**Topic Modelling Mathematics Publications – An Instructive Failure**

Zachary Fannin, Samuel Hansen, Shane Wolfram

*Introduction*

As both a methodological tool and a substantive research topic, classification pervades nearly all academic disciplines. No exception to this fact is mathematics, a subject area of interest to all of the members of our group. This interest led us to decide to focus on mathematical classification for our Digital Humanities Analytics Final Project. In this paper we will describe the path we took while investigating mathematical classification, the tools we used, and the results we obtained.

Before diving into our methodology and data, we should take a moment to look at the purpose and significance of classification. In an informal sense, classification is the process of grouping entities according to a common set of properties. This process allows humans to begin to understand their surroundings, recognizing or giving some rhyme or reason to them. Moreover, it facilitates information access and retrieval. Over time, different schemes have emerged to provide and enhance information resource classification; for mathematics, such a scheme is the Mathematical Subject Classification (MSC) system (Mathematics Subject Classification, n.d.).

MSC is an alphanumeric scheme collaboratively created and maintained by Mathematical Reviews and zbMATH, two major mathematical review databases. Since the majority of mathematical publications are indexed in these two databases, most of them ask scholars to submit the MSC code(s) of their article’s discipline(s), but subject editors at the two databases are the people who determine the final classifications. MSC takes a hierarchical structure, comprising three major levels that allow for increasing specificity in the code for more particular subdisciplines. The first level is made up of two digits, the second a letter, and the final an additional two digits. For example, from the latest iteration of MSC completed in 2010 (MSC2010, n.d.):

03-XX = Mathematical logic and foundations

03EXX = Set theory

03E15 = Descriptive set theory

Our initial plan was to use the MSC classifications of mathematical articles to create a temporal citation analysis of mathematical papers. In order to accomplish this original plan, we attempted multiple times to contact MathSciNet (MR: Copyright Information and Terms of Use, n.d.) to request the use of their data for the project, but we never elicited so much as a response from them. Samuel even reached out to a personal contact who works with MathSciNet and was not able to make any headway. After a while, it became clear we would not be able to get our hands on the data we would need in order to accomplish our original plan and needed to move on. Since MathSciNet was not willing to provide us with their classification information, we decided to try our hand at classification ourselves. Instead of relying on authors and subject editors, though, we were going to try an automated process leveraging topic modelling.

What is topic modeling and why is it useful? According to Mohr and Bogdanov, topic modeling provides an “automated procedure for coding the content of a corpus of texts (including very large corpora) into a set of substantively meaningful coding categories called 'topics'” (Mohr, et al., 2013) Unlike when humans hand-code a text, topic modeling algorithms do not require require a set of presupposed categories. Rather, topic modeling uses an inductive approach, constructing a set of topics (the number of which is defined by the researcher) by identifying words that tend to cluster together whenever a given topic is being discussed.

The most common method of topic modeling, and the one that we used, is Latent Dirichlet Allocation (LDA). LDA defines a topic as a distribution over all words in a corpus, such that “words that are strongly associated with the document's dominant topics have a higher chance of being selected” for use in a document (ibid).

Many approaches to topic modelling can be thought of as naive: Instead of using purpose-built tools for the exact corpus of materials, ready-to-use tools are employed. This is an understandable approach since the difference in what is needed to analyze newspaper articles and fiction books is not likely to be great enough to warrant the cost to create a specific tool for each. We decided to approach our topic modelling of mathematical articles in such a manner, to see if naive topic modelling would work on a mathematical corpus and because we did not have the time or the expertise to create a mathematics specific topic modelling program. At the very least, we felt this would allow us to answer whether topic modeling could be effective for mathematical publications, since various mathematical symbols, variables, and even concepts can have multiple meanings depending on the context in which they are invoked.

*Topic Modelling Process*

Our first step toward topic modelling mathematical articles was to create a scholarship corpus. We discussed scraping papers from the arXiv or JStor, but before attempting this we met with Ian Ross of GeoDeepDive (GeoDeepDive: Project Overview, n.d.), a project of the University of Wisconsin-Madison and Stanford University. They have standing agreement with publishers such as Elsevier, Wiley, and Taylor & Francis, that allows them to pull in the full text of articles and then analyze those papers using the natural language processing toolkit Stanford CoreNLP (Stanford CoreNLP, n.d.). GeoDeepDive was originally created to pull in and analyze geology papers, but research remit was broad and their agreement with the publishers allowed them pull any journal they publish. Thus, when when we asked Ian if they would be able to help us create our mathematical corpus, they agreed. In the end, GeoDeepDive pulled in 22,397 papers from the six journals *Discrete Mathematics*, *Advances in Mathematics*, *Journal of Graph Theory*, *Journal of the London Mathematical Society*, *Applicable Analysis*, and *Quaestiones Mathematicae*. They provided us with a PostgreSQL table with the CoreNLP analysis output. While we did not delve into much of the NLP data GeoDeepDive provided for us, we did decided to work with both the tokenized sentence arrays comprising the full text of the paper, as well as sentence arrays of those tokens’ lemmatization. An example of their output is available from GeoDeepDive (GeoDeepDive: USGS Sample tgz Archive, n.d.)--see the file titled “sentences\_nlp352”. Using these arrays, we were able to rebuild the full text of the articles we needed in order to attempt our topic modeling.

To create our topic models, we wrote a Python script (Hansen, 2018d) that used both the gensim topic modelling for humans (gensim: topic modelling for humans, n.d.), Python library, and the MALLET Machine Learning for Language Toolkit (MALLET Topic Modeling, n.d.) on both the full text of the articles in our corpus and the CoreNLP lemmatized full text. The script was based on of a tutorial by Rare Technologies (RARE Technologies, n.d.). The first step in the script was to randomly split the corpus of articles into two groups: The first group consisted of ⅓ of the articles we used to create our training set; the second group comprised the remaining ⅔ of the articles we modeled. Once the articles were split, we had to process the training set into a useful form. We started this by querying the CoreNLP PostgreSQL table and rebuilding the full text of the articles from either the words or the lemmas column. We then ran them through gensim’s corpora.Dictionary function and filtered out stopwords. We used gensim’s MALLET wrapper to create a ten-topic topic model from the processed set. With the model in hand, the rest of the script rebuilt the remaining articles from the PostgreSQL table and applied the model to them. The outputs were a text file for the topic model itself with a line for each topic consisting of 15 words multiplied by the percentage they played in the topic added together, and a text file of the topic numbers multiplied by the percentage for that topic for each article. Due to the immense size of the corpus, we were unable to run it on our own machines. Once again, Ian Ross came to our aid, this time in the guise of the University of Wisconsin-Madison’s Center for High Throughput Computing (CHTC, n.d. needed). They ran our topic modeling script six times: three times for regular full text and three times for the lemmatization of the full text.

Once we had the text file outputs of the topic model scripts, the data still needed to be processed into more usable forms. We again used Python to accomplish this. First we translated the topic text files into a CSV (Hansen, 2018b), with each row representing a single part of a single topic for each run. Samuel went through this list and, using knowledge of mathematics language, coded all of these topic words (Hansen, 2018e), applying mathematical subject areas such as *algebra* and *graph theory* to any of the topic words specific to those areas. This coding was far from perfect, as it relied on the interpretation of a single person and, in the end, Samuel was able to code only 333 of the 1200 total topic words. Then, using the coded CSV and the article topic model text files, we also created a CSV for each run, which created a row for each article and the percentage each coded value applied to that article for that run (Hansen, 2018c).

In order to see whether our topic modeling attempt was accurate, we needed a data set to classify at least a subset of our corpus. The best classification of mathematical papers is the aforementioned Mathematical Subject Classification (MSC) applied by MathSciNet and zbMath. As the University of Wisconsin-Madison does not subscribe to zbMath, we decided to attempt to scrape the MSC values for the articles in our corpus off MathSciNet. We ran into many problems attempting to do this in an automated way. Firstly, unless one is using a wired network connection on campus, MathSciNet lives behind the library’s EZProxy and there is not an easy way to deal with this using the standards urllib (urllib documentation, n.d.) or requests (Requests: HTTP for Humans, n.d.); even when we used a wired computer, these libraries only returned a blank page. We made more headway using Selenium web browser automation (Selenium, n.d.) and had a functional script (Hansen, 2018c); however, when we tried to run it to scrape MSC values, we were identified as unusual activity and MathSciNet blocked our access after only obtaining 2981 values. Though this only covered around 13% of our total corpus of articles, we believed it would give us insight into the efficacy of our topic modelling. So we created another script which combined the MSC values for the articles with two code codes that had achieved the highest percentages when summed across the word, lemma, and all runs (Hansen, 2018a).

For visualizing our topic models, we planned on using D3 (Bostock, n.d.), the industry standard for web visualizations. It allows for more complexity than Excel charts and makes it easier to share visualizations *via* websites like Github and Glitch. In hindsight, Excel may have sufficed for our purposes, but struggling with D3 was a good learning experience. An initial idea was to use a chord diagram to show the strength of the relationship between topic-word pairings; however, despite Shane’s attempts to remix several existing chord diagrams on the web for this purpose, for some reason these attempts proved unsuccessful. Specifically, one of the diagrams contained outdated code that Shane was unable to correct to work in the latest D3 version (5.1), and another simply failed to show anything at all despite also not generating errors – so it was hard to know where to start debugging.

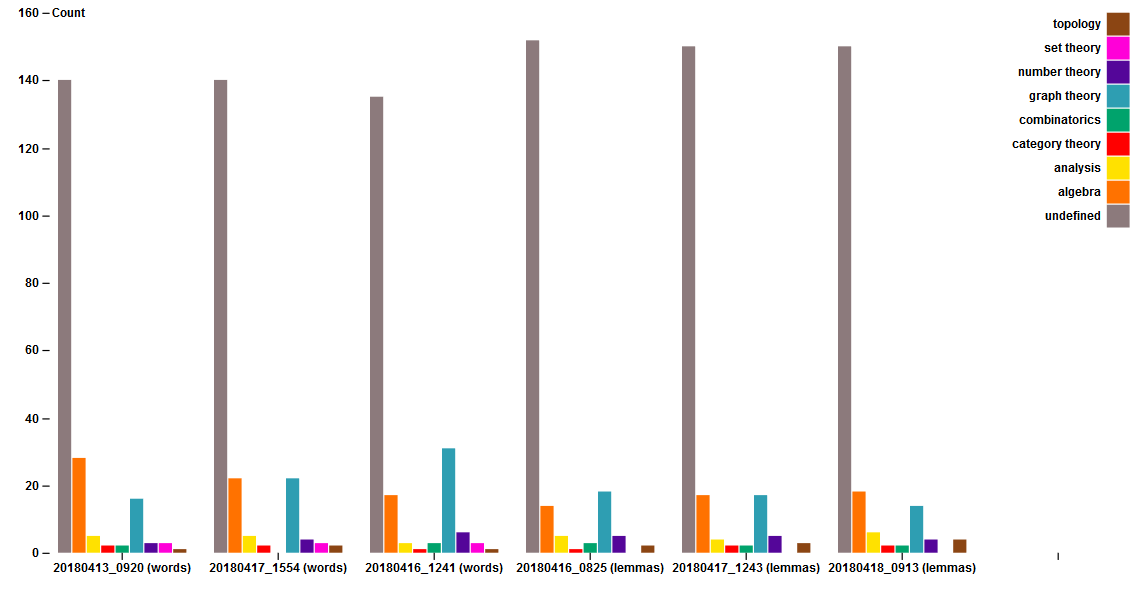
Fortunately, Shane's knowledge of D3 sufficed to make a couple bar charts to help visualize the results of our topic modelling. This was done by taking an existing bar chart off the web and modifying it slightly to fit our purposes. The first of these charts shows the distribution of papers by their highest percentage coded concepts for all of the runs. Perhaps the most interesting thing about this chart is that it clearly shows a difference in number of papers identified as “graph theory” between the word and lemma runs – though this difference (as is explained in the discussion) ended up not affecting the accuracy of the topic models. A second chart shows the distribution of papers with an “Undefined” highest percentage coded concept by their second highest percentage concept.

Additionally, pie charts were created using OpenOffice Calc to show the accuracy of our topic model for the papers for which we were able to get MSC codes before getting blocked from MathSciNet. We decided to use Calc for this because, surprisingly, there did not seem to be any decent D3 pie charts online (ones that included a legend, for instance) and Shane decided that it would be easier to just use Calc, which has a chart function very similar to Excel's, than to make a new chart in d3 or fix one of the ones he found. Altogether nine pie charts were made in this way, first using the data for all runs and then separating the word and lemma runs, in order to see if the difference in identification of “graph theory” articles between these runs would affect the accuracy of the topic models.

*Discussion*

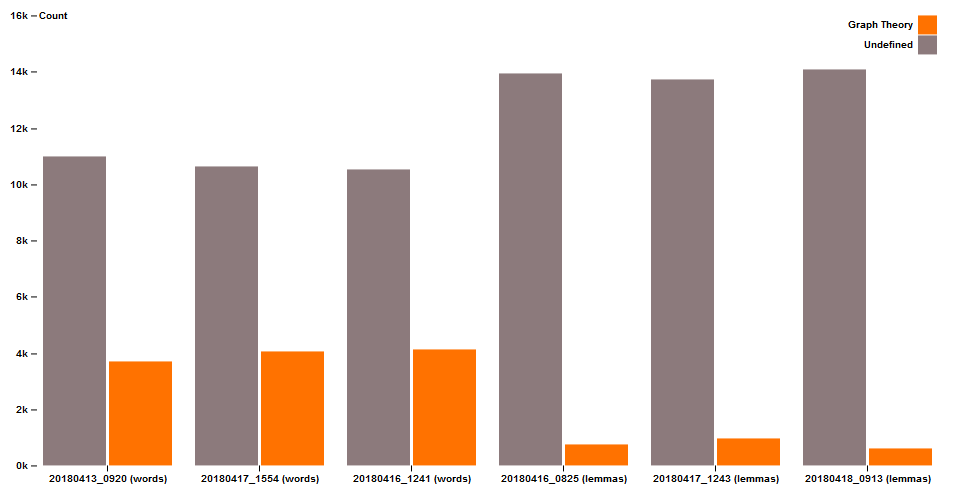
Before we delve into the results of the topic modelling, let us start with a discussion about the code values given to the fifteen words forming each topic. As mentioned in the Process section above, we were unable to code 867 of the words. During our analysis, we gave these words a code of ‘Undefined’. There were multiple reasons there were so many cases of ‘Undefined’ words. Some relate to the language of mathematics, which regularly reuses words, such as map, space, and metric *across* subfields, making them ambiguous and thus very hard to code for a single, specific area of mathematics. There were a lot of cases of weird artifacts such as ‘xi’, ‘el’, or ‘bn’ from the GeoDeepDive processing, likely related to handling equations and formulas. For the remaining 333 words we managed to identify under a specific area of mathematics, we used the codes ‘graph theory’(118), ‘algebra’(116), ‘analysis’(28), ‘number theory’(27), ‘topology’(13), ‘combinatorics’(12), ‘category theory’(10), and ‘set theory’(9). Words that were coded with the values ‘graph theory’ and ‘algebra’ were more likely to appear in the topics from runs based on the normal full text of the article versus the lemmatization (Figure 1).

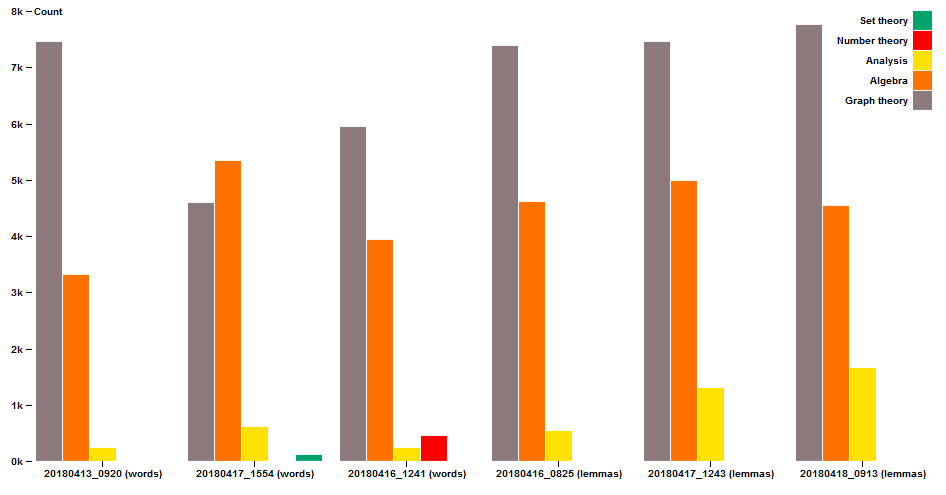
**Figure 1. Topic model words per code**

****

Shifting to the topic modelling of articles, only two of the coded values were identified as the highest percentage code for any of the papers, ‘Undefined’ and ‘graph theory’. Of the 88058 total runs, ‘Undefined’ was the highest percentage code 73881 times and ‘graph theory’ only 14,177 times. Related to ‘graph theory’ appearing as a code for topic modelling words in the lemma runs, ‘graph theory’ ends up being the highest percentage code at a much lower rate in lemma runs(Figure 2). As for the second highest percentage code, if the highest percentage code was ‘graph theory’, the second highest percentage code was, unsurprisingly, ‘Undefined’. When the highest percentage code was ‘Undefined’, on the other hand, the second highest percentage code was ‘graph theory’ 36421 times and ‘algebra ‘30824’ times, with all the other codes having totals at least an order of magnitude lower. Looking at the second highest percentage code on a per-run basis, we found that ‘graph theory’ was more likely in lemma than words type runs, likely related to ‘graph theory’ having a lower occurrence as the highest percentage code in the lemma runs, and ‘algebra’ having a higher count than ‘graph theory’ only once (Figure 3).

**Figure 2.** **Papers per highest percentage code, by run.**



**Figure 3. Papers per highest and second highest percentage code, by run.**

From these results, it became clear our topic model was unlikely to say anything about the corpus papers beyond relevance to graph theory. Therefore, we decided to see if the papers with either their highest percentage code or both highest and second highest percentage codes being graph theory also have an MSC value related to graph theory--in other words, they have an MSC value of the form 05Cxx or one of 68R10, 81Q30, 81T15, 82B20, 82C20, 90C35, 92E10, 94C15 (Mathematics Subject Classification - 05, n.d.). Looking across all runs, we had the MSC values for 2981 articles, of which 1413 were related to graph theory. When we looked at that set of articles for which we had MSC values and focused on the highest percentage code, we correctly identified 954 of the 1413 graph theory articles and misidentified only 28 non-graph theory papers as such. We did fail to identify 459 of the graph theory articles. When we looked at the set of articles for which we had MSC values and focused on the highest percentage code, we correctly identified 1404 of the 1413 graph theory articles and only failed to identify 9 of them. We did, however, misidentify 604 non-graph theory articles as such (Figure 4). These rates did not noticeably change when we looked at exclusively word or lemma runs as you can see in the probability tables(Figures 5,6).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Only Highest Percentage Code** | **words** | **words** | **lemmas** | **lemmas** | **allRuns** | **allRuns** |
| **p(gtTM)** | 970/2889 | 33.58% | 952/2878 | 33.08% | 982/2981 | 32.94% |
| **p(not gtTM)** | 1919/2889 | 66.42% | 1926/2878 | 66.92% | 1999/2981 | 67.06% |
| **p(gtMSC|gtTM)** | 942/970 | 97.11% | 925/952 | 97.16% | 954/982 | 97.15% |
| **p(not gtMSC|gtTM)** | 27/970 | 2.784% | 27/952 | 2.836% | 28/982 | 2.851% |
| **p(gtMSC|not gtTM)** | 420/1919 | 21.89% | 444/1926 | 23.05% | 459/1999 | 22.96% |
| **p(not gtMSC|not gtTM)** | 1499/1919 | 78.11% | 1482/1926 | 76.95% | 1540/1999 | 77.04% |
| **p(gtMSC)** | 1362/2889 | 47.14% | 1369/2878 | 47.57% | 1413/2981 | 47.4% |
| **p(not gtMSC)** | 1527/2889 | 52.86% | 1509/2878 | 52.43% | 1568/2981 | 52.6% |
| **p(gtTM|gtMSC)** | 942/1362 | 69.16% | 925/1369 | 67.57% | 954/1413 | 67.52% |
| **p(not gtTM|gtMSC)** | 420/1362 | 30.84% | 444/1369 | 32.43% | 459/1413 | 32.48% |
| **p(not gtTM|not gtMSC)** | 1499/1527 | 98.17% | 1482/1509 | 98.21% | 1540/1568 | 98.21% |
| **p(gtTM|not gtMSC)** | 28/1527 | 1.834% | 27/1509 | 1.789% | 28/1568 | 1.786% |

gtTM-Article’s highest percentage code is ‘graph theory’

gtMSC-Article has MSC value related to graph theory

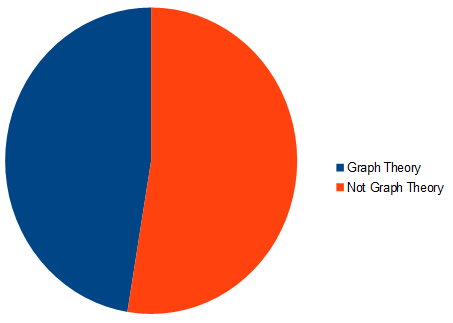
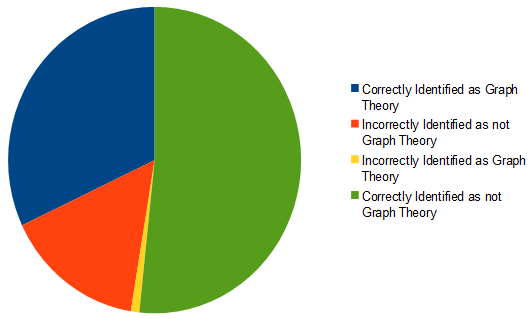
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Only Highest Percentage Code** | **words** | **words** | **lemmas** | **lemmas** | **allRuns** | **allRuns** |
| **p(gtTM)** | 1951/2889 | 67.53% | 1947/2878 | 67.65% | 2008/2981 | 67.36% |
| **p(not gtTM)** | 938/2889 | 32.47% | 931/2878 | 32.35% | 973/2981 | 32.64% |
| **p(gtMSC|gtTM)** | 1354/1951 | 69.4% | 1362/1947 | 69.95% | 1404/2008 | 69.92% |
| **p(not gtMSC|gtTM)** | 597/1951 | 30.6% | 585/1947 | 30.05% | 604/2008 | 30.08% |
| **p(gtMSC|not gtTM)** | 8/938 | 0.8529% | 7/931 | 0.751% | 9/973 | 0.925% |
| **p(not gtMSC|not gtTM)** | 930/938 | 99.15% | 924/931 | 99.25% | 964/973 | 99.08% |
| **p(gtMSC)** | 1362/2889 | 47.14% | 1369/2878 | 47.57% | 1413/2981 | 47.4% |
| **p(not gtMSC)** | 1527/2889 | 52.86% | 1509/2878 | 52.43% | 1568/2981 | 52.6% |
| **p(gtTM|gtMSC)** | 1354/1362 | 99.41% | 1362/1369 | 99.49% | 1404/1413 | 99.36% |
| **p(not gtTM|gtMSC)** | 8/1362 | 0.5874% | 7/1369 | 0.511% | 9/1413 | 0.6369% |
| **p(not gtTM|not gtMSC)** | 930/1527 | 60.9% | 924/1509 | 61.23% | 964/1568 | 61.48% |
| **p(gtTM|not gtMSC)** | 597/1527 | 39.1% | 585/1509 | 38.77 | 604/1568 | 38.52 |

gtTM-Article’s highest or second highest percentage code is ‘graph theory’

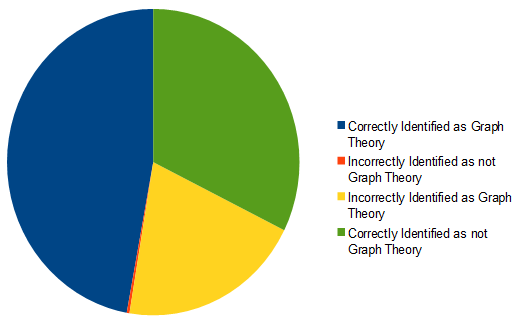
gtMSC-Article has MSC value related to graph theory

**Figure 4. Accuracy of Identification for Papers we had MSC values for(All Runs)**

Paper MSC Values: Highest percentage code:

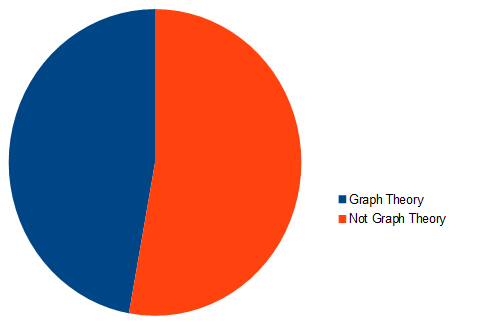
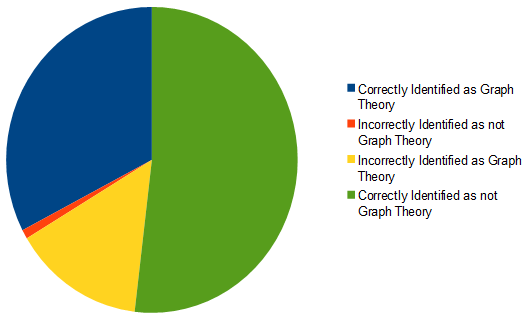


Highest and second highest percentage codes

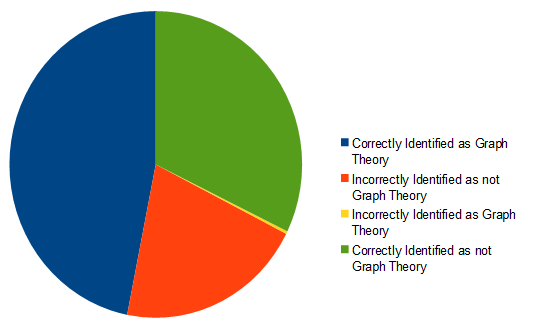


**Figure 5. Accuracy of Identification for Papers we had MSC values for(Word Only Runs)**

Paper MSC values: Highest percentage code:

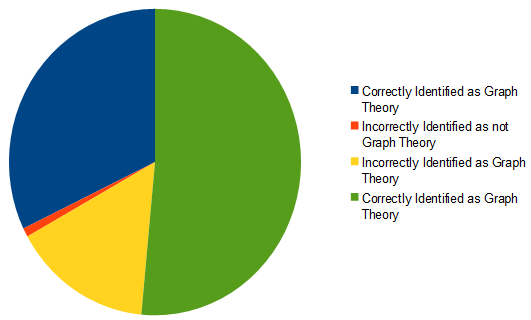
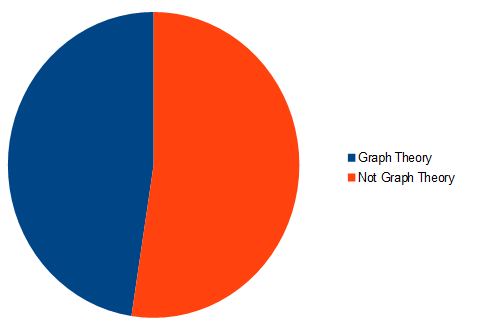


Highest and second highest percentage codes:

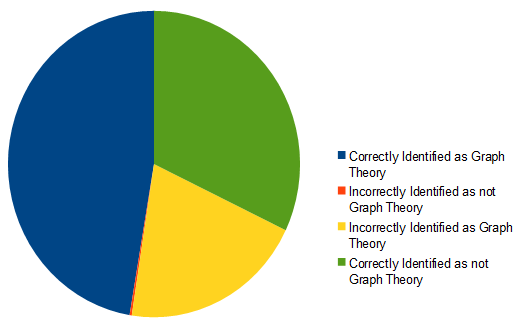


**Figure 6. Accuracy of Identification for Papers we had MSC values for(Lemma Only Runs)**

Paper MSC values: Highest percentage code:



Highest and second highest percentage codes:



*Conclusion and Next Steps*

Given the results from our topic model, we are comfortable answering the question, “Is naive topic modelling useful for classifying a mathematical corpus?” with a resounding no. Between the issues related to dealing with the equations and formulas within the papers from a technical standpoint and the problems of coding up topics filled with ambiguous words, it is clear that a naive, unfocused topic modelling approach is not helpful in classification.

Graph theory does appear to provide the naive approach with some hope. We believe this stems from the uniqueness of the language used in graph theory. This uniqueness manifested in two ways: Samuel noticed it was much easier to determine which words should be coded with ‘graph theory’; a less noticeable way was not clear until looking more closely at the coded list, where ‘graph theory’ as a code appears clustered together. This is because the words coded with the value belong to the same topic, meaning MALLET did actually identify graph theory topics from the training set in a way it did not for other areas of mathematics which are not noticeably clustered. Sadly, while our topic modelling did get closer to being effective with graph theory than with any other area it still left us with a decision between two bad alternatives. Either we could use only the highest percentage code and have the probability of a false positive be just below 2% but have the probability of accurately identifying a graph theory paper be below 70% or use the second highest percentage code as well and have the probability of accurately identifying a graph theory paper be greater than 99%, which would be great if it not for the probability of falsely identifying a paper as graph theory being nearly 40%.

There are many potential paths forward with this research. We could try to do the coding differently, perhaps by assigning codes at the topic level instead of the more granular words within a topic level. We could reach out to the mathematical community for help with the coding to get other perspectives on the potential meaning of the words comprising our subject. We have also discussed the potential of training the model using only papers from a known subject area (*e.g.*, only the articles about graph theory) to see if that could lead to higher accuracy and lower false positive rates in identifying papers from that subject area. We could even simply rerun our topic models after adding the weird processing artifact words in our current models to the filter/stopwords lists to see if removing them could help generate more useful models. At the very least, we are sure Samuel is going to be sending off an email soon to zbMath to see if they would be willing to share their data, since MathSciNet will be off limits for a while, thanks to the scraping process.

References:

CHTC. (n.d.). Retrieved May 7, 2018, from<http://chtc.cs.wisc.edu/>

Bostock, M. (n.d.). D3.js. Retrieved May 7, 2018, from<https://d3js.org/>

gensim: topic modelling for humans. (n.d.). Retrieved May 7, 2018, from<https://radimrehurek.com/gensim/>

GeoDeepDive: Project Overview. (n.d.). Retrieved May 7, 2018, from<https://geodeepdive.org/about.html>

GeoDeepDive: USGS Sample tgz Archive. (n.d.).

Hansen, S. (2018a). *analyzeTopicModellingDataUsingMSC.py*. Python. Retrieved from<https://github.com/samuelhansen/lis768/blob/master/finalProjectRepo/analyzeTopicModellingDataUsingMSC.py> (Original work published 2018)

Hansen, S. (2018b). *digestTopicModelingData.py*. Python. Retrieved from<https://github.com/samuelhansen/lis768/blob/master/finalProjectRepo/digestTopicModelingData.py> (Original work published 2018)

Hansen, S. (2018c). *topicModelingFromPostGresDB.py*. Python. Retrieved from<https://github.com/samuelhansen/lis768/blob/master/finalProjectRepo/topicModelingFromPostGresDB.py> (Original work published 2018)

Hansen, S. (2018d). *topicsCSVCoded.csv*. Retrieved from<https://github.com/samuelhansen/lis768/blob/master/finalProjectRepo/topModelResults/topicsCSVCoded.csv> (Original work published 2018)

Hansen, Samuel. (2018a). *mscScraper.py*. TeX. Retrieved from<https://github.com/samuelhansen/lis768/blob/master/finalProjectRepo/mscScraper.py> (Original work published 2018)

Hansen, Samuel. (2018b). *processTopicModellingData.py*. Python. Retrieved from<https://github.com/samuelhansen/lis768/blob/master/finalProjectRepo/processTopicModellingData.py> (Original work published 2018)

MALLET Topic Modeling. (n.d.). Retrieved May 7, 2018, from<http://mallet.cs.umass.edu/topics.php>

Mathematics Subject Classification. (n.d.). Retrieved May 7, 2018, from<https://zbmath.org/classification/>

Mathematics Subject Classification - 05. (n.d.). Retrieved May 7, 2018, from<https://zbmath.org/classification/?q=cc:05C>

Mohr, John W., and Petko Bogdanov. “Introduction—Topic Models: What They Are and Why They Matter.” *Poetics*, Topic Models and the Cultural Sciences, 41, no. 6 (December 1, 2013): 545–69. <https://doi.org/10.1016/j.poetic.2013.10.001>.

MR: Copyright Information and Terms of Use. (n.d.). Retrieved May 7, 2018, from<https://mathscinet-ams-org.ezproxy.library.wisc.edu/mathscinet/help/mathscinet_terms_of_use.html?version=2>

MSC2010. (n.d.). Retrieved May 7, 2018, from<http://msc2010.org/Default.html>

RARE Technologies. (n.d.). Tutorial on Mallet in Python. Retrieved May 7, 2018, from<https://rare-technologies.com/tutorial-on-mallet-in-python/#gensim_wrapper>

Requests: HTTP for Humans. (n.d.). Retrieved May 7, 2018, from<http://docs.python-requests.org/en/master/>

Selenium. (n.d.). Retrieved May 7, 2018, from<https://www.seleniumhq.org/>

Stanford CoreNLP. (n.d.). Retrieved May 7, 2018, from<https://stanfordnlp.github.io/CoreNLP/index.html>

urllib documentation. (n.d.). Retrieved May 7, 2018, from<https://docs.python.org/3/library/urllib.html>

zbMATH. (n.d.). Retrieved May 7, 2018, from<https://zbmath.org/>