Embedded Representations of Linguistic Style

Jonathan Bleiberg
Yale University / Department of Applied Mathematics
jonathan.bleiberg@yale.edu

Abstract

While various methods of document embedding have been established, existing methods focus largely on the semantic content of a text. This paper explores three methods for embedding the linguistic style of a text in a meaningful representation space. One of these methods relies on the hardcoding of embedding dimensions with metrics often used to evaluate a document's style, while the other two are neural network based algorithms that learn a distributed representation of document style. Hardcoding currently appears to perform the best of the three algorithms presented, however with additional fine tuning, synonym perturbation shows promise for even better performance.

1 Introduction

In recent years, considerable attention has been devoted to the establishment of methods that generate meaningful distributed representations of the semantic content of a document. The adoption of the most popular of these methods, Doc2Vec, can drastically improve the performance of simple logistic regression classifiers on sentiment analysis tasks while also serving as a viable method of information retrieval (Le and Mikolov, 2014). However, comparatively little attention has been paid to representing the linguistic style of a given text, irrespective of its semantic content.

In part, this may be a result of the inherent difficulty of defining the rather intuitive notion of linguistic style formally enough for the demands of computational linguistics. In this paper, we adopt the definition of "style as option" forwarded by J.R. Walpole (Walpole, 1980), which has seen some limited application in the field of computational stylistics (Khosmood and Levinson, 2008), and adapt it to generate vector representations of style.

Under the "style as option" framework, we take the linguistic style of a text to mean the active choice to employ one word, phrase, grammatical structure, etc. in the face of competing viable alternatives. For example, taking the case of diction, consider the sentence:

> "My neighbor's cat is bad."

In this sentence, the author could equally well have employed a different negative adjective to describe the cat, such as "terrible", "awful", or "unpleasant". Similarly, the author could have used a different noun to describe the cat, such as the more general term "pet" or the more formal term "feline", we arrive at the stylistically distinct sentence:

> My neighbor's feline friend is unpleasant.

Notably, while the style of the two sentences is easily distinguishable to a fluent reader of English, with the second assuming a far more formal character, the semantic content of the two sentences is almost identical. In this paper, we consider various approaches to isolating these authorial decisions which define linguistic style, while attempting to avoid the confounding variable of document content.

Much as with word embedding systems, the generation of stylistic embeddings serves a number of key purposes. For one, the majority of state of the art neural network based machine learning algorithms rely on a fixed length vectorized input (Le and Mikolov, 2014). If the dimensions of these input vectors are imbued with a some distributed representation of significance, it will

greatly improve the performance of the resulting model over that of a one hot encoding (Mikolov et al., 2013). Moreover, one can use the generated stylistic embeddings as the basis for a stylistic similarity metric (typically based on cosine similarity). These similarity relationships can be visualized in the 2D (or 3D) plane via the use of dimensionality reduction algorithms such as PCA and TSNE. Finally, stylistic embeddings serve as a bridge to the more difficult task of linguistic style transfer, on which even state of the art models currently receive less than satisfactory results. In this paper, we focus mainly on the second of these purposes and provide some suggestions for ways to improve existing language models with a stylistic embedding component and for possible future directions for work in linguistic style transfer.

2 Related Work

To the best of our knowledge, no existing papers focus on the generation of embeddings that exclusively concern a document's style. (Pavlick and Nenkova, 2015) develop a system of style scores for specific words, however, they base these scores on human judgements, raising scalability issues. As mentioned above, (Le and Mikolov, 2014) developed a system of encoding documents as embedding vectors, however, they did not focus on document style. Some work has also been done in the field of linguistic style transfer. (Khosmood and Levinson, 2008) rely on a manual approach, with set rules to transfer a text from one style to another. More recently, RNN based models have been applied to the task (Zhao et al., 2018), however, existing models tend to garble most inputs into meaningless phrases.

3 Algorithms

We present three algorithms for the generation of linguistic style embeddings, employing both hardcoding and neural network methods for training. One of these algorithms (partially as a semantic focused embedding that acts a standard of comparison) is a simple reinterpretation of Doc2Vec, while the other two, to the best of our knowledge, are novel approaches to the task of embedding generation.

3.1 Hardcoded Embeddings

To generate hard coded embeddings for linguistic style representation, we rely on a series of statistics that have been used by other authors to characterize text style. These include:

- Word level statistics: average characters per word, average syllables per word
- Sentence/Paragraph level statistics: average words per sentence/paragraph, average characters per paragraph, average sentences per paragraph
- Document level statistics: average word frequency, percentage of word hits in the `pyenchant` English dictionary, frequency of words over 6 characters long, frequency of words over two syllables long

These dimensions, adapted from the metrics used for style classification and transformation in (Khosmood and Levinson, 2008), represent the input statistics necessary to reconstruct existing measures of document style (in particular document formality), such as the Kincaid formula, ARI, Flesch Reading Ease, SMOG grade level, etc.

In our approach, each statistic, calculated over the whole of a document deemed to have uniform stylistic features, represents one dimension of a document's style embedding. When aggregated into a single vector, this gives a 10 dimensional embedding vector for each document. These vectors can then be used as is to represent a document style, or, perhaps more usefully, put through a dimensionality reduction algorithm and then visualized in fewer dimensions.

For this approach in particular, we recommend the use of dimensionality reduction techniques like PCA that identify the dimensions of maximum variance in order to generate a meaningful embedding. Given the highly supervised nature of this approach, it is likely that a number of the statistics compiled will not differ significantly between all sets of documents. As such, linear dimensionality reduction techniques can be a helpful tool for identifying the most important dimensions within a given corpus.

3.2 Doc2Vec

For Doc2Vec, we use the standard Distributed Memory of Paragraph Vectors (PV-DM) model of document embedding (Le and Mikolov, 2014). Each "document" represents a corpus of stylistically similar texts, which will together be assigned one embedding.

Under the PV-DM framework, a sequence of words from a given document and that document's id are each input to an embedding matrix. The resulting vectors are then concatenated or averaged, and finally fed to a softmax output layer. The gradients of the network are then calculated via backpropagation and the network is trained via stochastic gradient descent to output the next word in the sequence. As the network is trained, it begins to learn features of each document useful for the prediction task. One can then extract the document embedding matrix and interpret the weights as a set of document embeddings.

While Doc2Vec is currently a reasonably well established technique, the contribution of this paper is to reinterpret the Doc2Vec model under the "style as option" framework. Since Doc2Vec models are optimized for the task of predicting the next word in a sequence given an input document, each document embedding contains useful information about the next word that an author might pick in a given word context. This is equivalent to knowing some information about an author's diction, which, as demonstrated above, is a valuable indicator of document style. In particular, each embedding represents information related to a document's tendency to select one word rather than another in a specific context, which represents a specific instance of a stylistic choice under the "style as option" framework.

3.3 Synonym Perturbation

Finally, we present a new algorithm for learning specifically stylistic rather than content sensitive document embeddings. The technique relies on an underlying neural network architecture similar to that of the Doc2Vec PV-DM model, however, we optimize for a different prediction task to decrease the impact of document content on the resulting embeddings. In particular, we train a small neural network to predict a target word from a selection of context words and a synonym of the target word. This task aligns naturally with the style as option framework, for the network is trained to learn which synonym an author writing in a given style will choose over another in a given context. Moreover, by including the synonym in the prediction task, we hope to mitigate the effect of a document's semantic content on its embedding.

Specifically, the algorithm is as follows. Given a corpus C of documents $d_1, ..., d_n \in C$ and a vocabulary of words $w_i \in V$ the vocabulary, we perform the following steps:

Data: Documents $d_1, ..., d_n \in C$, each of uniform style; A synonym mapping $S : V \mapsto \mathcal{P}(V')$
Dataset preparation:
for $d_i \in C$ **do**
  for $w_j \in d_i$ **do**
    Compute the synonym set $S(w_j)$ for the target word w_j
    Randomly select a word $w' \in S(w_j)$
    Store the words w_{j-1}, w_{j+1} immediately before and after w_j along with w' and the document id i as the input data $(w_{j-1}, w', w_{j+1}, i)$
    Store w_j as the corresponding label
  end
end

Having prepared the data in this way, we then train a neural network on the data points $((w_{j-1}, w', w_{j+1}, i), w_j)$ as follows:

Training:
for *iter* ***in*** *num_iterations* **do**
  Pass the words w_{j-1}, w', w_{j+1} and the document id i to embedding matrices (the words share the same embedding matrix, while the documents have their own embedding matrix)
  Concatenate the resulting embedding vectors into a combined vector V
  Optionally pass V through a fully connected hidden layer
  Pass V (or the output of the hidden layer) through a softmax layer and compute the cross entropy loss between the output and the true label w_j
  Calculate gradients via backpropagation and perform gradient descent update on the network weights
end
return *The document embedding matrix*

This method has a number of hyperparameters that must be tuned properly in order for the model to succeed, including the context window size, the inclusion and/or size of the hidden layer, and the possibility of using pretrained word embeddings to initialize or obviate the need for the word embedding matrices. The most important hyperpa-

rameter however, is the specification of the synonym mapping S. Given the lack of a standardized synonym dictionary, we are forced to rely on a variety of heuristics to generate synonyms. One simple and effective approach is to leverage pretrained embeddings to generate synonym candidates. In particular:

Data: A set of pretrained embeddings $E : V \mapsto \mathbb{R}^p$; target words $w_1, ...w_n \in W$
for $w_i \in W$ **do**
  Compute the cosine similarity between w_i and $w' \in V$ for every word in the vocabulary:

$$\text{Sim}(w_i, w') = \frac{E(w_i) \cdot E(w')}{\|E(w_i)\| \, \|E(w')\|}$$

  Randomly select a word $w^\star$ from the three most similar vectors to w_i
  return $w^\star$
end

Though only performed once as a pretraining step, this algorithm does take $O(|W||V|)$ time, which can be computationally expensive if the document or embedding vocabulary sizes are large. One could potentially speed up the computation of synonyms by dividing the words in the embedding vocabulary into batches of size b and calculating cosine similarity only for the words within a given word's batch. This reduces the computational cost to $O(|W|b)$, at the cost of potentially less accurate synonym assignments.

Alternatively, one could also use an existing synonyms dictionary such as the `wordnet` dictionary availiable in the `nltk` library. This may speed up computation and potentially result in better synonyms.

4 Methodology

For each algorithm, we followed a similar methodology to evaluate their performance. We first trained each algorithm separately on both the Brown corpus and the selection from the `gutenberg` corpus available in the `nltk` python library. For the Brown corpus, each genre is treated as a single stylistic document, while for the `gutenberg` corpus, each book is treated as a single stylistic document (Note that due to time and computation constraints, we restricted our analysis to the first 5,000-10,000 words of each document for the synonym perturbation implementation). We then applied the PCA and TSNE (with perplexity parameter set to 2, due to the small number of datapoints) algorithms as implemented in the popular `sci-kit learn` framework to reduce the embeddings to a 2D representation. Finally, we visualize these embeddings in 2D space using the `matplotlib` library and evaluate the similarity groupings of the documents. Afterwards, we computed the three most similar (by cosine similarity) stylistic documents to the adventure and news subsets of the Brown corpus.

The statistics used in the hardcoded embeddings were collected from functions built on top of the `nltk corpus` object, available on the project's github repository. Syllable counts were derived from the `pyphen` library's English hyphenation dictionary and dictionary hits (a rough measure of the percentage of proper nouns and rare words) were calculated based on presence in the `pyenchant` American English dictionary.

The Doc2Vec embeddings were trained on the corpora via the standard PV-DM implementation available in the `gensim` library using an embedding size of 50.

For the synonym perturbation model, synonyms were obtained by first checking whether a synonym exists in the `nltk wordnet` corpus. If a synonym was available, we randomly selected from the list of synonyms provided. Otherwise, we randomly selected from the top 3 most similar words to the target word in the pretrained `'glove-wiki-gigaword-100'` available in the `gensim` library. Approximately one-fifth of the synonyms were derived from the `wordnet` dictionary, while the rest were derived from the pretrained embeddings.

We experimented with a variety of architectures for the synonym model, though to have a uniform standard of comparison, all models were first trained using an Adam optimizer for $\sim$5 epochs. In all cases, we precomputed the word embedding input activations using the `'glove-wiki-gigaword-100'` embedding set in order to save training time. Inputs were fed through a hidden layer of various sizes, then to a softmax output and compared to the true target words via the categorical entropy loss function available in Keras. The final stylistic document embeddings computed were of dimension 50.

Model	Adventure	News
Hardcoded	*romance*	*reviews*
	mystery	*editorial*
	fiction	*hobbies*
Doc2Vec	*romance*	*government*
	government	*editorial*
	mystery	*adventure*
Synonyms	*government*	*fiction*
	humor	*learned*
	lore	*humor*

Table 1: Top three most similar genres in the Brown corpus for each embedding model for adventure and news.

5 Results

5.1 Embedding Plots

For each model, we obtained both a TSNE plot and a PCA plot. Overall, the TSNE plots appeared more informative (and less cluttered), though of course great care must be taken when interpreting the results of a non-linear dimensionality reduction technique. We display the generated plots in Figure 1.

5.2 Similarity Task

For each model, we also obtained the three most similar documents to the adventure and news sections of the Brown corpus. Results are summarized in Table 1.

5.3 Other Metrics

For the synonym perturbation method, we were also able to use validation accuracy on a the synonym selection task to evaluate the succes of different model architectures. Validation accuracies were calculated on a held out set composed of 20% of the combined Brown/Gutenberg corpus dataset (to save time, only the first 10,000 words of each file/category were used in training and testing). The best such accuracy obtained was 48.89%, using a hidden layer with 75 nodes, dropout with a keep probability of 0.3, and batch normalization. Considering the impossible nature of the synonyms task (e.g. the input trigram "the bad cat" could just as easily have a target label of "terrible" and "horrible"), 48.89% accuracy is quite respectable performance.

6 Discussion

6.1 Analysis of Results

In the absence of a standardized metric for evaluating style embeddings like the analogy test often used in word embeddings, we are forced to rely on subjective evaluations of the plots and the genre similarity task.

Nonetheless, it appears that hardcoding currently performs better than the other algorithms. Analyzing Figure 1, it appears that hardcoding does an excellent job of segmenting the authors in the Gutenberg corpus, with all texts by the same author are placed into the same cluster. If we assume that style varies more between authors than between works by the same author, this is a highly encouraging sign. For the most part, the hardcoded embeddings also place the genres of the Brown corpus in reasonable groupings, with learned and belles-lettres together, and fiction, mystery, romance, and adventure forming a single cluster. The hardcoded embeddings also perform quite well on the genre similarity task, correctly placing news near editorials and adventure near romance, mystery, and fiction.

By comparison, the Doc2Vec embeddings perform slightly less well. It places most, but not all of the works by the same author in the same region of the embedding space (see the work by Chesterton in the bottom left corner). Some of the clusters generated for the Brown corpus are also bizarre, with reviews and romance lumped together. On the whole, however, the clusters generated are reasonable. On the similarity task, strangely, government is similar to both news and adventure, though the other two similar genres for both adventure and news seem reasonable. It is possible that this somewhat erratic behavior stems from the content dependence of Doc2Vec. Documents that have many similarly words by mere virtue of having some overlapping topics will be placed too close together by the Doc2Vec model (Note that with the chosen metrics, the hardcoded embeddings are content blind).

The synonym perturbation embeddings perform decently on the clustering task for the Brown corpus, with most of the generated clusters grouping together typically stylistically linked genres (adventure and fiction being the two exceptions). However, the synonym perturbation embeddings perform poorly on author clustering in the Gutenberg corpus, with the three Austen texts spread

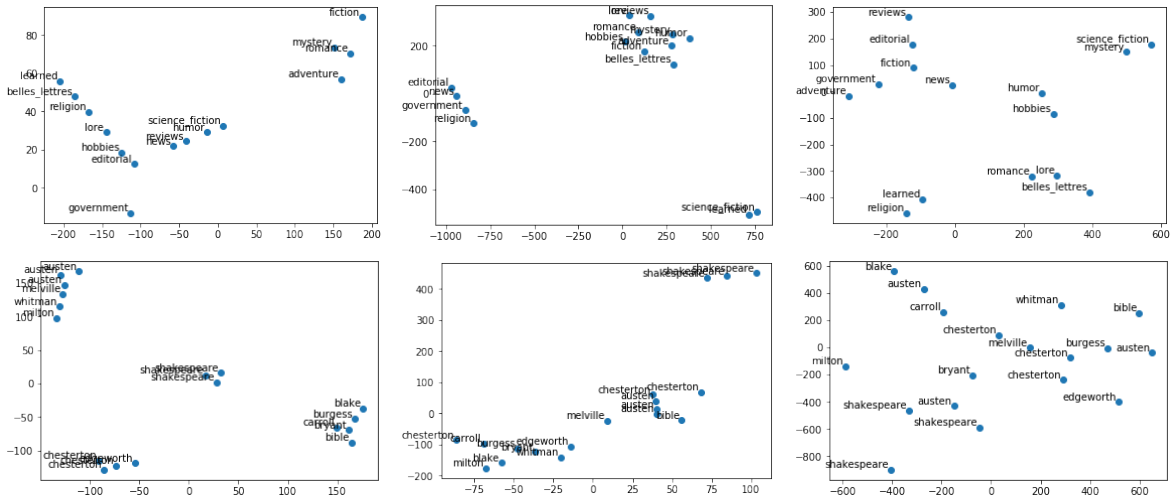

Figure 1: TSNE plots of embedding vectors on Brown corpus (Top) and Gutenberg corpus (Bottom) for the Hardcoded model (Left), the Doc2Vec model (Center), and the Synonym Perturbation model (Right).

across the embedding space. The embedding also do not perform particularly well on the genre similarity task, with adventure linked to government and news to fiction.

It is important to note that the embeddings construction approach described in the Methodology section above relies on several linguistic assumptions. Most critically, it assumed that the texts/categories have a uniform style, not varying between the different portions of the text. For the Brown corpus in particular, where each genre is composed of multiple documents, this assumption may not be valid. However, in the absence of a a standardized metric for style classification, this is a necessary simplifying assumption. Similarly, our evaluation approach relies on the stylistic uniformity of a given author to determine reasonable clusters, which may not always be a valid assumption. However, new methods of stylistic embedding evaluation must be developed in order to improve upon these heuristic style evaluations.

It is quite possible that with additional finetuning, the synonym perturbation model could significantly improve its performance. Due to the time constraints of the project (and personal computation constraints), we were unable to determine an optimal set of hyperparameters for the model. In particular, the models trained seemed quite prone to overfitting. We attempted to address this issue by varying the number of nodes in the hidden layer (or removing it altogether), adding Dropout and L2 regularization, and adding Batch Normalization. These tweaks led to significantly improved performance over the baseline model, however, the model still appeared to suffered from overfitting. We also varied the methods used to generate the synonyms, first using exclusively the cosine similarity method and eventually using wordnet synonyms when available. However, we did not have enough time or computational power to determine the optimal method of generating synonyms, so we defaulted to the faster wordnet approach. We were also unable to try out different sets of pretrained vectors for use in the similarity method, due to computational constraints and the need for at least some standard of comparison when tweaking the hyperparameters.

Moreover, while we initially attempted to use the entire Brown and Gutenberg corpora as training data for the synonym model, computational constrains forced us to restrict the analysis the first 10,000 words. The model would likely be improved by the inclusion of additional training data. We would have also liked to attempt to train the word embeddings along with the style embeddings, however, we considered it more time efficient to rely on precomputed activations.

6.2 Suggestions for Future Work

It is interesting to consider how the synonym perturbation model relates to the standard denoising autoencoder. In a sense, we can consider the generated synonyms as a form of noise added to the model inputs. This comparison is particularly apt when using the cosine similarity method of synonym generation, for this process is equivalent to

perturbing the input embeddings by a small angluar deviation. From these perturbed input embeddings, the model then attempts to recover the original target word, just as a denoising autoencoder seeks to remove the noise added to its inputs.

This comparison naturally lends itself to future extensions of the synonym perturbation model. Any model based on the perturbation of a critical feature of the text will result in a model that captures some element of style under the style as option paradigm. One simple extension is to perturb phrases rather than words. By using an RNN based seq2seq model, one could use synonym perturbation, word deletion/insertion, and other techniques to perturb an input phrase. By using such an architecture, one is free to change the length of the input when perturbing it and to use input phrases of varying length. This more complex model, if trained properly, would likely significantly outperform the basic trigram model proposed above. In addition to phrase perturbation, one also perturb the structure of a text by using POS tags in the input, or alternatively, train a model that can unscramble input words from the text given in random order.

In addition to these extensions, if the hyperparameters can be properly tuned and the validation accuracy of the model increased, the model could easily be applied to implement linguistic style transfer. In particular, a pretrained synonym perturbation model could be applied to a new text, treating the original text's words as input "synonyms" of words used in a target style. In this way, one could apply the model to iteratively output the synonomus words more likely to be used in the target style. The RNN seq2seq model described above could similarly be extended to transform the style of a text.

A final potential extension would be to make use of a GAN architecture to train a synonym perturbation model, which could improve performance at the cost of additional model complexity. In particular, given a set of synonyms and a target style, a generator network could generate words that are more likely to be used under the target style. Given these generated words (and optionally the input synonyms), the discriminator network would then be trained to determine whether the words come from the generator (i.e. are fake) or came from the actual text (i.e. are real). This model could then be applied in a similar way to that discussed above to the task of style transformation.

7 Conclusion

In conclusion, the hardcoded embeddings currently outperform both the Doc2Vec embeddings and the synonym perturbation based models. Nonetheless, the synonym perturbation model shows considerable promise for future work, both in fine tuning the current model and in developing new stylistic perturbation based model architectures.

References

Foaad Khosmood and Robert A Levinson. 2008. Automatic natural language style classification and transformation. In *BCS Corpus Profiling Workshop, London, UK*. sn.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

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.

Ellie Pavlick and Ani Nenkova. 2015. Inducing lexical style properties for paraphrase and genre differentiation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 218–224.

Jane R Walpole. 1980. Style as option. *College composition and communication*, 31(2):205–212.

Yanpeng Zhao, Wei Bi, Deng Cai, Xiaojiang Liu, Kewei Tu, and Shuming Shi. 2018. Language style transfer from sentences with arbitrary unknown styles. *arXiv preprint arXiv:1808.04071*.