OLD_Jupyter_Reproducible_paper-Copy1

June 16, 2020

1 Controllable text generator of spooky authors

1.0.1 Lucas Hideki Ueda

University of Campinas, Eletrical and Computational Engeeniring Faculty, lucashueda@gmail.com

Abstract Abstract— Natural Natural language processing (NLP) has undergone major changes with the introduction of embeddings, trasformers and so on. Generating text is one of the tasks of NLP and has undergone major advances in recent years with the GPT-2 and GPT-3. In this work we seek to study a way to control the generation of texts, based on 3 land story writers.Our source code is publicly available at https://github.com/lucashueda/reproducible_research .

Keywords: language modeling, variational autoencoder,normalizing flows, deep learning, spooky author.

1.0.2 1. Introduction

Language modelling tries to estimate a probability density function that can predict next token by the past ones [1]. This technique allows us to represent words in a latent space by using dense vectors with fixed dimensional. This dense vector is used in the literature with two main purposes, the first is a better representation of a phrase intead of an one hot vector and the second one is to control latent meanings by finding patterns in this latent space ([2], [4]).

Text generation it's a very hard task in NLP because of its "human" nature, it is very hard to find a way to generate text as a human since there is no specific pattern known in humans generated texts. Text DE-Generation is yet a big problem which makes algorithms very repetitive and with no reasonable meaning of the generated texts [3]. Additionally there is no main way to validate a text generator in terms of reasonable and coherence with human language in an automatic way. Additionally to control the text that is generated is one more problem in this task.

In this work we try to make a fully text generator system that is able to produce reasonable texts conditioned by an author embedding, i.e., by given as input an initial text and a author embedding the system is able to produce text as this target author. We use an dataframe that has texts from 3 spooky authors, Edgar Allan Poe, H.P. Lovecraft and Marry Shelley. This is a dataframe from a kaggle competition and has a lot of sentences of these 3 authors, we will use this dataframe to extract author's embeddings and generate our proposed method.

This work is organized as follows: Section 1 is the introduction, Section 2 we describe the methods that motives our proposed method, in Section 3 we describe the experiments made in the dataset

and the architectures tested, Section 4 and 5 we discuss and conclude our work. Further sections are just the source code and acknowledgements.

1.0.3 2. Methods

In this section we are going to summarize the motivations of our proposed method.

2.1. Language Modeling Language modelling is the process to estimate the probability density function that can predict a token given an array of past tokens [1]. This approach can be used to generate tokens, but the more gain about this technique was about word representation [2] where a dense vector could represent better a token than its one-hot vector, it allows a lot of evolution in NLP with GloVE, ElMO and more recently with Transformers. In this project we are going to use the meaning of language modelling to generate our tokens, we could use a sequence-to-sequence modelling instead of per token generation but for isolated effects purposes we choose towork with a simple way to do the token generation. Also we will do ablation using top-k and nucleus sampling [3] as decoding metodologies that are the best for our purpose.

2.2. Umap To visualize the ability of this latent representation to differentiate the writing style of each author, we will use UMAP, a methodology for reducing vector dimensionality. UMAP seeks, based on the initial data topology, to represent it efficiently in a smaller dimension (2D), which allows us to graphically visualize the topology of the generated representations.

1.0.4 3. Experiments

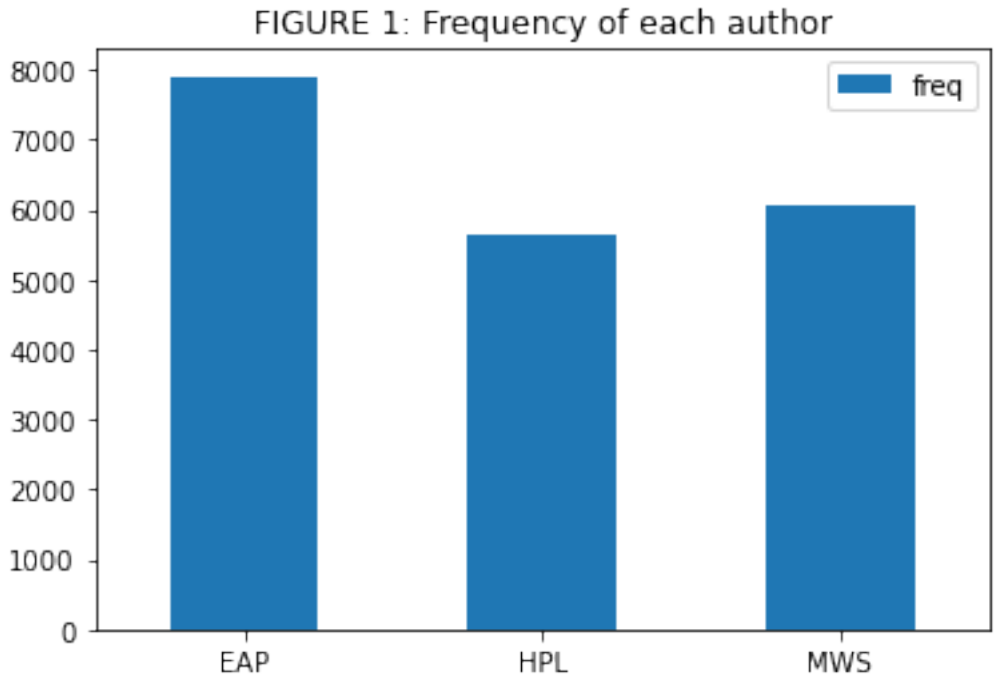
In this section we describe all of our experiments, the first one consist in an analysis on the dataset and the next two describing the models and the experiments dones.

3.1. Experiments: Spooky author dataset The dataset is provided by a kaggle competition that had as objective to classify a text with one of the three authors by texts, we will use the same dataset but for generative modelling purposes. It has 19579 observations such that almost a third by each author (Figure 1) and the number of words (tokens) goes from 2 to 861.

[1]:
```
# Import libs
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline

import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS
from nltk.util import ngrams
from collections import Counter
```

[7]:
```
base['freq'] = 1
count_per_author = base.groupby('author').agg({'freq': 'count'}).reset_index()
count_per_author.plot(kind='bar')
```

```
plt.title('FIGURE 1: Frequency of each author')
plt.xticks(np.arange(3), count_per_author['author'], rotation = 0)
plt.show()
```



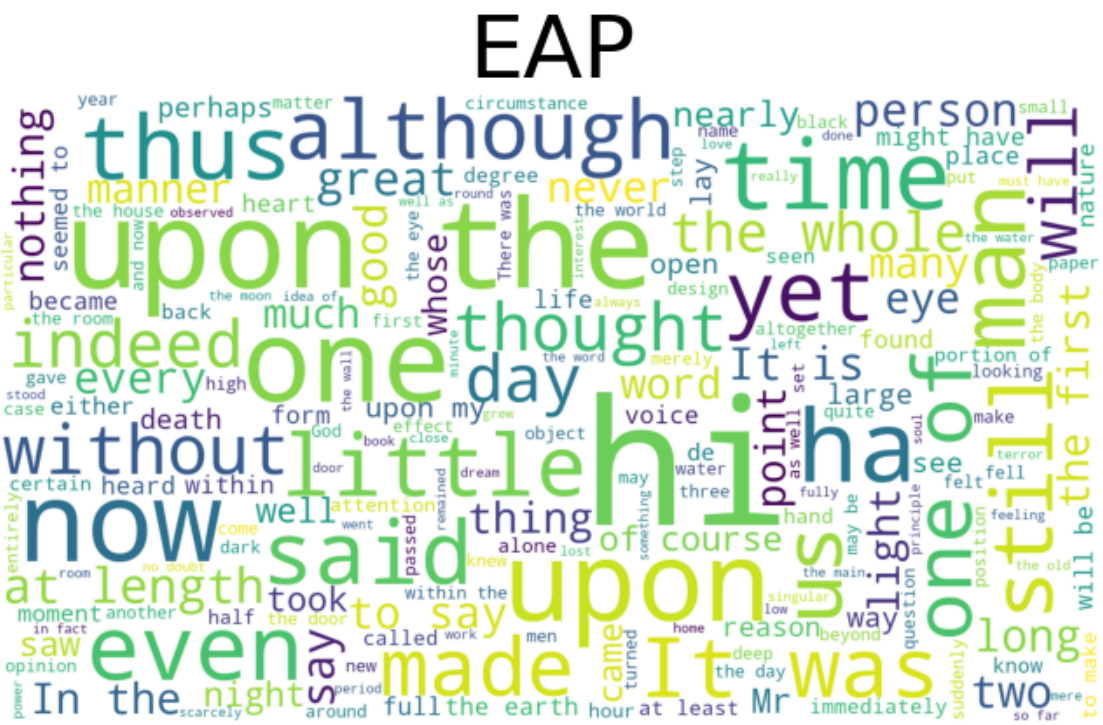
Our hypothesis using this dataset is that different authors intend to use different words (Figures 2,3,4)

[8]:
```
def cloud(text, title):
    # Setting figure parameters
    mpl.rcParams['figure.figsize']=(8.0,8.0)    #(6.0,4.0)
    #mpl.rcParams['font.size']=12                #10
    mpl.rcParams['savefig.dpi']=100             #72
    mpl.rcParams['figure.subplot.bottom']=.1

    # Processing Text
    stopwords = set(STOPWORDS) # Redundant
    wordcloud = WordCloud(width=1400, height=800,
                          background_color='white',
                          stopwords=stopwords,
                         ).generate(" ".join(text))

    # Output Visualization

    plt.figure(figsize=(10,5), facecolor='w')
    plt.imshow(wordcloud)
```

```
    plt.axis('off')
    plt.tight_layout(pad=0)
    plt.title(title, fontsize=50,color='k')
```

[9]:
```
x = 'EAP'
print(cloud(base[base.author == x]['text'].values,x))
```

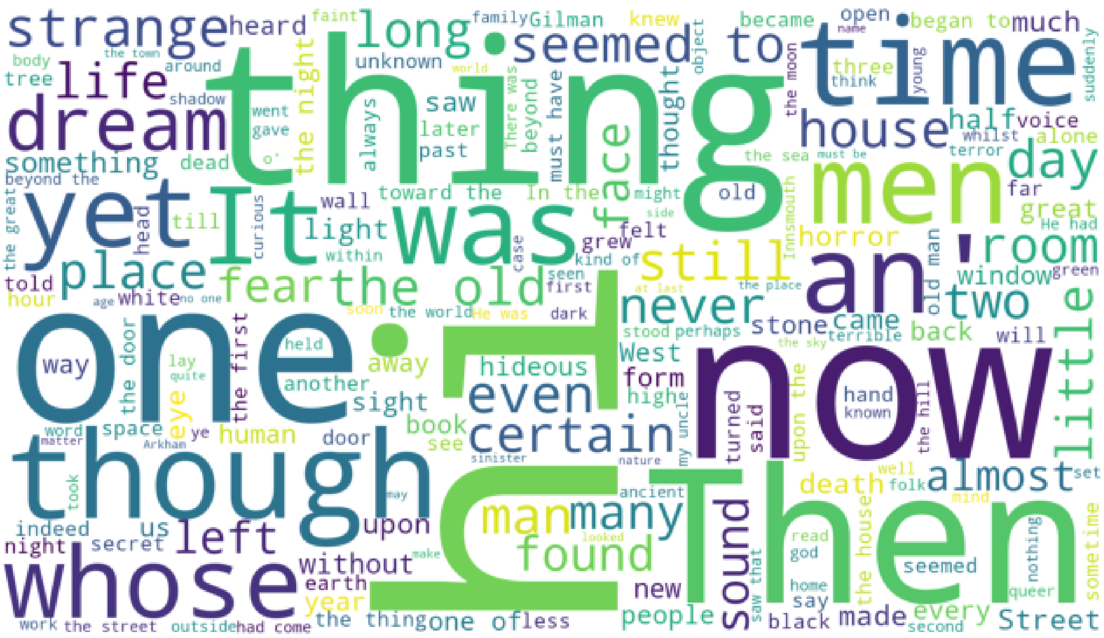
```
None
```



[10]:
```
x = 'HPL'
print(cloud(base[base.author == x]['text'].values,x))
```

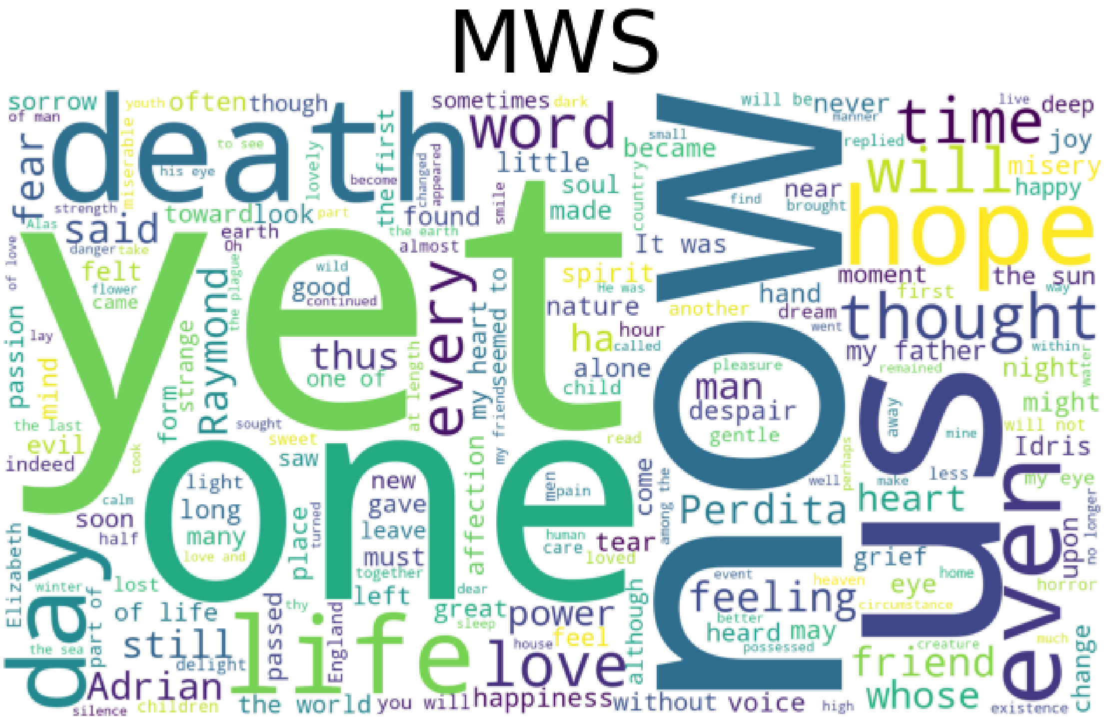
```
None
```

HPL




```python
x = 'MWS'
print(cloud(base[base.author == x]['text'].values,x))
```

None



3.2. Experiments: Vanilla model As a baseline we propose a simple Vanilla model that take as input a text and an one-hot vector that indicates what is the target author. With this baseline we intend to verify the real difference by our main approach in front of a simple one.

3.3. Experiments: Proposed model Our proposed method is based in a Transformer variation that is memory efficient, the Reformer. We add an author embedding together with the first N tokens to generate a new token as show in Figure 5. We perform ablations using top-k and nucleus sampling decoders and evaluate using our own metric.

Figure 5: The proposed method. The VAE architecture is trained separated to generate the VAE encoder output and concatenate in Transformer encode.

1.0.5 4. Discussion

Our ablations on our proposed method shows that a Transformer approach reaches better results, so as with the embedding modelling, VAE's get better latent representations and normalizing flows helps to avoid only gaussian distribution.

1.0.6 5. Conclusion

This work shows that is possible to build an entire controlled system that generate text conditioned to an specific author.

1.0.7 6. Source code

All the code used to do this project and instructions to reproduce it are available in https://github.com/lucashueda/reproducible_research.

1.0.8 7. Acknowledgements

This project is part of Computational Reproducible Research course at Unicamp (1S2020).

1.0.9 References

[1] P. V. C. J. Yoshua Bengio, R´ejean Ducharme, "A neural probabilistic language model," online:http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf, 2003.

[2] G. C. J. D. Tomas Mikolov, Kai Chen, "Efficient estimation of word representations in vector space," online:https://arxiv.org/abs/1301.3781, 2003.

[3] L. D. M. F. Y. C. Ari Holtzman, Jan Buys, "The curious case of neural text degeneration," online:https://arxiv.org/abs/1904.09751, 2019.

[4] M. W. Diederik P Kingma, "Auto-encoding variational bayes," online:https://arxiv.org/abs/1312.6114, 2013.

[5] Y. M. Kei Akuzawa, Yusuke Iwasawa, "Expressive speech synthesis via modeling expressions with variational autoencoder," online:https://arxiv.org/abs/1804.02135, 2018.

[6] N. P. J. L.-T. Vatsal Aggarwal, Marius Cotescu and R. Barra-Chicote,"Using vaes and normalizing flows for one-shot text-to-speech synthesis of expressive speech," online:https://arxiv.org/pdf/1911.12760.pdf, 2020.

[7] S. M. Danilo Jimenez Rezende, "Variational inference with normalizing flows," online:http://proceedings.mlr.press/v37/rezende15.pdf, 2015.

[8] N. P. J. U.-L. J. A. N. G. L. K. I. P. Ashish Vaswani, Noam Shazeer,"Attention is all you need," online:https://arxiv.org/abs/1706.03762, 2017.

[9] A. L. Nikita Kitaev, Łukasz Kaiser, "Reformer: The efficient transformer," online:https://arxiv.org/abs/2001.04451,2020.

[10] R. R. Stanley Chen, Douglas Beeferman, "Evaluation metrics for language models," online:https://www.cs.cmu.edu/~roni/papers/eval-metrics-bntuw-9802.pdf, 2001.

[11] T. W. Kishore Papineni, Salim Roukos and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," online:https://www.aclweb.org/anthology/P02-1040.pdf, 2002.

[12] N. Craswell,Mean Reciprocal Rank, pp. 1703–1703. Boston, MA:Springer US, 2009.