Learning Robust Representations using a Change Point Framework

Ame Osotsi
osotsi.a@gmail.com
Penn State University
University Park, PA, USA

Qunhua Li
qunhua.li@psu.edu
Penn State University
University Park, PA, USA

ABSTRACT

This work proposes Change Point Embeddings (CAPE), word embeddings learned using a novel change-point objective framework. Based on a Bayesian change point model for time series, a unique feature of this objective function is that it trains representations to be optimal at distinguishing between contexts. We gain improved separation between context representations, and therefore achieve improved performance in the presence of domain shift - a change in the data distribution between an algorithm's training dataset, and a dataset it encounters when deployed. Without needing fine-tuning, this change point framework outperforms state-of-the-art models in clustering topics, and is also more robust against noise in topic classification. Code and trained models are available at this URL.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Information extraction**.

KEYWORDS

Word embeddings, Bayesian change-point model, robustness

ACM Reference Format:
Ame Osotsi and Qunhua Li. 2021. Learning Robust Representations using a Change Point Framework. In *MiLeTS '21: 7th SIGKDD Workshop on Mining and Learning from Time Series, August 15th, 2021, Virtual, Singapore*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Word embedding —- the mapping of words into numerical vector spaces —- is a basic building block in many natural language processing (NLP) tasks. Many embedding techniques have been developed, including pretrained embedding techniques (e.g. Word2Vec [24], GloVe [26] and fastText [23]). More recently, deep neural network based models (e.g. ELMO [27], BERT [9]) have emerged. These models are trained to encode complex word relationships, leading to improved performance in many NLP tasks. However, despite the addition of objectives aimed at understanding relationships between sentences [9], topic-level information is still underutilized in training.

We propose a novel embedding training method, change point embedding (CAPE), based on a probabilistic change-point model. The key idea of this method is to construct training passages with a known transition of topics and then model the change of topics using a change point model. By training the embeddings to optimize the detection of the change point, it incorporates topic information into the learning process, gearing the embedding towards learning topic-discriminative features.

The use of this statistical model results in optimally separated word representations. As well-separated representations are less likely to be corrupted by noise, it improves the tolerance of the representation against corruption introduced by noise. This is especially crucial for language models, as the majority of the world's languages do not have reliable textual or expert resources [11]. Even in high-resource areas, language itself changes between different domains and over time, and generalization performance is an issue in contemporary models [21, 34].

In systematic comparisons, we show that CAPE substantially improves performance in unsupervised tasks such as identification of topic transitions, including situations with domain shift, and improves robustness to noise in text classification. In addition, we also develop novel strategies for evaluating pre-trained language models in topic discovery, differentiation and robustness to noise.

2 THE MODEL

In the CAPE model, each word w_i is represented using a d-dimensional vector, θ_{w_i}, with an associated covariance matrix $\Sigma_{w_i} = \text{var}(\theta_{w_i})$. Now suppose a sentence j consists of words $w_1, \ldots, w_n$. We encode it using (x_j, Σ_j) as follows:

$$\begin{aligned} x_j &= \frac{1}{n}(\theta_{w_1} + \cdots \theta_{w_n}) \\ \Sigma_j &= \text{var}(x_j) = \frac{1}{n^2}(\Sigma_{w_1} + \cdots \Sigma_{w_n}) \end{aligned} \tag{1}$$

where

$$\Sigma_{w_i} = \begin{bmatrix} \sigma^2_{w_{i,1}} & & 0 \\ & \ddots & \\ 0 & & \sigma^2_{w_{i,d}} \end{bmatrix}_{d\times d}.$$

Here we assume all θ_{w_i} are independent, and all Σ_{w_i} diagonal to enforce independence across different dimensions of θ_{w_i}.

A linear relationship in (1) was chosen because it seemed to be crucial for the robustness of the results. Briefly, arbitrary nonlinearity (for example in the attention function) is great for its incredible flexibility in fitting a complex dataset, but it also significantly complicates interpolation/extrapolation. For example if we add a new word $\theta_{w_{n+1}}$ to the sentence j, there would be no guarantee the new

MiLeTS '21, August 15th, 2021, Virtual, Singapore

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

sentence embedding $x_{j'}$ would be close to x_j. In practice however, sentences that share many words tend to have similar meanings. Thus a linear relationship can be expected to provide stability.

We model x_j using a multivariate normal distribution [25] and use a normal prior over the unknown parameter $\boldsymbol{\mu}$ as follows:

$$\begin{aligned} x_j|\boldsymbol{\mu}, \Sigma_j &\sim \mathcal{N}(\boldsymbol{\mu}, \Sigma_j) \\ \boldsymbol{\mu} &\sim \mathcal{N}(\mu_0, \Sigma_0) \end{aligned} \tag{2}$$

To learn representations that incorporate topic-level information, we develop a training strategy that constructs passages with known transitions of topics, and formulate the learning process into a change-point detection framework. We construct and optimize an objective function, based on a Bayesian online change point model (BOCD)[3], to learn $(\hat{\theta}, \hat{\Sigma})$ that recognize and differentiate topics.

Below in 2.1 we give a brief overview of BOCD. We then define the change-point based objective function in 2.2.

2.1 An overview of the BOCD algorithm

The BOCD algorithm [3] is used to find 'change points', i.e. locations of abrupt changes in a sequence of observations, $x_1, \ldots, x_T$, in an online manner. Briefly, BOCD estimates change points by probabilistically estimating the "run length" distribution r_t, i.e. the estimated number of observations passed at time t since the last change point. If a change point occurs at t, $r_t = 0$; otherwise, $r_t = r_{t-1} + 1$, i.e. the run length grows. Specifically, denote the observations that have been seen so far as $x_{1:t} = \{x_1 \cdots x_t\}$, then the run length distribution at the t^{th} observation is $R_{t,r} = P(r_t = r|x_{1:t}) \propto P(r_t, x_{1:t})$, where $r \in [0, 1, \cdots t-1]$.

The joint distribution $P(r_t, x_{1:t})$ is modeled in terms of $P(r_{t-1}, x_{1:t-1})$ as follows:

$$P(r_t, x_{1:t}) = \begin{cases} \sum_{r_{t-1}} \frac{1}{\lambda} P(x_t|r_{t-1}, \boldsymbol{x}_t^{(r_{t-1})}; \eta_t^{(r_{t-1})}) P(r_{t-1}, x_{1:t-1}) & \\ \qquad \text{if } r_t = 0. & \\ (1 - \frac{1}{\lambda}) P(x_t|r_{t-1}, \boldsymbol{x}_t^{(r_{t-1})}; \eta_t^{(r_{t-1})}) P(r_{t-1}, x_{1:t-1}) & \\ \qquad \text{if } r_t = r_{t-1} + 1. & \\ 0 \qquad \text{otherwise.} & \end{cases} \tag{3}$$

where $\frac{1}{\lambda}$ is the prior probability of have a change point at time t and the posterior predictive distribution $P(x_t|r_{t-1}, \boldsymbol{x}_t^{(r_{t-1})}; \eta_t^{(r_{t-1})})$ is specified according to the data. Given $R_{t,r}$, one can estimate the most probable run length at t as $\text{argmax}_r R_{t,r}$. See Figure 5 for an example.

2.2 The change-point based objective function and learning framework

To learn representations that incorporate topic-level context information, we first randomly choose two different contexts from a very large corpus (e.g. two different Wikipedia paragraphs) and then concatenate a string of m observations from each context (e.g. 2 sentences from each paragraph). We then encode them using (θ, Σ). This creates a data stream with a known change point at the $(m+1)$st observation. Using BOCD, we want to realize posterior run-length distributions that reflect the correct location of the change point, so we update (θ, Σ) to improve performance at the objective function below.

$$\begin{aligned} obj = R_{1,1} + R_{2,2} + \cdots + R_{m,m} \; + \\ R_{m+1,1} + R_{m+2,2} + \cdots + R_{2m,m} \end{aligned} \tag{4}$$

Large values of (4) are achieved when the run length estimate increases over the first m observations, resets to 0, and increases again over the last m. This happens when the first m observations are represented as a compact cluster and the last m observations map to a distant compact cluster. Thus, maximizing (4) with respect to (θ, Σ) will learn representations that best differentiate contexts. See the Appendix for a brief proof. The full process is summarized in Algorithm 1.

Note that this task does not require human annotation; the paragraph information is already in most text datasets (e.g. Wikipedia).

Algorithm 1 Training algorithm

```
procedure TRAIN
    procedure INITIALIZE
        Randomly initialize the encoder parameters (θ, Σ)
    for K iterations do
        Pick 2 distinct contexts in the training set and m observations from each context.
        Encode the 2m observations using θ, Σ.                ▷ Eq 1
        Get R using BOCD and calculate the objective.         ▷ Eq 3, 4
        Improve obj (wrt θ, Σ) using stochastic gradient methods.
    return trained encoder parameters (θ̂, Σ̂)
```

3 EVALUATION OF EMBEDDINGS

In this evaluation, we evaluate the coherence of topics (3.1) and robustness to noise in topic classification (3.2). In addition, we also evaluate distances between sentence vectors (3.3), which implies discriminability between sentences.

3.1 Clustering assessment

Representations that successfully differentiate between different contexts are expected to have low separation between representations within the same contexts, and high separation between dissimilar contexts. We quantify the separation quality using modularity.

Modularity, originally developed for community detection, is designed to measure how well a graph separates into a specific partitioning [8]. Here we create a passage with $m = 3$ sentences each from 2 unique paragraphs. We represent each sentence in a data stream as a node in the graph, and define the weight of the edge between node u and node v as $A_{uv} = corr(x_u, x_v)$, where x_u and x_v are the respective encoded representations and $corr(x_u, x_v)$ is computed using Pearson correlation. We then calculate modularity, given by

$$Q = \frac{1}{|E|} \sum_{uv} \left[A_{uv} - \frac{k_u k_v}{|E|} \right] \delta(c_u, c_v), \tag{5}$$

where A_{uv} is the weight of the edge between nodes u and v, k_u and k_v are the sum of edge weights associated with nodes u and v respectively, $\delta(c_u, c_v)$ equals 1 iff u and v originate from same context, and zero otherwise, and $|E|$ is the total sum of edge weights in the graph, i.e. the sum of all values in the upper triangle of the adjacency matrix A. This measure computes the normalized difference between the observed edge weights A_{uv} and the expected weights by chance $\frac{k_u k_v}{|E|}$ for all pairs of observations from the same context, reflecting the strength of association beyond chance.

3.2 Robustness evaluation in classification tasks

The robustness of word embeddings against noise in the data is critical for generalizability and reproducibility. We evaluate the robustness of the trained word embeddings in the task of topic classification when the text to be classified contains added noise.

To proceed, we split a labeled dataset into training and test sets, and add a fixed amount of random noise (random words) to each element of the test set. After encoding the training and test sets using the trained word embeddings, we train a classifier on the training set, then run the classifier on the test set and report the classifier's accuracy as a function of the amount of added noise. (Note: when evaluating BERT, we used its internal classification capabilities.) This experiment measures how robust a classifier trained on the representation of clean data is against noisy data.

3.3 Dispersion of sentence vectors

Dispersion of sentence vectors in a dataset reflects how well sentence vectors span the embedding space. A higher value implies that sentence vectors are well separated, offering a higher tolerance to corruption introduced by noise.

To quantify vector dispersion in a dataset, we first define a normalized distance statistic. Given a collection of sentences and a sentence vector v from a dataset, we define the normalized distance from v to the collection as

$$\text{Normalized Distance}(v) = \frac{\begin{array}{c}\text{median distance from } v \text{ to other} \\ \text{sentence vectors in the collection}\end{array}}{\sqrt{\sum_i v_i^2}} \tag{6}$$

To measure the overall vector dispersion in a dataset, we sample random vector collections 100 times in a dataset, then compute the normalized distance for each collection, and report the median normalized distance.

4 EMPIRICAL STUDY SETUP

4.1 Datasets

We downloaded Wikipedia (July 2019) and cleaned it, keeping 24M paragraphs with at least 3 sentences with 4+ words each. The last 300K paragraphs were split off to make the Wikipedia test set, and the rest were used to make the training data. This training set had a full vocabulary of size 4.8M, but to save time and space we restricted training and testing to the 400K most common words in the training set. Each paragraph is treated as a distinct context.

To test the embedding's performance under domain shift, we selected Bookscorpus as another test set. Bookscorpus consists of large datasets of free books on the internet written by yet unpublished authors [16, 19, 35, 36]. Its narrative style is quite different from Wikipedia training data.

For the robustness analysis, we picked the AG News corpus [1], which consists of news articles collected from various sources online, each annotated by news type. Following [13], we concatenated each article's title and short description together (average length 42 words), and used that to predict the news type.

4.2 Training process

We maximized the objective function on the Wikipedia training set using Tensorflow's [2] implementation of an adaptive gradient descent algorithm (AdaGrad [10]). We trained 3 models corresponding to $d \in \{10, 100, 500\}$, denoted as CAPE(d), each for 7-8M iterations. The procedure took $\{39, 40, 60\}$ hours respectively on an i7 8700K CPU.

4.3 Testing process

We evaluate CAPE against three competing methods: the fastText 300-dimensional English embeddings (2018) [23], the 300-dimensional Wikipedia+Gigaword GloVe vectors [26], and the 1024 dimensional BERT embeddings (BERT-Large, Uncased, Whole Word Masking), extracted using bert-as-service [32]. All embeddings used the same average of vectors encoder model. We run the following analyses:

(A) We randomly sample 1000 paragraphs from Wikipedia and Bookcorpus respective test sets and compute normalized distance (3.3) to evaluate vector dispersion in these two datasets. For each dataset, we also construct passages with 2 distinct contexts and $m = 3$ sentences per context and calculate modularity (3.1) to evaluate topic discrimination (avg runtime: < 5m).

(B) We run the classification test (3.2) on the AG news dataset, evaluating resistance to added noise. We train a SVM classifier on the clean data, and add a fixed amount (0-100) of random noise words at random positions to each article in the test set. We then assess the classifier's accuracy for classifying the article's news type as a function of the amount of noise (avg runtime: < 10m per test).

5 RESULTS AND DISCUSSION

5.1 CAPE generates better topic discrimination and higher vector separation in the test sets

We evaluated the modularity and normalized distance on the test sets of Wikipedia passages and Bookcorpus passages (without fine tuning). Table 1 shows a numerical summary of the results. Figures 3-4 in Appendix B show the full distributions.

CAPE has significantly higher median modularities than the other methods, suggesting that it clusters paragraphs (i.e. topics) much more coherently. It also has significantly higher median normalized distance. This suggests that it has better separated sentence embedding, indicating a higher efficiency to span the embedding space. It is also observed that its modularities and normalized distance increase with the increase of its dimension. Though

Bookscorpus has very different narrative style from the training set (Wikipedia), CAPE's superiority is maintained. In contrast, BERT does not seem to transfer well to these tasks.

	Modularity		Normalized distance	
	Wikipedia	Bookscorpus	Wikipedia	Bookscorpus
CAPE (10)	0.304	0.163	1.173	0.910
CAPE (100)	0.396	0.286	1.300	1.243
CAPE (500)	**0.465**	**0.376**	**1.389**	**1.343**
Glove	0.128	0.107	0.714	0.652
FastText	0.146	0.124	0.820	0.763
BERT	0.097	0.094	0.537	0.534

Table 1: Summary of clustering evaluation results in Wikipedia and Bookscorpus. The medians of modularity and normalized distance are reported. No fine tuning was applied to Bookscorpus.

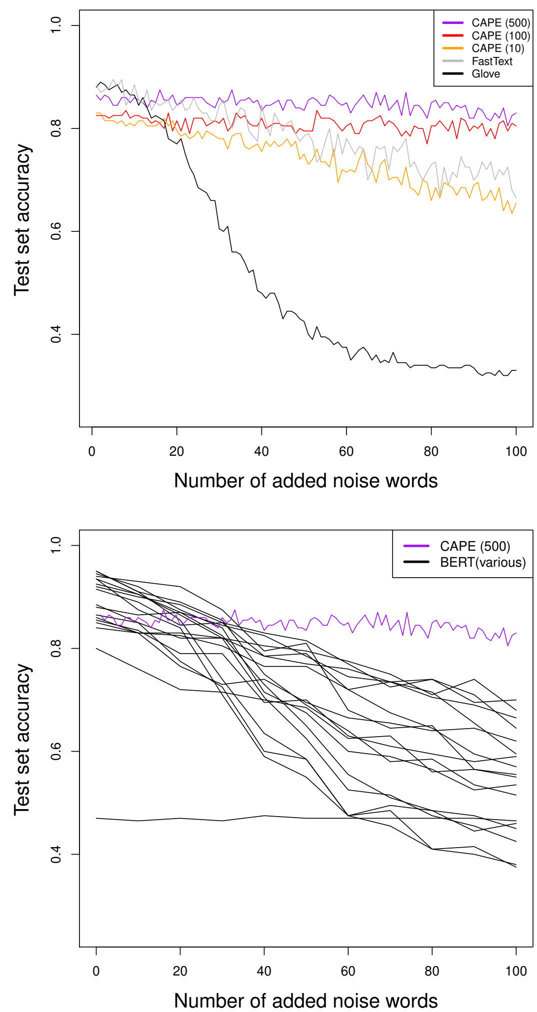


Figure 1: Accuracy in text classification as a function of added noise in the AG news dataset. Each line shows how a classifier trained (or fine-tuned) on a clean training set performs when evaluated on a test set with added noise. Top: CAPE, FastText and GloVe. Bottom: BERT with various fine-tuning settings.

5.2 CAPE is robust to noise in text classification

Next, we evaluated CAPE in text classification using the AG news data. We focused on the robustness of the embeddings against the distortion detailed in Section 3.2. Note that we only trained the classifier and did not fine-tune the encoder. When training BERT, we ran a grid search over the fine-tuning options recommended in the BERT paper (Appendix 3), varying batch size, learning rate and number of epochs.

As shown in Fig 1 (top), though all the three word embedding methods have broadly similar performance when no noise is added, the CAPE text classification has a much smaller accuracy reduction when noise is added. BERT's results (Fig 1 bottom) vary across fine-tuning settings. The overall relationship between their classification accuracy and the amount of added noise largely follow the general pattern as for FastText and GloVe, showing a larger accuracy reduction with added noise than CAPE. This indicates that CAPE is much more robust against this noise model than all the other three methods. With higher model dimension, CAPE shows improved classification accuracy in both the cases with and without added noise. CAPE's robustness is likely due to its property of generating more separated sentence vectors (5.1). Larger distance between sentence vectors offers higher tolerance to added noise.

6 CONCLUSION

In this paper we propose a novel change-point based framework for training and evaluating embeddings. CAPE, with its distributional assumptions and optimized separation through a change-point objective, strongly outperforms conventional models such as fastText, GLoVe or BERT at topic discrimination in the unsupervised setting. CAPE vectors are easily integrated into a statistical model.

The CAPE framework also demonstrates robust performance in topic classification when noise is present. We attribute CAPE's robustness to the linearity of the encoder, and the improved separation between the word embeddings, as seen using modularity and normalized distance. Other methods have closely packed vectors that are more likely to overlap, so they are more sensitive to distortions in the texts. Robustness is essential for generality and reproducibility of word embeddings in NLP tasks, as such tasks often involve changing language environments and adaptability to different contexts is highly desirable.

REFERENCES

[1] [n.d.]. AG News corpus. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html. Accessed: 2020-02-06.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning.. In OSDI, Vol. 16. 265–283.

[3] Ryan Prescott Adams and David JC MacKay. 2007. Bayesian online changepoint detection. arXiv preprint arXiv:0710.3742 (2007).

[4] Samaneh Aminikhanghahi and Diane J Cook. 2017. A survey of methods for time series change point detection. Knowledge and information systems 51, 2 (2017), 339–367.

[5] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. The Journal of Machine Learning Research 13, 1 (2012), 281–305.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020).

[7] Michael Byrd, Linh Nghiem, and Jing Cao. 2017. Lagged Exact Bayesian Online Changepoint Detection. arXiv preprint arXiv:1710.03276 (2017).

[8] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. 2014. Community detection via maximization of modularity and its variants. IEEE Transactions on Computational Social Systems 1, 1 (2014), 46–65.

[9] 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 (2018).

[10] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12, Jul (2011), 2121–2159.

[11] Ewan Dunbar, Xuan Nga Cao, Juan Benjumea, Julien Karadayi, Mathieu Bernard, Laurent Besacier, Xavier Anguera, and Emmanuel Dupoux. 2017. The zero resource speech challenge 2017. In 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). IEEE, 323–330.

[12] Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. arXiv preprint arXiv:1602.03483 (2016).

[13] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? natural language attack on text classification and entailment. arXiv preprint arXiv:1907.11932 (2019).

[14] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361 (2020).

[15] Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014).

[16] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. arXiv preprint arXiv:1506.06726 (2015).

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

[18] Jeremias Knoblauch and Theodoros Damoulas. 2018. Spatio-temporal Bayesian On-line Changepoint Detection with Model Selection. arXiv preprint arXiv:1805.05383 (2018).

[19] Sosuke Kobayashi. 2018. Homemade BookCorpus. https://github.com/soskek/bookcorpus.

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

[21] R Thomas McCoy, Junghyun Min, and Tal Linzen. 2019. Berts of a feather do not generalize together: Large variability in generalization across models with similar test set performance. arXiv preprint arXiv:1911.02969 (2019).

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013).

[23] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).

[24] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 746–751.

[25] Kevin P Murphy. 2007. Conjugate Bayesian analysis of the Gaussian distribution. def 1, $2\sigma 2$ (2007), 16.

[26] 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). 1532–1543.

[27] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018).

[28] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of EMNLP (Austin, Texas). Association for Computational Linguistics, 2383–2392.

[29] Ryan D Turner, Steven Bottone, and Clay J Stanek. 2013. Online variational approximations to non-exponential family change point models: with application to radar tracking. In Advances in Neural Information Processing Systems. 306–314.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv preprint arXiv:1706.03762 (2017).

[31] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In the Proceedings of ICLR.

[32] Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

[33] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237 (2019).

[34] Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. 2019. Learning and evaluating general linguistic intelligence. arXiv preprint arXiv:1901.11373 (2019).

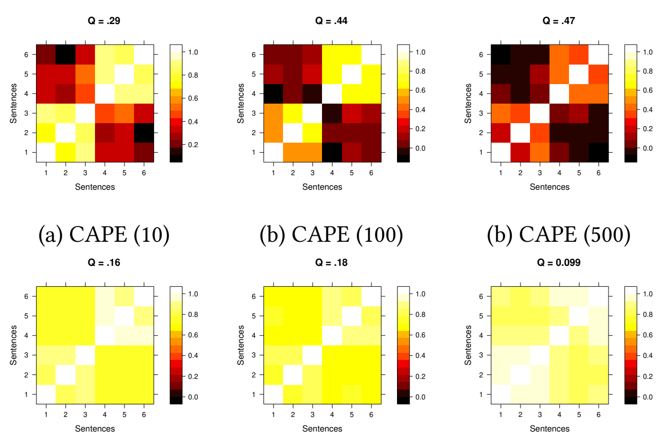
[35] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In arXiv preprint arXiv:1506.06724.

[36] 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 The IEEE International Conference on Computer Vision (ICCV).

A SEPARATION BETWEEN TOPICS IN AN EXAMPLE PASSAGE

Table 2: A sample passage of six sentences with two topics

Topic: Feral cats
A feral cat is a cat that lives outdoors and has had little or no human contact
They do not allow themselves to be handled or touched by humans, and will run away if they are able
They typically remain hidden from humans, although some feral cats become more comfortable with people who regularly feed them
Topic: Atlanta
Atlanta is the capital of, and the most populous city in, the US state of Georgia
With an estimated 2017 population of 486,290, it is also the 39th most populous city in the United States
The city serves as the cultural and economic center of the Atlanta metropolitan area, home to 5.8 million people and the ninth largest metropolitan area in the nation.



(a) CAPE (10) (b) CAPE (100) (b) CAPE (500)

(d) GloVe (e) fastText (f) BERT

Figure 2: Heatmaps show the correlations between all pairs of sentences from Table 2.

As an illustration, we compare the models on a Wikipedia passage (Table 2). The first three sentences are about feral cats, and the next three are about the city of Atlanta. Figure 2 shows the results.

In this example, CAPE achieves a much higher modularity score (Q = .29, .44, .47) than the other three word embedding methods (Q = 0.18, 0.16, and 0.14 for fastText, GloVe, and BERT, respectively), suggesting that CAPE has much better topic coherence and topic separation. CAPE also shows improved performance with increasing dimension.

B BOXPLOTS OF MODULARITY AND NORMALIZED DISTANCE IN WIKIPEDIA AND BOOKSCORPUS TEST SET

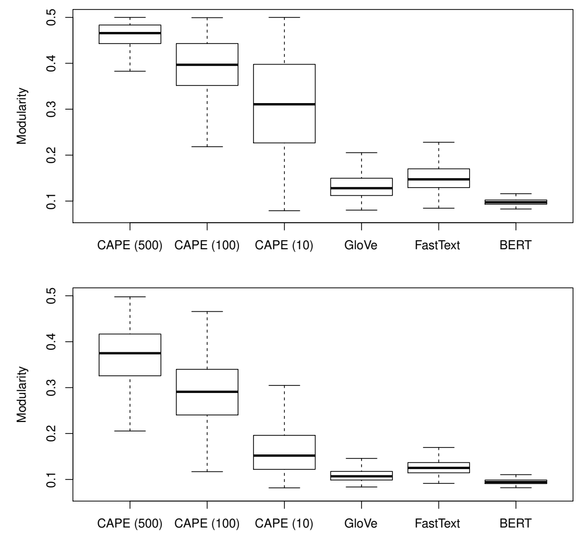


Figure 3: Distribution of modularity values in the Wikipedia (top) and Bookscorpus (bottom) test set.

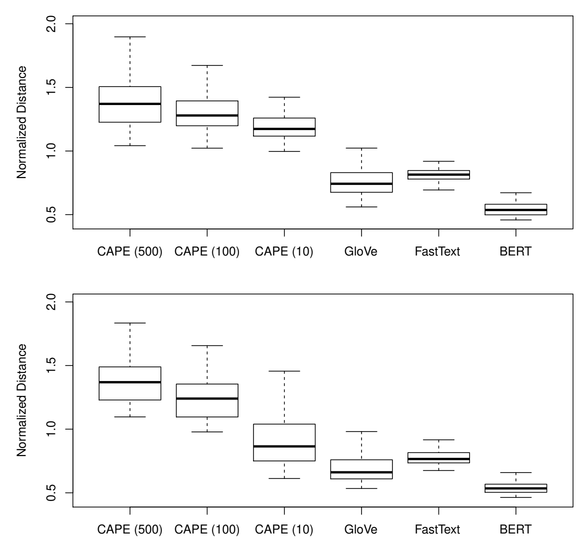


Figure 4: Distribution of normalized distance values in the Wikipedia (top) and Bookscorpus (bottom) test set.

C ADDITIONAL BOCD DETAILS

To estimate $R_{t,r}$, the algorithm processes the data stream in an online fashion. For each newly observed data point x_t, the algorithm uses the previously computed $P(r_{t-1}|x_{1:(t-1)})$ and the current data point x_t to estimate $P(r_t|x_{1:t})$ for all the possible run lengths ($r_t \in [0, 1, \cdots, t-1]$). The algorithm then updates all the parameters to take into account x_t, in the standard Bayesian manner, and the posterior is used as a prior for the next observed data point. When $P(x_t)$ is chosen to be a member of exponential family and conjugate priors are used, the parameters can be updated efficiently by conjugacy. Afterwards the algorithm returns the estimated run length probabilities, $R = \{R_{t,r} : t \in [0, 1, \cdots T], r_t \in [0, 1, \cdots t-1]\}$ and the estimated most probable change points. As $t^2/2$ $R_{r,t}$'s need to be computed, the computational complexity of BOCD is $O(t^2)$.

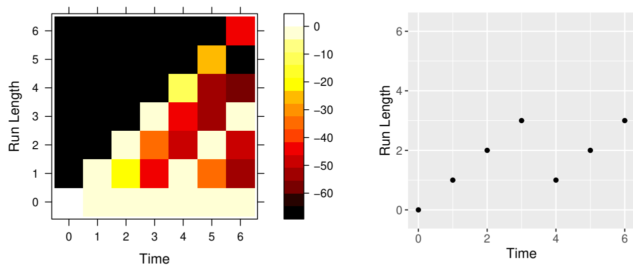


Figure 5: Visualization of the run length distributions for a simulated example of 6 observations ($m = 3$), with a change point at the 4th point. Left: Estimated log run-length probabilities, R. The color of the square at (t, r) corresponds to $\log P(r_t = r|x_{1:t})$. Right: Maximum a posteriori (MAP) run length estimates, $\max_r R_{t,r}$, which suggests the data can be divided into two segments: $\{x_1, x_2, x_3\}$ and $\{x_4, x_5, x_6\}$.

D ADDITIONAL MODEL SPECIFICATION AND HYPERPARAMETER OPTIMIZATION

By standard Bayesian conjugate computation in (2), the posterior predictive distribution $P(x_t|r_{t-1}, \boldsymbol{x}_t^{(r_{t-1})}, \eta_t^{(r_{t-1})})$ in (3) is a multivariate Normal distribution with run-specific parameters $\eta_t^{(r)} = \{\mu_t^{(r)}, \Sigma_t^{(r)}\}$. Below we give the initial parameters $\eta^{(0)} = \{\mu^{(0)}, \Sigma^{(0)}\}$ and Bayesian update procedures [25]. As in [3], we start with $R_{00} = P(r_0 = 0) = 1$.

$$\mu^{(0)} = (0, 0 \cdots 0)_d$$

$$\Sigma^{(0)} = \begin{bmatrix} (\sigma^2)^{(0)} & & \\ & \ddots & \\ & & (\sigma^2)^{(0)} \end{bmatrix}_{d\times d} \tag{7}$$

$$\Sigma_{t+1}^{(r+1)} = ((\Sigma_t^{(r)})^{-1} + \Sigma_t^{-1})^{-1}$$

$$\mu_{t+1}^{(r+1)} = \Sigma_{t+1}^{(r+1)}(\Sigma_t^{-1}x_t + (\Sigma_t^{(r)})^{-1}\mu_t^{(r)})$$

In this model the hyperparameters are

- The embedding size, d
- The number of sentences being compared, m
- The optimization algorithm (gradient descent variety), and its learning rate
- The batch size for gradient descent
- The strategy used to initialize (θ, Σ). We used a random-uniform initialization, where the ranges are hyperparameters.
- $(\sigma^2)^{(0)}$, the prior variance within BOCD.
- The constant hazard parameter λ within BOCD.

In this paper we present experimental results for $d \in \{10, 100, 500\}$. The number of sentences compared at each iteration was fixed at $m = 2$. Experiments with $m = 3$ found no appreciable difference except the models took longer to converge. The batch size was set at 25 for all experiments. Experiments showed that AdaGrad [10] was the best optimization algorithm. The parameter λ was fixed at 10. Preliminary experiments show that the choice of λ too made little difference.

For each d, the other hyperparameters were tuned using random search [5]. We picked random sets of hyperparameters, ran the algorithm for $5e4$ iterations with each set, then picked the hyperparameter set with the highest objective value at the end.

Other architectural choices include the preprocessing and tokenization strategies in Section 4.1. We used fairly straightforward tokenization techniques.

E SEPARATION OF EMBEDDING VECTORS REPRESENTING DIFFERENT CONTEXTS

We outline a brief argument to show that the optimization of (4) will separate vectors representing different contexts.

As described earlier in the setup of (4), where $m = 3$, the true run length increases over the first three observations, then resets to 0 and increases again over the last three observations. This means $R_{3,3} = P(r_3 = 3|x_{1:3})$ and $R_{4,1} = P(r_4 = 1|x_{1:4})$ will be maximized. Hence, $R_{4,4} = P(r_4 = 4|x_{1:4})$ will be minimized, since $\sum_{r=1}^4 R_{4,r}$=1. Consequently, $P(r_4, x_{1:4})$ will be small and $P(r_3, x_{1:3})$ will be large. By Eq 3 (middle case) for $t = 4$, $P(x_4|r_3 = 3, \boldsymbol{x}^{(3)} = x_{1:3}; \eta_4^{(3)})$ must be very small.

Based on the conjugate specification in (2) and (7), the posterior predictive distribution $P(x_4|r_3 = 3, \boldsymbol{x}^{(3)} = x_{1:3}; \eta_4^{(3)})$ is a multivariate Normal distribution with parameters found through conjugate Bayesian updating [25] over $x_{1:3}$ and $\Sigma_{1:3}$. It takes a small value when its kernel $(x_4 - \mu_4^{(3)})^{\mathrm{T}}(\Sigma_4^{(3)})^{-1}(x_4 - \mu_4^{(3)})$ is large. Note that,

$$\begin{aligned} (x_4 - \mu_4^{(3)})^{\mathrm{T}}(\Sigma_4^{(3)})^{-1}(x_4 - \mu_4^{(3)}) & \quad \text{Mahalanobis distance} \\ = (x_4 - \mu_4^{(3)})^{\mathrm{T}} P\Lambda^{-1}P^T(x_4 - \mu_4^{(3)}) & \quad \text{eigendecomposition} \end{aligned} \tag{8}$$

$$= \boldsymbol{b}^T\Lambda^{-1}\boldsymbol{b} \qquad \boldsymbol{b} = P^T(x_4 - \mu_4^{(3)}) \tag{9}$$

$$= \sum_i \frac{b_i^2}{\lambda_i} \tag{10}$$

So as $R_{4,4}$ is minimized through gradient descent, the kernel above is maximized, and it is preferential to move x_4 away from $\mu_4^{(3)}$ primarily in the direction associated with the smallest eigenvalue λ_i of $\Sigma_4^{(3)}$, because that yields the greatest increase in statistical (Mahalanobis) distance. This eigenvector is orthogonal to all the other eigenvectors, which together capture the major sources of variation in the distribution fit on $x_{1:3}$. Thus, the estimated x_4 is nearly orthogonal with $x_{1:3}$. As the algorithm converges, the representations of dissimilar contexts are distant and decorrelated.

Similar reasoning shows that representations in similar contexts (i.e. each of $\{x_1, x_2, x_3\}$, and $\{x_4, x_5, x_6\}$) will be close in Mahalanobis (and Euclidean) distance, and therefore more correlated. Together, this indicates that the optimization of (4) will separate vectors representing different contexts.

F RELATED WORK

F.1 Text representation

Due to their ability to capture syntactic and semantic information in words, pretrained word vectors are a core component in current NLP architectures. Numerous embeddings are available, for example Word2Vec [22, 24], GloVe [26] and fastText [23]. Word2Vec and FastText are trained using CBOW, a word prediction strategy, while GLoVE is trained on a combination of global word-word co-occurrence statistics and local context window methods. One drawback of these techniques is that the embedding representation is context independent, i.e. the vector representing a word stays fixed regardless of surrounding words. This makes them less adaptive to changes in the language environment.

More recently, deep neural network encoders have been shown to outperform embeddings across a broad range of diverse NLP tasks [28, 31]. These models, such as BERT [9], RoBERTa [20], XLNet [33], GPT-3 [6], demonstrate the efficacy of transformer models [30], in combination with millions to billions of parameters, extensive computational resources, and large amounts of training data [14].

By training them using word prediction strategies [17][12] in a large general corpus, these techniques generate contextual representations of input tokens that are infused with information of its neighborhood. The same word can have very different representations in different situations, thus taking context into account. After training, the full neural network models that were used to train these embeddings can be used for downstream tasks, unlike pre-trained embeddings which require additional structures (e.g classifiers) to make predictions. Relatively simple fine-tuning can be performed on these general models with task-specific data to improve performance in downstream tasks [15].

F.2 Change point analysis

Change point analysis is a powerful tool for detecting whether any changes take place in a data stream, and to locate where the changes occur [4]. Bayesian Online Changepoint Detection algorithm (BOCD) [3] identifies change points by modeling the probability distribution of the "run length", i.e. the elapsed time since the most recent change point. By leveraging conjugate-exponential models, it achieves efficient, exact and online Bayesian inference. It has been applied and extended in numerous ways, for example to include variational inference for non-exponential distributions [29], time lags that use future information to better detect change points [7], and non-stationary spatio-temporal processes [18].