North American Industry Classification System (NAICS) Recommender System

Holden Miller-Schaeffer

North American Industry Classification System (NAICS)

The North American Industry Classification System System (NAICS) was developed for use by Federal Statistical Agencies for the collection, analysis and publication of statistical data related to the US Economy. NAICS is a Self-Assigned System; no government official assigns a NAICS Code. What this means is a company selects the code that best depicts their primary business. If the Business Activities include more than one Unique Line of Business, the company may want to use more than one NAICS Code.

The Business Case

NAICS codes are used across a variety of industries to evaluate business operations, bank loans, and insurance risks. Since the code is self assigned, it may be difficult for users to know exactly what code they are supposed to use. This recommendation system is useful for users to query a description of their business operations and see a list of codes that are relevant to their search. This removes the hassle and research required for looking up the codes on their own. NAICS codes are broken into subgroups, with the most granular classification being 6 digits. The aim of this recommender system is to return the most accurate results for the 6 digit classifications.

Why a recommender system?

When originally framing this problem, I concluded that the main goal was to use a query of the description of business operations as input data to predict the most accurate classification. I went in with the idea of building a classification model, but I realized quickly that collecting a large enough dataset with labeled data was nearly impossible for the scope of this project. I reached out to various organizations, including the NAICS Association directly, and discovered that business names, descriptions, and NAICS classifications were hidden behind a significant paywall. This left me with no choice other than to reframe the problem, and thus I decided to build a recommender system that could learn from user feedback, and theoretically converge at a perfect state, where every recommendation returned is relevant to the user.

The Data

The two main datasets used in this system are the 2017 NAICS Index File and 2017 NAICS Descriptions files found on the US Census NAICS website. The 2017 NAICS codes were chosen over the recently updated 2022 NAICS codes, as many relevant industries have not had much time to update their classifications, and any information found online more likely pertains to the

2017 NAICS codes than the 2022 codes. The 2017 NAICS Index file contains a list of all NAICS codes, and a short description of keywords that describes the types of businesses within that classification. The second file, 2017 NAICS Descriptions contains in-depth business descriptions for each NAICS subclass. For example, there is a main description for all class codes that begin with *11*, another description for all classes that begin with *111*, all the way until the most granular 6-digit code.

Cleaning and Processing Data

Once loaded in, the two files look like this:

```
df = pd.read_csv(r'assets/2017_NAICS_Index_File.csv')
df.head()
```

	naics	description
0	111110	Soybean farming, field and seed production
1	111120	Canola farming, field and seed production
2	111120	Flaxseed farming, field and seed production
3	111120	Mustard seed farming, field and seed production
4	111120	Oilseed farming (except soybean), field and se...

```
des_df = pd.read_csv(r'assets/2017_NAICS_Descriptions.csv')
des_df.head()
```

	Code	Title	Description
0	11	Agriculture, Forestry, Fishing and HuntingT	The Sector as a Whole\n\nThe Agriculture, Fore...
1	111	Crop ProductionT	Industries in the Crop Production subsector gr...
2	1111	Oilseed and Grain FarmingT	This industry group comprises establishments p...
3	11111	Soybean FarmingT	See industry description for 111110.
4	111110	Soybean Farming	This industry comprises establishments primari...

In order to process the data, I created functions called *create_description_df* and *load_data()* that combine the relevant text into each class code into one long description containing all the relevant information. The result of these functions is a DataFrame with each NAICS code and corresponding concatenated description.

```
processed_df = load_data(df, des_df)
# view the results of the cleaned descriptions
processed_df.head()
```

	naics	description
0	111110	Soybean farming field and seed production Soy...
1	111120	Canola farming field and seed production Oils...
2	111130	Bean farming dry field and seed production D...
3	111140	Wheat farming field and seed production Wheat...
4	111150	Corn farming except sweet corn field and se...

Next, I created functions to remove special characters, numbers, and convert the descriptions to lowercase. Additionally, I created a column for all the stemmed versions of the words, and all the lemmatized versions of the words to be used later to test iterations . The end result looks like this:

```
processed_df.head()
```

	naics	description	clean_description	lemmatized	stemmed
0	111110	Soybean farming field and seed production Soy...	soybean farming field and seed production soy...	[soybean, farming, field, seed, production, so...	[soybean, farm, field, seed, product, soybean,...
1	111120	Canola farming field and seed production Oils...	canola farming field and seed production oils...	[canola, farming, field, seed, production, oil...	[canola, farm, field, seed, product, oilse, ex...
2	111130	Bean farming dry field and seed production D...	bean farming dry field and seed production d...	[bean, farming, dry, field, seed, production, ...	[bean, farm, dri, field, seed, product, dri, p...
3	111140	Wheat farming field and seed production Wheat...	wheat farming field and seed production wheat...	[wheat, farming, field, seed, production, whea...	[wheat, farm, field, seed, product, wheat, far...
4	111150	Corn farming except sweet corn field and se...	corn farming except sweet corn field and se...	[corn, farming, except, sweet, corn, field, se...	[corn, farm, except, sweet, corn, field, seed,...

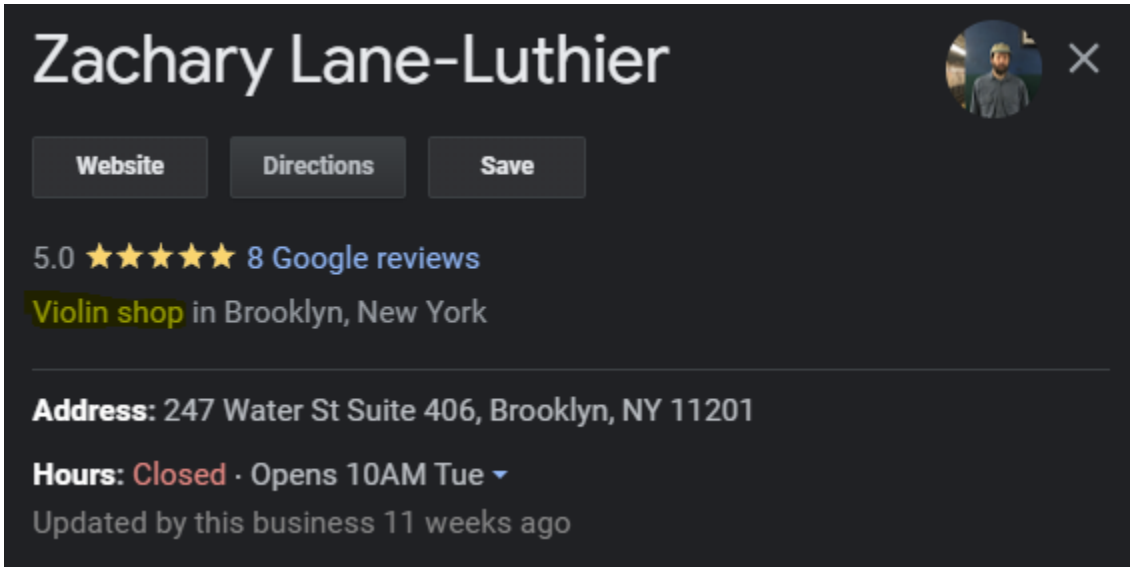
Stemming vs. Lemmatizing

There are two popular methods of normalizing text - lemmatization and stemming. Lemmatization takes the morphological base of a word. For example, “studies” and "studying", would be lemmatized to "study". Stemming takes the root of the word, for example, "studies" would become "studi" and "studying" would become "study". There are certain cases where one is more useful than the other, so I decided to try them both out in my testing, which I will explain the significance of later on in this blog post.

Queries and Relevance Judgments

In order for it to be effective, a recommender system needs labeled data in order to determine what documents are relevant to the user. In an item-based filter, this is usually determined by an individual user providing feedback on what documents are relevant to them. A collaborative filter returns documents to a user based on feedback from other users. However, this recommender system is neither of those. There is no one “individual” who is looking for things recommended to them, and there is no larger group that determines what is relevant to them. The only thing that matters in this type of recommender system is the query that is used and the relevant documents that correspond to that query.

Ultimately, I decided to use relevance judgments that I curated on my own. For each query, I found the most likely NAICS code and then filled in the subsequent relevant NAICS codes. I manually gathered a list of 1154 queries of business descriptions, and used those as the basis for filling in the relevant NAICS codes. In order to gather the queries, I looked through this list of general business descriptions, and then searched Google for similar businesses in those fields. After searching Google, I took the resulting business and used that as a query.



I originally started with a much larger list, but I had to trim it down so that it was manageable to observe the data. With this list, I then used a script that scraped Google for queries like, “Violin

shop NAICS Code", and took the result it found. Due to the way Google returns search results, I was left with 300 or so queries that my script could not find a NAICS code for, so I manually selected the best fitting code.

After I created my initial list with the most relevant NAICS code, I populated the relevant NAICS codes with the next most likely one in descending order. NAICS codes are split into subgroups (more information can be found here), so I considered anything relevant to be any NAICS code that is within the subgroup. First I searched back 1 digit – if any NAICS code started with the first 5 digits, it was considered the next most relevant, and then 4 digits. For example, the "musical instrument store" is most relevant for 45115, the next most relevant codes are ones that begin with 4511, and then the next most relevant codes begin with 451. I stopped at 2 digits, because any more than that created a list of relevance judgments with over a hundred relevant documents. This would have an adverse impact on the precision/recall scores.

This was the most efficient way I could create a list of relevant NAICS codes. However, this leaves a lot of room for error, because there are some like "Musical Instrument Manufacturing" that fall under 339992, and the relevant codes would be other types of manufacturing industries. However, our intuition tells us that a musical instrument manufacturer is probably more relevant to a musical instrument store than to another type of manufacturer.

Normally, relevance judgments are provided by a group of people, and I simply did not have the resources available to create thorough and in-depth judgments on my own, which is why I went with the solution that I chose. As you will see later, this played a part in the final results of my recommender system.

The final result looks like this:

	query	relevant_naics
0	Home improvement store	[444110, 444120, 444130, 444190]
1	Diesel fuel supplier	[424710, 424720]
2	Church	[813110]
3	Farm	[115116, 115111, 115112, 115113, 115114, 115115]
4	Seed supplier	[424910, 424920, 424930, 424940, 424950, 424990]

Recommender System Model Selection

For each model used, I tested with the lemmatized and stemmed versions of the text to note if there was any improvement between each version.

- **Bag of Words (BoW)** model uses the frequency that the tokens of a query appear in a document to retrieve and rank documents. It is common in many NLP tasks.
- **Term frequency–inverse document frequency (TFIDF)** model uses the weights of words in a query and how important they are to a document in order to retrieve and rank documents. Words that are unique to documents will be weighted higher, and words that appear in many documents will be weighted lower.
- **Latent Semantic Indexing** - This model finds the statistical co-occurrences of words that appear together to retrieve and rank documents. For example, queries like "musical instrument store" will rank higher for any class codes related to musical instruments, but will rank lower for any documents related to scientific instruments or other types of instruments.
- **Rocchio Feedback System -** I used the LSI model as a base for this feedback system, but the Rocchio feedback loop first presents a user with base recommendations from the LSI model, and then uses the feedback to update the query vectors, so that when it is queried again, the relevant documents will rank even higher than the first iteration.

Test Methods

For each model, I calculated the following statistics to measure the effectiveness of the system:

- **Precision/Recall@N** - these are the bread and butter of any recommender system. Precision is the proportion of documents retrieved that are considered relevant and recall is the proportion of all relevant documents that are retrieved. N is the cutoff for the number of retrieved documents at which to measure precision and recall.
- **Average Precision@N -** Average precision summarizes the area under the precision/recall curve. Average precision is high when both precision and recall are high, and low when either one of them is low.
- **Mean Average Precision (mAP@N) -** Mean average precision is the mean of all Average Precisions, which is a good measure of the the performance of the entire recommender system, since Precision/Recall and Average Precision can only be measured on individual queries. mAP@N summarizes how the system performs across all queries.
- **Normalized Discounted Cumulative Gain (NDCG)** - NDCG is a good measure of ranking quality in a recommender system. It measures the usefulness of a document based on its position in the retrieved documents. NDCG is higher when the documents are returned closest to the order they appear in the relevance judgments, and lower if they are in the incorrect order, or not in the list of documents at all.

Interpreting Results of Tests

Max Docs - Max docs is an argument for the maximum number of documents returned to the user. I chose 10 as the number for max docs, as any more than 10, and it defeats the purpose of what the recommender system is. A user will likely not look past the first 10 results, and anything beyond the first 10 results can effectively be considered not retrieved.

Stemming vs. Lemmatization

In the BoW model, the stemmed version of text performed slightly better than the lemmatized version with a mAP@N. This is because the frequency of the stemmed words increases, as there are more stems in common than lemmas. In the TFIDF model, the stemmed and lemmatized versions performed the same. However, in the LSI model, we see a significant increase in the mAP@N for the lemmatized version, as it makes sense that there is more semantic meaning between the lemmatized words than there are the stemmed words.

Determining Number of Concepts

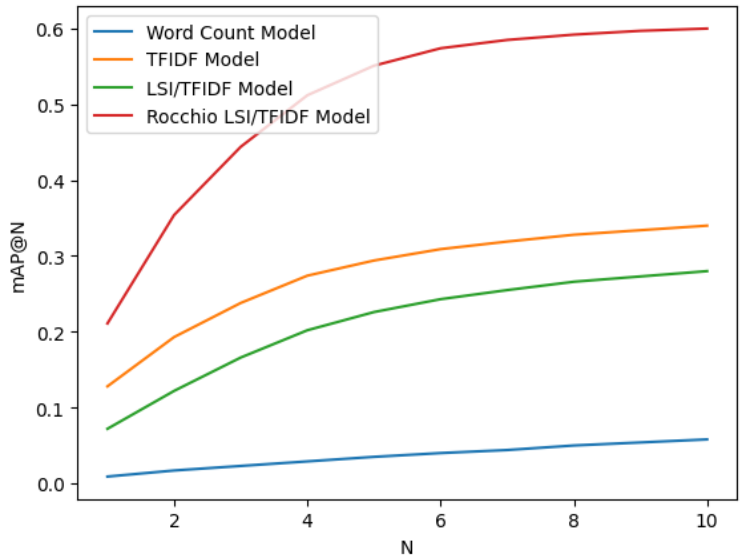
I initially ran the LSI model with the number of concepts = 100, but the results were lackluster. The mAP@N of 0.28 was actually less than the base TFIDF model of 0.34. In order to determine the ideal number of concepts, I ran the model with varying numbers, and ultimately discovered that 350 concepts produces the highest mAP@N of 0.363.

Rocchio Feedback

The Rocchio feedback system uses the LSI model with 350 concepts and trains on the relevance judgments for feedback provided to train the model to return more relevant results the next time the query is issued. The final result for mAP@N is 0.6, a significant increase from the other models.

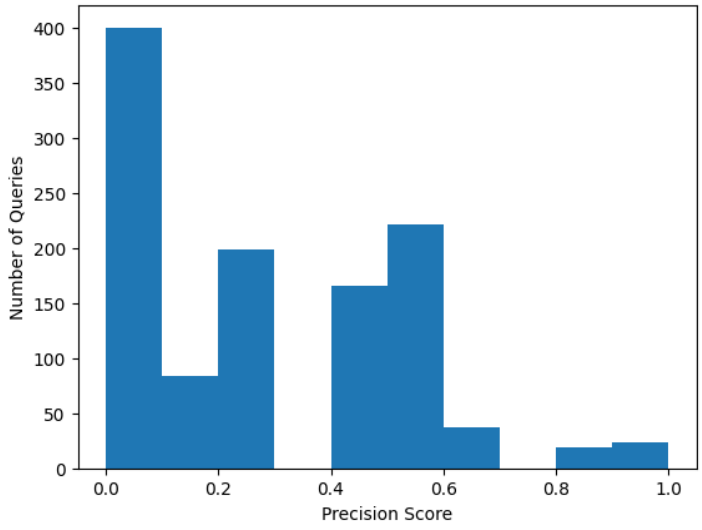
Determining N

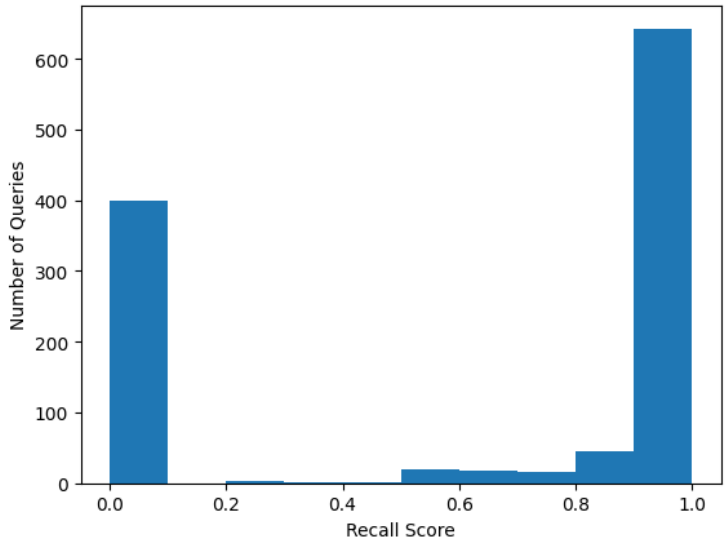
I experimented with different values for N, but I ultimately concluded that 10 is the ideal number for N, to match the max docs returned. In the case of this recommender system, N = 10 works because order is not of utmost importance. As long as the retrieved documents contain relevant documents, then the "goal" of this recommender system is complete. There is certainly room for improvement in that area, but as I said earlier, there is not always a single best matched NAICS code to a query, and therefore the cutoff should be measured within the range of the max docs returned. In the chart below, you can see how mAP@N increases as N increases, which is expected due to the nature of how the relevance judgments were created.



Precision/Recall

The below histograms show the distribution of the precision and recall scores across queries for the LSI model after the Rocchio updates are made.

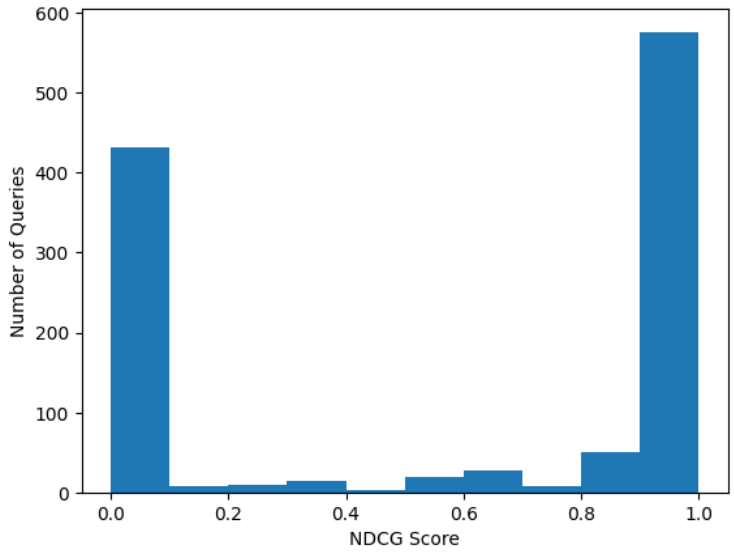




A precision of 0 suggests that there are a lot of queries in the data that the recommender system does not have high precision for. This is likely because there is a lot of overlap between queries, and also due to the method I used to create the relevance judgments. Many NAICS codes descriptions share similar industry descriptions, and a lot of words are similar across industries. Many of the words in queries might not even appear in the data set. For example, our intuition tells us that "violin shop" falls under 451140 - Musical Instrument and Supplies Stores, but if you look through the data, the word violin never appears in the description. A human can easily tell that a violin shop is a musical instrument store, but without seeing it in text, the recommender system cannot do the same. Improving results would require further engineering, as well as a professional review of relevant documents, with many users providing feedback. The consistently high recall scores suggest that a high amount of recommended documents appear in the list of relevant documents.

NDCG

NDCG is not nearly as important of a metric in this type of recommender system because order is not of utmost importance. One query could be equally relevant to more than one NAICS code, depending on the query that is issued. With the succinct queries I used from Google, NDCG is certainly a less important metric, but still good to measure nonetheless.



The overall NDCG scores are pretty high, and the documents are ordered pretty closely to the expected relevance judgments. The high number of 0 scores is for the same reason that precision and recall are 0.

The final scores for each query and model can be viewed in the code, where you can view and interpret the results yourself.

Model Results

Below you can view how the returned documents change for each model, and determine for yourself if you find them to be relevant or not for the query “musical instrument store.” All interpretations are based on what I consider to be the most relevant NAICS codes.

Bag of Words - Lemmatized

	naics	title
0	321999	All Other Miscellaneous Wood Product Manufactu...
1	453998	All Other Miscellaneous Store Retailers (excep...
2	423220	Home Furnishing Merchant Wholesalers
3	454390	Other Direct Selling Establishments
4	333111	Farm Machinery and Equipment Manufacturing
5	321920	Wood Container and Pallet Manufacturing
6	423390	Other Construction Material Merchant Wholesalers
7	623990	Other Residential Care Facilities
8	236117	New Housing For-Sale Builders
9	442299	All Other Home Furnishings Stores

Bag of Words - Stemmed

	naics	title
0	711510	Independent Artists, Writers, and Performers
1	711130	Musical Groups and Artists
2	711310	Promoters of Performing Arts, Sports, and Simi...
3	711320	Promoters of Performing Arts, Sports, and Simi...
4	711110	Theater Companies and Dinner Theaters
5	711219	Other Spectator Sports
6	339992	Musical Instrument Manufacturing
7	711410	Agents and Managers for Artists, Athletes, Ent...
8	711211	Sports Teams and Clubs
9	339999	All Other Miscellaneous Manufacturing

You can see that the lemmatized version returns very few, if any, results that a user will think is relevant to “musical instrument store.” It looks like the model performs a little better with the stemmed version, however, none of the results are the true NAICS code of 451140. The reason the stemmed version performs a little better here is because it is using the stem “music” of the word “musical” and finding classifications related to music. The lemmatized version finds classifications that have the highest count of appearances for “musical”, “instrument” and “store”, which is why we see completely different results.

TFIDF Lemmatized

[57]:

	naics	title
615	451140	Musical Instrument and Supplies Stores
613	451120	Hobby, Toy, and Game Stores
614	451130	Sewing, Needlework, and Piece Goods Stores
612	451110	Sporting Goods Stores
616	451211	Book Stores
617	451212	News Dealers and Newsstands
628	453991	Tobacco Stores
941	711130	Musical Groups and Artists
629	453998	All Other Miscellaneous Store Retailers (excep...
421	334519	Other Measuring and Controlling Device Manufac...

TFIDF Stemmed

[58]:

	naics	title
615	451140	Musical Instrument and Supplies Stores
613	451120	Hobby, Toy, and Game Stores
614	451130	Sewing, Needlework, and Piece Goods Stores
612	451110	Sporting Goods Stores
616	451211	Book Stores
617	451212	News Dealers and Newsstands
628	453991	Tobacco Stores
941	711130	Musical Groups and Artists
629	453998	All Other Miscellaneous Store Retailers (excep...
421	334519	Other Measuring and Controlling Device Manufac...

The lemmatized and stemmed version perform exactly the same in this scenario, which makes sense due to how TFIDF is weighted. You can see that TFIDF is able to use the weight of words and determines that 451140 is the top class. “Stores” is weighted heavily as well, as the next documents are other types of stores. The last document, 334519, is related to scientific instruments, which is where we see the weight of the word “instrument” come into play. Based on the top results, it looks like the TFIDF model does a pretty good job with this query.

For the LSI model, we will use the query “lawyer office.”

LSI Model - Lemmatized

	naics	title
908	621399	Offices of All Other Miscellaneous Health Prac...
907	621391	Offices of Podiatrists
906	621340	Offices of Physical, Occupational and Speech T...
789	541191	Title Abstract and Settlement Offices
790	541199	All Other Legal Services
787	541110	Offices of Lawyers
920	621999	All Other Miscellaneous Ambulatory Health Care...
904	621320	Offices of Optometrists
900	621111	Offices of Physicians (except Mental Health Sp...
901	621112	Offices of Physicians, Mental Health Specialists

LSI Model - Stemmed

	naics	title
908	621399	Offices of All Other Miscellaneous Health Prac...
907	621391	Offices of Podiatrists
906	621340	Offices of Physical, Occupational and Speech T...
789	541191	Title Abstract and Settlement Offices
790	541199	All Other Legal Services
787	541110	Offices of Lawyers
920	621999	All Other Miscellaneous Ambulatory Health Care...
904	621320	Offices of Optometrists
900	621111	Offices of Physicians (except Mental Health Sp...
901	621112	Offices of Physicians, Mental Health Specialists

The LSI model doesn't seem to perform all that well when the number of concepts = 100. The lemmatized version doesn't have enough concepts to pick up semantic meaning, and the stemmed version seems to mostly be using the TFIDF weight.

Rocchio Feedback - Lemmatized

	naics	title
787	541110	Offices of Lawyers
790	541199	All Other Legal Services
789	541191	Title Abstract and Settlement Offices
788	541120	Offices of Notaries
908	621399	Offices of All Other Miscellaneous Health Prac...
907	621391	Offices of Podiatrists
622	453210	Office Supplies and Stationery Stores
906	621340	Offices of Physical, Occupational and Speech T...
1036	922130	Legal Counsel and Prosecution
489	339940	Office Supplies (except Paper) Manufacturing

Rocchio Feedback - Stemmed

	naics	title
787	541110	Offices of Lawyers
790	541199	All Other Legal Services
789	541191	Title Abstract and Settlement Offices
788	541120	Offices of Notaries
908	621399	Offices of All Other Miscellaneous Health Prac...
907	621391	Offices of Podiatrists
622	453210	Office Supplies and Stationery Stores
906	621340	Offices of Physical, Occupational and Speech T...
1036	922130	Legal Counsel and Prosecution
489	339940	Office Supplies (except Paper) Manufacturing

Now with the number of concepts = 350 and the Rocchio updates performed, the recommender system correctly recommends better results for “lawyer office”. Both the lemmatized and stemmed version produce the same results, but given the higher mAP@N score for the lemmatized version, we know that there are some results where the lemmatized version squeezes out a better performance. It was tough to cherry pick the perfect query.

Summary

Queries

The recommender system exists in a vacuum. The queries used are only the results from Google, and not indicative of vastly differing queries that normal users would use. It would take many people and a lot of feedback for the recommender to converge at the optimal point. In addition, the recommender system fails with queries that contain words that are not in the broad scope of text. In order to improve this, I would need to improve the recommender system that continually feeds new documents to the user when they do not appear at all in the data. Since the system does not return documents with a weight of 0, they get left out. This means that eventually a user will query “violin shop”, and the correct NAICS code will appear. After feedback, the document vector will be updated to include violin shop as a valid query. This is not an ideal solution, and any proper solution would require a vast amount of user feedback in order to solve the problem.

Relevance Judgments

As I stated earlier, the relevant documents were determined by the closest matching industry codes up to two digits. While this works for the purpose of this project, this fails in practice because a query like “musical instruments” could apply to multiple NAICS codes. The relevance judgments only consider NAICS codes that are closely related 45115, which is the NAICS code for musical instrument stores. With more users creating relevance judgments, the accuracy of the relevance judgments would increase.

Failed Projects

My initial dream for this project was to create a functioning API that would allow users to query the name of a business and its address, and the application would return a list of NAICS recommendations. However, as the deadline came closer and closer, I realized that I simply did not have enough time to complete what I envisioned. I got too bogged down in creating proper queries and relevance judgments, and testing and interpreting my results. Unfortunately, due to a clerical error, I was placed in a group by myself. I was initially excited about working on a project by myself, but by the end, I was completely overwhelmed. It is unfortunate that I had to cut out a lot of improvements and final pieces to my project. With a group, I really believe that this project could be transformed into something special and I hope to continue working on the idea even after this class is completed.

Final Thoughts

Ultimately, I believe that a classification model is still the significant method of predicting the probability that a query belongs to each class, but as I stated in the beginning, the real limitation to this is acquiring accurately labeled data, which exists behind a paywall or an NDA, which I could not acquire for this project. The recommender system that I created produces decent results. A final mAP@N score of 0.6 is pretty good considering the limitations of the data and the methods I used to test the data, However, I believe a classification model could produce significantly better results.