w}_{k-1} \tag{4}$$

where,

$$\begin{gathered} \widetilde{w}_k = \begin{bmatrix} w_k \\ w_{k+1} \\ \vdots \\ w_{k+d-2} \\ w_{k+d-1} \end{bmatrix} \\ \widetilde{A} = \begin{bmatrix} 0 & I & 0 & \dots & 0 & 0 \\ 0 & 0 & I & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & I & 0 \\ A_1 & A_2 & A_3 & \dots & A_{d-1} & A_d \end{bmatrix} \end{gathered} \tag{5}$$

with I being a $m \times m$ identity matrix. Furthermore, the modified sentence matrix can be written as:

$$\widetilde{S}_1^N = [\widetilde{w}_1, \widetilde{w}_2, ..., \widetilde{w}_N] \tag{6}$$

With this relaxation, a particular word in a sentence is not only related to the preceding word, but to a number of preceding words in a window of size d, which is tunable, and $d = 1$ falls back to the first-order case. This, more realistic, relaxation to the original DMD algorithm is what motivates us to use HODMD in order to capture the transition dynamics in a sentence in order to construct sentence embeddings.

2.1.2 Generating sentence embeddings

The starting point is the following, which is a matrix form of equation 4:

$$\widetilde{S}_2^N \approx \widetilde{A}.\widetilde{S}_1^{N-1} \tag{7}$$

Performing SVD on S_1^{N-1} gives us:

$$\widetilde{S}_1^{N-1} = U.\Sigma.T^T \tag{8}$$

[1]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

Algorithm 1: HODMD algorithm for constructing a sentence embedding, *EigenSent*

```
  Data: Sequence of word vectors in a sentence
        $S_1^N = [w_1, w_2, ..., w_N]$, order
        parameter $d$, number of dynamic
        modes to choose $n$
  Result: Sentence embedding, $V$
1 Declare $\widetilde{S}_1^N = [\widetilde{w}_1, \widetilde{w}_2, ..., \widetilde{w}_N]$, where $\widetilde{w}_k$ is
    given by equation 5;
2 Perform SVD: $\widetilde{S}_1^{N-1} = U.\Sigma.T^T$;
3 higher-order Koopman operator:
    $\widetilde{A} \approx \widetilde{S}_2^N.T.\Sigma^{-1}.U^T$;
4 Eigendecomposition of Koopman operator:
    $[EigVec, EigVal] = eigendecomp(\widetilde{A})$,
    where $EigVec$ contains the eigenvectors,
    one per column, sorted by the magnitude of
    the eigenvalues in $EigVal$;
5 Sentence embedding:
    $V = EigVec_1 \oplus EigVec_2 ... \oplus EigVec_n$,
    where $a \oplus b$ signifies concatenation of
    vectors $a$ and $b$;
```

where Σ is the diagonal matrix containing the SVD singular values, sorted in decreasing order, while the columns in U and T are the spatial and temporal SVD-modes.

Using equations 7 and 8, we can derive:

$$\widetilde{A} \approx \widetilde{S}_2^N.T.\Sigma^{-1}.U^T \tag{9}$$

as, $T^T = T^{-1}$ and $U^T = U^{-1}$

Now that we have characterized the *higher-order Koopman operator*, $\widetilde{A}$, using equation 9, the dynamic modes and mode amplitudes can simply be calculated by obtaining its eigenvalues and eigenvectors using any eigendecomposition technique. Since the dynamic modes (or eigenvectors) corresponding to the largest dynamic mode amplitudes (or eigenvalues) capture the largest-scale dynamics present in the sequence of words, the top-K modes, as sorted by the mode amplitudes, are concatenated, to be used as the sentence embedding for the corresponding sentence.

The overall process is depicted in Algorithm 1.

For a chosen order d, the size of the sentence embedding is $m * d$.

2.2 Competing Methods

2.2.1 State-of-the-art

We compare our method, as explained in Algorithm 1, to three recent state-of-the-art methods. The first one, p-means (Rücklé et al., 2018), is a method that concatenates different *types* of means, known as *power-means* (Hardy et al., 1952), of the word embeddings in a sentence. The hypothesis of the authors of (Rücklé et al., 2018) is that the average of word vectors is only one type of order-statistic and there are several others available, which might add useful information when constructing sentence embeddings.

The second method, ELMo (Peters et al., 2018), trains a bi-directional LSTM, using word level and sub-word level features, with a language model objective on a large dataset, and then uses the representations of words from its internal layers to provide rich and contextual word embeddings. A pre-trained model, trained on the One Billion Words benchmark (Chelba et al., 2013), was used for our experiments, and an averaging bag-of-words scheme was employed to produce the sentence embeddings based on the word representation features from all three layers of the ELMo model.

The final approach, BERT (Devlin et al., 2019), aims to produce a general-purpose language model by training a deep network of bi-directional transformers with self attention, using a masked language-model objective. By taking bi-directionality into account, it improves on previous efforts, such as the *Generative Pre-trained Transformer* (Radford et al., 2018), which were unidirectional. For our experiments, a pre-trained model, trained on the concatenation of BooksCorpus (Zhu et al., 2015) and English Wikipedia, was used. Sentence embeddings were constructed by averaging the token representations from the second-to-last hidden layer of the model, as this approach produced good results in the original work.

2.2.2 Discrete Cosine Transform and PCA

Aside from comparing *EigenSent* to the state-of-the-art, it is also prudent to compare the proposed method to other approaches rooted in the frequency domain. Two very popular methods for summarizing or compressing information are Discrete Cosine Transform (DCT) (Ahmed et al., 1974) and Principal Components Analysis (PCA) (Hotelling, 1933).

DCT is a special case of the Fourier transform, which aims to decompose a signal into the frequencies that make up the signal. In the case of DCT the basis vectors are infinite-scale cosine

Dataset	#classes	#train docs	#test docs	#total docs
20 newsgroups (20-NG)	20	11293	7528	18821
Reuters-8 (R-8)	8	5485	2189	7674
Stanford Sentiment Treebank (SST-5)	5	8534	2209	10743

Table 1: Metadata information describing the datasets used in our experiments.

functions of increasing frequencies. The goal of DCT in the multidimensional case is to determine a set of vectors (or *components*) which can be used to linearly combine the cosine functions to retrieve the original signal. The DCT components are arranged in order of importance in recreating the original signal and the top-K components are concatenated to form the embedding of a corresponding sentence on which DCT is performed.

The comparison to PCA is only natural, as DMD works analogously to it. However, DMD contains information about the transition dynamics of a sequence, whereas PCA lacks this property (Schmid, 2010). This is because DMD is based on the eigendecomposition performed on the *Koopman operator* derived from the multidimensional signal, which captures the transition dynamics in that signal, whereas PCA is based on the covariance matrix produced from the signal, which does not. In a fashion similar to above, the top-K *principal components* are concatenated to form the sentence embedding.

We choose the aforementioned methods to benchmark *EigenSent* against because together they provide a significant coverage of logical competing ideas. The p-means method is based on the algebraic manipulation of the sequence of word embeddings, in order to create a sentence embedding, and does not require training, much like other methods such as (Arora et al., 2017). ELMo is another state-of-the-art method and it represents other such methods which leverage language models in order to capture contextual information to form embeddings. As for DCT and PCA, they are well-studied methods which are used for the spectral representation of a signal. DCT is a non-adaptive method, in a sense that it fixes the basis vectors to be cosine functions, whereas PCA is adaptive and learns a set of orthogonal bases.

3 Datasets and resources

3.1 Datasets

In order to compare the embeddings generated by our method to the benchmark methods, described in Section 2.2, we use three public datasets, pertaining to text and sentiment classification, of varying degrees of complexity.

The *20 newsgroups (20-NG)* and the *Reuters-8 (R-8)*[2] are popular text classification datasets which have been widely used in literature, comprising documents that appeared in the Reuters newswire in 1987. The former has 20 different conceptual classes for the textual content to be classified into, while the latter has 8.

The *Stanford Sentiment Treebank (SST-5)*[3] (Socher et al., 2013) is a dataset for sentiment categorization where a corpus of movie review excerpts from the rottentomatoes.com are categorized either 5 classes representing sentiments varying from *very positive* to *very negative*.

More metadata information about these datasets are provided in Table 1.

3.2 Software and resources

Resources: In order to construct sentence embeddings, all the competing methods except ELMo require a set of word vectors. In this work, we use the pre-trained set of word embeddings provided by Mikolov et al[4]. For experimenting with ELMo and evaluating it on the chosen datasets, we use a pre-trained model[5], trained on the One Billion Words benchmark (Chelba et al., 2013). In the case of BERT, we use the *BERT-Large, Uncased*[6] model, which is a 24-layer deep transformer network that was trained on Wikipedia and the BooksCorpus (Zhu et al., 2015) for 1 million update steps.

Implementations: For DCT, we use an implementation provided in the *SciPy* Python library (Jones et al., 2001–)[7], which uses *Fast Fourier Transform (FFT)* to get the cosine transform components, while PCA is made available in *scikit-*

[2]https://www.cs.umb.edu/~smimarog/textmining/datasets/

[3]https://nlp.stanford.edu/sentiment/index.html

[4]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

[5]https://tfhub.dev/google/elmo/2

[6]https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-24_H-1024_A-16.zip

[7]https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.fftpack.dct.html

Model	Configurations
pmeans	The power parameter, p, is varied between 1, [1,2], [1,3] and [1,6], where [1,6] means that the sentence embedding is produced by concatenating the power-means for the values 1, 2, 3, 4, 5 and 6.
BERT	Pre-trained model fetched as mentioned in Section 3.2; sentence embeddings were constructed based on word representation features from the second-to-last layer of the *BERT-Large, Uncased* model.
ELMo	Model downloaded as mentioned in Section 3.2 and sentence embeddings were constructed based on word representation features from all three layers of the ELMo model.
DCT	Components were varied from 1 to 6, after which performance plateaued or diminished.
PCA	Components were varied from 1 to 3, after which performance plateaued or diminished.
EigenSent	There are two components to tune for the HODMD algorithm: the window, d, and the number of components to retain, n, as described in Algorithm 1. d is varied between 1, 2, 3, [1-2], [1-3] and [1-6], while n is chosen as 1 or 2.
Linear SVM	L2 regularization parameter is varied between 0.001, 0.1, 1, 10 and 100.

Table 2: Algorithm configurations tested in our experiments.

learn package[8] (Pedregosa et al., 2011). We use the p-means implementation provided by Rücklé et al. themselves[9] (2018) and leverage *Tensorflow* graphs (Abadi et al., 2016) in order to use ELMo. For BERT, we leverage a fast in-memory message-queue based implementation, *bert-as-service*[10]. Finally, for an implementation of higher-order Dynamic Mode Decomposition (HODMD), we look towards a Python implementation by Demo et al.[11] (2018).

In order to foster reproducibility and openness, all of the experimental code is released[12] and results can easily be reproduced by re-running the provided code.

4 Experiments and results

4.1 Experiments

In this work, we perform extensive experiments to compare the performance of our *EigenSent* method to the other competing methods. Our experimental protocol is described clearly next:

1. We choose an algorithm (*EigenSent*, p-means, BERT, ELMo, DCT or PCA) and a set of hyperparameter values pertaining to it (e.g., the number of components to keep in PCA, or the powers in p-means) to evaluate.
2. Then, choose a dataset (20-NG, R-8 or SST-5). For every word in every sentence in the train and test splits of the dataset, retrieve the corresponding word vector using the pre-trained model stated in Section 3.2.
3. Followed by the application an algorithm-hyperparameter combination on the sequence of word vectors constructed in the previous step to create sentence embeddings.
4. Finally, we train a simple linear-kernel support vector machine (Cortes and Vapnik, 1995) using the created sentence embeddings corresponding to the train-split of a dataset, and evaluate the trained model on the test-split, by calculating *Precision*, *Recall* and their harmonic mean, the *F1*-score.

Table 2 holds more metadata details about the experiments performed, for the purposes of reproducibility.

4.2 Results

The results of experiments, corresponding to the configurations listed in Table 2 and shown in Table 3, are analyzed next.

4.2.1 Dataset analysis

Amongst the datasets, the Reuters-8 dataset is relatively easier to tackle, as shown by the consistently high *F1*-scores across all the methods and configurations. The 20 newsgroups dataset is slightly more complex, owing to the much larger number of classes that it contains.

Finally, the Stanford Sentiment Treebank dataset is observed to be very nuanced and it is much harder to manage high scores on it. As an example of the degree of complexity of the SST-5 dataset, consider the following training sentence, which is labeled as *very positive*: ”*The entire movie establishes a wonderfully creepy mood*”. While the word *wonderfully* is usually used with a

[8]`https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

[9]`https://github.com/UKPLab/arxiv2018-xling-sentence-embeddings/blob/master/model/sentence_embeddings.py`

[10]`https://bert-as-service.readthedocs.io`

[11]`https://mathlab.github.io/PyDMD/hodmd.html`

[12]`https://github.com/DeepK/hoDMD-experiments`

DCT		20-NG			R-8			SST-5		
Cmp.	Dim.	P	R	F1	P	R	F1	P	R	F1
1	300	45.98	43.94	44.41	47.99	43.76	45.22	22.39	21.00	20.78
2	600	55.03	53.98	54.25	57.37	56.60	56.71	23.30	23.89	23.14
3	900	57.43	55.91	56.35	72.01	72.82	72.17	25.03	25.34	24.70
4	1200	59.37	58.03	58.45	82.54	83.28	82.55	**30.11**	**30.09**	**29.53**
5	1500	60.88	59.03	59.64	87.56	88.07	87.42	28.39	28.51	28.21
6	1800	**61.07**	**59.16**	**59.78**	**90.41**	**90.78**	**90.38**	28.13	27.82	27.82

PCA		20-NG			R-8			SST-5		
Cmp.	Dim.	P	R	F1	P	R	F1	P	R	F1
1	300	52.95	49.87	50.39	80.39	77.43	78.38	**26.47**	**25.08**	**25.23**
2	600	**55.43**	**54.67**	**54.77**	82.86	81.82	82.10	26.39	24.99	25.14
3	900	53.93	53.00	53.25	**83.83**	**83.42**	**83.41**	25.13	23.94	24.18

pmeans		20-NG			R-8			SST-5		
Power(s)	Dim.	P	R	F1	P	R	F1	P	R	F1
1	300	68.72	68.19	68.25	96.34	96.30	96.27	27.88	26.44	24.81
[1-2]	600	71.27	70.69	70.82	**96.69**	**96.67**	**96.65**	32.14	31.55	31.49
[1-3]	900	71.85	71.37	71.48	96.13	96.11	96.07	33.32	**33.59**	33.03
[1-6]	1800	**72.20**	**71.65**	**71.79**	96.08	96.03	95.99	**33.77**	33.41	**33.26**

Language Model		20-NG			R-8			SST-5		
type	dim.	P	R	F1	P	R	F1	P	R	F1
BoW averaging of word vectors obtained from ELMo	1024	**71.20**	**71.79**	**71.36**	94.54	91.32	92.37	**42.35**	**41.51**	**41.54**
Average of token representations from the second-to-last hidden layer of BERT	1024	70.89	70.79	70.88	**95.52**	**95.39**	**95.39**	39.92	39.38	39.35

EigenSent			20-NG			R-8			SST-5		
n	d	Dim.	P	R	F1	P	R	F1	P	R	F1
1	1	300	60.00	59.00	59.01	93.71	93.15	93.33	33.80	31.98	32.12
1	2	300	61.62	60.39	60.47	94.62	94.20	94.30	34.54	32.58	32.88
1	3	300	61.48	59.99	60.05	94.43	93.92	94.05	34.14	30.99	31.29
1	[1-2]	600	63.69	62.92	62.97	95.03	96.03	95.99	34.49	32.95	33.17
1	[1-3]	900	64.47	64.05	64.10	95.27	95.11	95.14	34.67	33.00	33.18
1	[1-6]	1800	65.74	65.03	65.17	95.81	95.70	95.70	**35.32**	**33.69**	**33.91**
2	1	600	62.12	61.23	61.24	93.30	92.78	92.93	32.44	33.13	32.19
2	2	600	63.59	62.29	62.57	93.58	93.28	93.37	33.79	31.81	32.04
2	3	600	62.72	61.39	61.70	93.23	92.50	92.70	33.65	31.43	31.62
2	[1-2]	1200	65.47	64.54	64.76	95.30	95.20	95.19	33.79	32.38	32.71
2	[1-3]	1800	66.31	65.48	65.69	**95.91**	**95.80**	**95.76**	33.75	32.38	32.75
2	[1-6]	3600	**66.98**	**66.40**	**66.54**	96.85	95.73	95.71	32.81	31.64	31.98

Table 3: This shows the results of the experiments performed with the DCT, PCA, pmeans, ELMo, BERT and *EigenSent* on the 3 stated datasets. Note that [*a*,*b*] means that the sentence embedding is the result of concatenation of the embeddings produced by varying the corresponding hyperparameter from *a* to *b*. P, R and F1 indicate the percentage values of Precision, Recall and F-score. **Bold** indicates the best result in for that particular metric (column).

positive intent, *creepy* is most often negative. It is their combination, i.e., *wonderfully creepy*, which makes the description an example of a *very positive* sentiment.

4.2.2 Analysis of the competing methods

The results obtained for the benchmark methods can be observed to follow intuition. P-means, BERT and ELMo outperform PCA and DCT-based embedding creation techniques; the two latter methods do achieve respectable results, given that they are not attuned to creating embeddings, but are simply decomposing a sequence of word vectors into components, which we use to construct embeddings. It can be seen that the DCT-based embedding creation technique needs more components to achieve reasonable performance, as compared to PCA, because PCA learns the basis vectors in a data-driven way while DCT assumes cosine functions as bases. However, since it does not need to learn the bases, and therefore makes less errors than PCA, DCT is more performant than PCA when we utilize more components.

Among ELMo, BERT and p-means, ELMo performs better on the SST-5 dataset because it takes context into account in a much more sophisiticated way (owing to the bi-directional LSTM-based language model), than p-means. The performance of BERT is in-between the two. Both BERT and ELMo have not been fine-tuned in any way, for them to be fairly comparable to the other methods discussed in this work, none of which are task-specific.

4.2.3 Analysis of EigenSent

We thoroughly analyze our proposed method, next, from various different perspectives.

Choice of higher-order DMD vs standard DMD: Observing the results of the *EigenSent* method in Table 3, it is clear that the exploiting the *higher-order* assumption (see Equation 3) is beneficial, since the results are unanimously better for higher values of the order parameter, d.

Effect of adding more dynamic modes: Recall that in Algorithm 1, the number of dynamic modes to choose n was a parameter. This determines the number of eigenvectors that are retained after performing eigendecomposition on the Koopman operator (see Section 2.1.2). It can be observed that retaining the fundamental eigenvector, or the largest mode, is enough to secure a good performance, when it comes to constructing sentence embeddings with *EigenSent*, with small improvements made with choosing the first two, at the cost of embedding dimensionality.

Performance with respect to PCA and DCT: *EigenSent*, using HODMD, is consistently superior, as compared to the other spectral techniques tested in this work. This is because it captures information about the sequential behaviour of the word vectors which form a sentence, while the other methods do not.

Performance with respect to state-of-the-art: HODMD is designed to capture the *dynamics* in a multidimensional sequence but it does not directly capture the *scale*, which methods like p-means (or simply averaging the word vectors) do. This is reflected in the performance of *EigenSent* on the datasets tested with, as its performance is somewhat between ELMo (or BERT) and p-means. For the SST-5 dataset, which exhibits more complex behaviour and interplay amongst words, it performs better than p-means (and much better than simple averaging), because of its ability to capture the dynamics, which is probably the more important attribute in this case.

Concatenating with word vector average:

	20-NG			**R-8**			**SST-5**		
method	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 $\oplus$ avg.	**72.24**	*71.62*	*71.78*	**97.18**	**97.13**	**97.14**	**42.77**	**41.67**	**41.81**

Table 4: A summary table of methods studied in this paper and the best results obtained. In addition, the final row contains the result of the embeddings constructed by concatenating the average word vector embedding with the *EigenSent* embedding. **Bold** indicates the best result for a particular metric, while *italic* is the second-best.

Query Sentence	**Best Match**	**Cosine Similarity**
The storylines are woven together skilfully, the magnificent swooping aerial shots are breathtaking, and the overall experience is awesome.	The camera soars above the globe in dazzling panoramic shots that make the most of the large-screen format, before swooping down on a string of exotic locales, scooping the whole world up in a joyous communal festival of rhythm.	0.796
What 's most memorable about Circuit is that it's shot on digital video, whose tiny camera enables Shafer to navigate spaces both large ...and small... with considerable aplomb.	The large-format film is well suited to capture these musicians in full regalia and the incredible IMAX sound system lets you feel the beat down to your toes.	0.771
George Lucas returns as a visionary with a tale full of nuance and character dimension.	The script by David Koepp is perfectly serviceable and because he gives the story some soul ...he elevates the experience to a more mythic level.	0.758

Table 5: Examples of best-matching sentences based on the cosine-similarity between the embeddings obtained using *EigenSent*

The intuitions of *dynamics* and *scale*, corroborated with the performance observed in Table 3, as explained above, led us to combine the embeddings generated by *EigenSent* with those by simply averaging the word vectors, to capture both of these properties in a sentence.

The summary of results is provided in Table 4, where the best results for each method are provided. It also has an additional result where the most performant *EigenSent*-based embeddings have been concatenated with the average word vector embedding for a sentence. It can be readily seen that this concatenation significantly improves performance, as the resultant embeddings can now capture both the *scale* and *dynamics* of a sentence.

Examples of similar sentences with EigenSent: Apart from the extensive quantitative evaluation of the proposed method, we provide motivating examples of similar sentences from the Stanford Sentiment Treebank dataset, as deemed by our method, in Table 5. It can be noted that none of the sentence-pairs share common words, apart from stop-words, and the similarity is semantic. The first example shows sentences which are similar because they both praise the camerawork in a movie, while in the second example, the commonality is about the video format. In the last example, the sentences point to movies having interesting characters, soul and depth. All of these examples suggest that *EigenSent* can capture the very nuanced qualities of a sentence.

5 Conclusion

In this paper, we have proposed a novel method to construct sentence embeddings, by exploiting the dynamic properties of a sequence of word vectors that the sentence is made up of. We do this using a spectral decomposition method rooted in fluid-dynamics, known as higher-order Dynamic Mode Decomposition, which is known to capture the fundamental transition dynamics of a multidimensional signal. Thorough empirical validation of the proposed method, which we call *EigenSent*, against known state-of-the-art methods shows the promise of this technique in capturing the dynamics of a word vector sequence to distill sentence embeddings, which may be concatenated with word vector average embeddings to state-of-the-art performance.

The main contributions of the paper are:

1. We use signal summarization as an approach for creating sentence embeddings, a first to the best of our knowledge, using an algorithm from fluid dynamics called *higher-order Dynamic Mode Decomposition (HODMD).*
2. The rationale and intuition behind using the said method to capture the dynamic properties of a sentence are motivated, and the mathematical preliminaries of HODMD in the context of constructing sentence embeddings are clearly delineated.
3. A detailed experimental validation of *EigenSent*, is performed on three public datasets, of varying degrees of complexity and purpose, and against algorithms which are both state-of-the-art and diverse, to formulate general conclusions about *EigenSent*.
4. We postulate, and later observe, that our method can successfully capture the *dynamics* present in a sentence. In cases where *dynamics* alone does not capture the essence of a sentence, our embeddings may be concatenated with those obtained via word vector averaging to obtain state-of-the-art results.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pages 265–283.

Nasir Ahmed, T. Natarajan, and Kamisetty R. Rao. 1974. Discrete cosine transfom. *IEEE Transactions on Computers*, 23:90–93.

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

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. 2016. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS ONE*, 11:1–19.

Jose Camacho-Collados and Mohammad T. Pilehvar. 2018. From word to sense embeddings: A survey on vector representations of meaning. *Journal of Artificial Intelligence Research*, 63:743–788.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report.

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 (EMNLP)*, pages 670–680.

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

Nicola Demo, Marco Tezzele, and Gianluigi Rozza. 2018. PyDMD: Python Dynamic Mode Decomposition. *The Journal of Open Source Software*, 3.

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 North American Chapter of the Association for Computational Linguistics (NAACL)*.

Godfrey H. Hardy, John E. Littlewood, and George Pólya. 1952. *Inequalities*. Cambridge University Press.

Harold Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python.

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 28*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML)*, pages II–1188–II–1196.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

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.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv:1806.06259*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report.

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

Peter J. Schmid. 2010. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28.

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.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. In *International Conference on Learning Representations (ICLR)*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.