CookSmart

Course : DATA 515
Date : 2/23/2021
Team : Marc Mascarenhas
Sandeep Tiwari
Sandhya Tharanian

Project Overview

- What recipe can I make with my ingredients?
- Can Topic Modeling help?
- 50,000 Kaggle Recipe Dataset[1] to the rescue!

Technology Evaluation

For the purpose of this technology review, we chose to explore the below libraries for Latent Dirichlet Allocation (LDA).

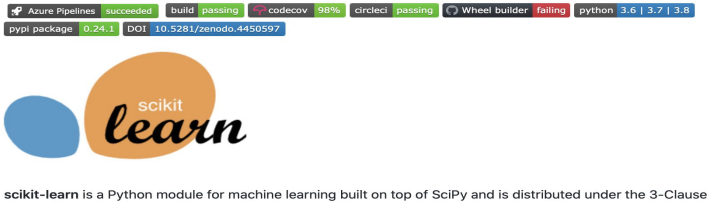
- Scikit learn
- Gensim

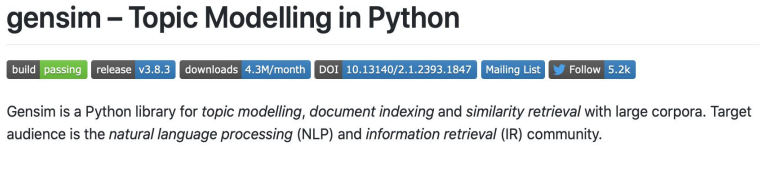
Evaluation framework

1. Documentation
2. Functionality
3. Ease of Implementation & Speed
4. Amount of hyperparameter tuning

Scikit learn vs Gensim

1. Documentation
   - Scikit learn has a consistent API interface and more familiar to us
   - Cover all badge





2. Functionality
   - Gensim has more functionality as it's a dedicated topic modeling library, however for the purposes of this project, both met our requirements.

Scikit learn vs Gensim

3. Ease of Implementation & Speed

Scikit Learn[2]

```
In [12]: now = datetime.now()

         count_vect = CountVectorizer(stop_words='english')
         doc_term_matrix =count_vect.fit_transform(
             data['ingredients'].values.astype('U'))


         LDA = LatentDirichletAllocation(n_components=8,
                                         random_state=42)
         LDA.fit(doc_term_matrix)

         print(datetime.now()-now)

         0:02:50.851897
```

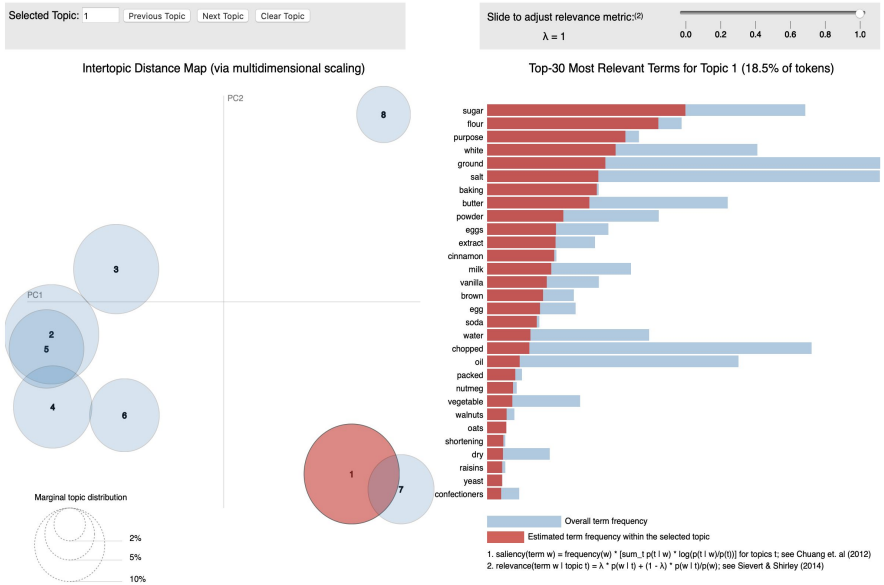
Gensim[3]

```
In [3]: data = pd.read_csv("../data/cleaned-data_recipe.csv")

        now = datetime.now()

        data.ingredients = data.ingredients.apply(ingredients_to_text)
        data_words =[recipe.split() for recipe in data.ingredients]


        id2word = corpora.Dictionary(data_words)
        corpus = [id2word.doc2bow(text) for text in data_words]


        lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                               id2word=id2word,
                                               num_topics=8,
                                               random_state=42
                                              )

        print(datetime.now()-now)

        0:00:18.352747
```

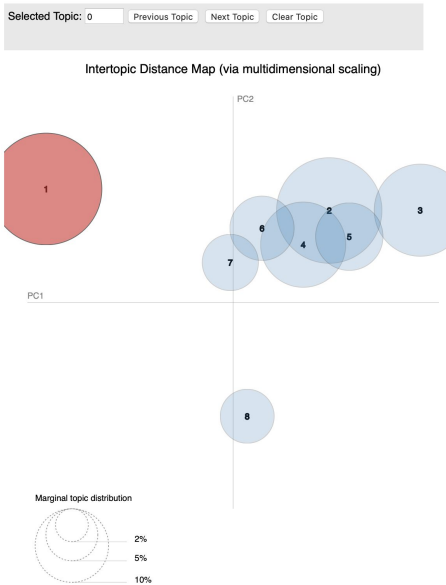
4. Amount of hyperparameter tuning
    - Scikit Learn worked better with default hyperparameters compared to Gensim.

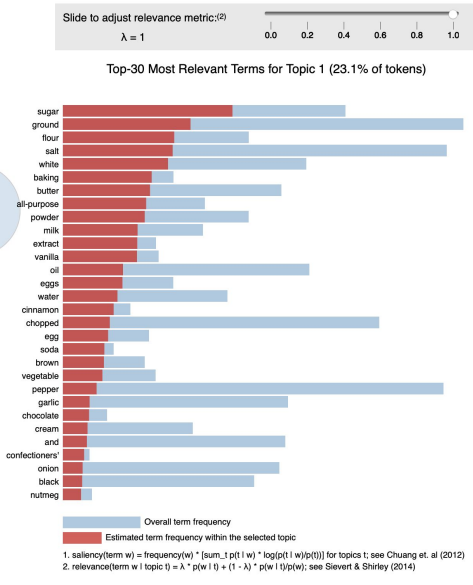
Scikit learn vs Gensim

Sklearn Implementation



Gensim Implementation





Why Scikit Learn?

Evaluation Framework	**Scikit Learn**	**Gensim**
Documentation	Recommended	x
Functionality	Recommended	Recommended
Ease of Implementation	Recommended	x
Speed	x	Recommended
Hyperparameter tuning	Recommended	x

References

[1] Dataset - https://www.kaggle.com/elisaxxygao/foodrecsysv1

[2] Scikit Learn LDA - https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html

[3] Gensim LDA - https://radimrehurek.com/gensim/models/ldamodel.html

THANK YOU!!

Appendix I

Gensim

Pros

- Gensim was able to run the model in 18.35 seconds.
- Has lot more built in functionality which makes it easy to evaluate performance metrics such as coherence or perplexity.
- Optimized for parallel processing.

Limitations

- Gensim relies on Python's multiprocessing libraries which has its limitations resulting in memory usage issues causing the multiprocessing to crash.
- Scikit learn's default lda model was better compared to Gensim's default lda model.
- Lot more tuning required to improve the model performance.
- Gensim doesn't have an implementation for NMF

Appendix II

Scikit learn

Pros

- Scikit Learn LDA model worked well with default hyperparameters.
- Scikit Learn being a well tested library provides API consistency which makes it almost easy to perform topic modeling using both LDA and NMF
- Scikit Learn also includes seeding options for NMF which greatly helps with algorithm convergence and offers both online and batch variants of LDA.

Limitations

- Scikit Learn was able to run the model in 170 seconds which was longer compared to Gensim.
- Evaluating the model performance using coherence or perplexity metrics is not a straightforward implementation in Scikit learn.