SIADS 694

Milestone II Report:

Wikipedia Text Complexity Classification & Topic Modelling

May 2021

Ai Zhong and Jim Creighton

Supervised Learning - Wikipedia Text Complexity Classification

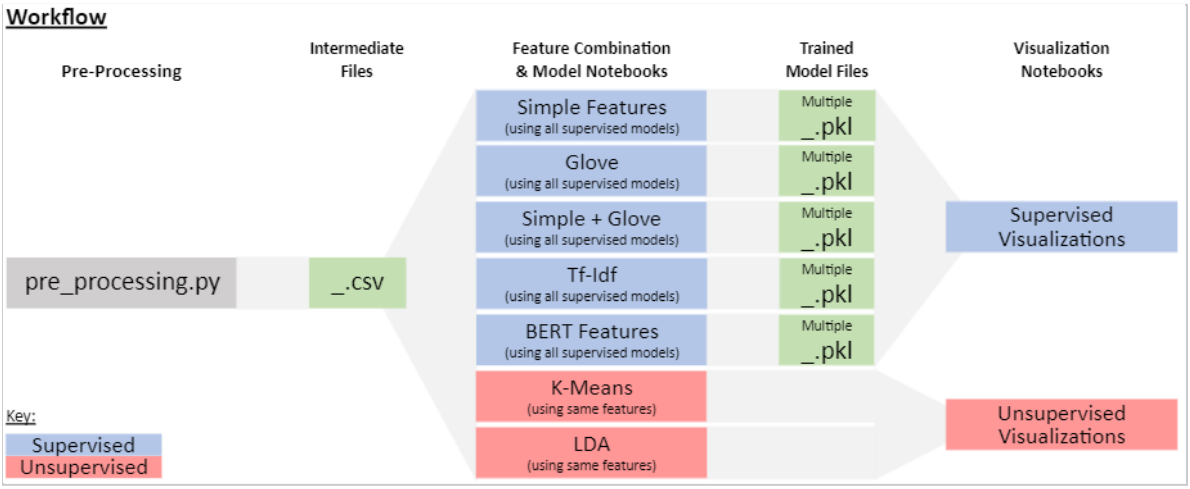
Motivation / Background

It is often necessary to provide simplified-language versions of text documents for language-learners, and one such example is the Simple Wikipedia website, with the stated aim of "providing an encyclopedia for people with different needs".

In this project, a set of documents (mostly single-sentence documents) were obtained from Wikipedia. Each is labelled 1 or 0, where 1 indicates a document needs to be simplified and 0 indicates it *doesn't* need to be simplified. Our goal was to train supervised learning models on this data set and then use the model to predict proper labels for an unlabeled test set.

Methods

The original training dataset "WikiLarge_Train.csv" contains 416,768 labeled documents. To save time while experimenting with different models, we chose a 100,000 document subset. After comparison, we used the full training set to train the chosen model and then predicted on the provided test set "WikiLarge_Test.csv" (119,092 unlabeled documents). We then submitted predictions to Kaggle.



General supervised workflow: Data Preprocessing -> Feature Generation -> Model Training -> Evaluation

We selected 3 main models for evaluation and experimented with BERT :

- Support-Vector Machines
- Random Forest
- Neural Network (Multi-Layer Perceptron)
- BERT (experimenting only)

We have 4 different feature combinations for the 3 main models, while BERT has its own feature generation process. The 4 feature sets are:

- 11 simple numerical features
- Pre-trained word vectors
- 11 simple numerical features + Pre-trained word vectors (a combination of the above two)
- Term Frequency - Inverse Document Frequency (TF-IDF)

For coding files, each feature set has one jupyter notebook. In this way, all 3 models are trained and tested across the same training and testing data, and the same preprocessing steps. (Note: We have 2 sample sizes for model experimentation, so notebooks have _10k & _100k suffixes for easier navigation.)

Source Code: https://github.com/ai-zhong/ai_and_jim_siads694

To illustrate our chosen models and reasons for selection:

- Linear SVM (Support-Vector Machines) Classifier: SVM is well known for its "kernel trick", which makes the computation process more efficient in high dimensional space, which is usually the case with text vectorization. Linear SVM works well if the 2 classes are separable, but fails when there

are items that belong to both class 1 and class 2. Also, the model takes more time to run with a large dataset.

- Random Forest: Being one of the most famous algorithms in the ensemble family, random forest outperforms other models under a lot of scenarios. Also, random forest is robust on handling big data as well as large feature variables. Furthermore, it doesn't require feature scaling as a preprocessing step, which can be useful when we combine different features (e.g. word vectors + simple numerical features). On the other hand, there are lots of parameters in this model, and it can be time consuming to do parameter tuning, given the current data size.
- Neural Network (MultiLayer Perceptron): It's well-known that neural networks can build very complex models, especially for large datasets. However, it can take a long time to train, and is easy to overfit. Therefore, we limited the complexity of the model by only adopting simple structures instead of using complex structures such as CNN or RNN or a large number of layers.
- BERT: BERT is built for Natural Language tasks and has been popular in recent years. This was a great opportunity for us to test an industry standard model and learn to not 'rebuild the wheel' when solving real-world problems. Since we only experimented with BERT, we did not tune the model or change any structure, but only made use of this pre-structured framework.

Features Selection:

- Simple Text Features Tested: Average Word Length, Average Word Syllables, Number of words in the document, and Number of Verbs in the document were each captured as a single feature for each document. The hypothesis was that these simple non-semantic features might correlate with perceptions of text complexity. **(4 features)**
- From External References:
  - The Dale Chall 3000 Word List is a 3000-word reference of basic English terms against which we compared individual documents to assess the document difficulty. Based on "The New Dale-Chall Readability Formula", the raw score for each document was calculated, and if the % of difficult words was greater than 5%, we used Dale Chall's adjusted score. **(1 feature)**
  - The Age of Acquisition reference is a 51,715-word file with multiple measurements of a word's attributes that potentially reflect the difficulty level of the word. The columns we chose to test for our data were: *Freq_pm, Letters, Nsyll, AoA_Kup_lem, and Perc_known_lem*. **(5 features)**
  - From Concreteness Rating (quantitative), document average was calculated based on 6 column measurements: *Bigram*, *Conc.M*, *Unknown*, *Percent_known*, *Total*, and *SUBTLEX*. Each was captured as single features (average for the document) and included based upon the belief that they are an indication of whether the sentences are simpler or more complex. **(6 features)**

  Note: It was correctly anticipated that some of the 16 features above would be highly correlated. We therefore used a correlation matrix heatmap to identify high correlations. We reduced it to 10 features, of which the highest correlation was *syllable_per_word* vs. *aoa_kup_lem* at 0.41. See Appendix A for correlation heatmaps and the final 10 features.

- Tf-Idf: Documents were tokenized, lemmatized, stop-words-removed, lowercased, and then featurized as Tf-Idf vectors. With the *max_features* set to be 10,000, documents were vectorized to a sparse matrix of size (n,10,000), where n is the number of documents. This can reduce noise as well as make the training process more efficient.
- GloVe (Pre-trained Word2Vec): We used 'glove.6B.100d.txt' glove vectors, where vectors are pre-trained on 6 Billion tokens with 400K uncased vocabulary, and a 100 dimension embedding for each word vector. We applied these vectors to each token in the document and calculated the average of all tokens' vectors. Lastly, each document was represented as a (1, 100) vector. If no words were available in that document, a (1,100) of zeros was used instead.

Our code was composed of a feeder file (pre_processing.py) that pre-process all documents and calculates all numerical scores for each of them. As mentioned, there are 4 feature sets and 4 notebooks for model running.

Parameter Tuning: Because we believed we would see greater performance improvement by focusing on feature selection, we spent more effort on feature combination selection than parameter tuning. Still, we then performed grid search on the selected RandomForest Model, and the best parameters were chosen based on model accuracy. See more in the Model Evaluation Section.

Challenges & Resolution: The main challenge was that running time for all models was long. To solve this problem, we first invested in Google Colab Pro for extra computation power. Secondly, we used samples from the training data instead of the whole dataset. Thirdly, we compromised even more when the number of features got large by using an even smaller sample size. Another challenge was to arrange the code properly to reduce repetitive preprocessing. We finally chose to save each numerical score calculated from external sources as a separate csv file, and reload them when needed.

Evaluation

Metrics: 1) Accuracy Score, 2) F1 Score and 3) ROC-AUC;

- Accuracy Score: Accuracy measures the proportion of correct predictions (both true positives and true negatives) versus the total number of cases examined. Accuracy is not a good metric when the 2 classes are imbalanced, because when there is a large difference between true positives and true negatives, accuracy score can stay high while the model performs poorly. However, in "*WikiLarge_Train.csv"*, we have a balanced binary class. Therefore, accuracy can serve as an ideal metric to evaluate model performance.
- F1 Score: F1 Score is a harmonic mean of precision and recall, which gives a better evaluation of the false negatives and false positives. This measurement is usually adopted when the classes are imbalanced. We included this measurement to have a well-rounded view of model performance, especially when the subsamples are slightly uneven between positive and negative classes.
- ROC-AUC: ROC-AUC curve is a performance measurement for classification at various threshold settings. ROC is a probability curve and AUC represents the separability of the 2 classes with the current model.The higher the AUC, the better the model is performing.

Evaluation Plots:

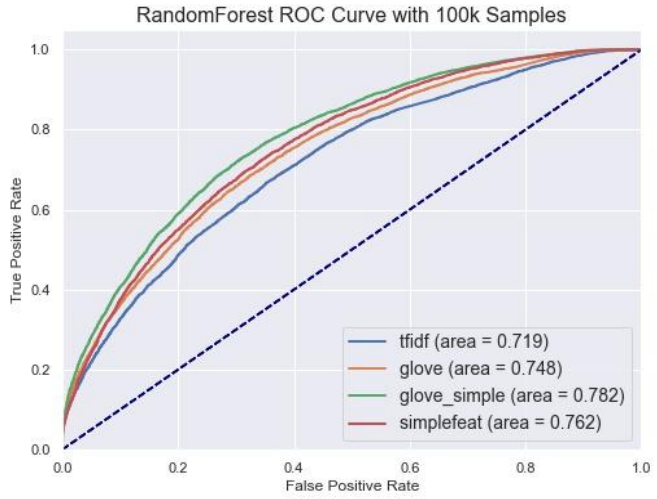
Due to resource and time constraints, we conducted most of our testing on a subsample of 100,000 and then trained the "best" model on the entire data set. However, an even smaller sample of 10,000 was used when SVM was introduced to the comparison (see below).

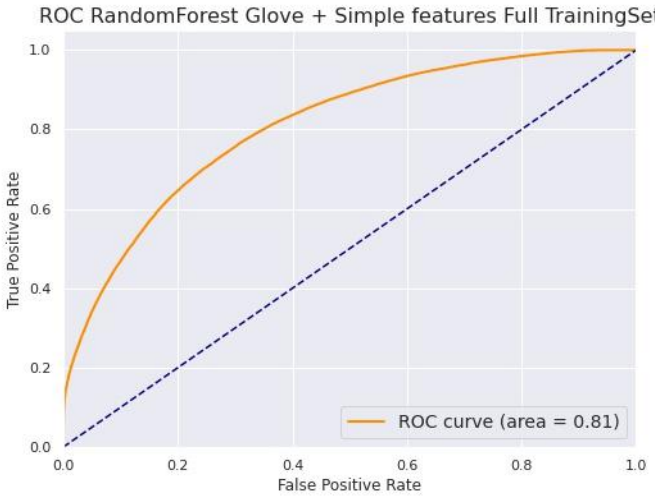
We made several comparisons as we evaluated our supervised models.

- First, we compared accuracy vs F1 scores. As seen in the Appendix B plots , F1 Scores do not tell a significantly different story than accuracy scores when looking across models, which is expected under a *balanced* binary classification problem. We therefore focused on accuracy going forward.
- We compared across models. As seen below (and in Appendix B), Random Forest had higher accuracy compared to MLP models for the four feature sets. Because SVM is so inefficient with large datasets, we chose an even smaller sample of 10,000 to run across all 3 models for final comparison (though we first confirmed that 10k &100k yielded similar results - see Appendix C).

Under the small sample environment, Random Forest still performed better than the other two models overall. Therefore, a more detailed comparison of this model's performance across different feature sets was calculated with ROC-AUC curves plotted in the section below.

Feature Sets Evaluation with ROC-AUC Curve**:** From the lower left ROC-AUC curves, we see that Glove+Simple Features gave us the highest AUC amongst all of the features - 0.78 . Therefore, our "best" final model and feature choice was RandomForest with Glove + 10 Simple Numerical Features. When trained on the full "WikiLarge_Train" (with train test split of 7:3) data, it yielded an AUC of 0.81.





Randomized Grid Search: After choosing the best model and variables, grid search was carried out for optimization. Due to time limits, we limited the grid search range to *n_estimators* = [300, 350, 400, 450, 500], and *max_depth*=[30, 40, 50]; The grid search was carried out with 100,000 samples. Mean test score (below) didn't change drastically with the changes of *n_estimator* and *max_depth*.

n Estimators	Max Depth	Mean test accuracy score	STD of Test Score	Rank of Test Score
500	40	0.69832	0.00094	4
350	50	0.69833	0.00167	3
300	30	0.69848	0.00204	2
450	50	0.69923	0.00281	1
300	40	0.69816	0.002	5

Feature Importance: After training the RF model with the full training data, we evaluated feature importance. As the input is a concatenation of numerical features and GloVe vectors, we can at least rank the importance of numerical features. Top 10 features are at right.

Random Forest Feature Importance		
feature index	Importance	feature
8	0.09755	dale_chall_score
0	0.02405	aoa (AoA_Kup_lem)
9	0.01681	conc_subtlex_score
6	0.01392	syllable_per_word
2	0.01201	verb2 (count of verbs in a sentence)
7	0.01164	conc_mean_score
79	0.01162	GloVe features
4	0.01127	aoa_perc_known_lem
65	0.01094	GloVe features
68	0.01043	GloVe features

Tradeoffs and Sensitivity: With the current parameter range, RandomForest seemed less sensitive to changes in *n_estimator* but more sensitive to *max_depth*. With more time, a larger range of values (and larger data set) should be tested. This will require more time/ computation. Also, with larger *n_estimator* and *max_depth*, the model gave higher training scores but was prone to overfitting.

What About BERT?: The BERT model is not listed with the above models because it has its own preprocessing steps and, further, tensorflow wraps features into a preprocessing model so that we do not need to worry about the detailed steps. Obviously this is a double-edged sword. It modularized the work and made everything smooth, but it made the model more black box and hard to debug. In this project, we trained the BERT model with the whole training set, predicted on the test set, and submitted to Kaggle for accuracy score evaluation. BERT achieved an accuracy score of 0.789. (*reference: https://bit.ly/3c2CiPI*). To give a fair comparison with the other 3 models, BERT was used on the testing data from the full training dataset with a split of 7:3 (same as final Random Forest). The accuracy score is 87.02 and the f1 score is 87.64, compared to accuracy score of 72.85 and f1 score of

72.97 from RandomForest. The ROC-AUC curve was plotted and attached in Appendix B. (*reference: https://bit.ly/3c148eW*). BERT outperformed our "best" traditional model, but is more black-box.

Analysis of Failures: *(We created evaluation of failure models in the supervised notebooks)*

1. Overfitting/Underfitting (*reference*: *https://bit.ly/2SGUl6R*): With RandomForest, it is easy to overfit when the *n_estimator* and *max_depth* are large, especially with a small subsample size (10k). We found the model to perform relatively poorly (fail) at some parameter choices versus others. With iterative testing, *max_depth* influences the model more than *n_estimators*. When *max_depth* increases, the training accuracy score increases, but the testing accuracy doesn't seem to always increase. To illustrate, when *max_depth*=10, training score is 0.7815, while testing score is 0.669; when max_depth=15, training score increases to 0.88, and testing score stays the same. On the other hand, when changing n_estimator from 50 to 200, both training scores and testing scores changed minimally.
2. Feature Scaling:*(reference: https://bit.ly/3uyFPvh*) For models like SVM and MLP, it's better to scale the numerical features with StandardScaler() or MinMaxScaler() before feeding in the model. Scaled and unscaled inputs were tested with the MLP model, and the accuracy score decreased from 67.18 to 61.27 (more failures of prediction), which is a significant change.
3. Model Selection (SVM): We selected SVM as a candidate model because it is known for efficiency and handling large numbers of features. However, we found SVM to be inefficient with a large number of data points, because kernelized SVMs try to compute the distance between every point. Therefore running time and required memory blows up when data points increase. *Linear* SVM can speed up the process, but unfortunately we chose Sklearn's *sklearn.svm.SVC* instead of *sklearn.svm.LinearSVC*. Though LinearSVC is said to be similar to SVC(kernel='linear'), their implementations are actually different. LinearSVC is implemented with liblinear, compared to libsvm with SVC. Liblinear is more flexible and scales better to large data points.
4. Tf-Idf: This feature set gave us relatively poor results (more prediction failures), which might be due to lack of preprocessing. As with Tf-Idf vectorization, the input matrix has a large number of features. A more proper way is to do dimension reduction first and then feed into the model. When feeding the original vectorized data into the model, the large dimensions can give poor results.

Unsupervised Learning - Wikipedia Topic Modelling

Motivation / Background

Without prior knowledge of the contents of the set of Wikipedia documents, we thought it would be interesting to use unsupervised learning methods to cluster the documents by topic in semantic space. In doing so, we could identify whether there are strong topic clusters (at what cluster counts), what topics emerge, and which models succeed best at this task.

Data Source

We used the same 119,000-document Wikipedia data set from our supervised evaluation. Since this was an unsupervised task, we only used the *text* here.

Unsupervised Learning Models Chosen

Two methods were selected: (1) Tf-Idf vectorization with K-Means and (2) CountVectorizer vectorization with LDA. LDA was chosen because it is known as the go-to method for topic modelling (a probabilistic generative model using soft clustering) and generally produces good results. The combination of Tf-Idf vectorization / K-Means was chosen because Tf-Idf is known as another effective method of vectorization into semantic space and K-Means (a hard clustering algorithm) works well with sizeable data sets. Our research indicated that K-Means/Tf-Idf topic modeling usually performs similarly (but not as well) as LDA.

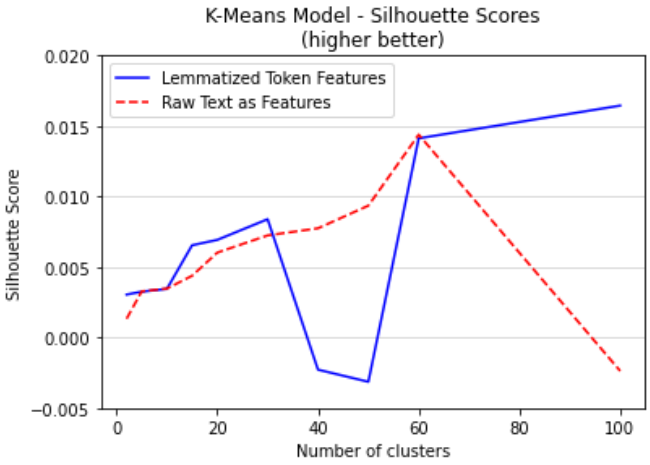
K-Means Methods and Evaluation

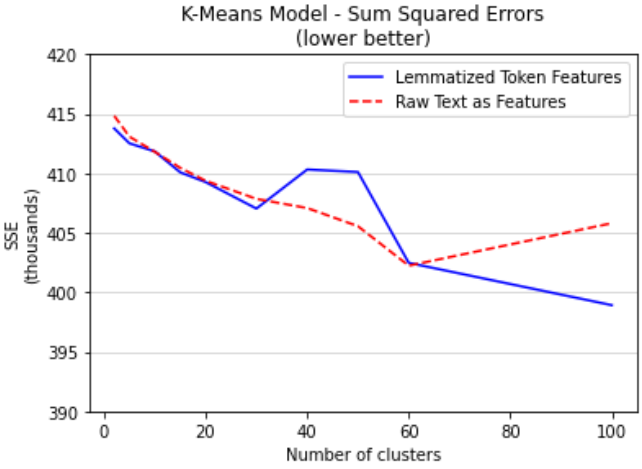
Source Code: https://bit.ly/3fvkoH9

Approach: Like our supervised analysis, we tokenized and lemmatized text of each document with a pre-processing file (pre_processing.py). Stop word removal and TF-Idf vectorization were performed on joined tokens of each document. Vectorized documents were clustered with K-Means.

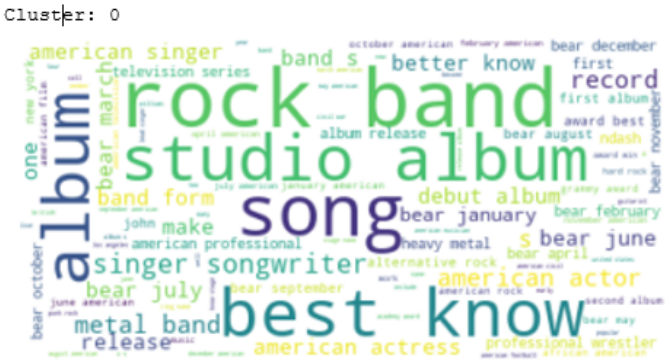
Feature Selection/Tuning and Evaluation:

- As a feature-selection effort, we applied Tf-Idf to the full document text strings (untokenized, unlemmatized, stop-words-included) and compared it to the results of the tokenized / lemmatized / stop-words-excluded joined text described above. Human evaluation of word clouds showed that the latter tokenized/lemmatized approach created more semantically cohesive results. Silhouette Score and Sum-of-Squared-Errors (SSE) evaluation for the un-lemmatized text were slightly worse than the lemmatized text at clusters below 40, but above that the data was less conclusive (see plot below for comparison). We focused on lemmatized tokens for the rest of the analysis.
- For parameter-tuning, K-Means was run at clusters ranging from 2 to 100 clusters and a Silhouette Score and Sum-of-Squared-Errors (SSE) were collected for each. All Silhouette Scores were small and SSE were high. Even though each improved marginally with larger cluster sizes, they still remained at levels indicating that, by those measures, they were poor clusters. No discernible "elbows" (inflections) emerged in the plots. (see plots below)
- Word Clouds were generated at multiple cluster levels (5, 10, 50, 100) and, despite poor Silhouette/SSE scores, the clusters' extracted terms often made good semantic sense to a human evaluator. A cluster count of 10 appeared to create semi-clear groupings and is used as an example. (Two examples below; All 10 word clouds in Appendix D). The two example word clouds demonstrate groupings around topics such as "music" and "football".





Example K-Means Clusters (from 10-cluster model):



Challenges: The biggest challenge of our K-Means approach (aside from poor Silhouette/SSE metrics) was that iterating through cluster size models and calculating Silhouette/SSE metrics took 1-2 hours per iteration (initiated two at a time due to system time out). Because the Tf-Idf matrix had 1.6M features, we tried SVD reduction to speed up the process, but it did not improve speed. The *max_iter*

parameter was also reduced, but created limited time improvement. We did not want to reduce the data set size for any of the unsupervised modelling, so we accepted the longer run times.

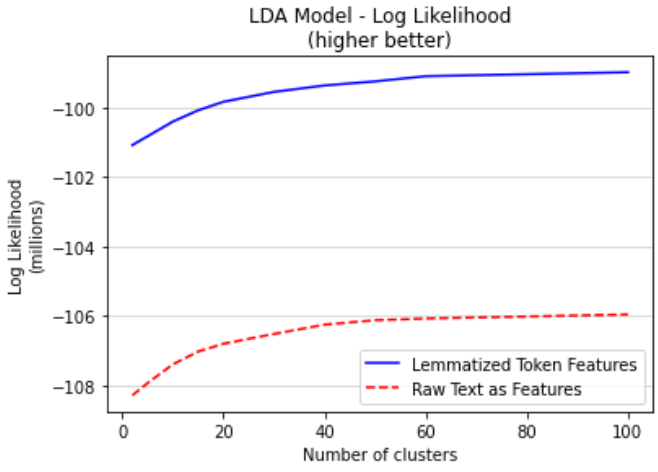
LDA Methods and Evaluation

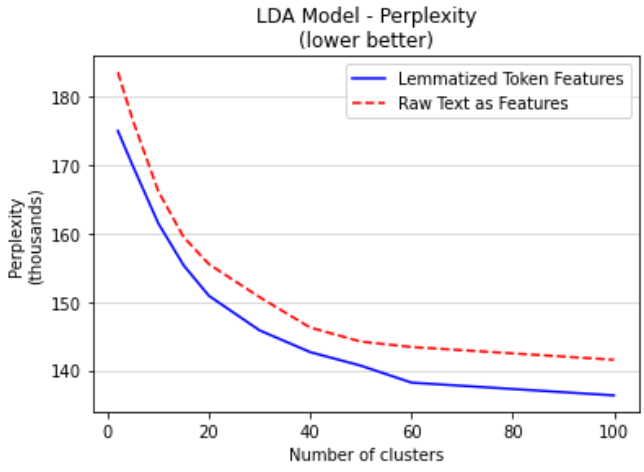
Source Code: https://bit.ly/3vEWsXO

Approach: Similar to our K-Means analysis, we loaded in the tokenized and lemmatized text of each document from a pre-processing file (pre_processing.py). Stop words were removed and CountVectorizer vectorization was performed on the joined tokens of each document. The vectorized documents were then clustered using LDA.

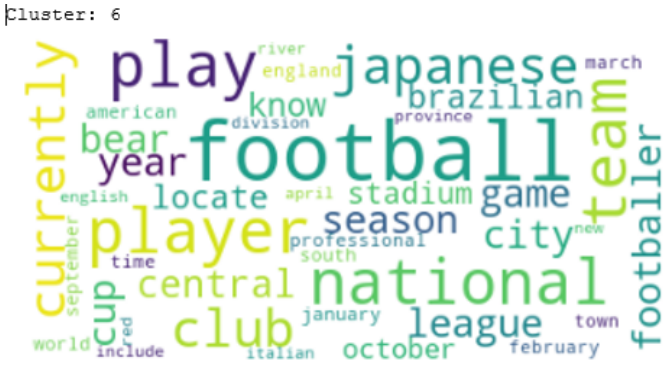
Feature Selection/Tuning and Evaluation:

- As a feature-selection effort, we applied CountVectorizer to the original full document text strings (untokenized, unlemmatized, stop-words-included) and compared it to the results of the tokenized / lemmatized / stop-words-excluded joined text described above. Through word cloud evaluation it was apparent, like with K-Means, that the lemmatized tokens provided more semantically cohesive clusters. Log likelihood and perplexity evaluation for the un-lemmatized tokens were roughly 5% worse (see plot below). We therefore used lemmatized tokens for the rest of the analysis.
- For parameter-tuning, LDA was run at cluster sizes ranging from 2 to 100. Unlike with K-Means, silhouette scores and SSE are not commonly utilized to evaluate LDA cluster quality. It was determined that log likelihood and perplexity are preferred measures for evaluating LDA and therefore both were collected for each. All log likelihood metrics were small and perplexity metrics were high. Both improved marginally with larger cluster sizes, and both appeared to converge to a steady-state value once they approached about 60 clusters. The actual values indicate that, by these metrics, these were not high-quality clusters. (see plots below)
- Additional LDA tuning was performed with the alpha and beta parameters. It was found that the default model performed better than with higher values of alpha (which assumes documents are composed of more topics) and higher values of beta (which assumes topics are composed of a large number of words in the corpus). (see Appendix G for plots)
- Word Clouds were again generated at multiple cluster levels. Despite poor log likelihood / perplexity metrics, the clusters' extracted topics again made good human sense. A cluster count of 10 created semi-clear groupings and is used as an example. (Two examples below; All 10 cluster word clouds in Appendix E). The two example word clouds demonstrate groupings around topics such as "football" and "weather". Of note: It's not surprising that humans recognized topics despite perplexity scores, since perplexity is known to often not correlate with human evaluation.
- PyLDAvis was utilized as a visualization and cluster quality evaluation tool. All 10 examples are in Appendix F, but the sample below is for the "US City"-themed cluster. PyLDAvis shows the same tokens as WordCloud, but also shows that the top 5 tokens represent a disproportionate share of the terms in this example. It also identifies that, with 10 clusters, documents are relatively equally distributed (as shown by approx equal diameters). Lastly, in the dimensions of the two principal components upon which the clusters are plotted, there is no overlap and clusters are well-distributed. This tells us the topics do not overlap and are easy to distinguish.

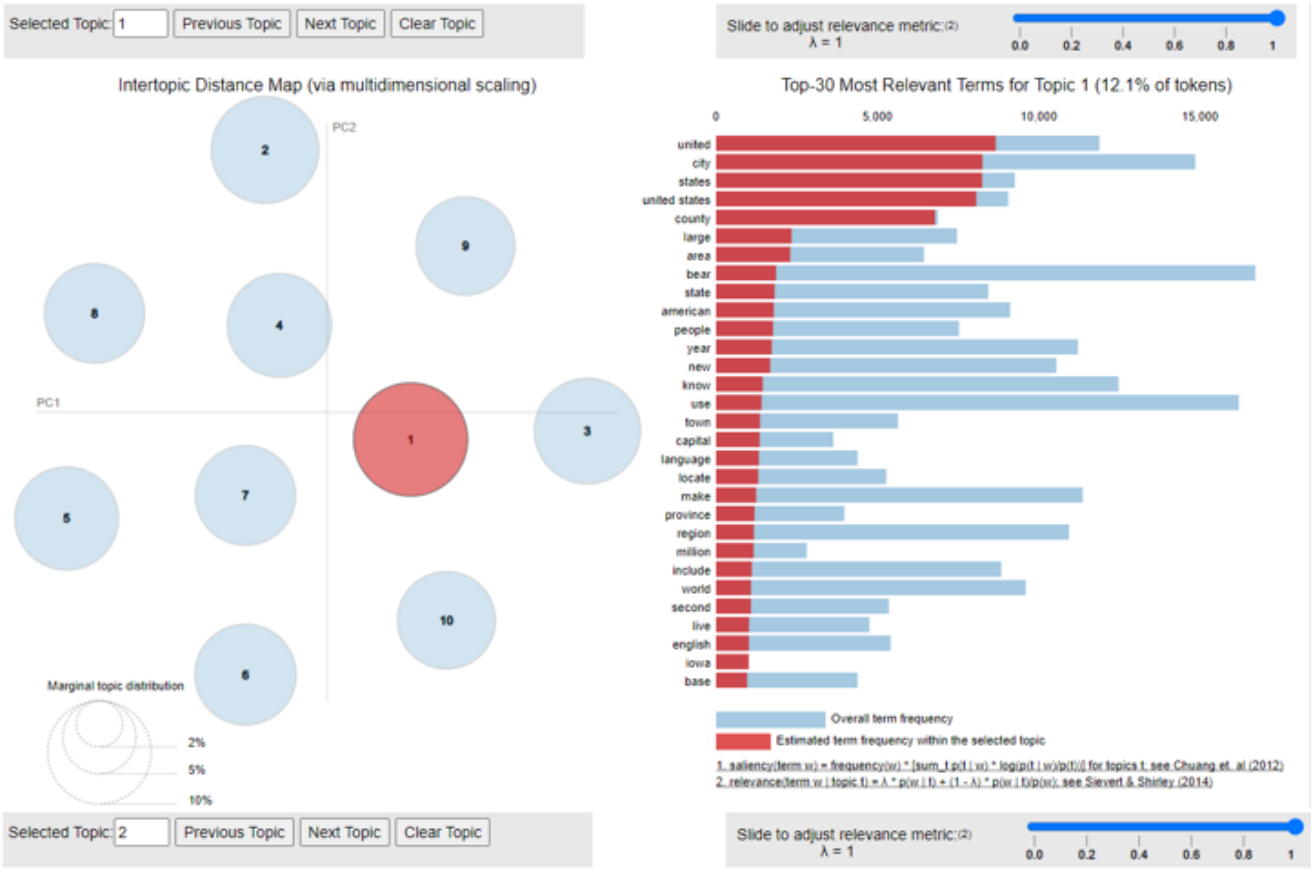




Example LDA Clusters (from 10-cluster model):



PyLDAvis Example



Challenges: Our challenges with LDA mirrored K-Means. Specifically, we encountered poor quality scores and long metric calculation run times for log likelihood and perplexity. System timeout during log likelihood and perplexity calculations limited each model run to three cluster sizes, but we chose to perform the analysis with multiple consecutive sessions rather than reduce the data set since we wanted a fair comparison with our K-Means approach.

Discussion

Supervised Model Learnings

Learnings:

- Based on what is described in the documentation of sklearn.svm.SVC: *"The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using LinearSVC or SGDClassifier instead, possibly after a Nystroem transformer."* Therefore, after spending a great deal of time running the SVM model we realized that our choice might need more time and computational resources. Things become different when data is large.
- Feature selection should be carried out as early as possible. Additionally, we should use all features at first instead of trying different feature columns from the external resources. Although some models are resilient to redundant features, good feature selection improves model performance across all models.
- Proper management of codes is critical especially with large datasets. Repetitive running of preprocessing steps can also consume a large amount of time. Whenever possible it's good practice to modularize the code, save intermediate files, and keep a clean work procedure.
- The best performance BERT model is not discussed in detail, since we are only experimenting without much understanding. In real life it is good not to 'reinvent the wheel' when there are resources available, but it also means when there is a problem you have no way to debug if you do not spend time to learn the real logic behind it.

Follow-on Work:

- Models may behave differently when data size is at a different scale. In this project we tried sub-sample sizes of 10,000 and 100,000. With more time, we would love to study how the model changes with different data sizes.
- Due to time constraints, randomized grid search was only carried out for the RandomForest model with a small sample size. We would love to apply grid search with more models and data, and finally compare performance.
- With more time, we want to study the BERT model more in depth, including how it preprocesses all the data and the mechanisms it adopts to make the process fast and modular.

<u>Unsupervised Model Learnings</u>

Learnings:

- Overall, it appears K-Means/Tf-Idf and CountVectorizer/LDA are feasible approaches to topic modelling and generate coherent topic clusters when interpreted by humans. This is despite the fact that traditional quality metrics (Silhouette/SSE for K-Means and Log Likelihood/Perplexity for LDA) returned values that, independently, would suggest poor model performance. With human evaluation as the standard for quality assessment, having a domain expert available to assess the clusters becomes important. Relying on human evaluation makes models more difficult to optimize.
- Both models continued to improve their quality metrics (Silhouette/SSE for K-Means and Log Likelihood/Perplexity for LDA) as cluster count increased. Despite this similarity, K-Means appeared to continue improving at a constant rate, while LDA appeared to converge at a steady-state value after about 60 clusters. Even though these metrics suggest greater quality with higher cluster counts, the utility of adding more clusters diminishes if the goal is to summarize a set with topics for an audience. Audience comprehension becomes difficult in excess of a dozen or so clusters (and eventually you reach one cluster per document, which is of little value).
- Despite expectations from literature that LDA would outperform K-Means, we did not find that. The poor metrics scores and differing metrics across models made it difficult. Word Cloud evaluation (same cluster counts and features) didn't reveal a clear winner.
- The discovery of French language was unexpected. Both models appeared to cluster French terms as two clusters. The clusters appear to be different French regions, but it is unclear if they are treated separately due to language or meaning. In PyLDAvis, topics 3&4 are the French topics and, notably, are at roughly opposite ends of PC2, which suggests they are different semantically (at least in French). But they also sit among many English clusters (significance of which is unclear).
- Models had overlapping topics upon which they clustered, but each also solely found unique topics. For example, both models identify "football", "United States cities", and "hockey" clusters. K-Means solely found "music" and LDA solely found "weather". Within the two French clusters, K-Means found clear "Northwest France" and "Southwest France", while LDA's clusters aren't as clear.
- This project demonstrated PyLDAvis' power, the ability to quickly evaluate (1) distribution of terms and clusters, and (2) cluster overlap. It was more valuable than expected.

Follow-on Work:

- With more time, we would try feature representations beyond lemmatized token and raw text variations. Specifically, we would try pre-trained word vectors, such as GloVe (but literature review indicates GloVe+LDA is less effective than the new LDA2vec).
- We would consider running higher cluster sizes to see if metrics improve. This would be especially interesting with K-Means, since our metrics didn't converge to a steady-state. Further, we would run K-Means models with multiple *random_seeds* initializations, and compare clusters across seeds, since varying K-Means initializations change clusters.
- We would attempt tuning more K-Means and LDA parameters to see if it improves quality metrics.
- Lastly, we would try model flavors, specifically K-Means Mini-batching (reportedly more efficient).

Ethical Considerations (for Supervised and Unsupervised):

Our unsupervised analysis involved clustering topics of public wikipedia articles, but no decisions (or publications) are being based upon it. Therefore, our clustering work would have limited ethical concerns. Our supervised work, however, is inherently making a judgement about whether an article needs to be simplified, which would impact whether it gets simplified. Ethical issues that generally arise (related to both supervised/unsupervised):

- Origins of the Wikipedia sample aren't clearly documented, including how, when, and who sampled it. Hidden selection bias may impact performance in specific applications.
- Wikipedia is an open-source project which allows anyone to contribute articles, with limited oversight. Therefore, the topics, points of view, and writing style are certainly biased toward individuals who would likely contribute to such a project. Therefore, models trained off the articles might perform best only on this kind of writing. As an example, it's unclear whether the models would perform effectively when applied to other languages.
- The generalizability of the models to serve the needs of broader communities will be impacted by feature engineering choices. For example, some featurization is dependent upon the language of the target text (like BERT). When applied to other languages it may be less effective, which could unfairly impact subpopulations that may benefit.
- More generally, the topic of "text created for language-learners" inherently resides in a policy arena filled with considerations of children/disadvantaged groups (e.g. native-born US children, the poorly-educated, immigrant populations). Ethically, one must ensure that the unique needs of all groups are being considered (no group is ignored) when this data and its conclusions are utilized.

Tradeoffs (for Supervised and Unsupervised):

There are multiple tradeoffs encountered in supervised and unsupervised learning.

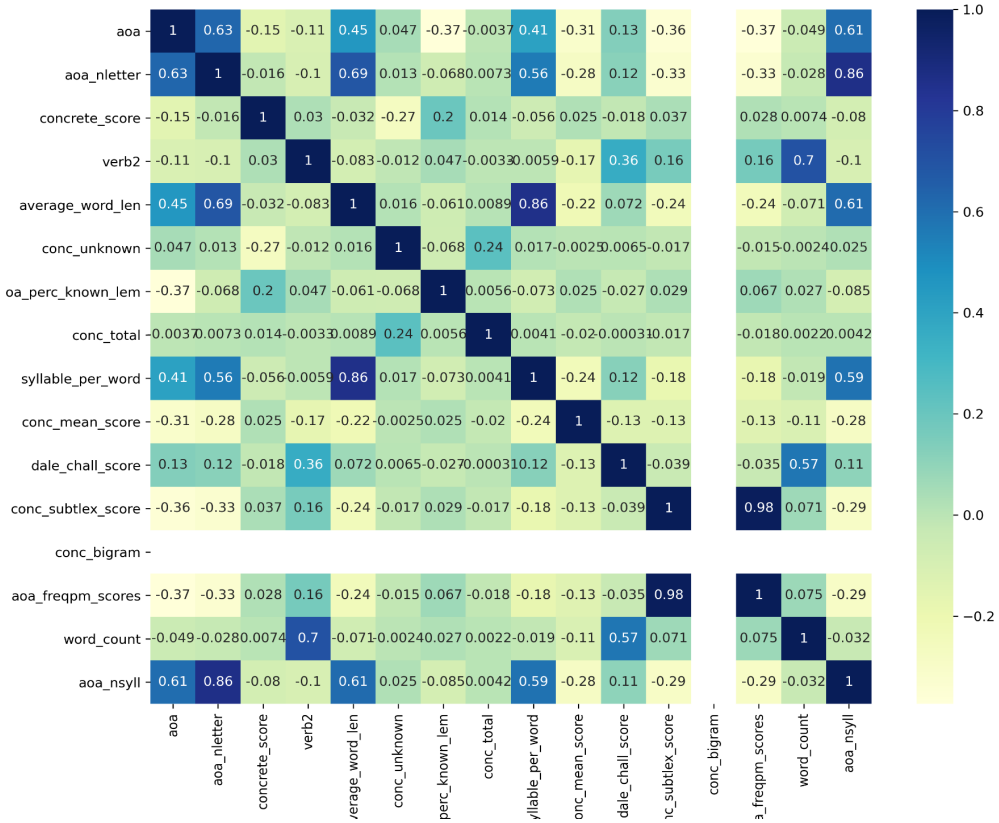
- One that we encountered was that of "accuracy-like" metrics (precision, recall, F1, etc.) by keeping all data versus the time and computational expense of using the full set. In our supervised work, we eventually chose to evaluate our many feature combinations on a subsample of the full data and then train the winning model on the full set. For our unsupervised work, though we didn't measure time differential, we may have reduced our computational time if we had reduced to just a subset of the data (but perhaps at some expense of topic cluster quality).
- A second tradeoff apparent in our work is between model performance and generalizability. As seen, two of our two highest-performing supervised models, BERT and Random Forest with Glove+Simple features, both relied upon language-specific featurization (both the BERT and Glove implementations were based upon English-only pre-trained files). We'd therefore achieve best performance on English test cases, but this means that our trained models will likely perform worse if they are fed non-English when used in the wild.
- We demonstrated that there is often a tradeoff between explainability and performance. Here, BERT outperformed Random Forest. BERT is a deep learning model with large weight matrices that is harder to explain, whereas Random Forest is more explainable (feature importance etc.).

Statement of Work

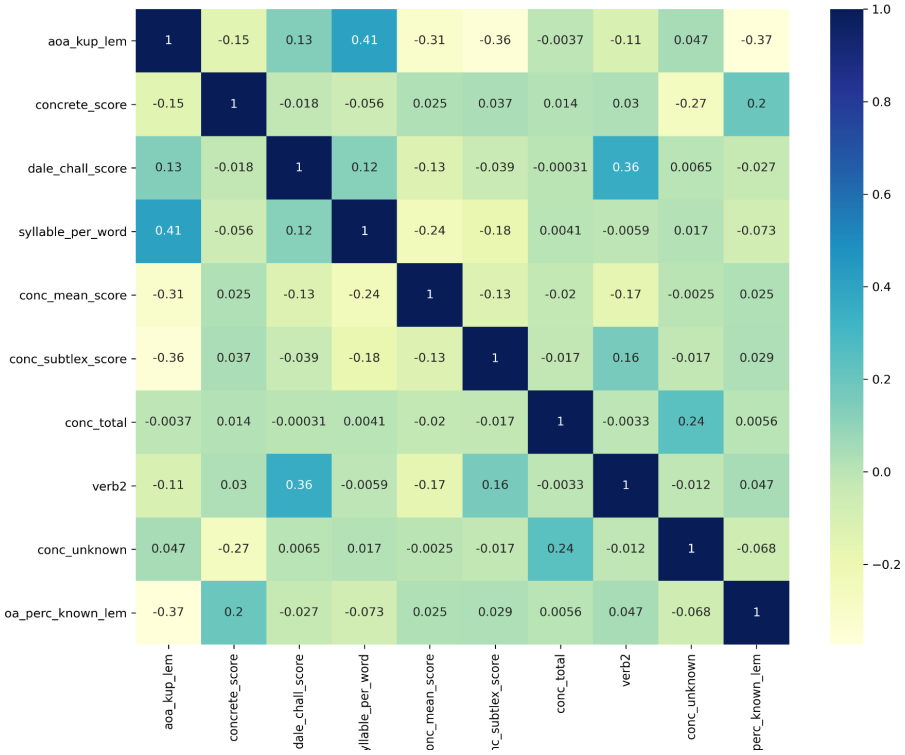
Ai and Jim collaborated through the project and each played roles in planning, modelling, visualization, and reporting. Though there were no pre-defined lines of responsibility, Ai performed more of the supervised learning work and Jim performed more of the unsupervised learning work. Regular meetings were held to update each other on respective work and provide feedback.

Supervised Modelling - Appendix A
Simple Feature Correlation

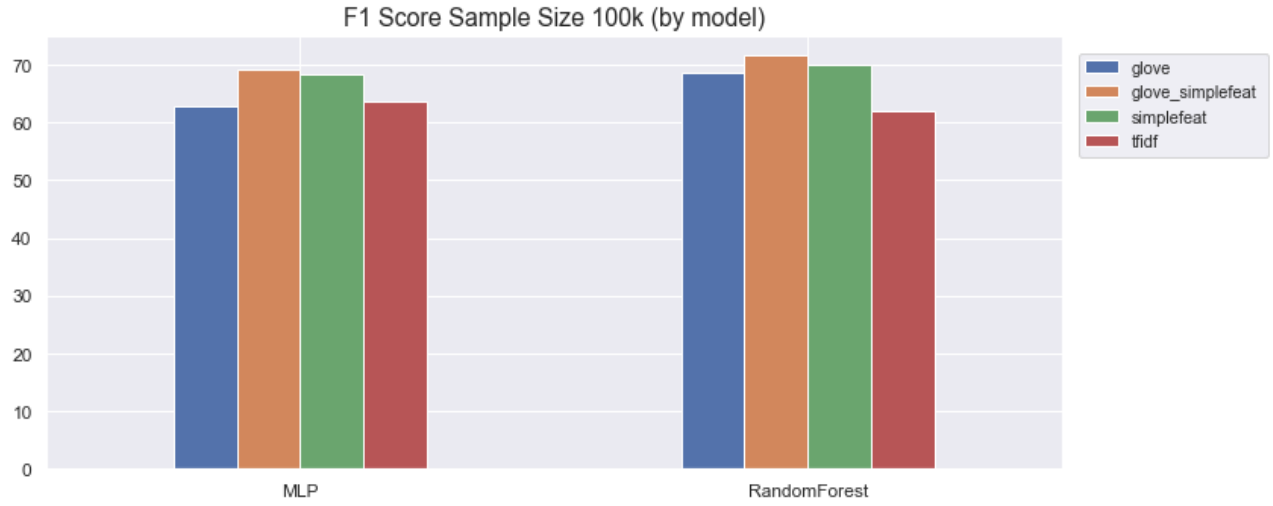
It was correctly anticipated that some of the 16 "Simple Text" and "External Reference" features above would be highly correlated (and the Bigram feature also seems to be Nan). We therefore used a correlation matrix heatmap to identify high correlations.



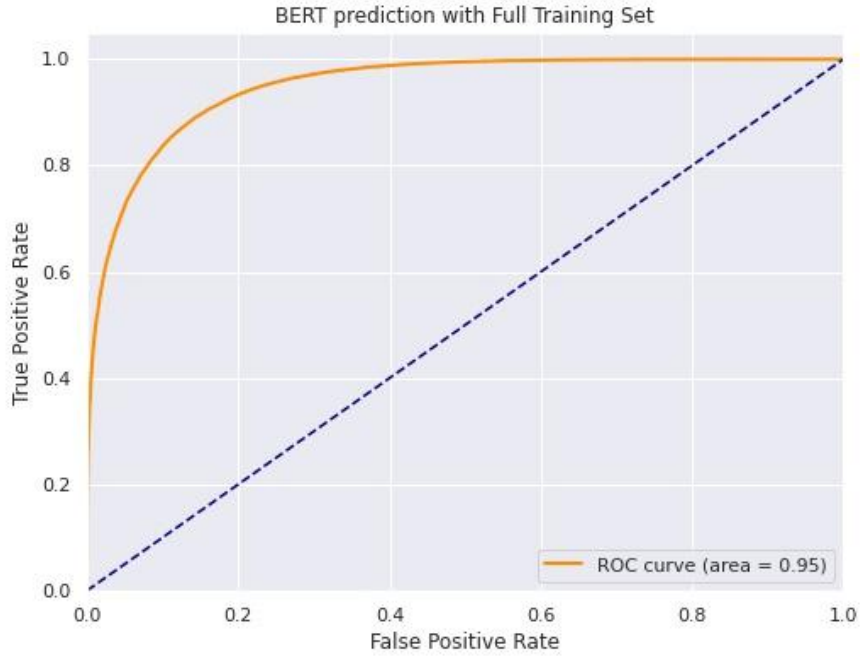
We therefore reduced to 10 features, of which the highest correlation was "syllable_per_word" vs. "aoa_kup_lem" at 0.41



Supervised Modelling - Appendix B
Accuracy & F1 Scores for 100K Samples for MLP and RandomForest



BERT ROC-AUC curve with 30 / 70 Split of Full Training Dataset



Supervised Modelling - Appendix C
Sample Size Comparison

Model Performance (with 100,000 samples) Accuracy and F1 Score Table

Model Performance with 100,000 Samples			
accuracy	**f1 score**	**model**	**feat**
64.06	62.810	MLP	glove
68.005	68.753	RandomForest	glove
63.825	69.253	MLP	glove_simplefeat
70.925	71.547	RandomForest	glove_simplefeat
67.185	68.275	MLP	simplefeat
68.965	70.105	RandomForest	simplefeat
63.77	63.777	MLP	tfidf
64.98	62.079	RandomForest	tfidf

Model Performance (with 10,000 samples) Accuracy and F1 Score Table

Model Performance with 10,000 Samples			
accuracy	**f1 score**	**model**	**feat**
60.100	58.567	MLP	glove
56.050	59.660	SVM	glove
64.750	66.122	RandomForest	glove
60.650	61.962	MLP	glove_simplefeat
62.150	57.013	SVM	glove_simplefeat
68.400	68.928	RandomForest	glove_simplefeat
65.550	66.795	MLP	simplefeat
65.200	63.900	SVM	simplefeat
67.100	68.213	RandomForest	simplefeat
57.250	58.596	MLP	tfidf
57.800	58.546	SVM	tfidf
60.700	58.147	RandomForest	tfidf

Unsupervised Modelling - Appendix D
Word Clouds for K-Means Modelling with 10 Clusters

Cluster: 0

Cluster: 1

Cluster: 2

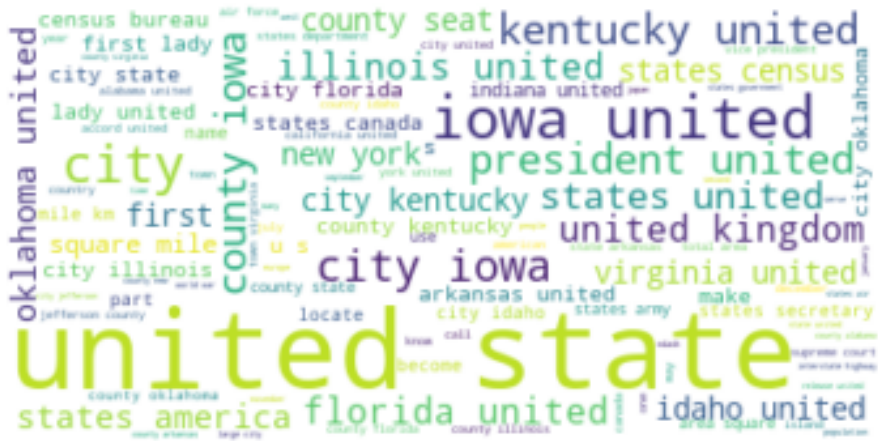
Cluster: 3

Cluster: 4

Cluster: 5

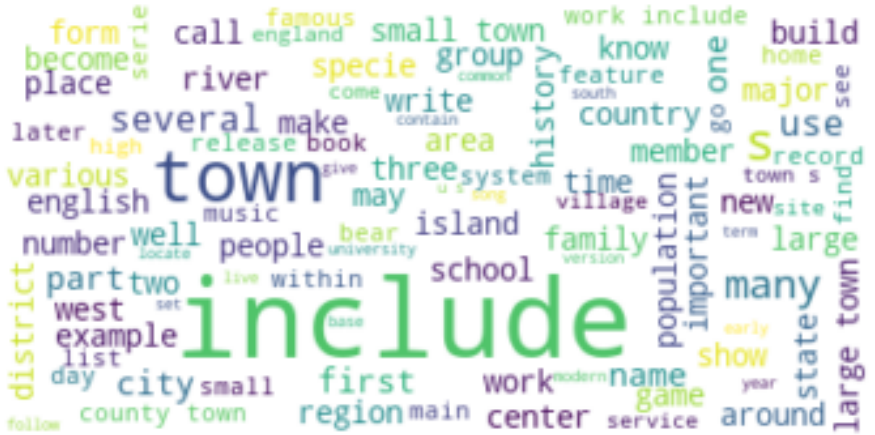
Cluster: 6

Cluster: 7



Cluster: 8

Cluster: 9



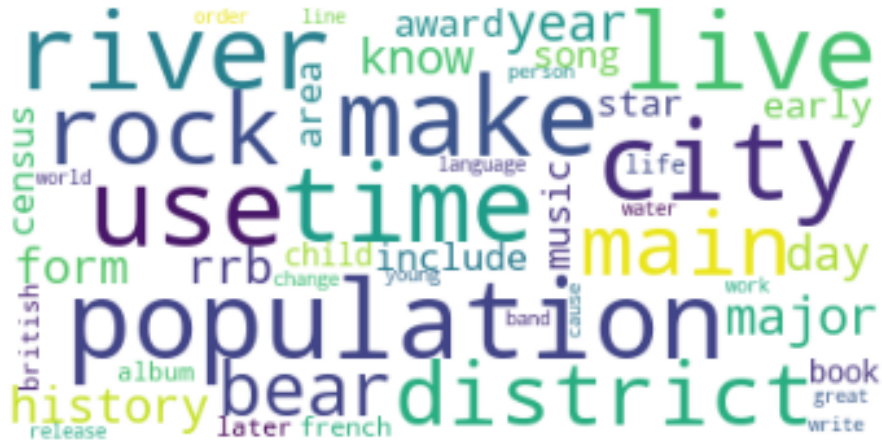
Unsupervised Modelling - Appendix E
Word Clouds for LDA Modelling with 10 Clusters

Cluster: 0

Cluster: 1

Cluster: 2

Cluster: 3

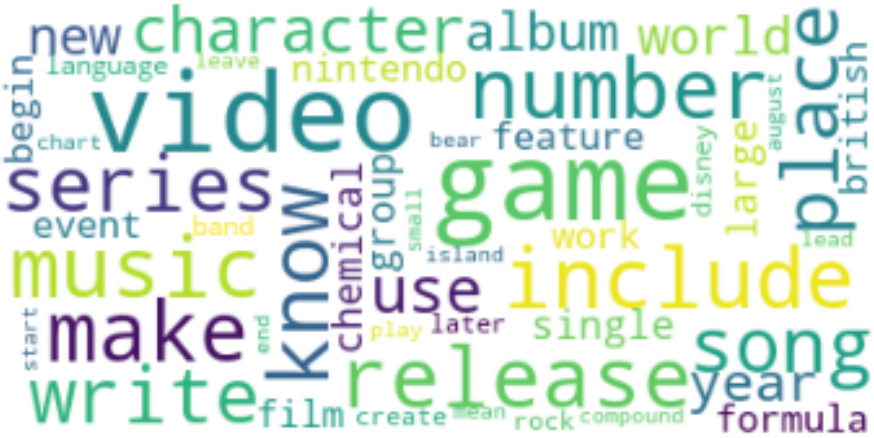


Cluster: 4

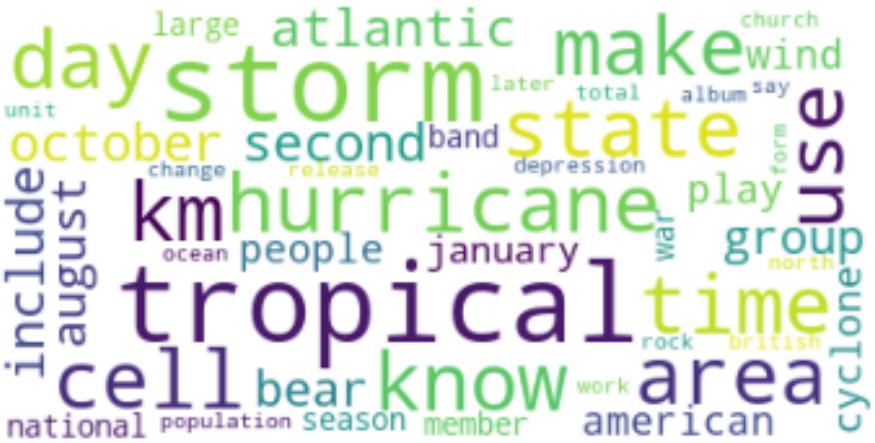
Cluster: 5

Cluster: 6

Cluster: 7

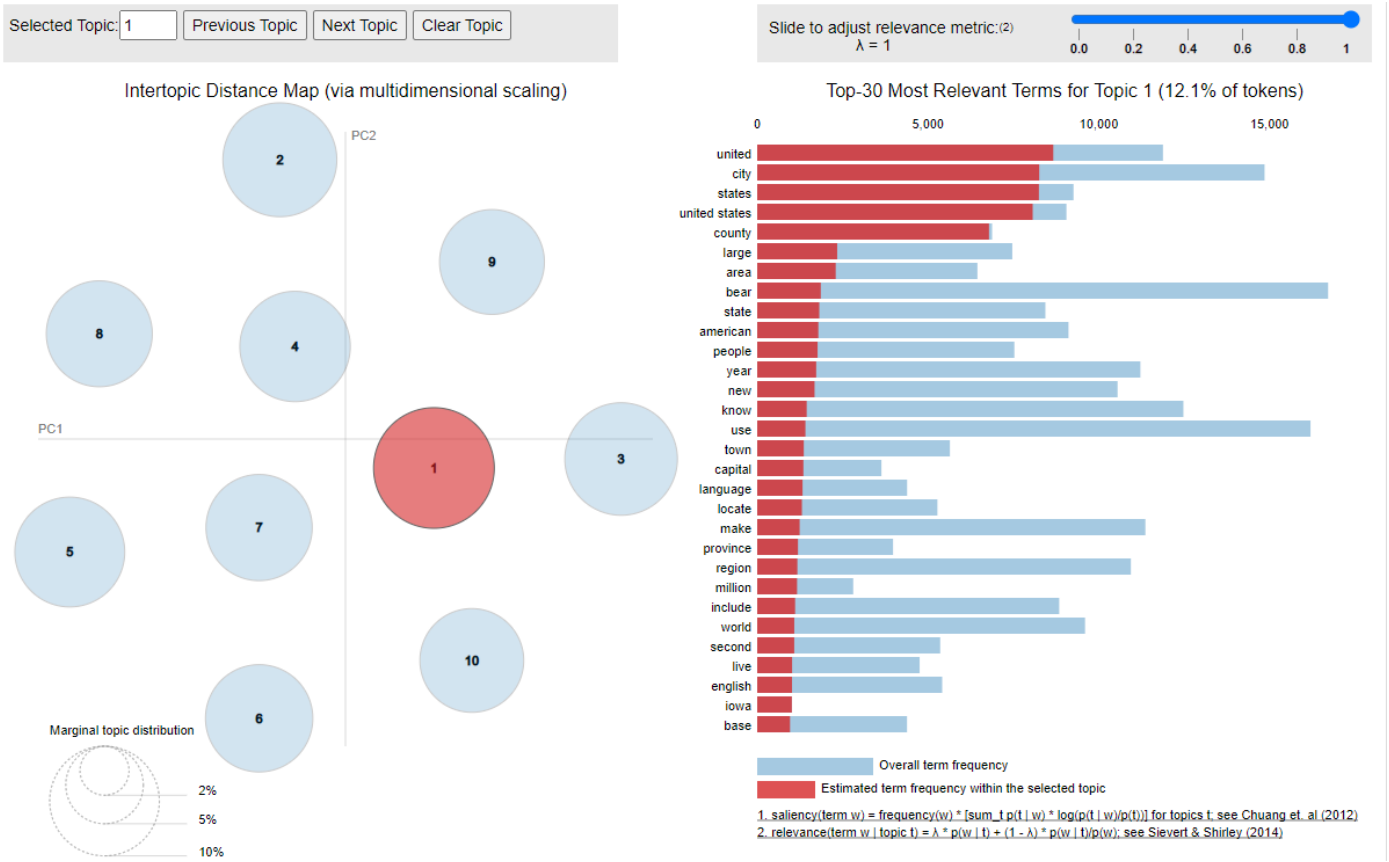


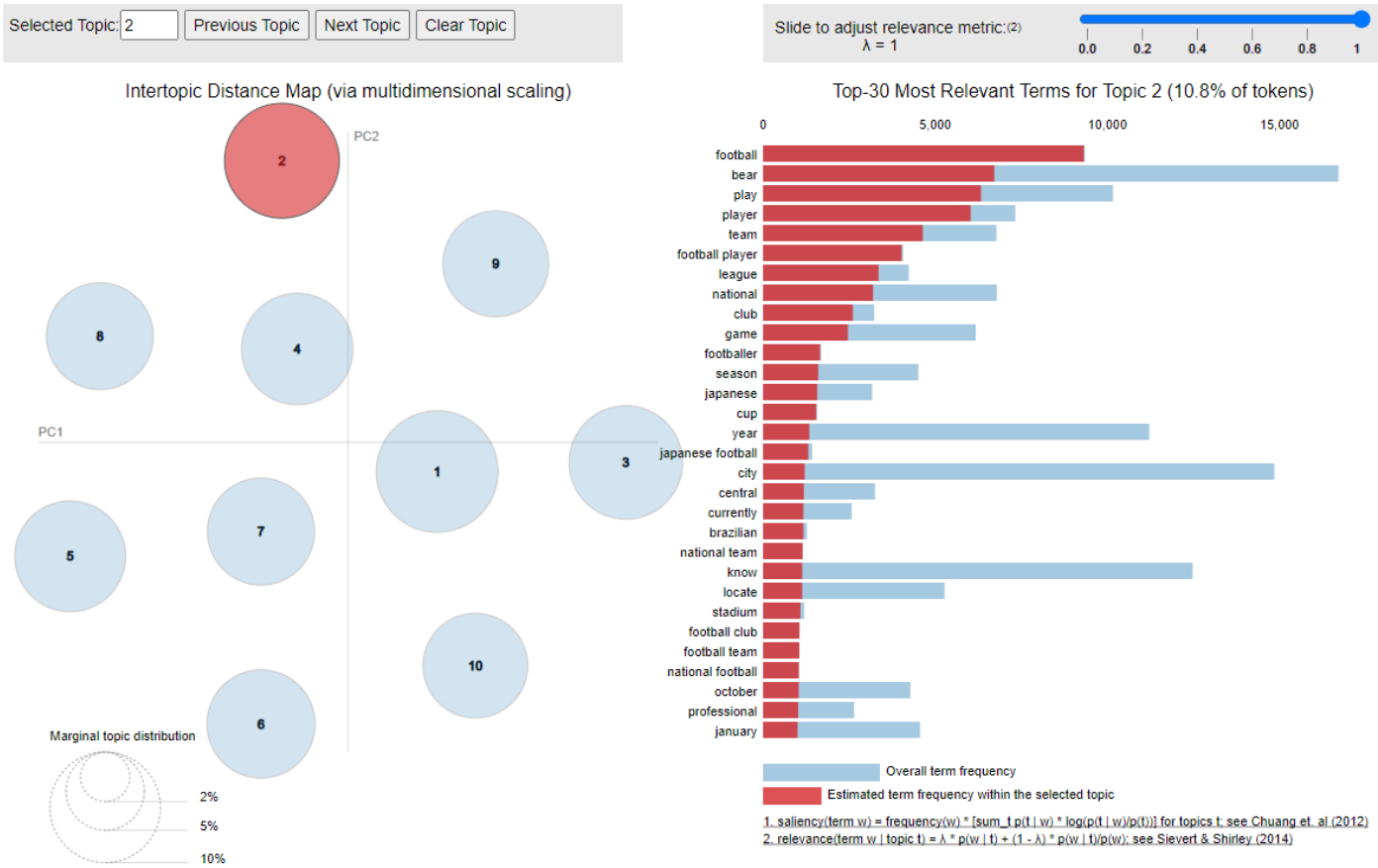
Cluster: 8



Cluster: 9

Unsupervised Modelling - Appendix F
PyLDAvis for LDA Modelling with 10 Clusters

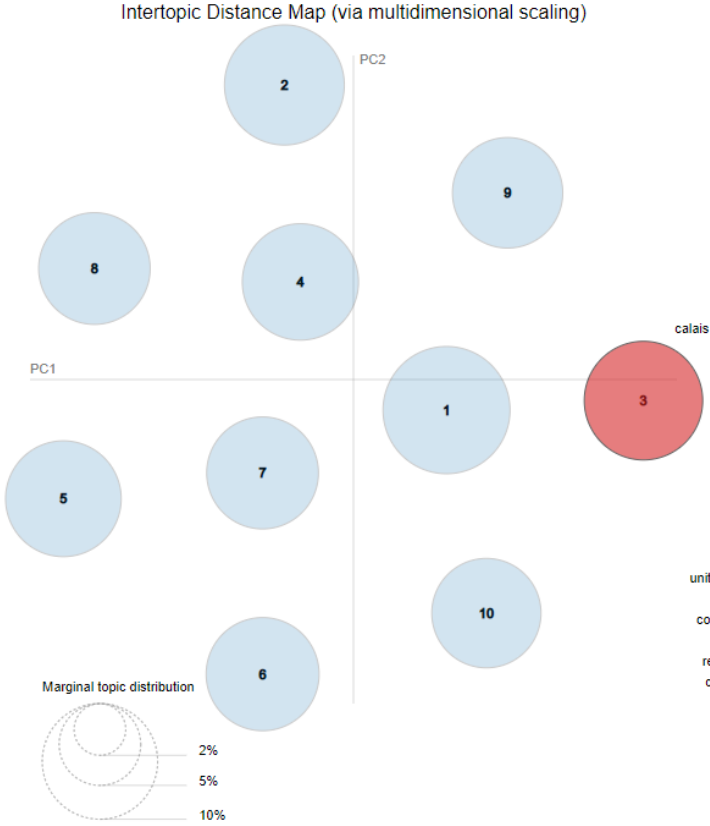




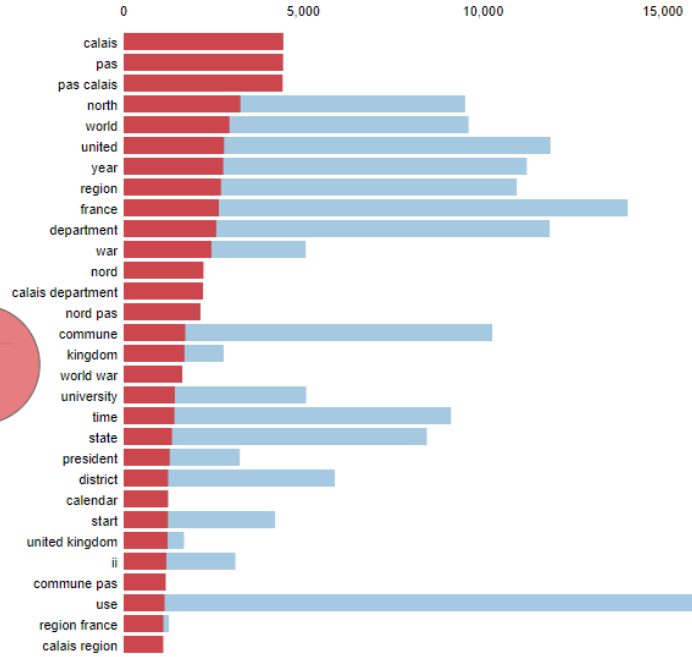
Selected Topic: 3 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)



Top-30 Most Relevant Terms for Topic 3 (10.5% of tokens)



Overall term frequency
Estimated term frequency within the selected topic

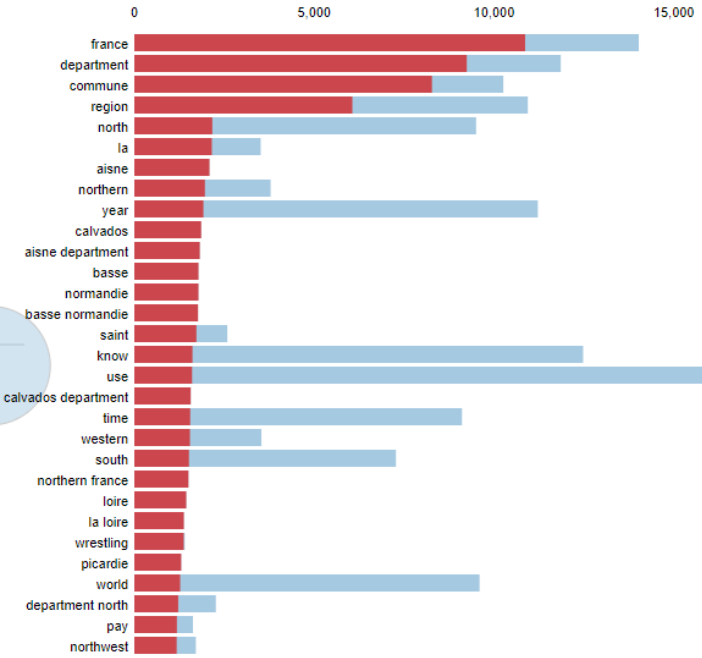
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Selected Topic: 4 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)
λ = 1
0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)

Top-30 Most Relevant Terms for Topic 4 (10.1% of tokens)



Overall term frequency
Estimated term frequency within the selected topic

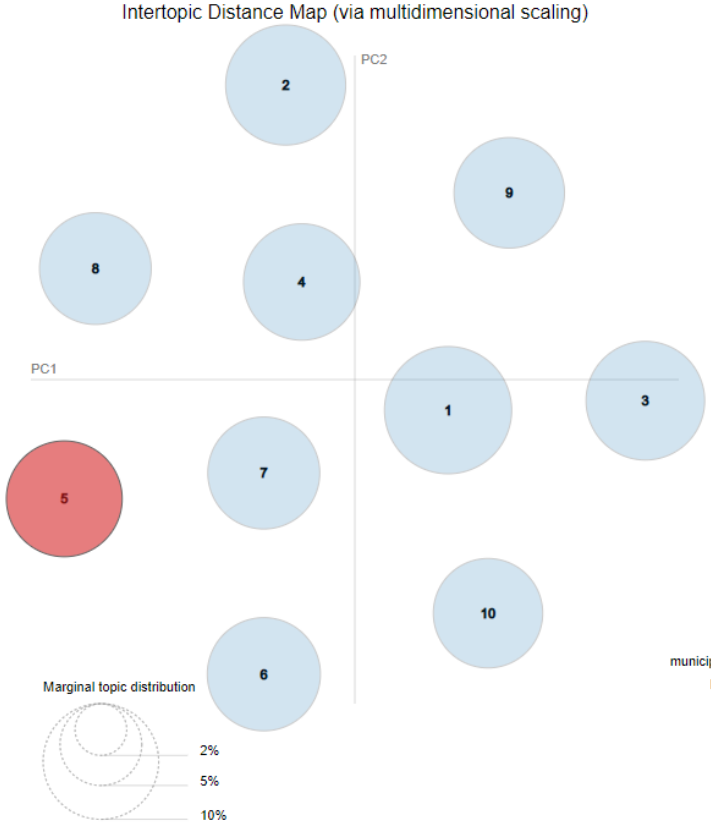
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Selected Topic: 5 | Previous Topic | Next Topic | Clear Topic

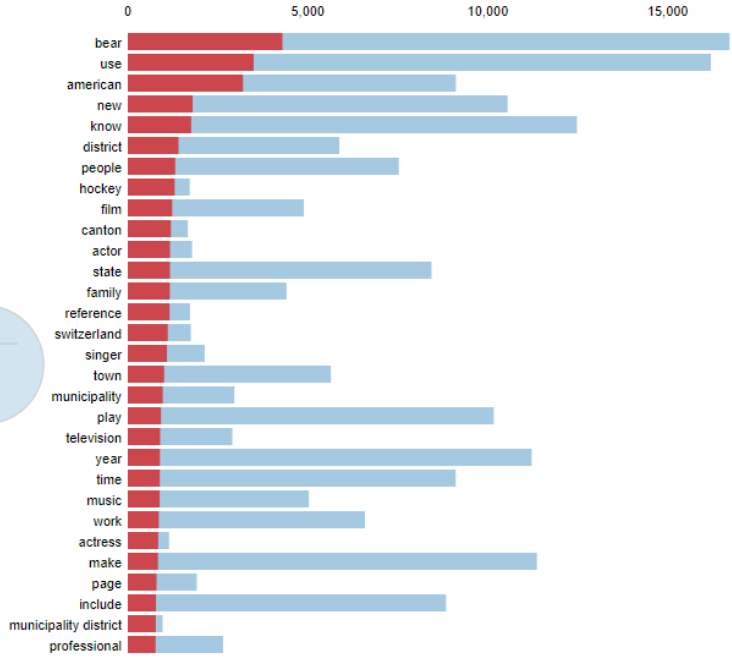
Slide to adjust relevance metric:(2)
λ = 1

0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)



Top-30 Most Relevant Terms for Topic 5 (10% of tokens)



Overall term frequency
Estimated term frequency within the selected topic

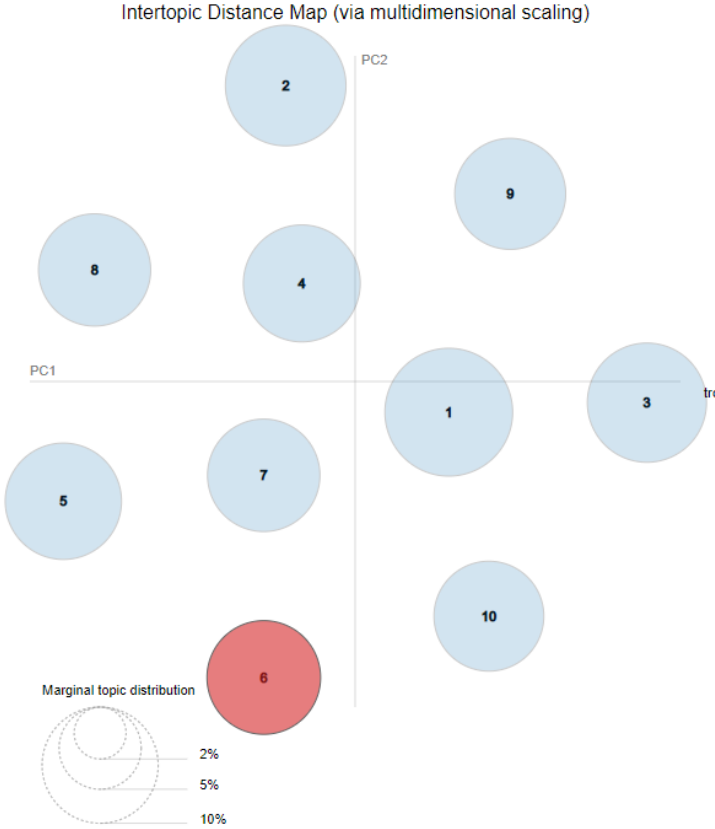
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Selected Topic: 6 | Previous Topic | Next Topic | Clear Topic

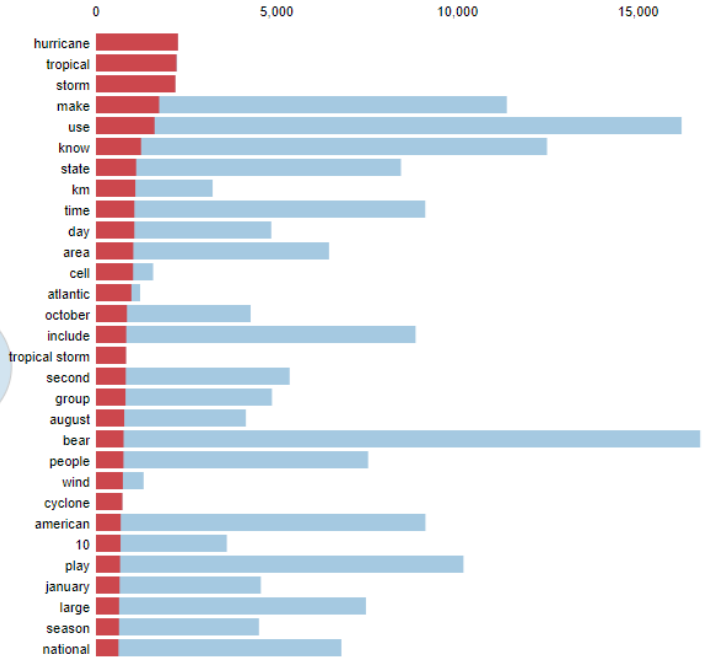
Slide to adjust relevance metric:(2)
λ = 1

0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)



Top-30 Most Relevant Terms for Topic 6 (9.6% of tokens)



Overall term frequency
Estimated term frequency within the selected topic

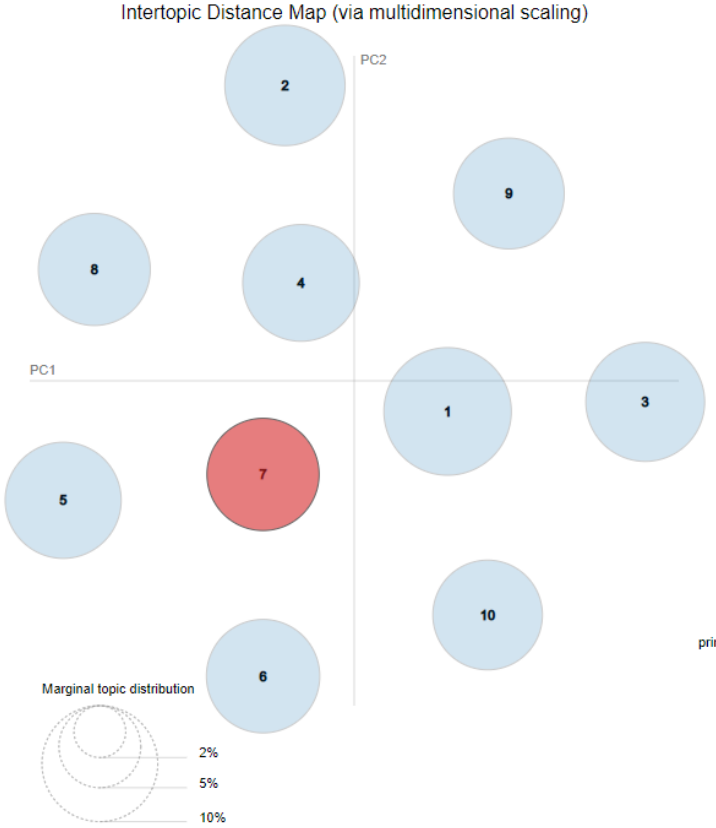
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Selected Topic: 7 | Previous Topic | Next Topic | Clear Topic

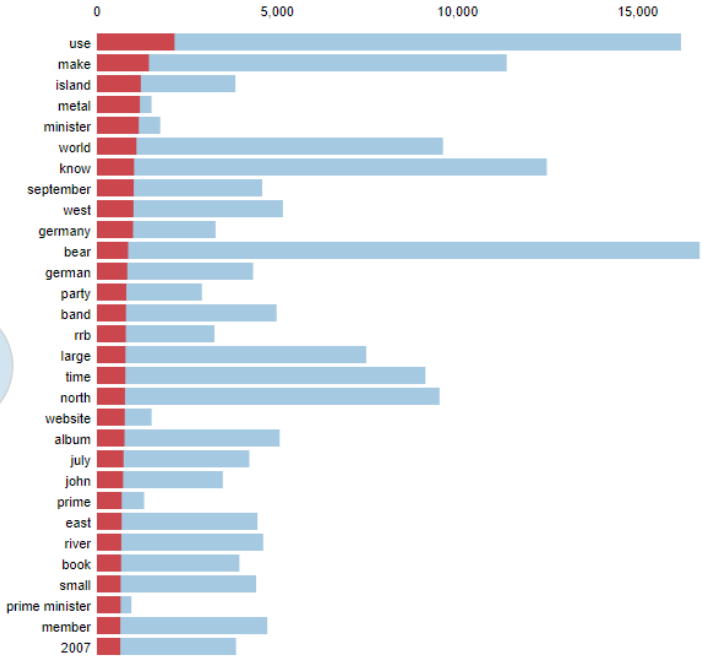
Slide to adjust relevance metric:(2)
λ = 1

0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)



Top-30 Most Relevant Terms for Topic 7 (9.4% of tokens)



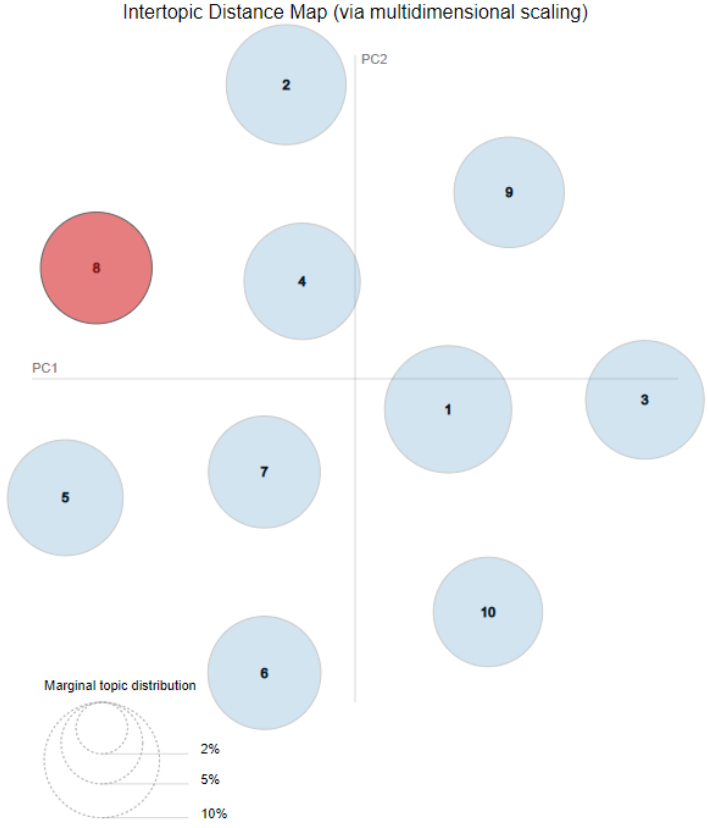
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Selected Topic: 8 | Previous Topic | Next Topic | Clear Topic

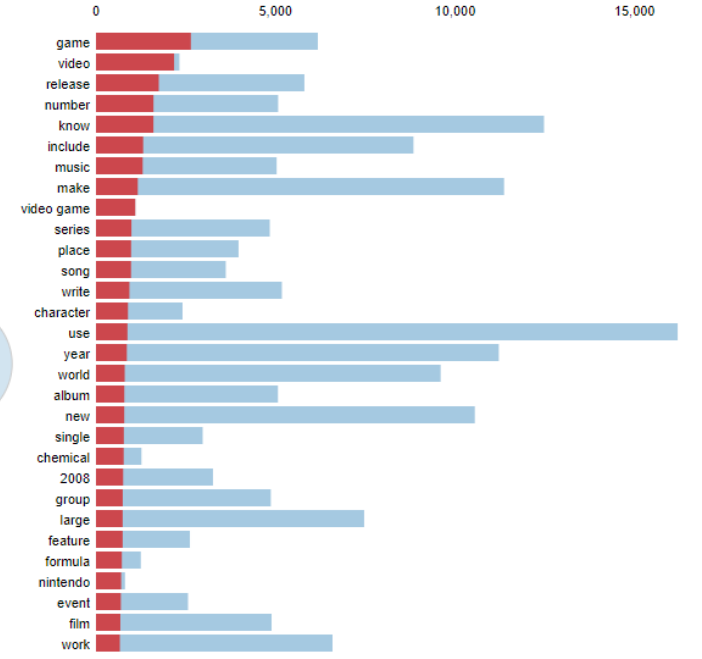
Slide to adjust relevance metric:(2)
λ = 1

0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)



Top-30 Most Relevant Terms for Topic 8 (9.3% of tokens)



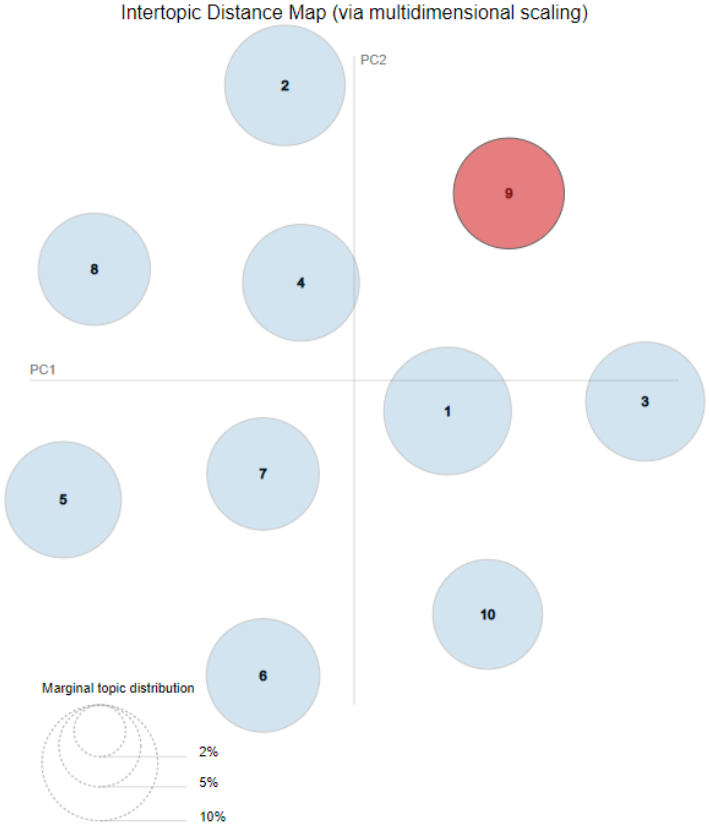
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Selected Topic: 9 Previous Topic Next Topic Clear Topic

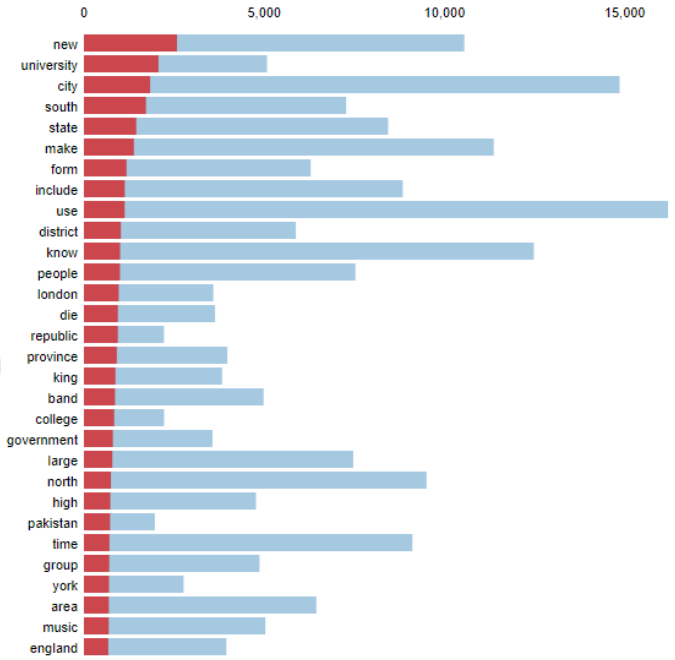
Slide to adjust relevance metric:(2)
λ = 1

0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)



Top-30 Most Relevant Terms for Topic 9 (9.1% of tokens)



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

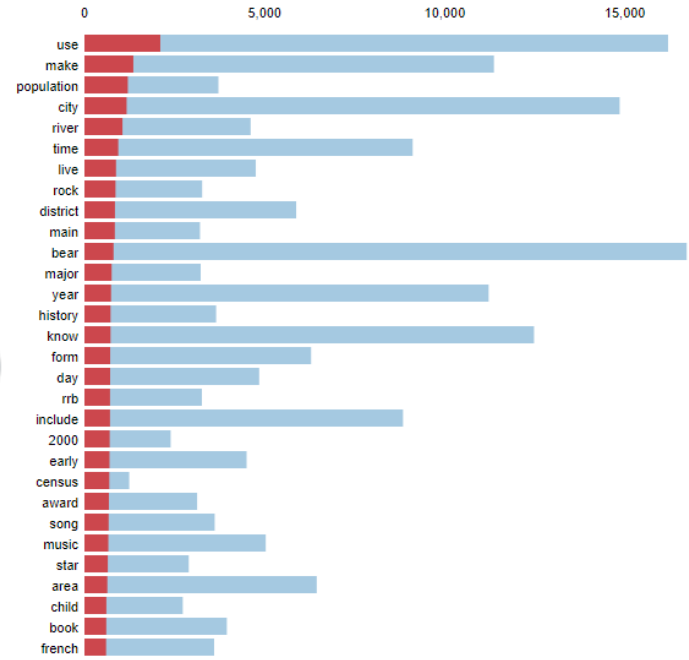
Selected Topic: 10 Previous Topic Next Topic Clear Topic

Slide to adjust relevance metric:(2)
λ = 1

0.0 0.2 0.4 0.6 0.8 1

Intertopic Distance Map (via multidimensional scaling)

Top-30 Most Relevant Terms for Topic 10 (9% of tokens)



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Unsupervised Modelling - Appendix G
Alpha and Beta Parameter Tuning for LDA Modelling with 10 Clusters

LDA tuning was performed with the alpha and beta parameters. It was found that the default model performed better than with higher values of alpha (which assumes documents are composed of more topics) and higher values of beta (which assumes topics are composed of a large number of words in the corpus). Default values for alpha and beta are 1/n_components, so in the evaluations below, the default (at 10 clusters) was 0.1. All value increases of alpha and beta reduced cluster quality.

[Note: alpha and beta were tested independently, such that one was held at the default value while the other was being varied.]