CS230

Patent Technology Classification and Citation Level Projection

Hanning Su*
Melbourne Business School
University of Melbourne
hanning@stanford.edu

Abstract

For various parties in the society, it is essential to know the potential impact of new technologies. This project shows that deep learning on the patent title + abstraction text can help them to get informed. With a deep LSTM RNN model trained using the large scale datasets I assembled and the word vectors generated from the skip-gram model, I achieve an accuracy of 71.4% in the technology classification, and 99% precision and 83% recall scores in identifying top1% cited patents in the test set

1 Introduction

A patent is a document through which an inventor claims the inventorship of his or her creation to the wider world. One (another inventor, business competitor, or some other stakeholder) could naturally ask two questions: 1. What kind of technology the inventor is presenting? 2. within its technology field, is the new work going to have a significant impact? My project aims to give a deep learning solution to these two important questions.

For the first task, I implemented a two layers LSTM RNN model. The model takes a matrix representation (word vectors stacked vertically) of the title and abstraction texts of a given patent as input and gives a output through a softmax layer predicting the technology class (i.e. subsection id) that the patent belongs to. In total, there are 125 distinct technology classes under the CPC (Cooperative Patent Classification) scheme, which the United States Patent and Trademark Office is currently using.

For the second task, I used the same two layers LSTM RNN model architecture taking matrix representation of patent texts as input. The output is a prediction of the group label based on the number of citations the patent will receive in five years after issuance. The group labels are based on the quantiles of the distribution of citation count distribution in each technology class. There are in total 4 groups including top 1%, top 10% but short of top 1%, not in the top 10% but cited at least once, and never cited at all. One can think of labels as short(mid)-term citation levels.

2 Related work

In recent years, a growing body of literature is utilizing machine/deep learning techniques to classify patent. Li et al (2018) built an classifier based on a CNN architecture to automatically extract features from patent text. Shalaby et al (2018) developed an LSTM approach based on fixed hierarchy vectors.

*Alternative E-mail: suhanning1997@gmail.com

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Hu et al (2018) proposed an patent keyword extraction algorithm based on the distributed skip-gram model for patent classification. My project adds to this strand of works by developing RNN classifier taking skip-gram word vectors as inputs.
In this project, the citation refers to the forward citation i.e. how many other patents cite the focal patent as prior art. I take the straightforward interpretation that citation measures the its impact in its technological field (Aristodemou et al, 2018). Past researches have noted that forward citation can measure patent quality, economic value (Squicciarini et al, 2013), and knowledge spillover (Sharma and Tripathi, 2017).
For the citation level projection task, the prior works are relatively scarce. Lee et al (2017) used feed-forward neural network to identify emerging technologies. Lin et al (2018) introduced a CNN based approach that integrated patent text materials to evaluate patent quality. My rough search indicates that I could be the first to use LSTM RNN architecture to predict the patent citation level.

3 Dataset and Features

For this project, citation level labels are created as follows.
1. From the patent dataset from PatentView, I extracted all the pairs of id of citing patent and id of cited patent.
2. I attached the grant date to each of the citing patents and of the cited patents.
3. For each cited patent, I create a new variable (maxdate) by adding 5 years to its grant date. Eliminate all its citing patents with grant date later than the maxdate. Then, I count all the remaining citing patents to get a raw within 5 year citation count.
4. I grouped the cited patent by the 3 digits CPC subsection code (technology class) that is listed first. (CPC orders subsection codes in decreasing order of importance, here I only take the subsection code listed first into account.) Then I calculated 99 and 90 percentiles of the citation count distribution within each of the technology class.
5. Finally, those patents with raw citation count greater that the 99 percentile get labeled as **top1%**. Each of those with count short of top1% but greater than 90%, gets a **top10%** label. Each of those with count of at least one but short of 90% percentile gets a **btm90** (bottom 90) label. Each of those without within 5 year citation has the **zerocite** label.
6. Remove examples without full 5 year window to receive citation.
The labelling of citation level has no fixed rule. Different labeling strategies encapsulate different information about a given patent. The strategy I adopted mostly capture a patent's short to midterm impact within the field it contributes to. It is necessary to mention that my labelling strategy may not be a good indicator of a patent's long term impact, since a patent can still receive citation after 50 years of its issuance date (Hall et al 2001).
The rest of data processing involve joining the dataset containing CPC subsection ids of patents to the table containing the title + abstraction text.

Summary of resulting dataset

Citation level dataset: 5159188 examples. (Prepared for the citation level projection task)
Technology class dataset: 6779660 examples. (Prepared for the technology class classification task)
title + abstraction text dataset: 7131735 examples. (Prepared for Skip-gram embedding model training)

Final processing before training

In deriving the word vectors, I used the entire 7131725 training examples to train the skip-gram model, ending up with 92652 distinct word vectors.

To speed up the training, citation level dataset is grouped by technology class and 15% of examples in each group are randomly sampled, and then the resulting subset is further divided randomly into train (98%) and test set (2%). I also used the remaining 85% of the citation level dataset for further test. Therefore, in the end, I have a training set of 996609 examples, a small test set of 20339, and a large test set of 5762712 examples to further verify the generalizability of the trained model.

Since citation level projection task is essentially a task of learning from the past and generalizing into the future, in order to check how far my model generalize, I divide the original dataset into patents issued in 2010 and before and those issued after 2010. I used pre-2010 set for training and divide the post-2010 set based on issuance year so that the performance of the model for each year after 2010 can be evaluated.

Due to time and computational power limitations of the project, hyperparameters are mostly at default setting, so I did not create any development set.

The implementation of the procedure using Keras generator to feed millions of examples into models involve some further data processing. A generator is an object that feed the large dataset in small batches into deep learning models to avoid depletion of RAM. In my project, I used RNN models, which require feeding texts in matrix form. Therefore, I define my generator such that it converts the texts from pandas dataframe read from a csv file into matrices. For this conversion process, I implemented some further data processing: 1. the non-English and non-number characters and infrequent words are removed. 2. The remaining texts are tokenized, their corresponding word vectors are extracted from trained skip-gram model, and the vectors are stacked vertically. (For technology classification, I let the matrix to have maximum 100 rows. Examples with less than 1 tokenized words are removed. For citation level projection task, I let the matrix to have maximum 200 rows. Examples with less than 50 tokenized words are removed.) I zero padded (truncated) the matrix in cases where the number of tokens are less(more) than 100 or 200 respectively. 3. 100 such matrices are bundled into a minibatch and fed into and RNN models by the generator. 4. At the end of each training epoch, the indexes for generating minibatches are shuffled to ensure robustness of the model.

4 Methods

First I train a skip-gram model with negative sampling to get a mapping of all the words in all the existing patent title + abstraction texts to vectors. Negative sampling refers to the generation of a new dataset. For each given context word $\mathbf{c}$, we pair it up with each of the target words $\mathbf{t}$'s within a window of certain size. Each of the resulting pairs have label 1. Then, we pair up the context with multiple randomly selected words from text corpus. The resulting pairs are labeled as 0. We then define a logistic regression with the task of predicting whether the pair is a context-target pair (label 1) or not (label 0. In mathematical notation, for given any pair of word c and t, the response is a random indicator $Y := \mathbb{I}(\text{c and t form a context-target pair})$, the model is :

$$\mathbb{E}(Y) = \mathbb{P}(Y = 1|c, t) = \sigma(\theta_{\mathbf{t}}^{\mathbf{T}}\mathbf{e_c})$$

We train the model and keep the $\mathbf{e_c}$ as word vectors.
I initially tried out CNN and FCN (A form of CNN without fully connected layers). The idea behind both of those architectures is to treat the input matrix representation of text as a one channel image and to perform convolution on the "image", and perhaps the learned parameters in the convolution layers can capture increasingly complex features just like they do in image classification tasks. However, it turns out that both of the architectures have convergence issues, the cost does not drop after tens of epochs, so I will not elaborate on them more in this and next section.
I end up implementing a deep RNN using LSTM units. A deep RNN is stacking up multiple many to many single layer RNN, each of which taking as input the output sequence of the layer below, and then stack on top a many to one single layer RNN outputing to a fully connected layer and then to a softmax layer giving the final prediction. Such a structure allows inputs with various lengths. The units in the network LSTM (long short term memroy), which capable of learning long-term dependencies in the input texts, which may be helpful for my tasks. The performance evaluation will be based on the confusion matrix $\mathbf{M}$. The row indexes i's of $\mathbf{M}$ represent the ground true labels and the column indexes j's the predicted labels. We use the following definitions of precision and recall:

$$\text{Precision}_i = \frac{\mathbf{M_{ii}}}{\sum_j \mathbf{M_{ji}}} \qquad \text{Recall}_i = \frac{\mathbf{M_{ii}}}{\sum_j \mathbf{M_{ij}}}$$

Note that in figure 1 2 3 4, I plot confusion matrices normalized row-wise and column-wise. The diagonal elements of them are exactly recall and precision scores respectively. Finally, the accuracy metric is simply the fraction of all the test examples for which our model gives correct prediction.

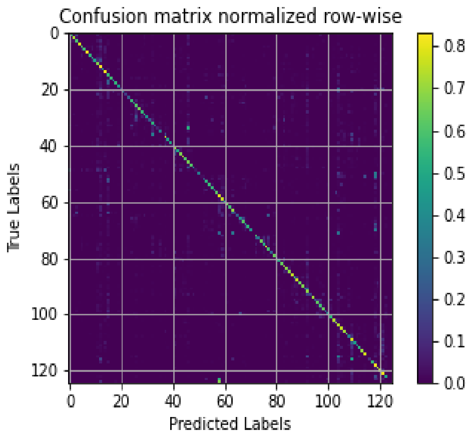

Figure 1

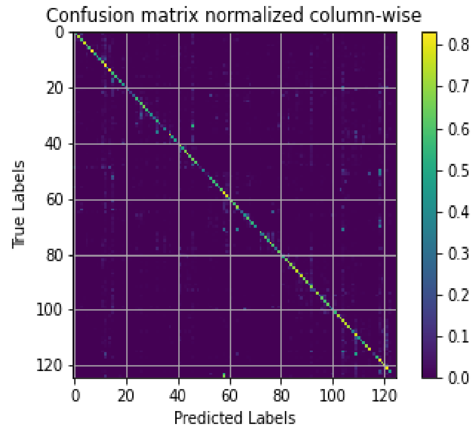

Figure 2

5 Experiments/Results/Discussion

In this project, I first trained the skip-gram model using the hyperparameters setting of Li et al (2018). A simple evaluation based on cosine similarity shows that similar words are close in the vector space. (See Apendix A for for examples)
For the technology classification task, the RNN model achieves an accuracy of 71.4% on the small test set. Then the model is tested on the large test set of over five million examples, it gets an accuracy of 71.37%, demonstrating remarkable generalizability of our model.
The figure 1 and figure 2 are the confusion matrices generated from the test result on the large test set. We can see that most bright spots are on the diagonal, demonstrating for most of 125 tech classes, the model has precision and recall within the range of 0.5 to 0.8. For most of classes, our model can figure most of the true positives and does not confuse between classes.
Our model perform poorly on underrepresented tech classes, often producing zero positive predictions, especially those with less than 1000 training examples. (Please see the appendix B for precision recall scores for all 125 classes).

For the citation level projection task, the RNN model trained on the pre-2010 examples get tested on 5 separate test sets containing patent examples with issuance year of 2011, 2012, 2013, 2014 and 2015 respectively. The resulting accuracy scores are 59%, 57%, 55%, 53%, and 51% respectively, demonstrating a steady decline of model generalizability as it tries to generalize further into the future.
One of the most import goals of projecting citation level is to find out the patents that are seminal in their respective fields. Therefore, we examine whether our model can pinpoint them. If we look at diagonals of the confusion matrices in figure 3 and figure 4. Throughout the five testing years, the only class for which the model maintain both high level of precision and recall is the top1% cited class. In fact, the performance is extremely steady, showing no sign of decline over the five testing years. We can thus be confident that the for at least five years into the future, the model can correctly declare over 80% of all the actual top1% (recall score over 80%) cited patents, and within those that it identify as top1%, about 99% (precision score over 99%) of them are true positives.
Curiously, the model performs very poorly for the top10% but not top 1% class. One explanation is that the patents with top impact factor and the rest of patents have substantial difference in title + abstraction text, while moderately impactful patents do not differentiate themselves substantially from those receiving a few to no citations.
Finally, the model can figure most of the bottom 90% cited patents out, but it does not distinguish them well enough from zero cited patents and top10%.

6 Conclusion/Future Work

The goal of this project is to predict a patent's technological field and its impact within its field. For the former task, I achieved 71% accuracy and precision/recall score falling between 50% and 80% for

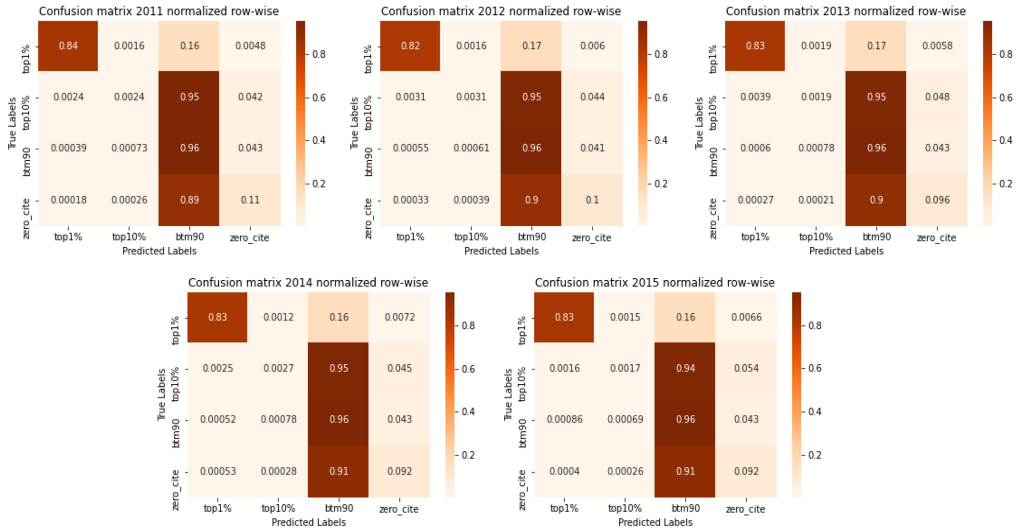

Figure 3

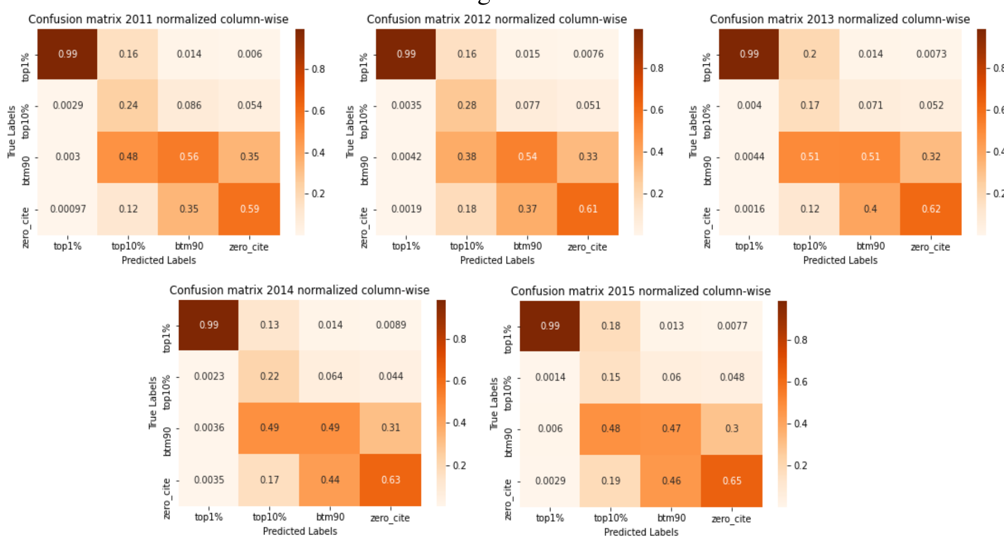

Figure 4

most classes. For patents with sufficient amount length of title + abstraction text (at least 50 tokens), the model for the second task achieves over 50% accuracy at least 5 years going into the future and is especially capable of discover the most impactful patents (those top 1% most cited).
For the future work, the existing models have a vast room for hyperparameter tuning so that I can choose a optimal set of hyperparameters so that the model converge better to the minimum cost.
In addition, given that the questions I want to address are closely related, it is reasonable to construct a multitasking model performing the two classification tasks simultaneously in the future. The balancing of weights of the two tasks can be a very interesting question to explore.
The patent citation data has truncation issue, for example, a patent in 2018 does not have a five year citation count because it is less than 5 years from now (March 2021). Therefore, currently, to estimate impact of patent granted in 2021, our model can only learn from data up to 2016 (at most), which to a certain extent must have negatively impact the quality of estimation. Devising a way to incorporate more recent data into the model training is important for the model to be useful in the real world.
Finally, the patent document include far more information (such as claims, descriptions and graphs) than the title + abstraction task. I would also like to explore methods to incorporate them into patent classification tasks to get even better performance.

7 Contributions

This is project is entirely my own work.

References

Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).

Aristodemou, Leonidas, and Frank Tietze. "Citations as a measure of technological impact: A review of forward citation-based measures." World Patent Information 53 (2018): 39-44.

Hall, Bronwyn H., Adam B. Jaffe, and Manuel Trajtenberg. The NBER patent citation data file: Lessons, insights and methodological tools. No. w8498. National Bureau of Economic Research, 2001.

Hollet et al. Keras. GitHub. Retrieved from https://github.com/fchollet/keras (2015)

Hu, Jie, et al. "Patent keyword extraction algorithm based on distributed representation for patent classification." Entropy 20.2 (2018): 104.

Lee, Changyong, et al. "Early identification of emerging technologies: A machine learning approach using multiple patent indicators." Technological Forecasting and Social Change 127 (2018): 291-303.

Li, Shaobo, et al. "DeepPatent: patent classification with convolutional neural networks and word embedding." Scientometrics 117.2 (2018): 721-744.

Lin, Hongjie, et al. "Patent quality valuation with deep learning models." International Conference on Database Systems for Advanced Applications. Springer, Cham, 2018.

Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.

Řehůřek, Radim, and Petr Sojka. "Gensim—statistical semantics in python." Retrieved from genism. org (2011).

Squicciarini, Mariagrazia, Hélène Dernis, and Chiara Criscuolo. "Measuring patent quality: Indicators of technological and economic value." (2013).

Shalaby, Marawan, et al. "An lstm approach to patent classification based on fixed hierarchy vectors." Proceedings of the 2018 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2018.

Sharma, P., and R. C. Tripathi. "Patent citation: A technique for measuring the knowledge flow of information and innovation." World Patent Information 51 (2017): 31-42.

Appendix A

Demonstrating the similar words have vectors with small Euclidiean distance.

```
similar words to hand
 [('hands', 0.666816771030426), ('grip', 0.6162362694740295), ('handheld', 0.5833435654640198), ('hel
d', 0.5777618288993835), ('thumb', 0.5683989524841309), ('finger', 0.5674130916595459), ('fingertip',
0.5662369728088379), ('wrist', 0.5559874176979065), ('glove', 0.5510510802268982), ('handle', 0.54713
58299255371)]
similar words to clean
 [('cleaned', 0.7214415073394775), ('cleaning', 0.6857302784919739), ('dirty', 0.6056607365608215),
('cleans', 0.5931688547134399), ('cleanse', 0.5638104677200317), ('cleaner', 0.5545836687088013), ('d
isinfect', 0.5516268014907837), ('scrubbed', 0.5446197390556335), ('soiled', 0.538492739200592), ('sa
nitize', 0.5377720594406128)]
similar words to space
 [('spaces', 0.7825511693954468), ('partition', 0.5470077991485596), ('cavity', 0.5142579674720764),
('void', 0.5132923722267151), ('accommodation', 0.5112019181251526), ('workspace', 0.502509832382202
1), ('inside', 0.498166561126709), ('partitions', 0.49671226739883423), ('gap', 0.4830506443977356),
('slot', 0.4824589490890503)]
similar words to grass
 [('turf', 0.757792055606842), ('lawn', 0.7408095598220825), ('mulch', 0.7095460891723633), ('clippin
gs', 0.7085317373275757), ('vegetation', 0.7040784358978271), ('mowing', 0.6913439035415649), ('mulch
ing', 0.6605969667434692), ('mower', 0.6520372629165649), ('weed', 0.6388605833053589), ('lawns', 0.6
275840401649475)]
similar words to ocean
 [('sea', 0.8188051581382751), ('seabed', 0.7292262315750122), ('seafloor', 0.7203499674797058), ('of
fshore', 0.6848180294036865), ('river', 0.6737468838691711), ('oceanic', 0.6656750440597534), ('moore
d', 0.6502468585968018), ('ship', 0.6439331769943237), ('buoy', 0.642741858959198), ('coastal', 0.634
8066329956055)]
similar words to brain
 [('cerebral', 0.7809474468231201), ('myocardial', 0.7458490133285522), ('neurological', 0.7089080214
500427), ('nerve', 0.7044943571090698), ('myocardium', 0.6921192407608032), ('ischemic', 0.6906895041
465759), ('nerves', 0.6824162006378174), ('ischemia', 0.6759458780288696), ('cardiac', 0.661411762237
5488), ('neuronal', 0.6598320007324219)]
similar words to eye
 [('eyes', 0.8023295998573303), ('ocular', 0.7203682661056519), ('eyeball', 0.7054115533828735), ('re
tina', 0.6774422526359558), ('cornea', 0.6543327569961548), ('sclera', 0.6167570352554321), ('fundu
s', 0.6149924397468567), ('eyelid', 0.6007823944091797), ('corneal', 0.5859100818634033), ('anterio
r', 0.5739483833312988)]
similar words to vacuum
 [('suction', 0.705642819404602), ('subatmospheric', 0.6161016225814819), ('evacuated', 0.61263561248
7793), ('evacuating', 0.6018824577331543), ('vaccum', 0.5907180309295654), ('pressure', 0.57641953229
90417), ('chamber', 0.5627647638320923), ('cleaner', 0.5611953735351562), ('canister', 0.548358559608
4595), ('vacuuming', 0.5364481806755066)]
similar words to network
 [('networks', 0.8499081134796143), ('gateway', 0.7442525625228882), ('networking', 0.710473775863647
5), ('internet', 0.6981134414672852), ('peer', 0.685767650604248), ('router', 0.6834752559661865),
('lan', 0.6818010807037354), ('connectivity', 0.6749587059020996), ('communications', 0.6709034442901
611), ('infrastructure', 0.667117714881897)]
```

Appendix B

Classification report part I

	precision	recall	f1-score	support
0	0.81	0.73	0.77	81474
1	0.56	0.56	0.56	4023
2	0.58	0.73	0.65	4369
3	0.66	0.60	0.63	20884
4	0.78	0.77	0.78	6259
5	0.57	0.61	0.59	10284
6	0.81	0.57	0.67	3671
7	0.82	0.83	0.82	9283
8	0.70	0.54	0.61	6726
9	0.62	0.50	0.56	16397
10	0.69	0.48	0.57	4134
11	0.63	0.68	0.65	72757
12	0.82	0.83	0.83	411278
13	0.58	0.42	0.49	7839
14	0.79	0.82	0.80	71811
15	0.52	0.57	0.54	87163
16	0.70	0.57	0.63	5044
17	0.58	0.39	0.47	4579
18	0.70	0.55	0.62	2736
19	0.55	0.46	0.50	22369
20	0.49	0.15	0.22	1992
21	0.58	0.29	0.38	4582
22	0.35	0.14	0.20	5675
23	0.43	0.25	0.32	1780
24	0.59	0.56	0.58	19320
25	0.74	0.49	0.59	14145
26	0.60	0.61	0.60	49371
27	0.74	0.66	0.70	16204
28	0.62	0.56	0.59	28884
29	0.64	0.52	0.57	11177
30	0.50	0.43	0.46	6627
31	0.57	0.24	0.33	4371
32	0.61	0.54	0.58	48252
33	0.56	0.30	0.39	3905
34	0.61	0.07	0.13	2876
35	0.46	0.35	0.40	23804
36	0.00	0.00	0.00	512
37	0.75	0.79	0.77	65203
38	0.58	0.48	0.52	7668
39	0.70	0.49	0.58	4268
40	0.35	0.08	0.12	3918
41	0.68	0.70	0.69	153756
42	0.62	0.49	0.55	7466
43	0.62	0.67	0.64	42956
44	0.64	0.74	0.68	19918
45	0.67	0.61	0.64	23695
46	0.67	0.66	0.66	109829
47	0.69	0.61	0.65	14853
48	0.53	0.32	0.40	8051
49	0.91	0.04	0.08	720
50	0.60	0.35	0.44	4887
51	0.22	0.02	0.04	13344
52	0.51	0.31	0.38	17204
53	0.56	0.54	0.55	15573
54	0.72	0.57	0.64	15276
55	0.54	0.56	0.55	17615
56	0.73	0.40	0.52	2608
57	0.67	0.62	0.65	2375
58	0.79	0.80	0.79	226399
59	0.67	0.77	0.72	105183
60	0.61	0.54	0.58	54140
61	0.70	0.65	0.67	27168
62	0.70	0.58	0.63	15312
63	0.70	0.70	0.70	75620
64	0.33	0.15	0.21	869
65	0.81	0.25	0.38	570
66	0.44	0.43	0.44	7100
67	0.56	0.61	0.59	16063
68	0.47	0.49	0.48	25060
69	0.66	0.55	0.60	12096
70	0.77	0.43	0.55	6430

Classification report part II

71	0.00	0.00	0.00	677
72	0.57	0.61	0.59	9104
73	0.58	0.33	0.42	2557
74	0.86	0.65	0.74	3825
75	0.54	0.52	0.53	6848
76	0.86	0.81	0.83	5618
77	0.61	0.59	0.60	16022
78	0.69	0.27	0.39	832
79	0.64	0.64	0.64	13260
80	0.65	0.45	0.53	12164
81	0.56	0.49	0.52	16389
82	0.52	0.64	0.58	9331
83	0.58	0.71	0.64	37910
84	0.69	0.68	0.69	23645
85	0.59	0.69	0.63	11948
86	0.76	0.76	0.76	46438
87	0.66	0.57	0.61	44758
88	0.71	0.76	0.73	68241
89	0.62	0.65	0.63	9326
90	0.68	0.67	0.68	30764
91	0.41	0.35	0.38	7358
92	0.62	0.69	0.66	141583
93	0.53	0.17	0.26	3633
94	0.61	0.71	0.66	24711
95	0.47	0.30	0.37	2026
96	0.64	0.63	0.64	15522
97	0.51	0.59	0.54	20470
98	0.68	0.64	0.66	18655
99	0.53	0.33	0.41	3964
100	0.40	0.36	0.38	4545
101	0.56	0.55	0.55	12287
102	0.78	0.74	0.76	21381
103	0.61	0.71	0.65	9165
104	0.68	0.74	0.71	305209
105	0.69	0.76	0.72	133375
106	0.78	0.72	0.75	121631
107	0.78	0.61	0.69	7312
108	0.42	0.36	0.39	41269
109	0.76	0.78	0.77	550145
110	0.77	0.46	0.57	32810
111	0.45	0.44	0.44	26970
112	0.66	0.53	0.59	54261
113	0.74	0.70	0.72	33232
114	0.79	0.78	0.78	139254
115	0.00	0.00	0.00	165
116	0.49	0.11	0.18	4763
117	0.74	0.55	0.63	11382
118	0.80	0.82	0.81	543745
119	0.65	0.72	0.68	106375
120	0.64	0.74	0.69	102615
121	0.79	0.81	0.80	598050
122	0.59	0.48	0.53	75341
123	0.00	0.00	0.00	17
124	0.00	0.00	0.00	72
accuracy			0.71	5762700
macro avg	0.61	0.52	0.54	5762700
weighted avg	0.71	0.71	0.71	5762700