Multi-class Multi-label Patent Classification with Hybrid Neural Networks

Saurav Datta, Arvindh Ganesan, Christina Papadimitriou
Summer 2018

Code for Data parsing:
https://github.com/cpapadimitriou/W266-Final-Project/blob/master/preparation/parse_xml.py

Code for Data Processing, EDA and CNN and Capsule Network model:
https://github.com/cpapadimitriou/W266-Final-Project/blob/master/Main_Notebook.ipynb

Code for BILSTM *error* analysis
https://github.com/cpapadimitriou/W266-Final-Project/blob/BILSTM/Main_Notebook_New.ipynb

Abstract

Patent automatic classification (PAC) tasks have drawn a lot of research interest in the recent years as the worldwide patent application submissions keep increasing. Most of the currently used approaches for automatic patent classifications are dependent on the knowledge of domain experts. In this project we propose a hybrid neural network model for multi-label multi-class patent classification, which will be able to capture important features and long term dependencies. First we ran a baseline model, namely a Convolutional Neural Network (CNN) to establish a baseline for our classification task. Then we modeled a BI-directional Long-Short-Term memory (LSTM) network, which is similar to the one discussed in the paper but with tweaks such as using TF-IDF methodology to select salient words from the patent documents. Finally, we propose a Capsule Network model architecture as an improvement over the other two models. We have described the Capsule Network model architecture in our report here.

Keywords: text feature extraction, multi-label patent classification, hybrid neural networks

1 Introduction

Patent classification is the process of categorizing a patent or a patent application into corresponding categories. The World Intellectual Property Organization (WIPO) has created the International Patent Classification (IPC) system as a standard taxonomy for patent classification. When a patent application is submitted to a local patent office, the patent examiners classify the patents and use the classification labels to search for previous inventions in the same field. This procedure helps them decide whether a patent should be granted or not, and hence it is a crucial part of the patent filing process. Patent filing is a field where domain experts are involved and for which companies charge a fee. Some of the companies in this space are: patentsearchinternational, planetpatent, cpaglobal. There are ongoing attempts to automate this search as much as possible, the most prominent being the tie-up between Google and USPTO.

Patent Automatic Classification (PAC) tasks have the objective of finding the best candidate IPC labels, as patent examiners do. Considering the time-consuming and labor-intensive nature of patent classification, automating this process, and doing so accurately, will have significant benefits in terms of money and time. However, the complicated hierarchical classification scheme and the highly technical terminology of patent documents, make the development of a high performance PAC system challenging.

Patent classification, from the perspective of Natural Language Processing, is interesting because of the following reasons:

1) It is a multi-label multi-class classification problem.
2) Classifying a patent means capturing the intent of the invention.
3) Unique features in patent documents make traditional information retrieval methods used for Web or ad hoc search inappropriate or at least of limited applicability.

This project adopts a deep learning algorithm in order to create a high-performance PAC system.

2 Background

2.1 Patent Classification

Patent classification is mainly based on the IPC taxonomy, which is a hierarchical structure consisting of sections, classes, subclasses, groups and subgroups. The purpose of a PAC system is to find the set of IPC labels that best represent the patent.

A number of algorithms have been proposed for developing a PAC system. They include SVM classifiers, parse network of winnows (SNoW), Bayesian classifiers, and neural network classifiers. A couple of them worth mentioning are:

1. Verberne [1] conducted a series of classification experiments with the linguistic classification system (LCS) based on Naive Bayes, Winnow and SVMlight in the context of Conference and Labs of the Evaluation Forum Intellectual Property (CLEF-IP) 2011. The classification precision score of 74.43%.
2. Li, Tate et al. [2] proposed a two-layered feed-forward neural network and employed the Levenberg–Marquardt algorithm to train the network for 1948 patent documents from United States Patent Classification (USPC) 360/324. The authors used a stemming approach, the Brown Corpus, to handle most irregular words. They achieved an accuracy of 73.38% and 77.12% on two category sets, respectively.
3. Finally, Hierarchical Feature Extraction Model for Multi-Label Mechanical Patent Classification is proposed by Jie Hu et al [3]. This architecture includes a CNN for feature extraction and a Bi-LSTM for multi label classification. This model leverages the local lexical level and global sentence level features effectively. They achieved a precision of 81.55%.

2.2 Feature Extraction using Deep Learning

In the recent past, there has been a lot of developments in NLP using deep learning algorithms. As word embeddings started producing interesting results, it also gave way to the idea of creating higher level feature representation of contexts, sentences, global features, etc. CNN is a popular deep learning algorithm to map a sequence of words (word embeddings) to a high-dimensional, nonlinear features. CNN uses a series of convolutional filters on a sequence of words to extract *n*-gram features at different positions of text. Xiang et al. [4] built a character-level CNN model for several large-scale datasets to show that it could achieve state-of-the-art or competitive results.

CNN does well with local lexical patterns, it does not do as well when it comes to a long range dependencies. This is where RNNs come in. RNNs are designed to work well with long range sequential dependencies. RNN has been successfully applied to a variety of problems: speech recognition, language modeling, translation, image captioning, question answering, text summarization etc. RNN does suffer from the vanishing / exploding gradient issue when the gap between two-time steps becomes too large. LSTMs overcome this issue by controlling the information that that is let through the subsequent time steps through the notion of gates.A bidirectional LSTM network is a variation of LSTM which consist of two separate LSTM networks to store the context in both directions;

Ying et al. [5] combined CNN and BiLSTM to extract Chinese events from unstructured data. They employed CNN and LSTM to capture both lexical-level and sentence-level features. The proposed method achieved competitive performance in several aspects on the Automatic Content Extraction (ACE) 2005 dataset.

Siwei Lai et al [6] used recurrent convolutional neural network for text classification that (as per the proposal) to learn more of the contextual / semantic information. They applied a bi-directional recurrent structure, which may introduce considerably less noise compared to a traditional window- based neural network, to capture the contextual information to the greatest extent possible when learning word representations.

3 Methods

3.1 Data

3.1.1 Dataset

The dataset used in this project is a subset of the CLEF-IP 2011 dataset. The CLEF-IP 2011 dataset consists of 1.35 million patents from the European Patent Office (EPO) and the World Intellectual Property Organization (WIPO), filed between 1978 and 2009. These 1.35 million patents contain three kinds of language contents, namely English, German, and French. The patent data was in XML format and had to be parsed to be used for the patent classification task. Each patent document is identified by its application number. For each document, we decided to extract the following sections: title, abstract, claims, description and labels. The labels were mapped from label codes to label descriptions using the patent labels mapping. The data can be found here:

- Patent data: http://www.ifs.tuwien.ac.at/~clef-ip/download/2011/index.shtml
- Patent Labels: http://www.wipo.int/ipc/itos4ipc/ITSupport_and_download_area/20110101/IPC_scheme_title_list/EN_ipc_section_C_title_list_20110101.txt

Generally speaking, the title of a patent indicates the name of the patent, and the abstract gives a brief technical description of the innovation. The claim section's main function is to protect the inventors' rights. The description section describes the process, the machine, manufacture, composition of matter, or improvement invented, the background of the invention, and a description of its application. Finally, the classification part presents one or multiple class labels. A patent document also includes additional sections, such as meta-information about the inventor, which were excluded from our project's scope.

In our analysis, we only consider documents that are written in English, and contain at least one IPC classification label. Additionally, documents that were missing any of the sections mentioned above (title, abstract, claims, and description), or had less than 200 words in total, were excluded. Sections with multiple paragraphs were merged into one text, and in cases where multiple document versions existed for a single patent (recognized by the unique patent application number), only the first seen patent was considered. After the data cleaning process, our dataset consists of 78,172 patent documents, of which 62,537 were used in the train set and 15,635 in the test set.

The four sections of the patent document mentioned above (title, abstract, claims, and description) were concatenated into one text. Figure 1 below shows the distribution of the document length in our dataset after concatenation, measured by the number of words in each document. The median document length is 3,488 words and the average 4,433 (positively skewed distribution).

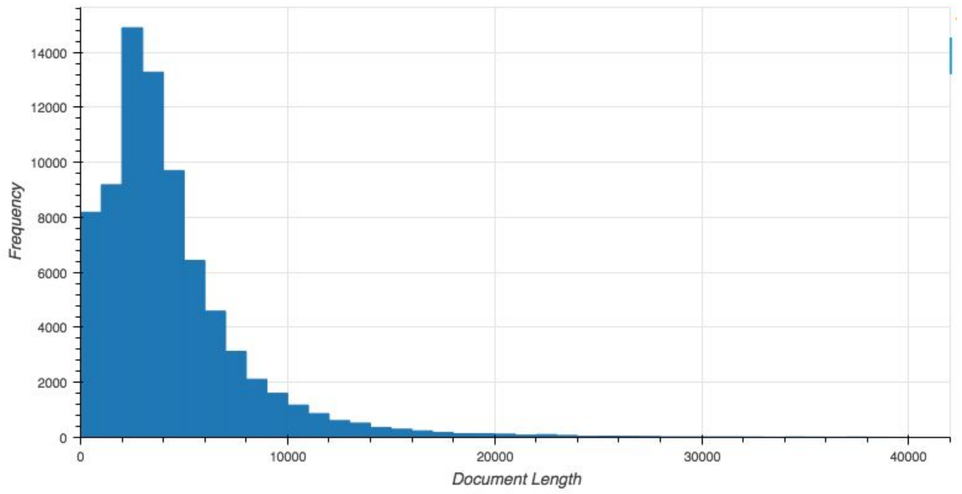


Figure 1. Document Length histogram

3.1.2 Feature Processing

The next step was to pre-process the full text (title, abstract, claims, description) in all patent documents in order to enhance performance for our classification task. The processing of the patent document text included the following steps:

1. Removing punctuation and symbols
2. Lower-casing all words
3. Removing stop words
4. Lemmatizing (i.e. converting a word into its root word)
5. Removing digits
6. Removing URLs

3.1.3 TF-IDF for Feature Selection

TF-IDF or term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is comprised of two components - term frequency and inverse document frequency.

Term frequency (TF) of a word is simply the number of times a word occurs in a given document. Inverse document frequency (IDF) of a word is the log of total number of documents over number of documents that contain that specific word. This means that if a word occurs in all of the documents, then it’s IDF value is zero but if the term appears in just one document, then it’s IDF value will be large. In other words, IDF is an indicator of importance of a word considering the whole corpus. Thus, the combination of sheer number of times a word has appears in a document and its overall importance in the corpus can play a very crucial role in determining the importance of a word in a given document.

TF-IDF Calculation process on patent documents:

150 words from each section

We combined the title, claim, abstract and description sections of the patents to form the patent corpus. We then fed this corpus to a TF-IDF vectorizer to get the TF-IDF for each of the words for a given patent document. Then, we applied the TF-IDF vectorizer transform function to the individual sections of the patent document to get the TF-IDF for each of the individual sections. Then for each patent document and for each section, we extracted the top 150 words based on the corresponding TF-IDF ranking. In the end, for each patent document, each of the sections - claim, title, abstract and description had up to 150 words.

We combined the title, claim, abstract and description sections of the patents to form the patent corpus. We then fed this corpus to a TF-IDF vectorizer to get the TF-IDF for each of the words for a given patent document. For each patent document, we then extracted the top 600 words with the highest TF-IDF score.

3.1.4 Encoding Labels

As mentioned before, the IPC is a hierarchical classification system. For this project we have chosen the subclass level, which is the 3rd level in the IPC hierarchy, and the most commonly used in other PAC applications according to our literature review. On average, each patent in our dataset had 1.4 classification labels at the subclass level. The maximum number of labels was 8 and the minimum 1, and there were 96 unique labels across our dataset.

A crucial step for our multi-class multi-label classification task, where each patent can have more than one labels from a pool of 96 possible classes, was to transform the target variable (y) from a list of labels to a binarized matrix. This matrix had 96 columns (each represented a label), and each row represented one patent document. The matrix was filled with 0's and 1's, indicating the labels for each patent document. For example, if a document was classified with 3 labels, each row in the matrix would have 1's under these 3 labels, and zeros everywhere else.

Figure 2 shows the number of times that each label occurs across all patent documents. It is important to notice that we are dealing with a classification problem that has unbalanced classes (i.e. some labels occur more often than others). For example, some classes occur almost 6000 thousand times, while others 6 only times. Our error analysis showed that our https://stackoverflow.com/questions/48485870/multi-label-classification-with-class-weights-in-kerasCNN baseline model "suffered" from the unbalanced class problem, resulting in always predicting the most frequent class with the highest probability. To deal with this issue we tried applying weights to the classes during training based on frequency. However, balancing the classes in a multi-label classification problem is a complex task, and our simple implementation did not yield better model performance.
Per the discussion on similar problems [7] , it seems we need to weigh our metrics too. Additional measures such as Cohen's Kappa and Receiver Operating Characteristic (ROC) curve [8] have been suggested as providing a better measurement in such situations.

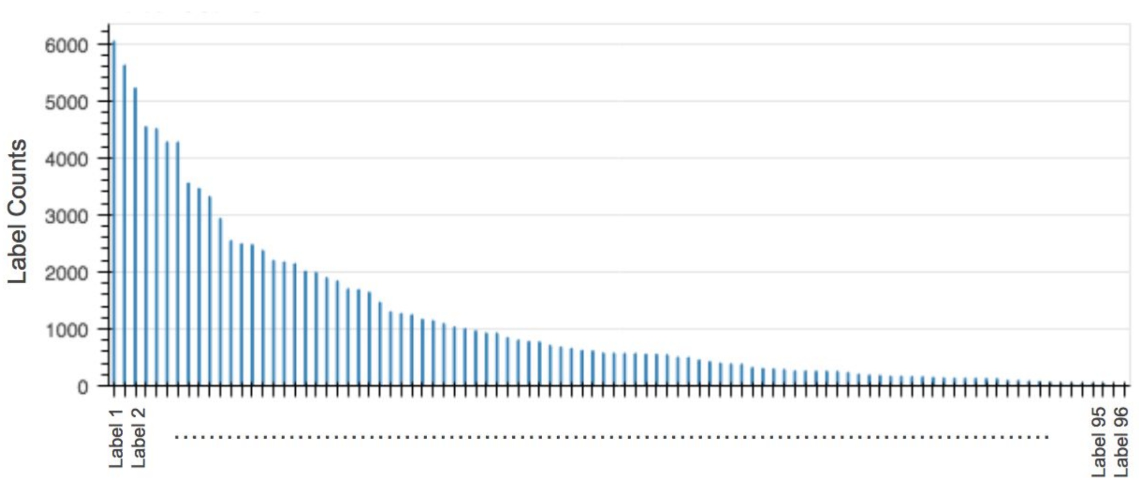


Figure 2. Label counts across patent documents

3.2 Word Embeddings

We used word embeddings to map the words in the patent document into dense vector representations. Word embeddings represent each word well from both syntactic and semantic perspectives.

3.2.1 Pre-trained Word Embeddings

At first, we used the wiki news pre-trained word embeddings, trained in a corpora of 1 million words, with embedding dimension of 300. However, the pre-trained word embeddings are likely not capturing the technical and complex language that is used in patent documents.

3.2.2 Training Word Embeddings with Word2Vec

Patent documents often contain language that is not part of general language but rather constitutes specialized technical terms. These technical terms may contain crucial information about the categories that we are trying to classify. Thus, we decided to train our own word embeddings using the CBOW word2vec algorithm.

After preprocessing the patent document to remove punctuations, white spaces and converting the text to lowercase, each of the four sections - title, claim, abstract and description were combined to form a patent corpus. This corpus was then fed to a CBOW algorithm to train word embeddings. We kept the embedding dimension size to 100.
The total vocabulary size was 581702. Thus, the output of the CBOW model is an embedding matrix with dimensions 581702 x 100.

3.3 Models

The input to our models is the concatenated text from patent documents (including sections: title, abstract, claims, description) and padded to sequence length 600. The output of our models is the probabilities for the 96 IPC labels for each document. In order to get these probabilities we use as a final layer a fully-connected layer with sigmoid activation.
It is important to mention that we are using sigmoid activation instead of softmax. Softmax would not be appropriate for a multi-label classification because it implies that the probability for a class is dependent on the probabilities for the other classes (i.e. they add up to one). In our case the probabilities for each class were calculated through sigmoid activation.

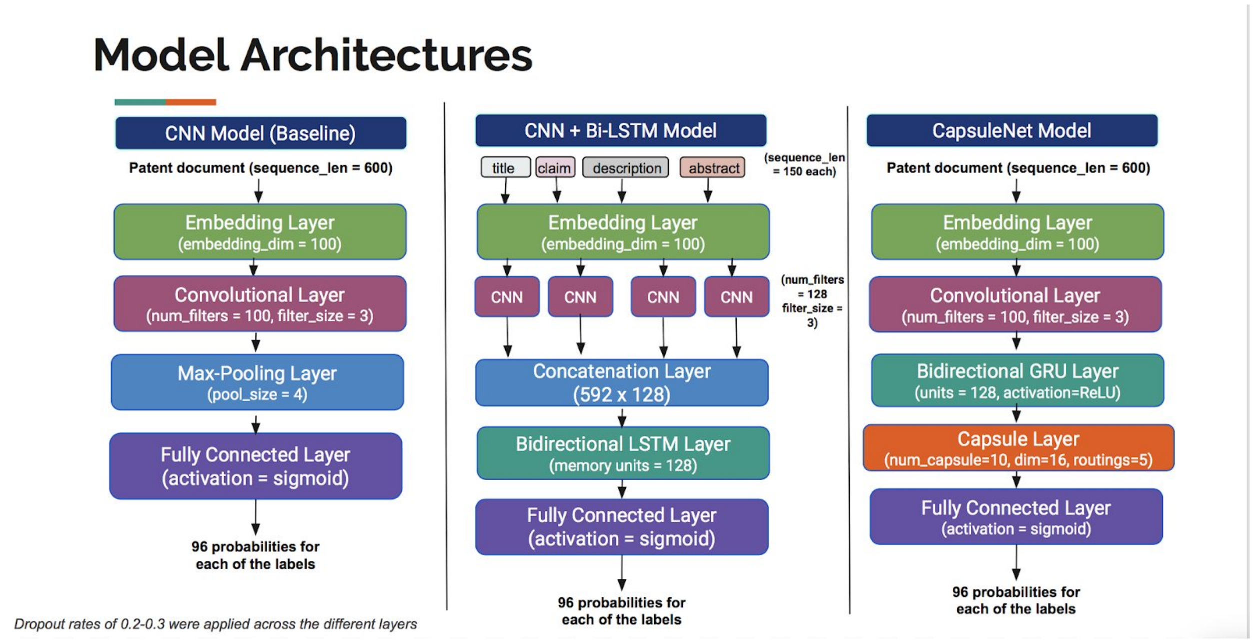

Model Architectures
CNN Model (Baseline)
Patent document (sequence_len = 600)
Embedding Layer (embedding_dim = 100)
Convolutional Layer (num_filters = 100, filter_size = 3)
Max-Pooling Layer (pool_size = 4)
Fully Connected Layer (activation = sigmoid)
96 probabilities for each of the labels
Dropout rates of 0.2-0.3 were applied across the different layers
CNN + Bi-LSTM Model
title
claim
description
abstract
(sequence_len = 150 each)
Embedding Layer (embedding_dim = 100)
CNN
CNN
CNN
CNN
(num_filters = 128 filter_size = 3)
Concatenation Layer (592 x 128)
Bidirectional LSTM Layer (memory units = 128)
Fully Connected Layer (activation = sigmoid)
96 probabilities for each of the labels
CapsuleNet Model
Patent document (sequence_len = 600)
Embedding Layer (embedding_dim = 100)
Convolutional Layer (num_filters = 100, filter_size = 3)
Bidirectional GRU Layer (units = 128, activation=ReLU)
Capsule Layer (num_capsule=10, dim=16, routings=5)
Fully Connected Layer (activation = sigmoid)
96 probabilities for each of the labels

Figure 3. Model Architectures

3.3.1 Baseline Model

Our goal for the baseline model was to extract simple n-gram features from the patent text by using a single layer CNN model. The CNN model includes an embedding layer (using the pre-trained word embeddings mentioned above), a convolutional layer with 100 filters, filter size 3 by 100 (kernel size by *embedding_dim)*, which uses a ReLU activation and is followed by a max-pooling function. A fully-connected layer with sigmoid activation was used to calculate the probabilities of the 96 target classes. Finally, we use an Adaptive Moment Estimation (ADAM) to minimize the cross entropy loss function.

There are a couple of disadvantages that this model suffered from -

1. CNN does not capture long range dependencies. Patent documents often have long sentences with the subjects far removed from specific verbs / actions / adjectives.

2. We also noticed the classes were unbalanced and might have potentially impacted the model's results. This was evident from the fact that the model always predicted the category that appeared the most.

3.3.2 CNN + BiLSTM

Our goal with the CNN - BILSTM architecture is to negate some of the disadvantages we have seen with our baseline CNN model. LSTM does well with long range dependencies. BI-LSTM does this in both the directions.

Local Feature extraction using CNN

As with our baseline model, we employ CNN to extract local n-gram features. In this case, we read three words at a time. The various sections of the patent document has specific purposes and therefore we decided to treat the four sections in consideration (title, claim, abstract and description) independently using four different CNN channels. Unlike our baseline model, we do not employ max pooling as we wanted to retain as much information as possible for the BI-LSTM layer. We simply concatenate the output of the four CNN channels. So, our input vector to BI-LSTM should have the key local features from the salient sections of the patent document.

Classification using Bi-LSTM
For the classification problem, We intend to use LSTM as it can handle sequence data and capture long-term dependencies. LSTM also overcomes the issue of vanishing / exploding gradients. We specifically will use the Bi-LSTM that can process the context in both the directions [left to right and right to left]. We employed 128 cells for both the forward and the backward layers. The output of the BI-LSTM is passed to a fully connected layer with a sigmoid activation function. The sigmoid turns gives independent probabilities to the 96 class labels that we try to predict.

Model Architecture

1. After all the preprocessing of the patent data, we get the input texts from the four sections - (title, claim, abstract and description) of the document.

2. For each of the patent document, we then used the TF-IDF ranking to get the best 150 words from each of the sections.
3. We then used the embedding matrix that we trained to get the word embeddings that correspond to the input data. So for each section, we had a matrix of 150 x 100, considering the embedding dimension of 100.
4. Now, each of these sections were passed through a convolutional layer with 128 filters, with each filter having a kernel size of 3. These CNN layers used the RELU activation function and thus we only carry forward the most critical information with any negative value clamped to zero. The output of the CNN layer are four feature maps each of size - 148 x 128.
5. We then concatenate the output of the four feature maps and feed it to a BI-LSTM. This results in a matrix of size 592 x 128.
6. The BI-LSTM has 128 cells for both the forward and the backward layers. The output of the BI-LSTM is a 1 x 256 vector.
7. We finally employ a fully connected layer with a sigmoid activation function to get the 96 class probabilities.

3.3.3 Capsule Network (CapsNet)

Capsule networks (CapsNet) have gained relevance in recent time, and are said to have overcome some of the drawbacks of CNN's. Since our baseline, as well as the second model, are based of off CNN, we thought it would be good to put CapsNet to the test.

CNN's have a pooling layer which aggregates the output from a group of neurons to a single neuron to the higher layer. In our case, it is a max pooling layer which selects the most active feature. CNN's then hierarchically build such pattern extraction pipelines at multiple levels. Since the most active feature from a group of layers is promoted, some of the information is lost. Essentially, the higher layers have no say in which of the features from lower level is chosen. Capsule network addresses this issue by introduce an iterative dynamic routing process to decide the credit attribution between nodes from lower and higher layers.

Model Architecture:

a) After all the preprocessing of the patent data, we concatenate the input texts from the four sections - (title, claim, abstract and description) of the document. We refer to this as "full_text".
b) We use TF-IDF to get the top 600 words from each patent document. Documents shorter than 600 words are padded.
c) We then used the embedding matrix that we trained to get the word embeddings that correspond to the input data. So, we arrive at a matrix of 600 x 100, still considering the embedding dimension of 100.

d) We have a Dropout layer to prevent overfitting.
e) We now introduce a 1D CNN layer with kernel_size 4 , number of filters as 100 and a RELU activation.
f) We do not have a max pooling layer, like our other two models.
g) We placed a bidirectional GRU layer here with 128 units, instead of a bidirectional LSTM like our second model. We read that for smaller datasets, such as ours, GRU is more efficient. There is scope here for comparing GRU against LSTM.
h) Next, we place our Capsule layer. The main differences [9] compared to CNN are:

		Capsule	**CNN**
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\hat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\\|\mathbf{s}_j\\|^2}{1+\\|\mathbf{s}_j\\|^2}\frac{\mathbf{s}_j}{\\|\mathbf{s}_j\\|}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)

i) We finally employ a sigmoid activation function to get the 96 class probabilities.

Capsule layer details:

1. The Weighting/Sum is referred to as dynamic routing. This is done for r iterations, which is a hyper-parameter.
2. The non-linear activation is a squashing function.
3. If the prediction of a lower layer is in agreement with the activation vector of a higher layer, their dot product will be higher. As a result, the weight of such lower layers are increased in the output vector. This continues for r iterations.

We have referred to [10] when designing the Capsule Network architecture.

3.4 Evaluation Metrics

For the evaluation of our multi-label classification models we used the recall, precision and F1 scores as metrics. Precision shows the ratio of the predicted labels that are true labels (i.e. the ratio of how much of the predicted is correct). Recall shows the ratio of the true labels that were predicted correctly (i.e. the ratio of how many of the actual labels were predicted).

Specifically, we evaluate the models based on these metrics when predicting the top 1, top 2, and top 10 IPC labels for each document (considering that the average number of labels per document in our dataset is 1.4). For example, to obtain the top 2 predicted labels for a document, we took as predicted labels the 2 labels with the highest probabilities coming out of the output layer. These predicted labels were then compared to the actual ones to obtain the precision and recall at the individual document level (see equations below):

$$Precision = \frac{TP}{TP + FP} = \frac{trueLabels \cap predictionLabels}{predictionLabels}$$

$$Recall = \frac{TP}{TP + FN} = \frac{trueLabels \cap predictionLabels}{trueLabels}$$

Where:
True Positives (TP) is the number of labels predicted by our approach (prediction labels) that matched the IPC labels (true labels), without taking the exact order into account.
False Positives (FP) is the labels predicted by our approach (prediction labels) that do not match the true IPC labels.
False Negatives (FN) is the labels that should have been predicted by our approach, but were not.
True Negatives (TN) is the labels that, correctly, were not predicted by our approach.

After calculating the above metrics for each patent document, we calculate the final Precision, Recall and F1-score across all documents as follows:

$$Precision_{total} = \frac{1}{TotalSamples} \sum_{n=i}^{TotalSamples} Precision_i$$

$$Recall_{total} = \frac{1}{TotalSamples} \sum_{n=i}^{TotalSamples} Recall_i$$

$$F1_{total} = 2 * \frac{Precision_{total} * Recall_{total}}{Precision_{total} + Recall_{total}}$$

4 Results and discussion

We ran a couple of different baseline algorithms to test their performance. As this is a multi-class multi-label prediction, we calculated precision, recall and f1 score for the models for a top 1, top 2 and top 10 labels.

Here are our results:

		Top 1 Label			Top 2 Labels			Top 10 Labels		
	Epochs	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Per the reference paper)	40	81.55%	55.02%	63.06%	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
CNN (Baseline)	5	17.00%	13.20%	14.20%	11.30%	17.20%	13.10%	6.40%	48.50%	11.10%
CNN + Bi-LSTM	20	71.00%	59.20%	64.90%	47.70%	73.70%	57.90%	12.80%	93.00%	22.50%
Capsule Network	20	79.20%	65.50%	71.70%	53.10%	82.00%	64.50%	13.40%	97.30%	23.60%

Figure 4. Results Table for CNN, CCN+Bi-LSTM, and CapsNet Models

We can see that the Capsule Network model performed the best among our 3 models. For top-1 label classification, it exceeded the F1-score of our reference by around 8% even though we used less data and trained for fewer epochs .

The second best model is the CNN+BiLSTM with a top-1 label F1-score of 64.90%. It would be interesting to see the results by merely training all the models for the same number of epochs (40) as our reference model.

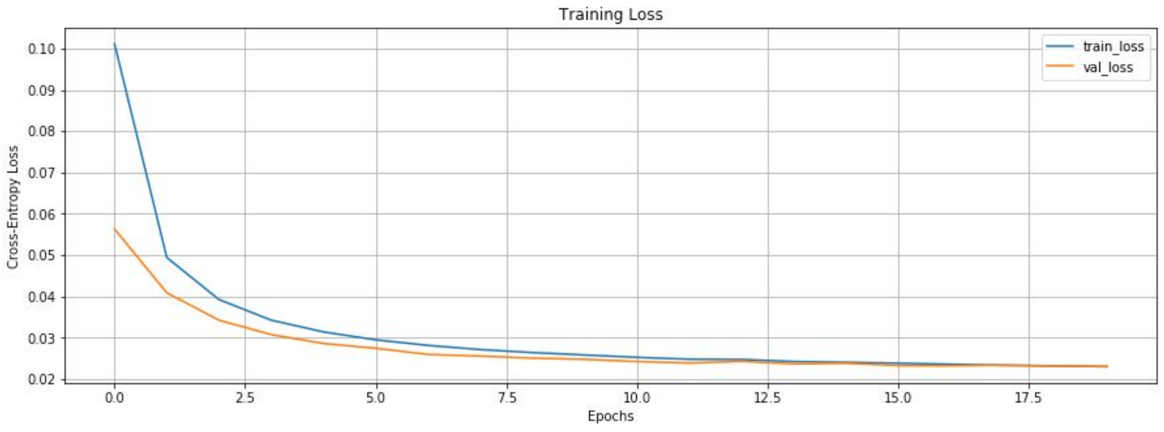


Figure 5. Train vs Valuation Loss for Capsule Network

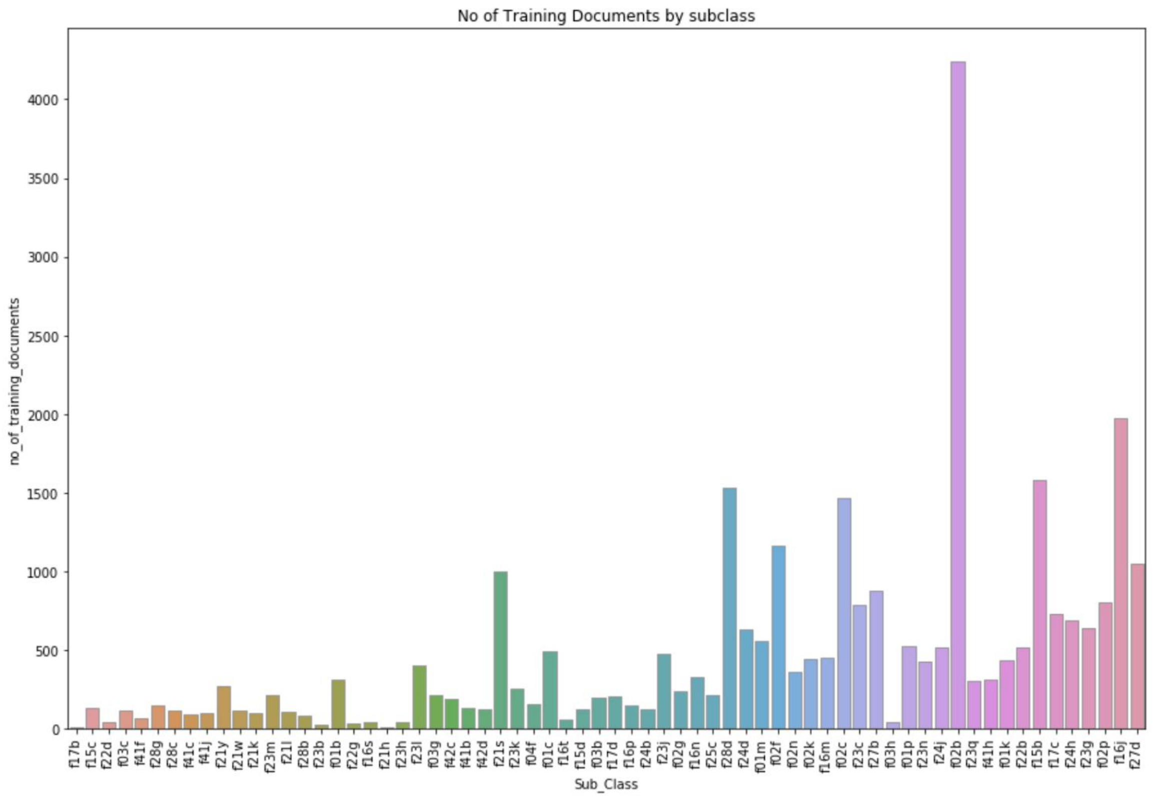
Looking at figure 4, we see that the loss for training and validation seem to decrease as the epochs increase. The lines converge around epoch 20. We stop here to ensure that the model is not overfitting.

4.1 Error Analysis

In our error analysis for BI-LSTM model, we observed the following:

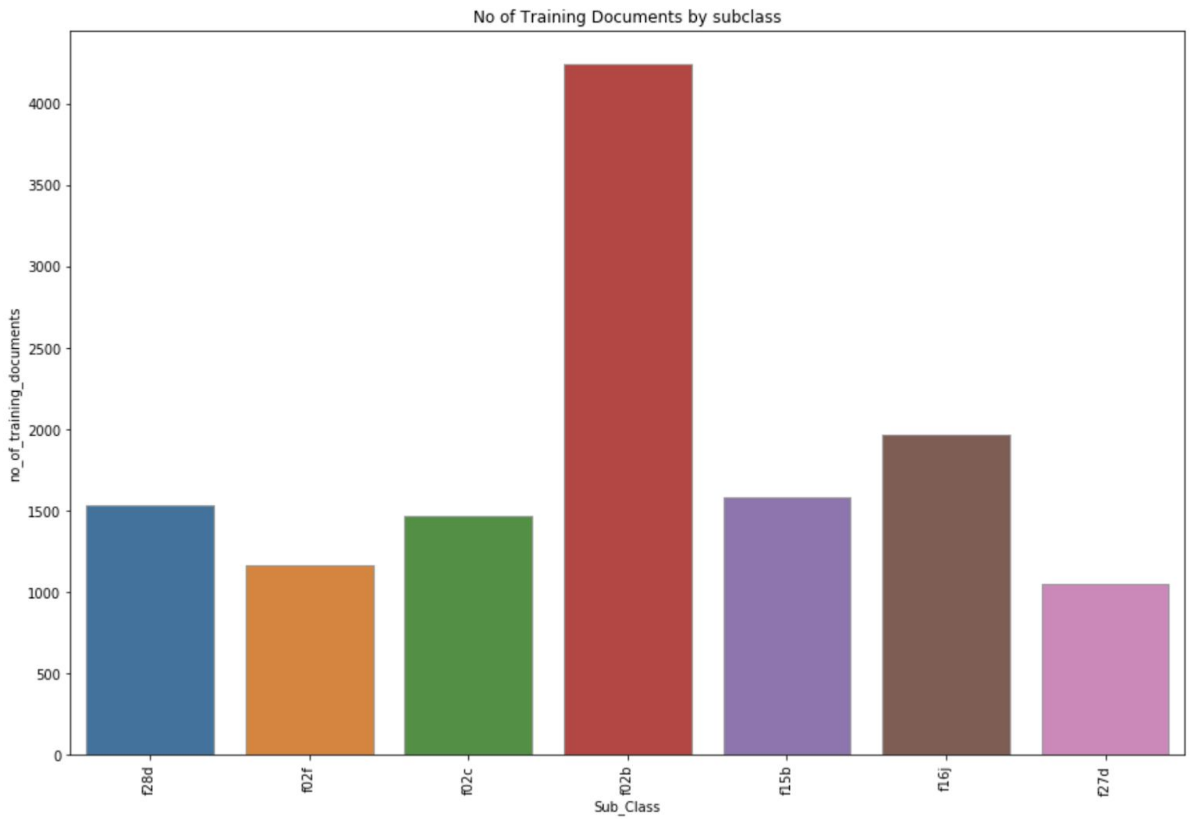
- There are 21 labels that were never classified correctly by the model.
- There were 66 labels that were classified at the error rate of more than 50%

The vast majority of these classes suffered from class imbalance.



The figure above shows that has most of these classes has fewer than 500 documents. Of course there are a few exceptions such as ‘f02b’, ‘f16j’, etc for which there was a reasonable number of training data.

Let’s zoom in on these exceptions. The figure below is a list of classes that has more than 1000 documents and error rate of more than 50%.



Let's take the example of 'f02b', which has decent training data and hone in on a specific example.

Patent document -> https://patents.google.com/patent/US6572956/en

Patent title : *Weatherable multilayer resinous articles and method for their preparation*

This patent is classified across multiple subclasses. These subclasses are also across multiple categories - B ["PERFORMING OPERATIONS; TRANSPORTING"] and F [MECHANICAL].

Since our scope was to classify only F [MECHANICAL], we dropped subclasses from other categories.
In our CLEF dataset, this patent was assigned to the subclass *F02B* but the predicted class was *F16L*. The subclass *F02B* pertains to internal combustion engine and related ignition methods. The predicted subclass *F16L* pertains to pipes, joints, fitting, protective coating, etc. **It turns out that this patent is grossly misclassified in the CLEF dataset (2011) even at the class level as this has nothing to do with internal combustion engine**. But this patent does deal with a substance for coating and thus got classified as *F16L*. **In this case, our model's prediction was much more in alignment with the actual class of B.**

There were some of the other factors that also contributed to error:

1. **Complex jargon**
    - Patent documents are often complex and technical details and language are sometimes intentionally vague in order to minimize the chance of a rejection. This means that lot of times, the patents could be assigned an incorrect sub-class.
2. **Inaccurate labelling**
    - Also patents are sometimes classified across multiple subclasses but they are truly only applicable for one of the classes. Consider the following record:

```
“combustion engine valve operating internal relief pressure valve air
chamber common passage chambers intakeside engine control exhaustside
embodiment volume revolution cylinders detecting cylinder accumulator
member defining intake cam supply connected valves invention fig hole
rotatbly leading operating internal combustion number value portion
shows head passages connect thereof present check sealability
intermediate detected casing set transitional regulator secured
responsiveness annular pistons openings obliqued provided piston
exhaust shaft oil plurality received figs unit operational bolt
portions controlled tenth projection ninth structure slidably members
end ceiling eighth commonly operation sleeves state followability
multistages equal connecting means changing seventh airsupply
cylindrical partition sixth respect bearing block operatively
releasing poor precisely described swiftly step complicated fifth
controlling avoided embodiments supplied hinders object joint
interposed lifter slide previous pressurized type utmost release
wasteful released stage bottomed referring deviated fourcylinder
fourth ringlike defines leaked largediameter opening return closing
saved multicylinder opened variations mounted port subtracting
restrain difference inputted seal achieve furthermore systems”
```

- This patent really deals with valves,air chambers, pistons and not related to the overall workings of an internal combustion engine.Our model predicts the class to be 'f01l' ['CYCLICALLY OPERATING VALVES FOR MACHINES OR ENGINES'] correctly. But it is also classified as 'f02b'.But as mentioned, 'f02b' has to do with internal combustion engines from a fuel injection standpoint.

3. Incorrect language labels

- Some of the documents were in German or French, but tagged as English language in the XML document.

5 Possible enhancements for a business use case

5.1 Enhance word embeddings

We would like to improve the richness of the input features by having a word embedding that best represents the semantic and contextual relationships from a technology perspective.

5.2 Balance the class

We would like to increase the input data to a 100k patent documents so that we have a more balanced class representation, or use other machine learning techniques (such as bootstrapping) to deal with the unbalanced classification problem.

5.3 Supporting other languages

Currently we are considering sections of English language only. Expanding the scope to support other languages would vastly expand the use cases of our experiment.

5.4 Detect language

We are currently going by the language tag in the XML document to determine the language. However, many documents are incorrectly tagged as EN - English language.   We may choose to detect the language ourselves for a more accurate classification.

6 Conclusion

With some more tuning, we feel our experiment can help make a difference for patent examiners, as well as applicants. If used in conjunction with patent retrieval, it will significantly reduce the time taken to determine whether prior art exists for an application.

The Capsule Network model showed promising results. Per our knowledge, this is the first application of Capsule Network to patent classification. With more research into the workings, we are sure Capsule Network will be the model to beat for classification problems.

7 References

1. Verberne, S.; D'hondt, E. Patent Classification Experiments with the Linguistic Classification System LCS in CLEF-IP 2011. In Proceedings of the CLEF 2011 Notebook Papers/Labs/Workshop, Amsterdam, The Netherlands, 19–22 September 2011.

2. Li, Z.; Tate, D.; Lane, C.; Adams, C. A framework for automatic TRIZ level of invention estimation of patents using natural language processing, knowledge-transfer and patent citation metrics. *Comput. Aided Des.* 2012,*44*, 987–1010.
3. J. Hu, S.Li, J. Hu, G. Yang, A Hierarchical Feature Extraction Model for Multi-Label Mechanical Patent Classification, 2018

4. Xiang Zhang et al Character-level convolutional networks for text classification.
5. Zeng, Y.; Yang, H.; Feng, Y. A convolutional BiLSTM neural network model for Chinese event extraction. In Proceedings of the International Conference on Computer Processing of Oriental Languages, Kunming, China, 2–6 December 2016; Springer: Berlin, Germany, 2016.

6. Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao - Recurrent Convolutional Neural Networks for Text Classification
7. https://stackoverflow.com/questions/48485870/multi-label-classification-with-class-weights-in-keras
8. https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
9. https://medium.com/ai³-theory-practice-business/understanding-hintons-capsule-networks-part-ii-how-capsules-work-153b6ade9f66
10. https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-capsule-networks