FROM PILOT TO PRODUCTION

1

2

3

Copy the de-duplication/semantic similarity Python code from the base CDO Council Pilot Toolkit “Read Me” package.

If SBERT/BigBird does not scale, replace with Python packages [FAISS](https://ai.facebook.com/tools/faiss/) or [ANNOY](https://github.com/spotify/annoy) which are designed for high throughput similarity search, but not as accurate.

To scale, select desired platform (e.g., Floydhub, Azure, AWS, etc.) with appropriate memory capacity. For <1,000 comments, use 8GB. For more, use 32GB or 60-64GB.

Run the models and assess results. If there are errors, return to step 2. If you would like to further refine the results, proceed to the customization steps.

DEDUPLICATION & SEMANTIC SIMILARITY

hLDA

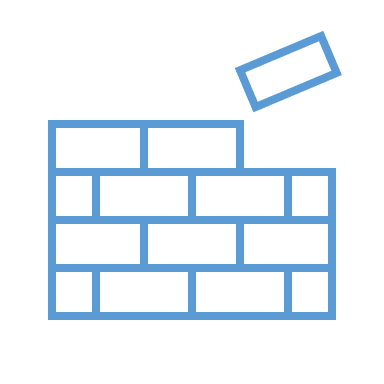
TOPIC MODELING

Comparing comments to each other grows quickly: if you have N comments, and you compare each comment to one another, it will correlate to N^2 comparisons. One ruling with even 1,473 comments would end up having 1,084,128 total comparisons. The ruling with the most comments in our pilot dataset has 54,737 comments, which would be 1,498,042,216 comparisons. Reducing the data size helped reduce computational complexity for the purposes of this pilot.

COMPLEXITY

hLDA is extremely memory intensive. Our Floydhub CPU environments offer options of 8GB and 32GB. We encountered memory errors using 8GB. Depending on the number of comments an agency is looking to analyze, memory errors may be encountered at 32GB as well. Depending on an agency’s comment count, we recommend using a server 32GB or higher.

Copy the hLDA Python code from the base CDO Council Pilot Toolkit “Read Me” package.



The semantic similarity and hLDA models were fine-tuned and performed well using a sample of around 500 comments under each rule from various agencies. For the purpose of this pilot and prototype, this reduced the computational complexities described below.

For GSA or agencies that would like to implement these models in production, they can scale the models using the following steps.

If agencies wish to tune the base **DEDUPLICATION & SEMANTIC SIMILARITY MODEL**, an agency’s data science team would:

Manually identify examples of comment pairs that are semantically similar.

Label the data so that they are identified as semantically similar.

Note: Labels should be created for two categories:

* + 1. Pairs of comments that are semantically similar (labeled “1,” which is the cosine similarity metric indicating most semantically similar)
    2. Comments that are not semantically similar (labeled “0,” to indicate they are not at all semantically similar).
       - Note: model performance will be improved if “not similar” comment pairs are at least talking about the same topic.

Using the CDO Council Pilot Toolkit developed in this pilot, copy the BigBird de-duplication/semantic similarity Python code from the “Read Me” package.

Import this code and existing BigBird model into their GPU or preferred platform.

Train the generalizable BigBird model on agency-specific documents with domain-specific terminology, using the labeled comment pairs in step 1.

* + To do this, encode the text with the BigBird model. This produces text embeddings that can be used as a feature vector (predictor variables) for a classification model.
  + The classification model would predict semantic similarity using the language model representation as input data.

Run the models and review and validate results.

Once there is a language model representation of comments and labels for comments, a classification model (e.g., logistic regression, random forest) can be created using the data scientist’s preferred language.

Refine as needed.

1

2

4

5

6

3

7

8

We suggest that agencies first implement the base toolset and evaluate the results and cost/benefit before attempting customization.

CUSTOMIZATION STEPS

In order to **customize the base toolset** provided through this pilot, agencies would need an internal data science team or contractor with knowledge of natural language processing and the selected models, and access to a GPU.

***Additional detail on the Deduplication and Semantic Similarity attached files***

The attached programs are for different uses:

1. “example-bb-transfer-learning.py” – script to fine tune the language model. Fine tuning the language model isn’t required, but if the agency finds the existing language model is underperforming based on their expectations, then fine-tuning is a good idea. This requires the agency to produce new data to train.
2. “embedtext.py” – contains a function “make\_embed\_as\_df” which takes a list of comments, a list of the corresponding IDs for the comments, and a language model, and returns a pandas dataframe of the IDs, the comments, and the embeddings for each comment.
3. “example 1-encode data general use.py” – example of using the language model to create embeddings for every comment in a dataset. Uses function defined in “embedtext.py”. The embeddings created here are used in examples 2A and 2B.
4. “example 2A-pair comments with 0.85 cosine similarity threshold.py”- using the data created by “example 1-encode data general use.py”, create a list of paired comments who’s embeddings have a cosine similarity score of 0.85 or higher.
5. “example 2B-clustering with embeddings.py”- using the data created by “example 1-encode data general use.py”, clusters close comments together using the DBSCAN algorithm.

The files most agencies would use is “embedtext.py” and can closely follow the code in “example 1-encode data general use.py” to produce their own text embeddings based on the agency’s own comments. Both examples 2A and 2B are both examples of how to use the embeddings produced by “embedtext.py” and example 1, the goal of both 2A and 2B is using the embeddings to identify duplicate and near duplicate comments.

**Model Fine-Tuning Special Note**

If an agency wants to fine-tune the model, they first have to have data to train the model on. This data will be created by the agency and must contain pairs of comments as well as a label indicating if the comment pair is semantically similar or not. Example:

|  |  |  |
| --- | --- | --- |
| Comment1 | Comment2 | Label |
| I really like this new regulation! This will improve my community. | This new regulation is going to improve the lives of many people in my community! | 1 |
| This new regulation is really well-crafted! | This proposed ruling has some great ideas, but there isn't anything that addresses issue X. Please include it! | 0 |

The pair of comments in the first row is a clear semantic match as the comments are praising the regulation; we label it 1 for “semantic match”. The second row isn’t; the first comment is praising the regulation but the second comment is a partial endorsement, so we label it 0 for “not a semantic match”.

Labeling pairs of comments is less of a technical issue and is more so a human decision one. The language model gets a sense of what comments are semantically similar based on the data it is given and will use similar judgement that reflects how pairs of comments are labeled in the fine-tuning data.

If agencies wish to further tune the base **HLDA TOPIC MODEL**, they would build on the unsupervised topic modeling technique developed in this pilot to create a weakly supervised topic model. To do this, an agency’s data science team would need to:

Work with comment processing SMEs to review a set of agency-specific comments and provide desired category labels for the topics found in these comments.

* + 1. These labels will help validate the results of the model.
    2. Labels should clearly indicate examples of topics.

Using the CDO Council Pilot Toolkit developed in this pilot, copy the hLDA Python code from the “Read Me” package.

Import this code into their GPU or preferred platform.

Train the generalizable hLDA model on agency-specific documents with domain-specific terminology, using the labeled comments in step 1.

* + 1. To do this, encode the text to produce text embeddings that can be used as input data (predictive variables)
    2. Once there is a language model representation of comments and labels for comments, a classification model (e.g., logistic regression, random forest) can be created using the data scientist’s preferred language.
    3. Using the language model representation as input data, the classification model would predict comment topic labels.

Run the models and review and validate results.

Tune model parameters (e.g., for different topic areas) to influence performance as needed.

* + 1. Agencies may also consider creating separate models for each topic area, training on a more discrete set to improve algorithms for specific subject areas.
    2. Agencies can use the same code for each language model representation and classification model but different data and labels related to the topic area.

1

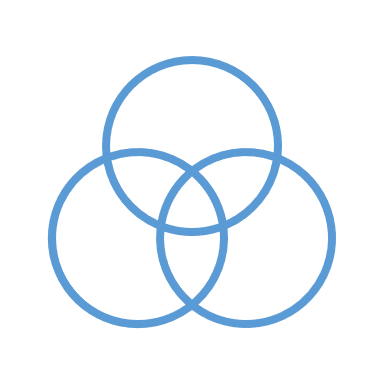
2

4

5

6

3



***Additional detail on the hLDA attached files***

Files included:  
\* hlda.py  
\* lda\_vis.ipynb   
\* optimseed\_equity\_keywords.ipynb