# Technology Review

Xiaolong Yang

## Topic Selection:

Recommendation combined Topic Models & Neural Language Models on Wikipedia data.

## 1.What is the Purpose of the project?

The tools aimed to build recommendation system for Wikipedia.

Wikipedia is a free online encyclopedia. It is one of the famous Big Text data source and used in many text mining related project, such as Google Knowledge Graph, Wikipedia Search Engine. I am building the tool to automatedly recommend relevant Wiki page on user’s interested page, such as recommend President Trump on President Obama’s page.

Each Wikipedia page is a topic, you can easily find related information within the topic. Take President Obama as example, you can find his personal details, early life and career, presidential campaigns, presidency and so on in his wiki page. Wikipedia use hyperlink in the page to indicate other topics related to current topic. In the President Obama example, you can find “United States presidential election, 2008” within the wiki page, which is highly related events. However, there are also meaningless topics such as “English”, “Seattle”, which is the language he knew or where he stayed for a while.

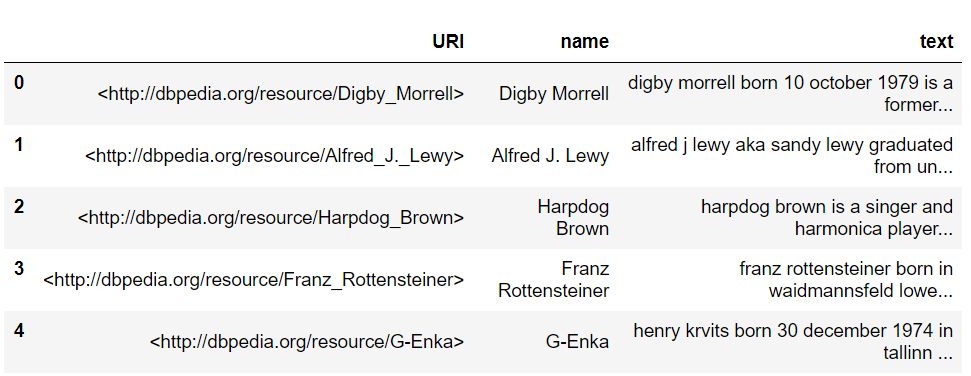
In this case, if I can develop a tool to show merely the highly related topics of user interested, it would save user’s time to explore the document.

## 2. What is the data source?

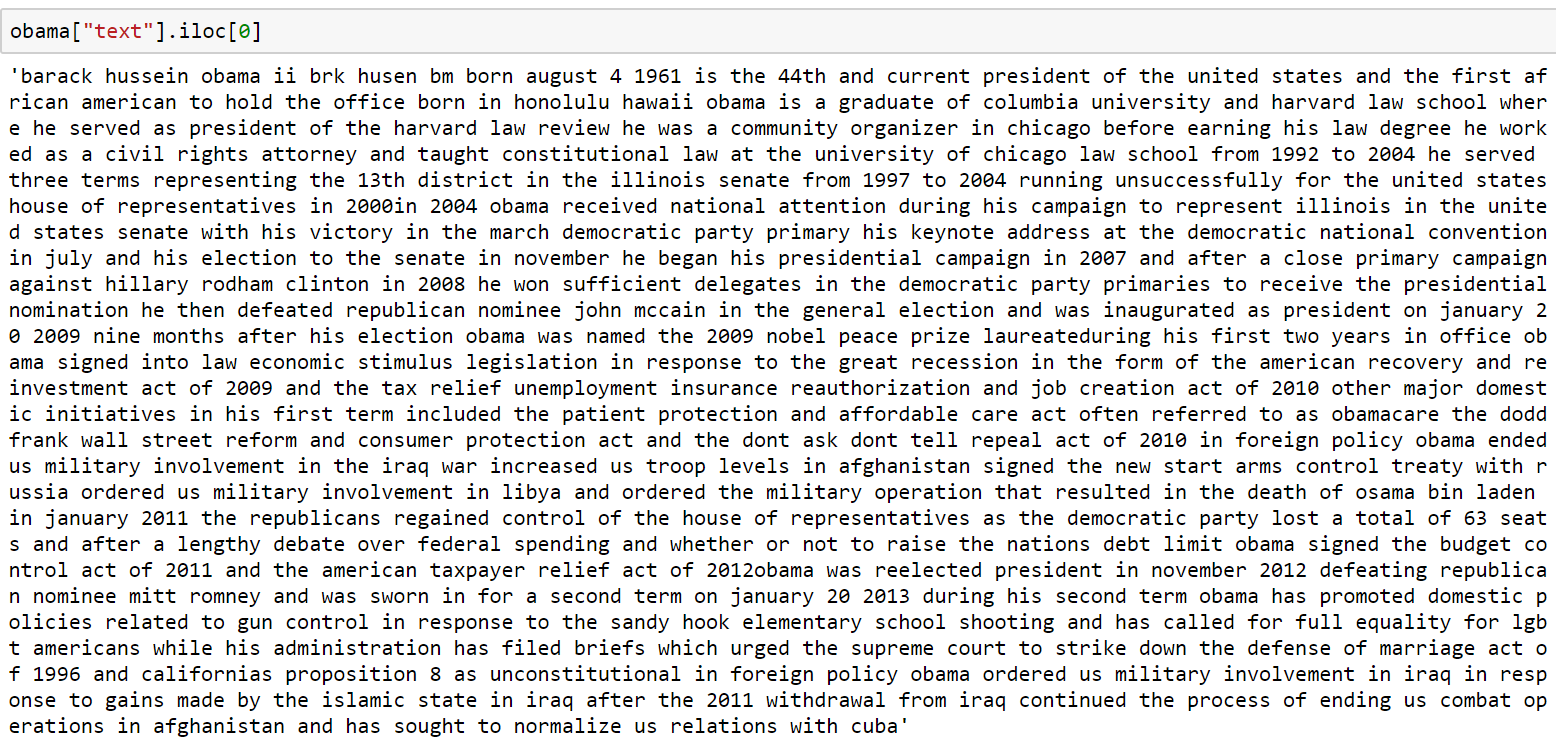
In the beginning of the project, I tried to use the whole dataset from Wikipedia. <https://dumps.wikimedia.org/enwiki/20170901/>. However, the dataset is as large as 14 GB and I can merely finish my project based on the raw data. After then, I found there are 3 categories in the dataset: page-meta (All pages), page-article (Articles, templates, media/file descriptions, and primary meta-pages), abstract (extracted page abstracts for Yahoo). I tried to directly analyze the abstract data (4.9 GB in total), but it was hard to find took too much time for each iteration.

After then, I narrow my search to a manageable data source of Wikipedia, but still capable to recommend “Clinton” to “Obama”. In other words, I tried to find dataset contained only “people” Wiki page. There is a data called “people\_wiki.gl” from University of Washington <https://d396qusza40orc.cloudfront.net/phoenixassets/people_wiki.csv>, which met the requirement and only 150 MB.

There are totally 59071 people in the dataset, each people had 3 columns: URI, name, and text. The “Name” is the name of people, the “Text” is the wiki page related to the people. Here is an example how the data looks like:



Here is an example of Barack Obama’s text:



## 3. What tech used to text retrieval?

I followed the same steps as MP1, tokenize, remove stop words and stem, and uni-gram.

Here is an example of tokenize on Barack Obama’s text:



Here is an example of stop words removal on Barack Obama’s text:



Here is an example of stemming on Barack Obama’s text:



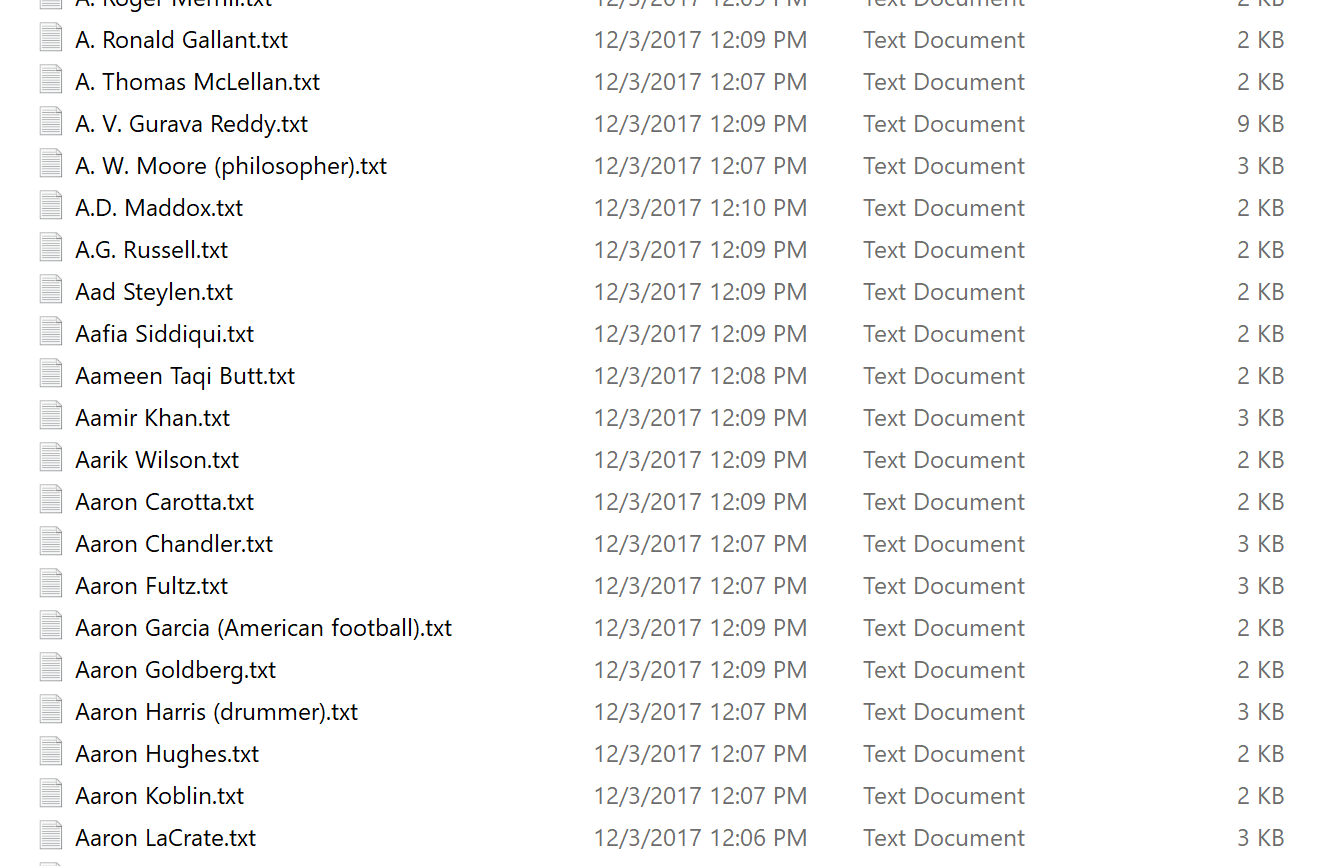
Here is an example of uni-gram on Barack Obama’s text:



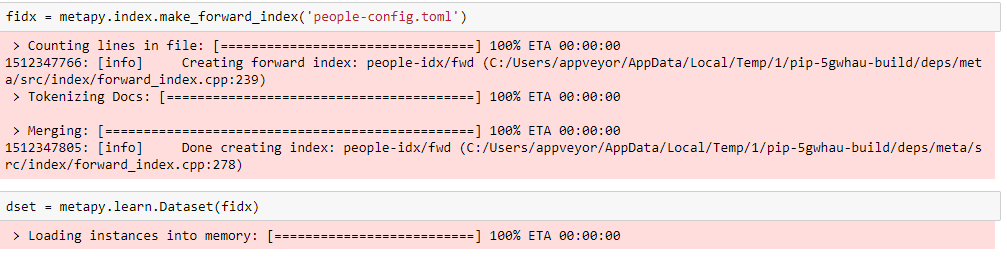
## 3. How to transfer data to meet Metapy requirement?

Metapy stored all process data in disk index. Before we started modeling, we need first index the data. However, since the data I got was stored as a big data frame, the first step is to divide the data into different text file.

Take a glimpse how it looks like:



And then combined into a data called data.dat. I also defined config files in the people.toml and transfer the data into index format:



Here is how it looks like in file system:



## 4. What tech used for recommendation?

I built my recommender with two techs, Topic Models & Neural Language Models. The main difference between these models is the contextual information they use: LSA and topic models use *documents* as contexts, and Neural language models and distributional semantic models instead use *words* as contexts. These different contextual representations capture different types of semantic similarity; the document-based models capture semantic relatedness (e.g. “boat” – “water”) while the word-based models capture semantic similarity (e.g. “boat” – “ship”).

For President Obama example, I tried to find “Politics” topic using topic models, and “Trump”, “Clinton” words using neural language models.

I tried to do the same thing as the following documents, and it took so much time for training. Right now, I was still working on pruning the parameters for better result.

Meta for topic-modeling Example

<https://github.com/meta-toolkit/metapy/blob/master/tutorials/5-topic-modeling.ipynb>

Meta for Word2vec Example

<https://meta-toolkit.org/word-embeddings.html>

Tensor flow Word2vec Example:

<https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/udacity/5_word2vec.ipynb>

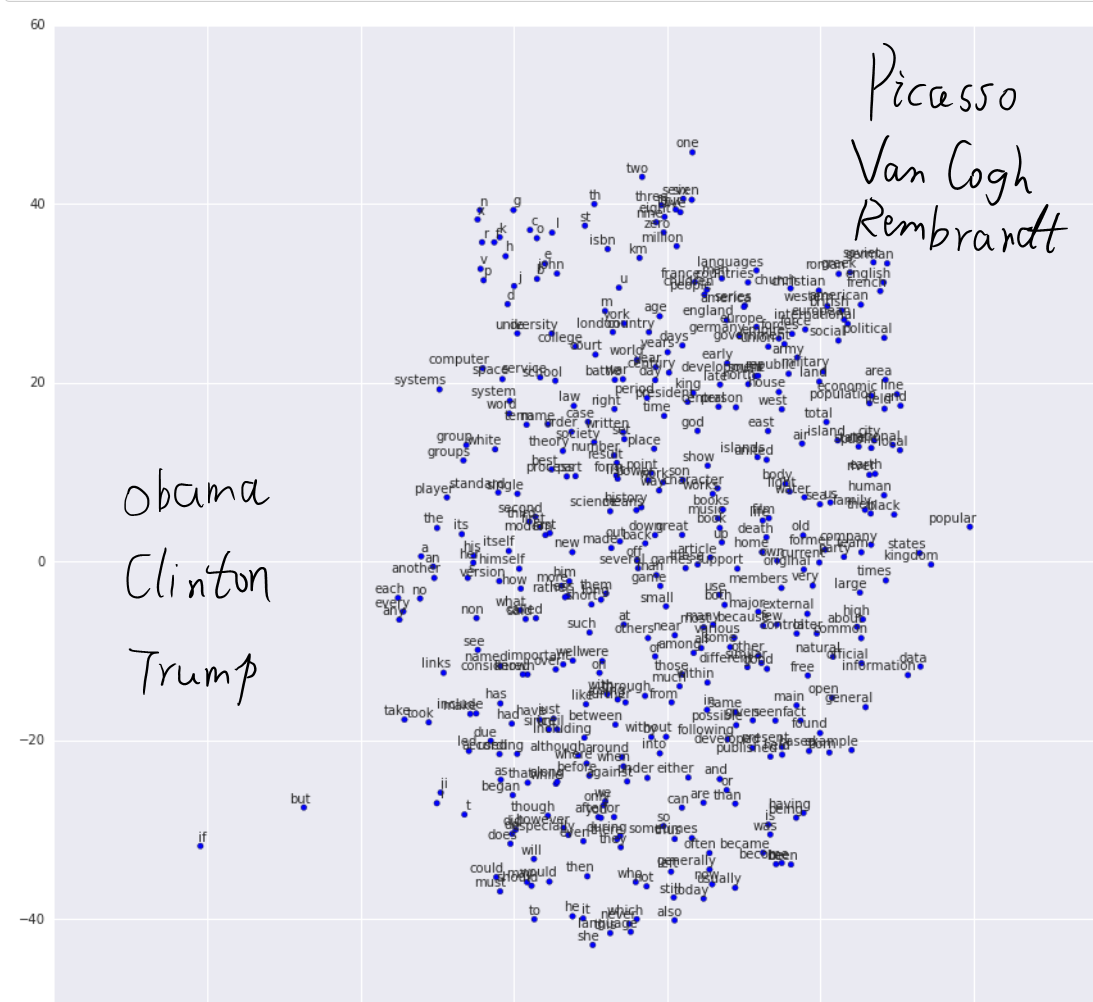
I tried to work on both LDA and word embedding on Metapy, however, MeTA did not have the tutorial to implement the word embedding on python. Thus, I chose to use Tensor flow to train the model.

Expected result for topic modeling

model.topic\_distribution(“obama”) {Artist: 0.021341, Politic: 0.978659}

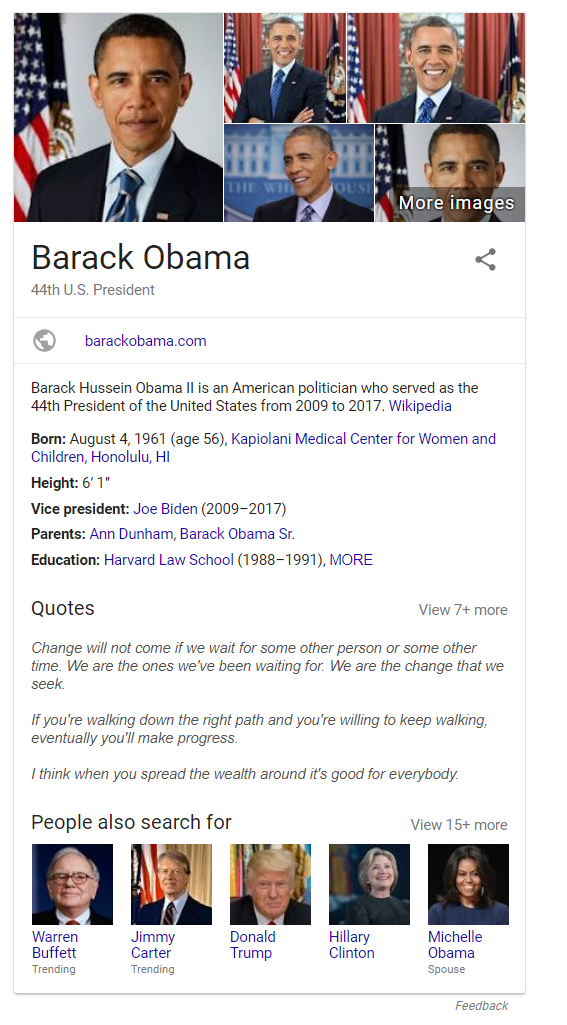
model.topic\_distribution(“Picasso”) {Artist: 0.978659, Politic: 0.021341}

Expected result for word2vec



## 5. How to represent the result?

The google graph show the result as:



I would use Bootstrap to show the result as:

