Unsupervised Text Classification & Clustering:
What are folks doing these days?

Rachael Tatman, Kaggle
PostDate

datetime the forum post was made

28Apr10  6Jun19
Problem: I can't keep reading all the forum posts on Kaggle with my human eyeballs
Problem: I can't keep reading all the forum posts on Kaggle with my human eyeballs

Solution: Unsupervised clustering to summarize common topics & user concerns
Problem: I can't keep reading all the forum posts on Kaggle with my human eyeballs

Solution: Unsupervised clustering to summarize common topics & user concerns
Some ground rules:

- **Needs to be in Python or R**
  - I’m livecoding the project in Kernels & those are the only two languages we support
  - I just don’t want to use Java or C++ or Matlab whatever

- **Needs to be fast to retrain or add new classes**
  - New topics emerge very quickly (specific bugs, competition shakeups, ML papers)
  - I'll probably have to re-run it daily or weekly
  - Eventually... streaming?

- **Want to avoid large/weird dependencies**
  - “Oh, that’s just some .jar I downloaded from a random website. The code doesn’t run without it and I’m sure it’s fine to just stick in our codebase.”

- **Clusters/topics should be easily interpretable**
I asked on Twitter!

Lots of good ideas!

Three main bins:

- End-to-end solutions
- Suggestions for feature engineering + clustering
- Misc. tips & tricks (ex: embeddings -> PCA -> remove 1st principle component)

What are y'all's current favorite unsupervised classification/clustering approaches for text? So far I've looked at:

- LDA
- Embeddings (doc2vec) + clustering (k-means)
- Unsupervised keyword extraction (YAKE)

Is there something else I should consider?
End-to-end solutions

- **Gensim**
  - ✓ In Python, no weird dependencies
  - ✓ Old standby that incorporates a looot of different methods
  - ✓ Don’t need whole corpus in memory (but mine’s not that big)
  - ❌ Under LGPL (probably fine for prototyping, but might need to meet with legal if I’m using it for work stuff)

- **BigARTM**
  - ✓ Can incorporate multiple objectives at once (sparsening, smoothing, decorrelation, etc.)
  - ✗ Weird dependency/install process (it’s a C++ library with a Python API)

- **TopSBM**
  - ✓ Came highly recommended: “Scary good”
  - ✗ Weird dependency (graph-tool, which is C++ with a Python wrapper)
Feature Engineering: Words to numbers

● Traditional Topic Modelling Approaches
  ○ **LDA**: Slow, hard to interpret, not my fave
  ○ **pLSA**: Cheaper version of LSA, tends to overfit
  ○ **tf-idf**: Hard to interpret, my texts (forums posts) are too short

● Embeddings
  ○ **GloVe**: considers context, can’t handle new words
  ○ **Word2vec**: doesn’t handle small corpuses very well, very fast to train
  ○ **fasttext**: can handle out of vocabulary words (extension of word2vec)

● Contextual embeddings (don’t think I have enough data to train my own…)
  ○ **ELMO, BERT, etc.**: I consider these more of a replacement for language models
  ○ **USE embeddings**: Not super familiar with this but looks useful for applying to sentence similarity
Feature Engineering: Words to numbers

- Traditional Topic Modelling Approaches
  - **LDA**: Slow, hard to interpret, not my fave
  - **pLSA**: Cheaper version of LSA, tends to overfit
  - **tf-idf**: Hard to interpret, my texts (forums posts) are too short

- **Embeddings**
  - **GloVe**: considers context, can’t handle new words
  - **Word2vec**: doesn’t handle small corpuses very well, very fast to train
  - **fasttext**: can handle out of vocabulary words (extension of word2vec)

- Contextual embeddings (don’t think I have enough data to train my own…)
  - **ELMO, BERT, etc.**: I consider these more of a replacement for language models
  - **USE embeddings**: Not super familiar with this but looks useful for applying to sentence similarity
Feature Engineering: Dimensionality Reduction

- **UMAP:**
  - Recommended to me by, among other people, Leland McInnes, the researcher who developed it 😄 (he suggested using hellinger distance)
  - Similar to t-SNE but can also be used for non-linear dimension reduction
  - Something about manifolds? (The math’s a little over my head, tbh)

- **PCA:**
  - OG dimensionality reduction (paper is from 1901!) but on its own maybe not the best
  - Trick: remove first principal component as a way to reduce the weight of “expected” words
    - (from Arora (2018) 'A simple but tough to beat baseline for sentence embeddings')
Feature Engineering: Dimensionality Reduction

- **UMAP:**
  - Recommended to me by, among other people, Leland McInnes, the researcher who developed it 😂 (he suggested using hellinger distance)
  - Similar to t-SNE but can also be used for non-linear dimension reduction
  - Something about manifolds? (The math’s a little over my head, tbh)

- **PCA:**
  - OG dimensionality reduction (paper is from 1901!) but on its own maybe not the best
  - Trick: remove first principal component as a way to reduce the weight of “expected” words
    - (from Arora (2018) 'A simple but tough to beat baseline for sentence embeddings')
Wildcard!

- **Unsupervised keyword extraction:** **YAKE**
  - Extracts keywords from single texts
  - Could use it as dimensionality reduction
  - Keywords -> embeddings -> clustering?
  - One of their sample texts is about the Kaggle acquisition! 😊
  - Haven’t played around with it, but came highly recommended
  - `pip install git+https://github.com/LIAAD/yake`
Wildcard!

- **Lda2vec**
  - Embeddings + topic models trained simultaneously
  - Developed at StitchFix 3ish years ago
  - Still pretty experimental but could be helpful
  - Under MIT license
  - Has a tutorial notebook
  - Might be very slow???
Clustering:

- **Brown Clusters**
  - Doesn't require feature engineering; can take words directly
  - Hierarchical clusters (could be useful for visualization/exploration)
  - Can be actively updated (wouldn’t have to retrain)

- **DBSCAN/H(ierarchical)DBSCAN**
  - Could take embeddings
  - Clusters assumed to be of similar densities

- **Spectral clustering**
  - Doesn’t make assumptions about spatial distribution of data
  - In sklearn
Clustering:

- **Brown Clusters**
  - Doesn’t require feature engineering; can take words directly
  - Hierarchical clusters (could be useful for visualization/exploration)
  - Can be actively updated (wouldn’t have to retrain)

- **DBSCAN / H(ierarchical)DBSCAN**
  - Could take embeddings
  - Clusters assumed to be of similar densities

- **Spectral clustering**
  - Doesn’t make assumptions about spatial distribution of data
  - In sklearn
Next stage: Experiments

- word2vec
- fasttext
- USE

- UMAP
- PCA - 1st

- HDBSCAN
- Spectral Clustering

- YAKE
- Brown Clustering

- Id2vec
Next stage: Experiments

- word2vec
- fasttext
- USE

- UMAP
- PCA - 1st
  - HDBSCAN
  - Spectral Clustering

- YAKE
  - Brown Clustering

- Ida2vec

Trying first
Future work

● Slackbot!
  ○ For now, I'll probably run the code in Kernels

● Other things I want to do as part of this project
  ● Identify questions I'm likely to answer
    ○ Extend to arbitrary user
  ● Build an alerting system that flags sudden new trends on the forums (competition drama, major bug, etc.)
    ○ I doooon't want to handle streaming data :weary:
Thanks!
I’m very open to feedback/suggestions :)

@rctatman