Automatic classification of patents into CPC categories using NLP

Tolu Bukola

August 2020

The patent processing challenge

- The U.S. patent office (USPTO) has issued over 6 million patents. Every year > 500,000 applications are filed.
- A first critical step in patent processing is sorting patents into the right topic areas so that they can be read by relevant experts and compared to the existing corpus. This can be accelerated by machine learning.
- ML-based classification methods can be extended to assist patent research, key for any IP –driven company.
- The goal of this project is to classify patents into one of 9 Cooperative Patent Classification ('CPC') topic areas based on the text of a patent's abstract and descriptions. Topic areas are specified by the first letter of the CPC:
  - A - Human Necessities
  - B – Performing Operations; Transporting
  - C – Chemistry; Metallurgy
  - D – Textiles; Paper
  - E – Fixed Constructions
  - F – Mechanical Engineering; Lighting; Heating; Weapons; Blasting
  - G – Physics
  - H – Electricity
  - Y – New developments; Cross Sectional Technologies

Machine learning can accelerate the labor-intensive patent review process

Data gathering and preparation

- Google Patents Public data is a 120M row database of patents dating back to the 18th century. It includes full text for U.S. patents and bibliographical data for international patents. It is stored in Google Cloud and accessed via SQL in BigQuery, Google's enterprise data warehouse.
- Performed 9 SQL queries (~1TB each) on Google Patents data, downloading 1000 random examples of a patent from each of the 9 classes since 2000. Combined these patents into one large dataframe.
- Created target column "label", which is the first letter of the cpc_code.
- Two columns will be used for classification:
  - **abstract_localized_text:** the abstract is a summary of the patent. These are on average 110 words long.
  - **description_localized_text:** the full patent description. These are on average 12,000 words long.

```
RangeIndex: 9000 entries, 0 to 8999
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      9000 non-null   int64
 1   application_number              9000 non-null   object
 2   publication_date                9000 non-null   int64
 3   title_localized_text            9000 non-null   object
 4   abstract_localized_text         9000 non-null   object
 5   claims_localized_text           9000 non-null   object
 6   description_localized_text      9000 non-null   object
 7   f0_                             9000 non-null   object
 8   assignee_harmonized_country_code 8211 non-null   object
 9   ipc_code                        9000 non-null   object
 10  cpc_code                        9000 non-null   object
 11  label                           9000 non-null   object
dtypes: int64(2), object(10)
memory usage: 843.9+ KB
```

Data was gathered from Google Cloud's BigQuery using SQL

NLP representation of data: Term Frequency, Inverse Document Frequency

- **Key NLP terms/concepts:**
  - **"Bag-of-words"**: documents represented as de-contextualized set of words ("record the play" vs. "play the record")
  - **Document:** individual collection of words (e.g. a patent abstract or description)
  - **Corpus:** set of documents (e.g. full set of patent abstracts or descriptions)
  - **Term frequency:** proportion of times each term is seen in a document
  - **Document frequency:** proportion of documents containing a term
  - **Term Frequency, Inverse Document Frequency:** A composite metric that rewards terms that occur most often within a document and penalizes terms that occur most often across most documents. Logically it favors terms that are most unique to a document
- **Step 1:** Transform input X matrices into TF-IDF matrices. Parameter tuning determines the number of features:
  - **Min-df: minimum number of documents a term can occur in to be considered relevant**
  - Max-df: minimum number of documents a term can occur in to be considered relevant
  - Stop words: set of words that are removed from the vocabulary
- **Step 2:** Train and evaluate labels, y, on TF-IDF input matrices.

Features matrices were built with Tfid transformers

Summary of parameter tuning and model selection

Model no.	Min_df	Model	Features	Train score	Test Score
Abstract text input					
1	0%	Logistic	15,188	76%	57%
2	1%	Logistic	690	61%	51%
3	2%	**Logistic**	**346**	**53%**	**49%**
4	5%	Logistic	98	39%	36%
5	2%	**Linear SVM**	**98**	**55%**	**48%**
6	2%	Poly SVM	98	95%	46%
7	2%	Radial SVM	98	77%	49%
8	2%	**Sigmoid SVM**	**98**	**46%**	**47%**
9	2%	**KNN (K = 20)**	**98**	**46%**	**40%**
Description text input					
10	0%	Logistic	340,000	78%	65%
11	5%	Logistic	2,500	71%	62%
12	10%	Logistic	1,500	64%	59%
13	20%	**Logistic**	**640**	**61%**	**57%**
14	20%	**Linear SVM**	**640**	**62%**	**57%**
15	20%	Poly SVM	640	91%	57%
16	20%	Radial SVM	640	78%	59%
17	20%	**Sigmoid SVM**	**640**	**54%**	**55%**
18	20%	**KNN (K = 20)**	**640**	**52%**	**51%**

- Models (10,11) with no minimum document frequency demonstrate dramatic over-fit.
- Models based on description text score higher than models based on abstract text even with the same number of features (2,13), indicating description text contains extra information.
- Polynomial and radial decision boundaries lead to dramatic overfitting (6,7,15,16).
- Non-parametric models close the performance gap when more data is added (17,18).
- Accuracy of ~51% is well above the 11% base-line from guessing among 9 classes.

Logistic, SVM, and KNN models show strong performance

Model Evaluation (1/3): Evaluation of logistic models

Logistic Model on abstract text w/ 346 features

	precision	recall	f1-score	support
A	0.54	0.47	0.50	195
B	0.34	0.29	0.32	211
C	0.47	0.69	0.56	179
D	0.65	0.64	0.64	191
E	0.59	0.67	0.63	203
F	0.59	0.64	0.62	207
G	0.48	0.49	0.48	209
H	0.54	0.54	0.54	204
Y	0.30	0.19	0.23	201
accuracy			0.51	1800
macro avg	0.50	0.51	0.50	1800
weighted avg	0.50	0.51	0.50	1800

Logistic Model on description w/ 641 features

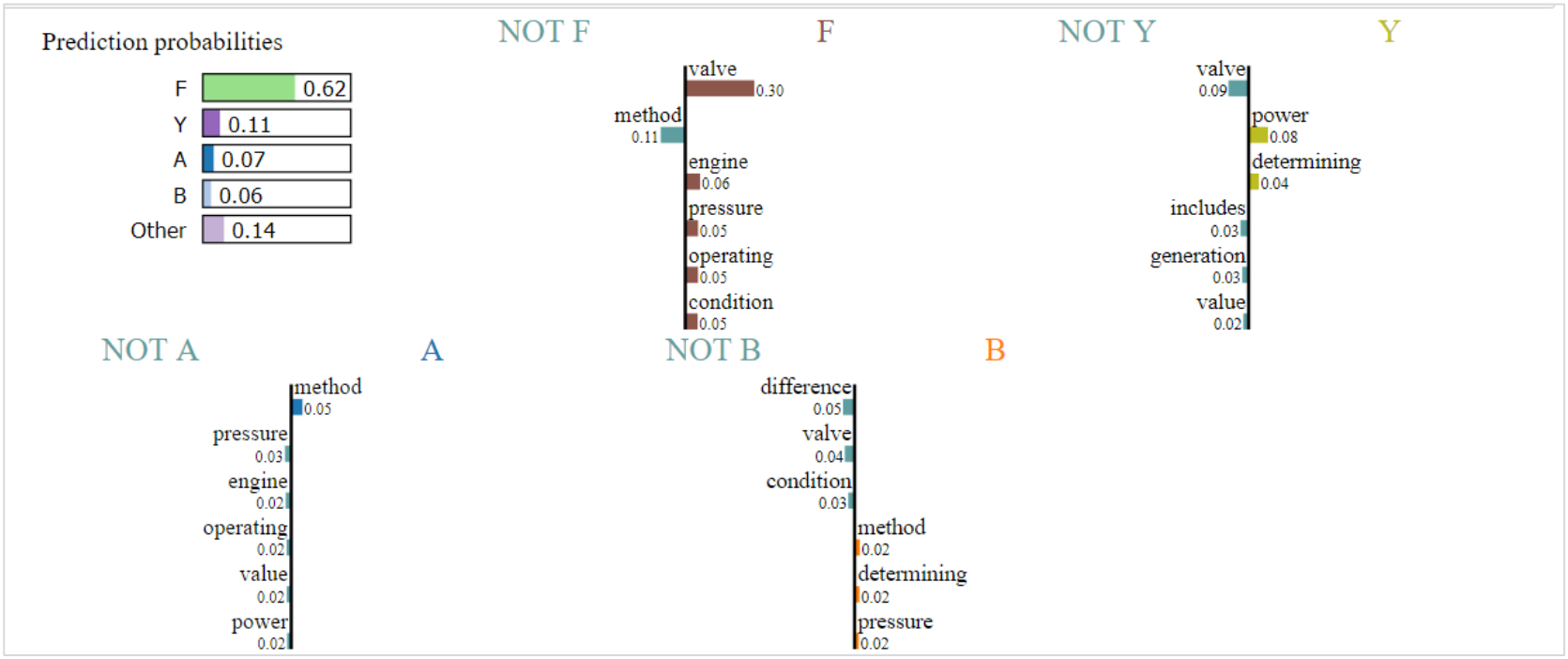
	precision	recall	f1-score	support
A	0.66	0.58	0.62	195
B	0.40	0.26	0.32	211
C	0.53	0.74	0.62	179
D	0.69	0.75	0.72	191
E	0.59	0.71	0.65	203
F	0.64	0.73	0.68	207
G	0.54	0.58	0.56	209
H	0.58	0.62	0.60	204
Y	0.30	0.16	0.21	201
accuracy			0.57	1800
macro avg	0.55	0.57	0.55	1800
weighted avg	0.55	0.57	0.55	1800

- Recall is lowest for group Y, “cross-sectional-technologies”, which is a composite of topic areas.
- Precision is low for group Y and group B, “performing operations/transporting”. Group B consists of technologies which assist in transforming (mixing, boiling, melting) or transporting materials. It is thus, like group Y, a multidisciplinary composite grouping.
- Accuracy of ~51% is well above the 11% base-line from guessing among 9 classes.
- Adding more data (longer descriptive text) improved predictions on every category except the non-specific groups B and Y.

Logistic model is highly efficacious but overfit

LIME (Local Interpretable Model-Agnostic Explanations) example

'A method for diagnosing a waste gate valve malfunction in a power generation system is presented. The method includes determining an actual pressure differential across a throttle valve. The method further includes determining an estimated pressure differential across the throttle valve based on one or more first operating parameters of the power generation system. Furthermore, the method includes determining an absolute difference between the actual pressure differential and the estimated pressure differential. Moreover, the method also includes comparing the absolute difference with a threshold value and if the absolute difference is greater than the threshold value, determining an operating condition of the throttle valve. Additionally, the method includes determining whether the waste gate valve has malfunctioned based on the determined operating condition of the throttle valve. An engine controller and a power generation system employing the method are also presented.'



LIME output for patent predicted class F(mechanical engineering)

Unresolved issues and additional work

- Train the model over the entire dataset of ~10M+ patents using Google Cloud.
- Consider improvements to TFIDF and bag of words:
  - LSTM (long-term-short-term) neural nets to consider word sequence
  - BERT (bidirectional encoder representation) neural network technique
  - Okapi BM25 best matching to estimate relevance of documents to search query
- Apply neural network models instead of logistic, SVM, KNN.
- Create flask app to provide classification given patent text.

There is significant scope for model improvement

Model Evaluation (2/3): Comparison of SVM models

Linear SVM on abstract text w/ 346 features

	precision	recall	f1-score	support
A	0.47	0.44	0.45	195
B	0.37	0.27	0.32	211
C	0.47	0.64	0.54	179
D	0.52	0.55	0.54	191
E	0.47	0.57	0.51	203
F	0.54	0.60	0.57	207
G	0.53	0.50	0.51	209
H	0.54	0.57	0.56	204
Y	0.32	0.19	0.24	201
accuracy			0.48	1800
macro avg	0.47	0.48	0.47	1800
weighted avg	0.47	0.48	0.47	1800

Linear SVM on abstract text w/ 640 features

	precision	recall	f1-score	support
A	0.69	0.56	0.62	195
B	0.36	0.26	0.30	211
C	0.51	0.78	0.62	179
D	0.72	0.76	0.74	191
E	0.59	0.71	0.65	203
F	0.62	0.72	0.67	207
G	0.53	0.59	0.56	209
H	0.60	0.62	0.61	204
Y	0.33	0.14	0.20	201
accuracy			0.57	1800
macro avg	0.55	0.57	0.55	1800
weighted avg	0.55	0.57	0.55	1800

Sigmoid SVM on abstract text w/ 346 features

	precision	recall	f1-score	support
A	0.46	0.46	0.46	195
B	0.37	0.28	0.32	211
C	0.44	0.61	0.51	179
D	0.51	0.52	0.52	191
E	0.46	0.55	0.50	203
F	0.54	0.59	0.57	207
G	0.50	0.47	0.48	209
H	0.54	0.58	0.56	204
Y	0.29	0.17	0.22	201
accuracy			0.47	1800
macro avg	0.46	0.47	0.46	1800
weighted avg	0.46	0.47	0.46	1800

Sigmoid SVM on abstract text w/ 640 features

	precision	recall	f1-score	support
A	0.65	0.55	0.59	195
B	0.33	0.23	0.27	211
C	0.49	0.76	0.60	179
D	0.70	0.75	0.73	191
E	0.59	0.69	0.64	203
F	0.60	0.72	0.66	207
G	0.53	0.57	0.55	209
H	0.59	0.63	0.61	204
Y	0.30	0.12	0.17	201
accuracy			0.55	1800
macro avg	0.53	0.56	0.53	1800
weighted avg	0.53	0.55	0.53	1800

SVM models improve dramatically with addition of data

Model Evaluation (3/3): Comparison of KNN models

K= 20NN on abstract text w/ 346 features

	precision	recall	f1-score	support
A	0.41	0.28	0.33	195
B	0.31	0.20	0.24	211
C	0.35	0.67	0.46	179
D	0.46	0.46	0.46	191
E	0.43	0.50	0.46	203
F	0.46	0.52	0.49	207
G	0.43	0.44	0.44	209
H	0.44	0.50	0.47	204
Y	0.29	0.13	0.18	201
accuracy			0.41	1800
macro avg	0.40	0.41	0.39	1800
weighted avg	0.40	0.41	0.39	1800

K= 20NN on abstract text w/ 640 features

	precision	recall	f1-score	support
A	0.49	0.41	0.45	195
B	0.43	0.25	0.31	211
C	0.47	0.71	0.56	179
D	0.66	0.72	0.69	191
E	0.50	0.60	0.55	203
F	0.53	0.63	0.57	207
G	0.53	0.56	0.54	209
H	0.52	0.62	0.56	204
Y	0.41	0.17	0.24	201
accuracy			0.51	1800
macro avg	0.50	0.52	0.50	1800
weighted avg	0.50	0.51	0.50	1800

KNN models improve dramatically with addition of data