Zero-shot Patent Classification

Luke Mizuhashi (lukemiz)

Patents today are manually assigned labels that indicate what part of intellectual property space a patent protects and in which legal systems that protection is provided. This labeling process constitutes a significant portion of the entire patent application process and, currently, globally, demand for patent application classification services far outpaces the supply of applicable labor.

The purpose of this project is to examine techniques for accurately and efficiently labeling patents using machine learning.

Literature Review

Asudani did an incredible review of dozens of papers, labeling which embedding and training method was used, the task they accomplished, and the value of their evaluation metric. From Asudani's review:

> Graph Convolutional Network (GCN) is suggested as a method for classifying text. The Text-GCN learns the embed- ding for both words and documents after initializing with a one-hot representation of each. The experimental outcome demonstrates Text-GCN's tolerance to minimal training data in text classification. Text-GCN can effectively use the limited labeled documents and collect information on global word co-occurrence (Yao et al. 2019). For text categorization, the graph-of-docs paradigm is proposed to represent numerous documents as a single graph. The suggested method recognizes a term's significance across the board in a document collection and encourages the inclusion of relationship edges across documents. Experimental results demonstrate that the suggested model outperforms the baseline models with an accuracy score of 97.5% (Giarelis et al. 2020).

This is particularly interesting in the context of patent embeddings, because there are some nodes in the CPC and related hierarchies with very few documents. Also, the world's legal systems tend to naturally create a graph of semantic relationships between patent documents that is encoded in each patent by the polysemous jargon mentioned above.

Asudani also mentions quite a few instances of classification, sentiment analysis, and word segmentation tasks completed with >= 95% accuracy using relatively simple embeddings like Bag of Words (BoW) and word2vec—on languages that don't use Latin alphabets, particularly with languages that use Chinese characters. (Xu,)

Which raises an interesting question: is it easier to map Chinese characters to a semantic vector space than it is to map Latin characters? And if so, is this because Chinese characters tend to be used as ideograms (that is, direct encoding of semantics) while Latin characters are used to encode sounds?

Risch used a recurrent network to train fastText word embeddings using WIPO-alpha, USPTO-5M, and USPTO-2M. Risch also provides quite a few references to older research in training patent specific word embeddings.

Dataset

Google's Public Patent Dataset was used for this project. Google ingests, processes, and publishes the dataset once a quarter. It includes abstracts, claims, and descriptions of given patents along with other metadata like classification codes from around the world for the given patents, assignees, languages, and so on.

When a version of the data is fresh, it has no timestamp in its title. When it is deprecated, it is frozen and renamed with a timestamp in its title. It doesn't change significantly from version to version unless the team publishes some new code to production.

To avoid having source data change underneath me, I used the freshest deprecated data, table `publications_202307`. Furthermore, to avoid paying massive fees for repeated Google Cloud BigQuery access, I queried the table once and recorded the result locally as JSON, selecting one thousand random rows with:

```
SELECT * FROM publications_202307
ARRAY_LENGTH(description_localized) > 0 ORDER BY RAND() LIMIT
1000
```

The Google Patents Dataset uses the `description_localized` field to store the data most similar to what a patent examiner will need to classify on new patent applications. So, the population for the random sample was limited to rows containing localized descriptions in order to ensure that the resulting vector space is populated with vectors that can associate descriptive text with resultant patent classification codes.

Data sampled from the Google Patent Public Dataset was then sent to the OpenAI Embedding Service for conversion to embeddings.

The Google Patent Public Dataset is hosted on Google Cloud Big Query, a database engine that doesn't use explicit primary keys in the sense that many structured databases do. This query was run on the dataset to determine that the `publication_number` column of the `publications_202307` table is used as a primary key and unique sample ID:

```
SELECT COUNT(*) as total_rows, COUNT(DISTINCT {column_name}) as
distinct_rows FROM publications_202307
```

Where `{column_name}`is a variable that varies over all of the names of all of the columns in the table. `publication_number` was the only column with a one-to-one mapping of rows to values.

So, each sample and set of associated word embeddings were indexed by Publication Number, the identifying number assigned to a patent by one of the world's patent offices. It's interesting to note that each invention may be awarded a patent by more than one patent office, so each invention may have more than one publication number. But, this captures nicely the reality that the semantics of a single invention varies depending on the context in which it is patented. So, publication numbers capture this fanning out of invention to localized semantics nicely.

Data Preparation

Samples were divided into two non-overlapping subsets: Model Samples and Query Samples.

Model Samples were converted to word embeddings using OpenAI's Embedding Service. To make sure the direction of each vector is influenced by both the sample's patent text and its classification codes, every API request included a set of base data taken from the associated patent sample as well as a subset of non base data. Examples of base data include classification codes, patent assignees, assignment dates, and so on. Examples of non base data include localized patent descriptions, patent claims, and abstracts.

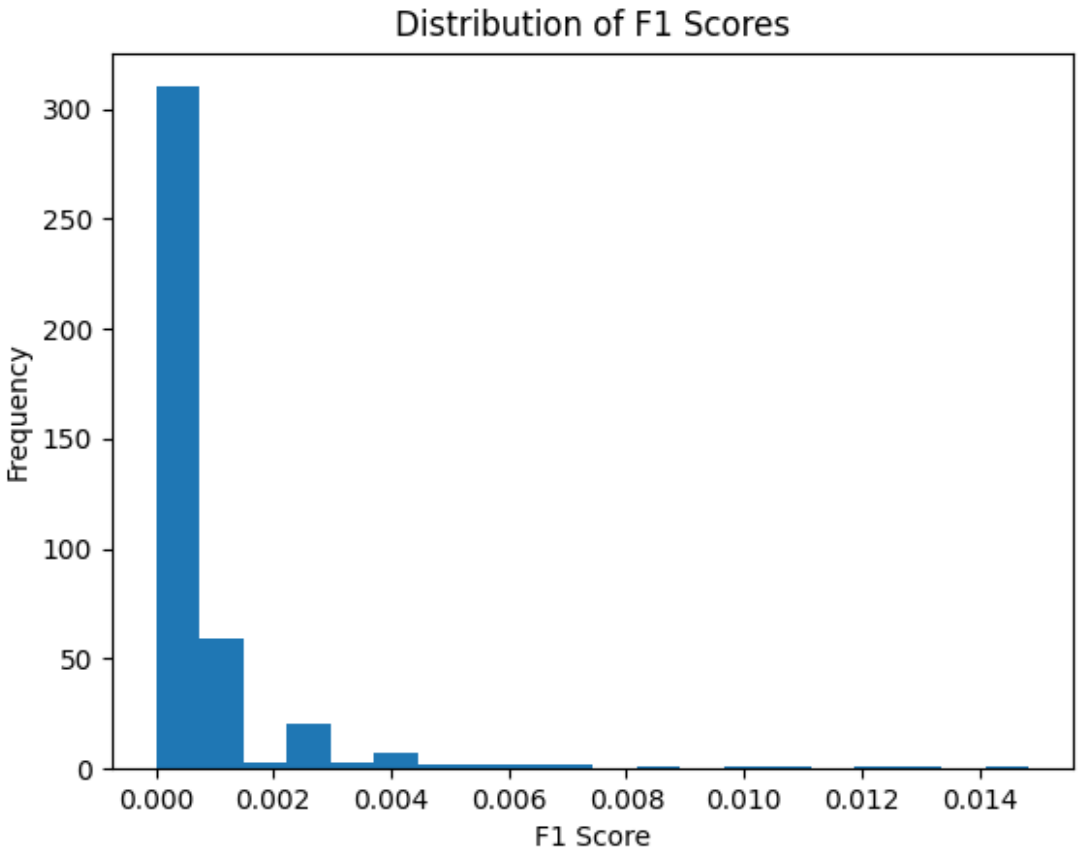
To minimize financial costs, each Model Sample sent to the embedding API consisted of base model tokens merged with non-base model tokens as described above, converted to JSON, and then from that JSON most whitespace and characters related to data structure were removed.

Calls to OpenAI's embedding service are limited to 8,192 tokens per API call. Roughly half of the thousand samples taken from the Google Patent Public Dataset map to more than 8,192 tokens. So, samples were only sent to the embedding service for use in the model if their merged base and non-base tokens had less than 8,192 tokens, resulting in embeddings for 584 Model Samples and 416 Query Samples.

Query Samples were prepared by extracting data from their `description_localized` field in all available languages. Each description in each language was partitioned into several overlapping partitions such that each partition maps to 8,192 or fewer tokens. Each partition was then sent to the OpenAI Embedding Service, resulting in at least one embedding per Query Sample.

Cosine similarity was then used to compare the resulting embeddings in the query to the embeddings in the Model Set. Response Embeddings from the Model Set within some distance of a vector in the query were returned. Response Embeddings were then mapped to the samples from which they were created. All patent classifications were extracted from those samples and compared with the patent classifications of the samples that produced the query using the standard calculation of F1 Score.

Running this process resulted in an average F1 score of `0.00070961406366366699` across all queries, or essentially zero. Which, as baselines go, is not hard to beat.



Code

Code can be found here: https://github.com/LukeMizuhashi/nlp-dartboard/tree/sidestep

Future Work

Only sending samples that fit within OpenAI's Embedding Service's maximum token limit biases the study to patents about which the Google Patent Public Dataset has limited information. A script was written that ingests patent samples and uses binary search along with OpenAI's `tiktoken` tokenization library to partition each sample into several API requests, each with the maximum number of tokens possible. Each API response results in one embedding vector, so there are one or more embedding vectors per sample when using this script. There wasn't time to implement observability code to verify the script is working as intended, however. So, this script was not used in this study. Future studies should investigate the use of multiple vectors for samples that map to large token sets.

The Google Public Patent Dataset is one of the most complete collections of patent data in the world, spanning literature from several global patent offices and all classification codes and time periods. It also contributes far more tokens to the Common Crawl dataset than many other well

established sources of information (Dodge). Future work may entail analysis of the distribution of labels in samples taken from the dataset. This project used all code-related fields to calculate F1 scores. Future efforts that use Google Patent Public Dataset might benefit from only using code-related fields that are well populated in the source data itself.

Several patent offices around the world offered use of their patent APIs. Future work will use them. Unlike the Google Patent Public Dataset, these APIs often provide full patent text, diagrams, and associated documents like patent office actions and litigation.

Future work may also entail training word embeddings from scratch and the application of more complex techniques like transformers with attention and text-GCN.

Future work may entail sending samples of patent applications to various large language models to see how well they predict classification codes.

I would also spend time working on development of infrastructure that allows careful examination and validation of data as it passes through the code written for this project.

References

1. Asudani, D.S., Nagwani, N.K. & Singh, P. Impact of word embedding models on text analytics in deep learning environment: a review. Artif Intell Rev 56, 10345–10425 (2023). https://doi.org/10.1007/s10462-023-10419-1

2. Yao L, Mao C, Luo Y (2019) Graph Convolutional Networks for Text Classification. Thirty-Third AAAI Conf Artif Intell 19. https://doi.org/10.1609/aaai.v33i01.33017370

3. Giarelis N, Kanakaris N, Karacapilidis N (2020) On a novel representation of multiple textual documents in a single graph. Smart Innov Syst Technol 193:105–115. https://doi.org/10.1007/978-981-15-5925-9_9/TABLES/1

4. Xu D, Tian Z, Lai R et al (2020) Deep learning based emotion analysis of microblog texts. Inf Fusion 64:1–11. https://doi.org/10.1016/j.inffus.2020.06.002

5. Almuhareb A, Alsanie W, Al-thubaity A (2019) Arabic word segmentation with long short-term memory neural networks and word embedding. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2893460

6. Risch, J. and Krestel, R. (2019), "Domain-specific word embeddings for patent classification", Data Technologies and Applications, Vol. 53 No. 1, pp. 108-122. https://doi.org/10.1108/DTA-01-2019-0002

7. Google Patents. (2023). Patent Coverage. Available at: https://patents.google.com/coverage

8. Dodge, J., Sap, M., Marasovic, A., Agnew, W., Ilharco, G., Groeneveld, D., Mitchell, M., & Gardner, M. (2021). Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. arXiv:2104.08758v2 [cs.CL].

9. OpenAI. (2023). Embeddings - OpenAI API. Available at: https://platform.openai.com/docs/guides/embeddings

10. Google Cloud. (2023). Google Patents Public Data. Available at: https://console.cloud.google.com/marketplace/product/google_patents_public_datasets/google-patents-public-data

11. OpenAI. (2023). tiktoken. Available at: https://github.com/openai/tiktoken