IST 736 - Text Mining

Final Project Report

Tag Puncher - Generating Tags for Documents

Yashaswi Pandey

Aditya Pawar

Table of Contents

Abstract

The goal of this project is to generate the tags for a document based on its textual contents so that one can understand the different topics that are mentioned in the document or are related to it in some way. The audience for this project are students and professionals who take a lot of notes or keep a lot of text files on them to refer to, generating tags using our model will go will generate the tags for these texts, then we can filter for specific tags or to find out documents related to a particular topic, this will help to find the necessary documents quickly and also make us aware of those topics mentioned in other documents that we might not have noticed. To make this possible we used wikipedia articles as texts and the titles of other pages it is connected to are used as the tags for the document, to find the connected pages we see which articles on wikipedia are linked in the documents text. After creating our model and testing it on other wikipedia articles it had not seen yet, the model after training was highly precise but recall was quite low, as we lowered our confidence threshold the precision decreased while the recall increased. This result is quite common when dealing with the case of extreme multi labelling as there is a long tail of labels that only appear a few times. In general when tested it was able to predict accurate tags that were related to the text across a variety of categories like sports, history, geography, economics among others. In the end we were able to successfully train a model that when given a text can label it with most of the correct tags.

Introduction

Ever since humans have invented writing, they have written down thoughts, plans and other documents ranging from mundane to historically significant. A clay tablet that was dated to 1750 B.C contained a complaint to a copper merchant for sending the wrong grade of copper ore shows that information written down can survive for a long time and if useful can be of use to people centuries later, you just need to look at the work of ancient philosophers like Euclid, Archimedes, Pythagoras in the west and Aryabhatta, Confucius in the east among thousands of other people who have had a huge impact on modern life even though they existed long ago. The concepts that they brought to life, from the Pythagorean Theorem to inventing zero to writing five postulates that govern most of modern day geometry, we can see the impact and presence of these ideas even in the modern day, we are still seeing people develop upon these concepts and refer to these works even after thousands of years. There is a connection of information through the ages and with our project we aim to create a model that will help connect the information. In the digital age, people have digitized documents such as notes, journals, ebooks, research papers etc. There is a connection between the concepts mentioned in the different documents and there is an urgent need for people to be able to connect them so that they can access information that is relevant to them but might be hidden in large collections and can go unnoticed. We have seen the rise of note taking applications such as Roam Research that give people the ability to link their notes, they can then see how different notes are connected to each other based on the links to the other notes, this way the user might be able to find new connection of ideas or an old forgotten idea. We can see this is important to people especially students and researchers as there has been a rise in options for users for applications that allow them to connect their ideas such as Obsidian and Logseq which offer free tiers and open source as their selling points.The problem with this is that the user needs to write down everything and manually tag and connect their notes, the issue being that the user

might not know which things to connect to what when dealing with a new subject and would need to constantly refactor their notes to be able to properly link the contents of the notes.

Our project allows the user to run all the text through the model and get the appropriate tags, this will ensure that the user is able to find connections between all documents without having to go through each and add the tags manually, making the process of creating a web of knowledge easier and faster. The rest of this paper will discuss the research we did, how we transformed the data to suit our needs and how we trained a model that can successfully tag text documents with most of the relevant tags.

Literature review

During our research to complete our project we read a few papers related to what we wanted to achieve to help guide us in the correct direction and so that we could adapt the previous knowledge to our use case. Salton and Buckley (1988) proposed the term-inverse document frequency (TF-IDF) method for keyword extraction, this became the basis of early rule-based approaches to automate tag generation. These methods, while successful, struggled with context and semantics. Recent work by Gupta, Shah and Patel (2019) employed BERT to generate context aware tags for scientific articles, these outperformed the traditional method.

When it came to classifying multiple labels for a single document, Tsoumakas and Katakis (2007) provided strategies such as binary relevance and label powerset, these provided a great foundation but failed to capture label dependencies. Recent work by Liu, Jin and Wang (2021) using deep learning based methods were able to achieve state of the art results on the Reuters-21578 dataset which has 21,578 documents with each having multiple categories.

Bush's (1945) concept of "associative trails" talked about a system where information could be linked non-linearly and users could navigate the knowledge similar to human thought process. This concept can be seen as the inspiration for modern knowledge management systems like Roam Research and Obsidian, this method, while accurate, is time consuming and heavily dependent on the user's skill level. Shen, Chen, and Wang (2020) tried to use natural language processing to predict relationships between documents, but this method used syntactic patterns to infer these relationships which limits their ability to understand the semantics and the context and have difficulty adapting to domain specific or complex documents.

Finally we found out about folksonomy, described in 2005 by Thomas Vander Wal as user-created bottom-up categorical structure, in which users assigned their own tags to documents, like tagging images on Flikr, later he also stated it as "tagging that works". Our project aims to build a model that can learn from large scale structured data with organically generated connections using links which we will treat as the tags. With this we hope to bring the benefit of folksonomy, without needing to manually tag.

Methodology

We downloaded the entire structured dataset released by Wikimedia for the English Wikipedia. We then went through the entire dataset and created a page details file, this file contained details for each page in the dataset such as the number of words, number of links and the link density in the article, we also created another csv which contained a count of all the links in the dataset. We then trimmed down the number of links from twenty five million to 256k, we removed links that appeared rarely for us that was less than twenty five times as it would not get trained properly due to low training data, we also removed links that pointed to files and web pages like pdf, png, php. We then divided the dataset into training, testing by choosing the articles with the highest link density and we also made a validation data set that contained the articles with the lowest link density to test out our model and see if it could predict tags for those pages. We sampled the data in this way so that we could get enough labels while training and when testing being able to evaluate properly by comparing to existing labels. The sampling for validation made sense as we wanted to see if we could improve tagging for those articles using our trained model.

To approach this problem we focused on training a machine learning model on high quality data. We applied supervised learning frameworks where the articles from wikipedia served as the input text and the titles of the pages it linked to as the associated labels. The process utilized natural language processing and multi-label classification techniques so we could apply multiple labels per text. This suited our needs as it helps us get a large amount of high quality real world text which can help us model the complex relationships between the text and the labels as wikipedia is able to provide a diverse set of articles with millions of labels among them, making it an easy choice for a task such as this.

Initially we started with the idea of leveraging BERT's capabilities of generating contextual text embeddings to achieve a high performing model, this was in particular due to the inability of TF-IDF and other traditional approaches to understand the context and the semantics to generate high quality embedding to be used by the model. This idea had to be scrapped due to the high computational costs and the extra effort needed to increase the label output space of the model, also since most BERT based model still have a token length of 512, it means that we would need to truncate or split intelligently a lot of articles due to their length.

Our second approach was to use FastXML which trains multiple trees to create a model that can traverse it to make predictions, even though we were able to quickly train the model with little effort on our part the real challenge lay in understanding why it would not save the tree model to disk, this made loading it and evaluation an impossible task.

Finally we used PECOS X-Linear algorithm developed by amazon specifically as a scalable extreme multi-label classifier (XMC), this model allowed us to train a flat one vs all classifier or a hierarchical label cluster. The full form for PECOS is **P**rediction for **E**normous **CO**de **S**pace and can handle millions of labels making it extremely suitable for our needs of training a model on the wikipedia data and links. We ended up training several models with different parameters and architectures including both one vs all and the hierarchical label clustering. For preprocessing we changed the hyperparameters of our TF-IDF vectorizer to try and improve our model performance.

The first model was a hierarchical clustering model that generated label embeddings using pifa, which generates the embeddings using weighted combination of label and feature information. This model was able to train extremely quickly as the hierarchical clustering reduces the training time by not training each label separately. The second model had all parameters the same but instead of generating the label embeddings each label was trained in

an one vs all flat model, this took quite a while compared to the first model and barely performed better than the hierarchical model. We also trained a model where the vectorizers also created bigrams

Each of the decisions made was to help us get to our goal of training a model that would help label a text document with the relevant tags to help connect information.

Results

High Level Data Description

The data we got from wikipedia was in multiple jsonl files, after processing this data for our work we had two columns for our main use case, the text from the articles and its corresponding labels that we generated by finding all the pages the current page links to and using its title as our label, each text had multiple labels associated with it with a total of more than half a million unique labels.

We filtered the dataset to ensure that there were enough articles and enough labels per article and having enough words per article to help the model learn better with better data, this helped us generate a data set where the articles represented a diverse range of text. Each row has two main components, the text and the labels, the text is cleaned plain text wikipedia articles and the labels which were wikipedia page titles that were hyperlinked in the articles.

There were slightly more than half a million labels with minimum 50 occurrences and the mean being about 274 occurrences in the final training corpus, around 100k rows in the training data set and around 10k rows in the test data set. There were at least 25 labels for every article and all articles had to be at least 100 words.

For cleaning the data we first need to process each jsonl file, each row being a json file, so we loaded it, then in the first pass we extracted the name, the identifier, the url, the number of words, the number of links and the density of links in the file, another csv was created at the same time which stored each unique label and its counts. The urls in the unique label files were cleaned by mapping the url to the name of the article from the first file where present and by using regex to extract the last part of the url. We then removed any urls we did not need such as those with low occurrences or those that linked to files or non-wiki pages. We then sorted the dataframe with all the details and sorted it by the link density, the top hundred thousand articles became our training dataset, this was to provide a lot of links for each training example. The next thousand were saved as the test dataset, as we wanted to check on data that had similar amounts of tags to generate proper metrics. The thousand articles that had more than 100 words but very low link density were chosen to be the validation to check on our own whether the model was able to add tags to text very similar to what it might have been trained on. We also tested on articles by pasting them individually to rate the model.

The above cleaning approaches were made with the final dataset in mind from day one, where we knew that the dataset we needed before we could think about creating a dataset that had two main columns, a column with the text and a column for the labels stored as a list, this made the cleaning process quite intuitive with only the time needed to understand the structure of the downloaded dataset and how to pre-process it taking time, otherwise the dataset was created quite quickly and we were able to train a few models in the rest of the time, this became

quite the blessing as few of our models ran for over twelve hours before eventually the system ran out of memory and we had to adjust the parameters and hyperparameters before we began training again.

The reason we needed to get the data in the form mentioned earlier was that it made it easier for us to transform the dataset in any way we wanted depending on the model, eg for BERT we split the dataset after 450 words so that it would fit logical parts as one example in the model, this resulted in multiple rows for a single article in some cases. The FastXML model worked without too much transformation apart from making sparse matrices for our text and labels. For pecos we needed to factorize the labels and map it to the list to get a list of numbers then this was saved as a tsv with the first part being comma separated numbers representing the labels and the second part was the text.

Low Level Analysis

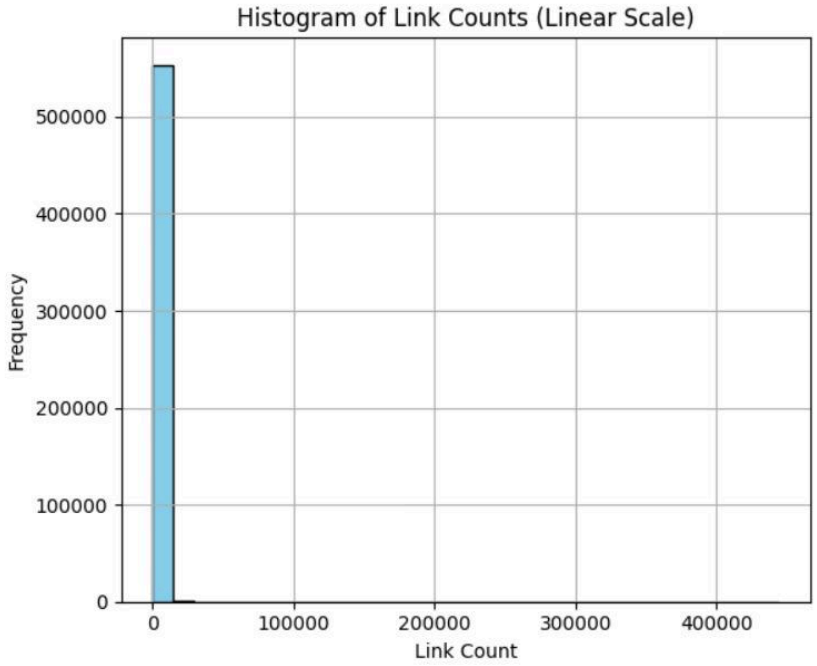
In the end we have three models that we tested and the model that we trained last with hundred thousand rows and bigram TF-IDF vectorizer had the best performance.

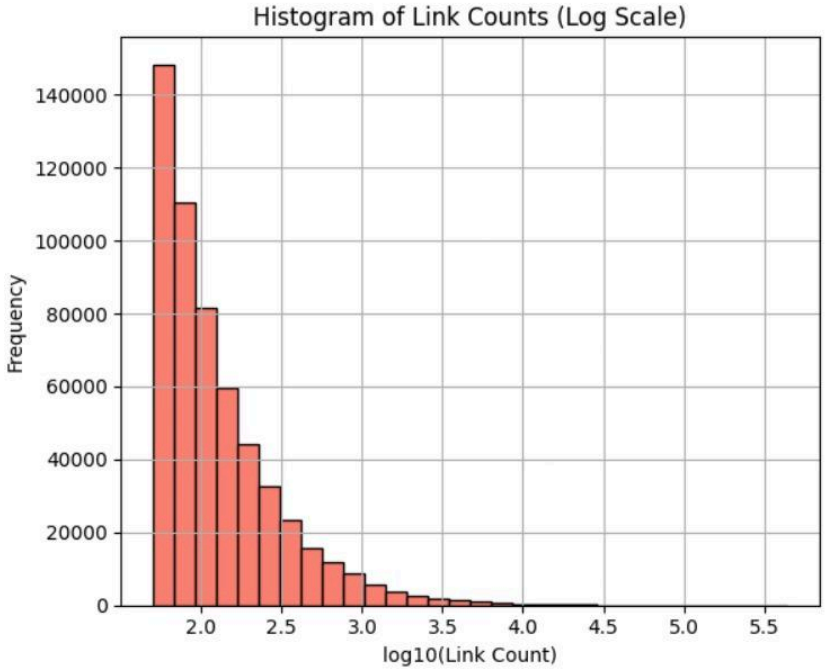
prec = 86.65 84.35 82.11 80.13 78.18 76.58 74.69 72.86 71.08 69.27
recall = 5.32 10.11 14.47 18.55 22.32 25.92 29.14 32.10 34.84 37.27

The high precision and low recal which shows that the model is accurate and can predict with a high precision, but it is not able to recall all relevant labels

The high precision and low recall is commonly seen in extreme multi-classification as there a a

few labels present a lot of times and a lot of labels present a lot of times, the distribution of labels can be seen below,





The other two models used a smaller dataset and used just the unigram TF-IDF vectorizer, none of these had the performance close to that of our final model, their performance was as follows

One vs All

prec = 77.65 77.11 76.11 75.23 74.39 73.50 72.85 72.02 71.34 70.56

recall = 2.65 5.20 7.63 9.98 12.26 14.47 16.66 18.73 20.77 22.71

Hierarchical embedding

prec = 76.58 75.40 74.20 73.17 72.18 71.25 70.32 69.56 68.75 68.08

recall = 2.58 5.04 7.37 9.64 11.82 13.93 15.94 17.92 19.84 21.73

As we can see that the one vs all model performed slightly better than the hierarchical embedding model, but compared to the minutes of training time it took for the hierarchical with unigram TF-IDF and the half an hour it took our best performing model based on the hierarchical embedding too, the one vs all took over four hours to train.

The results above show that the model can assign the labels accurately though it still misses some relevant tags and this is because of the low number of training examples, but for most documents it should be able to generate a few high quality labels out of the 500k it was trained on. The model ends up answering most of the questions and hypotheses we had before starting, but there is still a lot of area to improve, with providing better embeddings being the top priority.

Discussion

The analysis of the predictions made by the model help us to answer the main question posed which was to use folksonomy and bring it to the personal level by creating a model that can predict labels in a large label space and connect documents so that the user can access different information from various sources in the way the human brain thinks. This is done by giving the documents labels using the model and then checking how different documents are connected to each other based on these tags.

The limitations we ran into were that to train a model across a large label space requires a lot of memory to store and access the embeddings, this made us reduce the number of labels and the number of labels per document so we could train the model. Secondly, we have trained the model after the text was preprocessed with a TF-IDF vectorizer, using deep learning techniques to take advantage of context would provide a boost to the results.

Future research should focus on ways that can help us zero shot or one shot the rare label prediction as over eleven million links that we extracted occurred only once, thus making this the prime focus of any future work.

Conclusion

In conclusion, we were able to successfully train a model that after reading a text can predict labels associated with it from a large label space, we were able to answer the question of whether we can take user generated tags and apply it at a personal level to be able to tag different texts and then find out how they are connected or what information they contain. The unanswered questions are whether this can be scaled to be able to provide domain specific tagging and whether we can zero shot or one shot the rare labels.

Any future research should focus on bringing folksonomy to the personal levels so as to take advantage of the tagging that works.

References

Bush, V. (1945). As we may think. *The Atlantic Monthly*, *176*(1), 101–108.

Gupta, S., Shah, M., & Patel, K. (2019). Automatic tag generation for scientific articles using BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 4905–4910).

Liu, Y., Jin, D., & Wang, Y. (2021). Multi-label text classification using BERT-based ensemble learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(10), 8762–8770.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, *24*(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Shen, Y., Chen, M., & Wang, D. (2020). Automatic knowledge graph construction for personal note linking. In *Proceedings of the 2020 Conference on Computational Natural Language Learning (CoNLL)* (pp. 345–355).

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, *3*(3), 1–13. https://doi.org/10.4018/jdwm.2007070101

Vander Wal, T. (2005). Folksonomy coinage and definition. Retrieved from http://vanderwal.net/folksonomy.html

Zhang, M.-L., & Zhou, Z.-H. (2013). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837. https://doi.org/10.1109/TKDE.2013.39