NCBI Document Recommender

Andrew Y

Figure 1. National Center for Biotechnology Information

Abstract

For students, finding relevant research papers for their studies is a problem faced in every assignment outside of their expertise. The goal of this project is to create a useful tool to find papers useful for their desired topic. Scope-wise, the model will work with documents from the National Center for Biotechnology Information, or the NCBI, as they have the an extensive database of documents relating to biology, mainly bio-informatics, bio-medicine, and bio-technology. The outcome of the project is for the user to be able to input a keyword, and receive a list of documents that would be relevant to their work. To do this, our project will use the NCBI API to pull in documents from the database, and then utilize an unsupervised learning model to find a list of relevant documents to return.

ACM Reference Format:
Andrew Y. 2018. NCBI Document Recommender. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages.
https://doi.org/XXXXXXX.XXXXXXX

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The project is a model where a user will be able to input a keyword to find a list of documents whose topics are relevant to the keyword. The problem this model attempts to solve is to help minimize the effort required to find useful sources or references to use when one is attempting to do educational work outside of their field of expertise. For example, if a student was writing an essay about different allergies for a class, then they would be able to enter the keyword "hay fever" and be able to receive numerous published educational pieces around hay fever. While not life changing, this would help reduce the time spent browsing numerous documents and allow the user to allocate more time to other tasks of their assignment. To help keep the scope manageable, the project will be working with data from the National Center

of Biotechnology Information, or NCBI for short. They are a research institute that is a branch of the National Institutes of Health, and is known for housing a multitude of databases containing tools and services relating to biotechnology and biomedicine. They also house databases dedicated to bio-related literature, which is what this project will be pulling the data from. While their database does have an existing search feature, it is quite limited in its functionality, and does not have a way to return documents based on their relevancy. The potential contribution of this project is to provide an additional way for documents to be acquired in a way that would ease the burden of any aspiring researcher attempting to expand their scope of knowledge.

2 Related Works

As a recommender system, it can utilize obvious references to the renown Google search engine. In addition, NCBI already have their own search function for documents, a tool named Entrez, which is their dedicated text retrieval system. However, their existing search function only has four different ways to sort their result, which are article order, publication date, journal number, or PMC live data. The goal of the project is to create a different way to receive the results, one that utilizes the document contents rather than their metadata which is different from the existing methods. In doing so, we can ensure that the resulting list of documents will be of more use to the user than scrolling through documents listed in chronological order.

3 Proposed Work

This project has six main tasks to be completed. The first task is the utilization of Entrez. Entrez is NCBI's official in-house tool for data extraction, and comes with their own Python library that utilizes their API to pull data into our notebooks. It also comes with in-depth documentation with directions on how to use their tool for reliable data scraping. The only thing to keep an eye out for would be to make a conscious effort to use the API as little as possible to avoid clogging up their bandwidth with requests. To combat this, one can easily download the data and store it in a local repository for ease of access. Conveniently, Entrez allows us to also designate a key word to base our data around, and this greatly helps in narrowing down the range of document data that we receive. For instance, the dataset that this project utilizes for training uses the 'hay fever' keyword.

The second task is the data cleaning, data wrangling, and data pre-processing tasks. In regards to data warehousing, fortunately enough the NCBI organization does a perfect job of keeping the data well managed in their own "warehouse" so the data can be kept in its original format. This data set consists of individual documents as its rows, and document features as its columns. The document features mostly consist of metadata about the document, such as publication date, the journal that it belongs to, or the author. From these features, there are three main ones that will have the most focus: document abstract, document reference number, and document author. Document abstract is our biggest priority as it will be where the bulk of the modeling will utilize. The abstract is used over the actual text data as it is more feasible in regards to performance, and also helps combat over-fitting. This is due to it acting as a summary for the actual article text, and thus can be treated as a form of dimension reduction. As the abstract data is still also in text form, we will undergo the typical text pre-processing steps such as but not limited to, removing accents, word lemmatizing, and removing stop words. Once the pre-processing has been completed, it will be fed through both a Count Vectorizer and a TF-IDF Vectorizer model to create a document-term matrix. This is done as it is the most optimal format to feed text data into a machine learning model.

The document reference number feature is a count of how many times the document has been referenced by other articles in field. It will be utilized to help act as a metric for determining a document's significance or quality, as documents that are highly referenced are most likely well-written or well-researched enough if they are frequently referenced by other members in the community. However, there is an issue of a significant amount of documents not having a reference number; as not having a reference number is not the same as having a reference number of zero, we do not want to leave out all of these documents. Thus a regression model will also be created to predict the supposed reference numbers based on their abstract contents. By using the reference number for each document, it can be used to prune the documents that might not be as helpful if a threshold is set to only keep documents with a reference number above a certain amount.

The last kept feature is document author. This feature isn't particularly used at the moment but potentially could be used to help find similar documents if an author can be considered well-known in the field. This is a metric that will be revisited later if time permits.

Once these cleaning tasks have been completed, we can move onto the third task of Data Exploration and Visualization. This task is relatively straightforward, and consists of plotting various count plots, bar plots, and distribution plots to view the popularity or trends of the different features. For example, we can tell that Ciprandi G is the most popular author, and that the large majority of documents were published from the years 2005 - 2010. While most of these EDA did not end up being used in the actual model, they provided some insight of potential future ideas that can be done to improve the project as a whole.

Once these tasks have been completed, then the actual modeling can start. Main task four consists of constructing a supervised regression model to impute the missing reference numbers. As the reference numbers are considered count

data (or data that can be recorded by counting), we will be utilizing the Mean Poisson Deviance metric for evaluating the models due to its specialization in count data. A multitude of different regression models will be tested as well, mainly linear regression, poisson regression, decision tree regression, and random forest regression. Each model will be run twice on both the Count Vectorizer inputs and the TF-IDF Vectorizer inputs.

With the optimal regression model and parameters found, the missing reference numbers can be imputed. While this is happening, main task five can be started where the inputs are sanitized through a combination of normalization, scaling, and matrix factorization for preparation of the next step: clustering. Our goal is to group the documents into N distinct clusters so that we can have N distinct documents to pull. Both K-Means clustering and Hierarchical clustering will be tested, and the main evaluation metric of choice is Silhouette Score. This is due to silhouette's score focus in how distinct each cluster is, which we want to prioritize in order to have a wide variety of different documents to pull.

The final main task six is simply to pull our desired documents from the clusters. This is done through finding the document in each cluster with the highest reference number. With all the documents in hand, they will be returned to the user for their use.

4 Evaluation

For evaluations, we have two main modeling portions to focus on. The first is the supervised regression model for predicting and imputing the missing reference numbers, and the second is the unsupervised clustering model for grouping up the documents itself.

In terms of the supervised regression model, the evaluation metric of choice was the metric Mean Poisson Deviance. This is found through the 'sklearn.metrics' library, sand measures the mean poisson deviance regression loss. While admittedly the inner working of the statistics involved is a bit puzzling to the author of this project, from online research it is known that this metric is useful for machine learning modeling involving predicting count data.

For the inputs,they were ran through a Count Vectorizer and a TF-IDF Vectorizer, and both were used for every model, essentially running each model twice. From the results of the models, the Count Vectorizer inputs consistently performed better on the training data set split, while the TF-IDF inputs consistently performed better on the testing data set split. As test data performance is more prioritized, the TF-IDF inputs were chosen in the end as inputs for the rest of the project.

The modeling portion of the supervised regression model consisted of testing each input on various models and comparing their mean poisson deviance score. The models tested were linear regression, poisson regression, decision tree regression, and random forest regression. Performance wise, linear regression did horrendously bad. This is most likely due to how non-linear a document-term matrix is as an input, so it is understandable that there are no linear relations between term frequencies. This just left us with poisson regression, decision tree regression, and random forest regression. While all three had surprisingly close scores, the poisson regression model had a mean poisson deviance score that was a couple points better than the others, and thus it was chosen.

GridSearchCV was then ran on the poisson regressor to discover the best hyper-parameters and help attempt to push the score further, but the results ended up revealing that the default hyper-parameters turned out to be the highest performing ones. So thus the best model for imputing the missing reference numbers was a Poisson Regression model with TF-IDF inputs.

With the supervised model found and the missing values imputed, the clustering portion can begin. The evaluation metric that will be used is the Silhouette Score. The Silhouette Score in particular measures how clumped up the clusters are, as with the more clumped up the clusters, the less overlap there is between clusters. As our main priority is to have very distinct clusters, this metric suits the goal of this project greatly.

The clustering model tested will be K-Means Clustering and Hierarchical Clustering. In a situation such as this, k-means had consistently a better score than hierarchical. This is most likely due to the project preemptively letting the user set the number of documents that is desired. As the number of documents will match with the number of clusters we have, k-means clustering can play to its advantage by having its K value match the number of documents.

For the clustering model inputs, several inputs were tested. The base input was still the old TF-IDF input but further cleaning has been done to it to better suit a clustering model. With the models, they tested the inputs being scaled, normalized, scaled then normalized, normalized then scaled, and normalized then scaled then undergoing SVD. Scaling then normalizing the data had the best score of the regular cleaning, though adding in SVD caused an enormous boost in silhouette score. While testing various number of components for the SVD, the resulting optimal number turned out to be two. Thus our final model for the cluster was a K-Means cluster with the number of documents as the K, and having the inputs first scaled then normalized then SVD'd.

5 Discussion

Proposal Stage:

The first priority of the project timeline is to get the data ready. This will involve utilizing the Entrez API to pull data from the database, and doing extensive cleaning of the text data to better prepare it for modeling. The second milestone will be to ensure that our metric of determining whether or

not a document can be considered significant enough to be considered for returning. The third milestone will be to be able to create a model that will return a list of significant documents. Once theses steps are finished, we can undergo model optimization and performance tuning.

The potential challenges mainly resolve around evaluating the recommender system, in which the clustering method will act as a suitable backup plan as a way of improving performance.

Project Checkpoint Stage:

The project is proceeding smoothly, the time spent working with the data after the project proposal gave ample time to test different theories and potential problems. The data for the model that utilized the database site's tool that pulled the document data using the API has been acquired both consistently and reliably. The data wrangling has been completed as well, with the text data of the abstract cleaned into a way that best suits the TF-IDF Vectorizer input. The TF-IDF Vectorizer as well has been used to create a document-term matrix ready and prepped to be used for modeling.

Almost all of the problems that popped up during the proposal brainstorming stage have been dealt with. For the issue of finding a way to evaluate a recommender system when there is no feedback to improve off of, it was swapped to utilizing a clustering method instead that judged the documents based on their sum of reference numbers.

However, an unforeseen issue did appear when Exploratory Data Analysis was done as it turns out a significant number of documents did not have a reference number for the model to work with. As a missing reference number was different than documents that had a reference number of zero, simply leaving out these documents would be a waste. Thus came the idea of creating a second model, a regression one, to help predict the count of the reference number if the number was not missing.

As the modeling portion is still a work in progress, it will be discussed in the following report.

Final Checkpoint Stage:

Overall the project timeline was completed quite nicely, though it seemed less of a timeline and more of a glorified check list. However it did have its fair share of challenges that popped up, though most of them were just ideas from the proposal that seemed well-thought out at the time but did not work out in practice.

The first challenge was that the initial model for finding document could not be evaluated properly. As shown above, the first initial model was a recommender system, which made sense as this was a project meant to recommend documents. However, with the way the process of finding the documents was envisioned, finding a proper evaluation metric for it did not seem feasible. This was due to the lack of user data or preferences, no feedback from users, and no ground truths to evaluate our recommended predictions with. Thus lead to a swap from a recommender system to a clustering system.

In a similar fashion, the initial idea for finding documents to return ended up being quite redundant at best and disastrous at worst. The idea was to split the documents into numerous clusters with each cluster having the same size as the proposed number of documents to return. Then the sum total of all reference numbers in a cluster would be calculated, with the cluster having the largest sum being chosen and having all of their documents selected. The first problem came into the fact that having cluster size match the number of documents would mean an ungodly amount of clusters. For example, if we wanted a cluster of ten documents, and there are 9,000 documents, then we would have to calculate 900 different clusters. As as added bonus, it does not seem possible in sklearn to ensure that each cluster had the same size without creative external problem solving. Lastly, during a conversation with a colleague it was pointed out that if all of the documents came from the winning cluster, that would meant that they were all quite similar due to their close proximity. Thus having a list of similar documents with similar contents would mean that much of the information would be redundant. It was due to all this that the current method of having the number of documents match the number of clusters rather than the cluster size was swapped to.

The last challenge and the one that still hasn't really been solved is that the overall run-time is still much too long. Having to pull the data, clean the inputs at multiple points, and fit multiple models for a list of documents may not seem worth the wait for many users.

6 Conclusion

Although the code and commands have not been streamlined yet, the project is in a usable state. The main takeaways from this project is that recommender systems are difficult or rather near impossible to evaluate when there is no ground truth or feedback to build off of. Without evaluation, it is difficult to understand how good or how bad the model is doing.

There is also plenty of potential future work that can be done for this project. If time permits, the top priority would be to either refactor the code, or organize it into a proper app or application for ease of use. Expanding the project to be able to scrape data from organizations other than NCBI would be a plus as well, as is including other metadata in the model such as authors or key terms. Lastly just being able to further reduce run-time would be a big plus.

7 References

Entrez's Bio Python Package
https://biopython.org/docs/1.76/api/Bio.Entrez.html