Sparse Multi-label Patent Classification with Deep Learning

Alexander Mueller and **Kevin Stone** {alexmue, kevinstone}@berkeley.edu
Division of Data Science and Information
University of California
Berkeley, CA 94720-1776, USA

Abstract

This paper describes a multi-label classification model capable of categorizing patents into Cooperative Patent Classification (CPC) codes. Classifications are done on the CPC subclass level and are based on patent abstract texts from the USPTO 2M dataset. Our models use GloVe and BERT word embeddings with CNN+Dense layer architecture. The best algorithm uses a weighted binary cross-entropy loss function and achieves an F1 score of 59.2 for the challenging multi-label classification task for 632 sparse CPC subclasses. This result surpasses recent benchmarks using word2vec embeddings, but stays behind BERT-based implementations. We show metric sensitivity to data sub-setting with GloVe and BERT and provide insights into multi-label classification problems from specific patent examples.

Keywords: Sparse multi-label classification, BERT, GloVe, CNN, Patent classification

1. Introduction

More than 3.3M patents were filed worldwide in 2018 (WIPO, 2019). Patents are a strategic resource for innovation management, but their amount and the complexity of their taxonomy are creating significant challenges for the patent system and its users.

Patent classification is a core task in patent management and is typically undertaken almost exclusively by patent experts and patent examiners (Shaobo et al., 2018). One way to expedite the categorization and search process is to narrow down which codes apply to a new patent filing and then search for patents by code. Understanding which codes apply to a potential patent filing also allows companies to make a more informed decision on whether or not an investment in the patent is warranted.

The Cooperative Patent Classification (CPC) system has been jointly developed by the European Patent Office (EPO) and the United States Patent and Trademark Office (USPTO) and launched in 2013. The CPC system has over 250,000 categories (EPO and USPTO, 2015). Patents are assigned one or more CPC codes. Each classification term consists of a symbol such as ”B60W 20/00” (which represents ”Control systems specially adapted for hybrid vehicles”). The first letter is the ”section symbol” consisting of a letter from ”A” (”Human Necessities”), followed by ”B” (”Performing operations; transporting”) up to ”H” (”Electricity”). This is followed by a two-digit number to give a ”class symbol” (”B60” represents ”Vehicles in general”). The final letter makes up the ”subclass” (B60W represents ”Conjoint control of vehicle sub-units”). The subclass is then followed by a number of digits to indicate groups and subgroups.

We are focused on classifying patents on the CPC subclass level, which includes more than 600 labels. We postulate that the efficiency of patent analysis can be greatly improved if we use NLP to speed up the patent classification process.

2. Related work

Many algorithms have been used to classify patent documents, among which ANN, SVMs, and kNN are the most commonly used (De Clerc et al., 2019; Verberne et al., 2010). Only very few studies have successfully applied deep neural networks to solve the complex patent classification task. Shaobo et al. (2018) have used word2vec skip-gram embeddings with CNN on the USPTO 2M dataset, resulting in a F1 score (top 5) of 40.0 for CPC subclasses. Very recently, BERT has been applied to CPC subclass labeling on the USPTO 2M dataset, resulting in an F1 score (top 5) of 46.8 (Lee and Hsiang, 2019).

3. Multi-label classification

In contrast to conventional single-label classification, multi-label classification (MLC) allows an instance to belong to several classes simultaneously. MLC methods are increasingly required by applications with enormous label spaces, such as image and video annotation for multimedia search or categorization of Wikipedia articles (Wydmuch et al., 2018).

3.1 Problem formulation

Let X denote an instance space and $\mathcal{S} = \{\lambda_1, \lambda_2, ..., \lambda_m\}$ be a finite set of class labels. We assume that an instance $x \in \mathcal{X}$ is associated with a subset of relevant labels $S \in 2^{\mathcal{S}}$. In the set S of relevant labels we identify a binary vector $y = (y_1, y_2, ..., y_m)$ in which $y_{i=1} \Leftrightarrow \lambda_i \in S$. By $\mathcal{Y} = \{0,1\}^m$ we denote the set of possible labelings. Observations are assumed to be generated independently and identically from a probability distribution $P(X,Y)$ on $\mathcal{X} \times \mathcal{Y}$ (Dembczynski et al., 2012).

The MLC problem is then to learn a classifier $h : \mathcal{X} \to \mathcal{R}^m$ that, given training data of a set of observations $(x, y) \in \mathcal{X} \times \mathcal{Y}$, for a given instance $x \in X$ returns a vector:

$$h(x) = (h_1(x), h_2(x), ..., h_m(x))$$

The optimization problem can then be formulated as minimizing the loss L on multi-label predictions:

$$h^*(x) = \arg\min_h E_X[E_{Y|X} L(Y, h(x))]$$

3.2 Evaluation metrics

Sparse MLC requires different metrics than those used in traditional single-label classification. Especially in sparse multi-labeling problems, accuracy is not a suitable metric as a high score can be achieved simply by always predicting the negative class. In the literature, various metrics have been proposed, among them Hamming Loss, precision/recall, and F1 score. In this paper, we will primarily evaluate precision, recall and F1 score (macro-averaged over all instances), trying to achieve a balance between precision and recall. Unlike in document retrieval systems where users would like relevant results to be on the first page (high precision), we assume that in our application, IP lawyers and patent authors also prioritize recall to ensure they do not overlook a particular area.

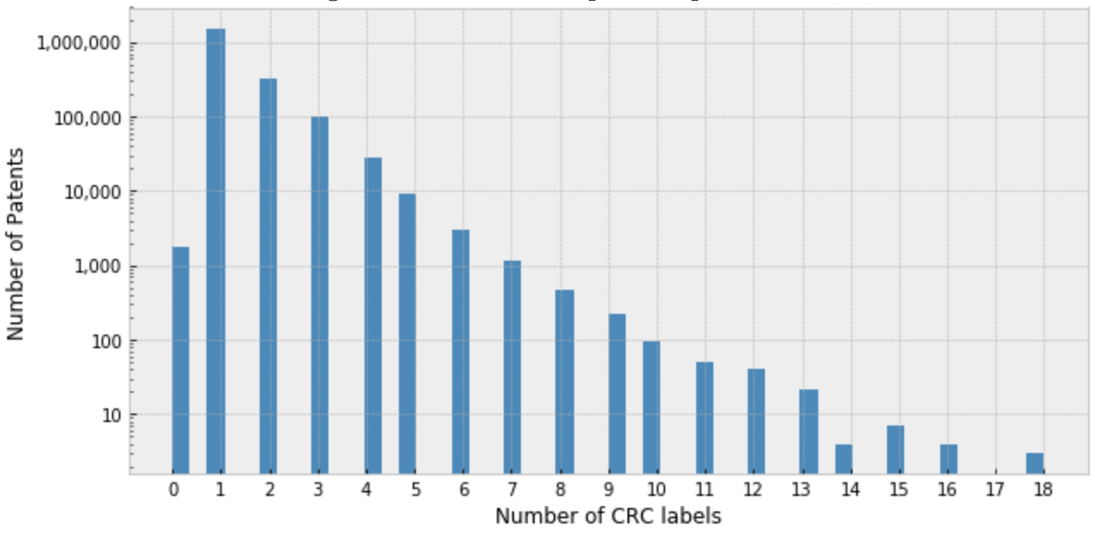
3.3 Dataset

To evaluate the performance of our classifier, we use the USPTO 2M benchmark dataset introduced by Shaobo et al. (2018). It contains 2M patents from 2006 to 2015, patent numbers, titles, abstract texts (218M tokens and a vocabulary of 237k) and their associated subclass labels. The longest abstract in the patent dataset has 509 words, and there are only eight words in the shortest abstract. On average there are 109 words in each abstract.

We notice that we face two large imbalances in the dataset:

- The set of labels is *extremely sparse* — only a tiny subset of the labels are positive. While there are 632 CPC subclass labels in total, each patent has on average only 1.3 CRC labels. The number of labels per patent decreases linearly on a log scale. Most patents have just one CRC label (see figure 1).

Figure 1: USPTO-2M patents per label



  The sparsity of the labels will likely make model training inefficient, as most labels are negatives. This could overwhelm training and therefore contribute no useful learning signal. When easily classified negatives comprise the majority of the cross entropy loss and dominate the gradient, the model might converge towards zero in early training periods. We will deal with this topic in section 4.1.

- The set of positive labels has an *imbalanced distribution* — the most common CPC class has more than 280k patents, and there is a long tail of CPC labels with fewer than 1000 patents. The imbalance within the set of active labels might cause the model to underfit, as there is an insufficient number of training signals available for seldomly used CPC labels. We therefore considered to take out labels with less than

1000 patents, which would reduce the label set from 630 to roughly 200. However, this would make the model incapable of inferring infrequent labels, which limits the practical applicability of the system. Therefore, we decided to keep the unbalanced label distribution.

4. Models

4.1 Objective function

We apply a multi-label binarizer to convert the list of CPC subclass labels to a multi-hot encoding $\{0, 1\}$ indicating the presence of class labels. Our objective function is *binary cross-entropy loss* (details see annex), which takes the independent sigmoid output for all predictions and sums up the loss over all binary classifications.

To address the aforementioned label sparsity problem, we propose to reshape the loss function and prevent it from learning only the negative class. We implemented the *weighted binary cross-entropy loss* function, where y is the label and $\hat{y}$ is the predicted probability for a given set of examples, and α is a multiplicative coefficient for the positive labels term in the loss expression.

$$H_{weighted}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \alpha * y_i * \log(\hat{y_i}) + (1 - y_i) * \log(1 - \hat{y_i})$$

When $\alpha > 1$, the weighted binary cross-entropy loss function increases the gradient for false-negative predictions and thereby pushes more predictions towards 1. This increases recall at the cost of precision, matching our purpose to more effectively identify sparse labels. A function implementation for the Keras library in Python and a plot of losses for different weight configurations are included in appendix A (figure 2).

Recent work has proposed alternative objective functions for sparse and imbalanced data sets, e.g. ”focal loss” (Lin et al., 2018), ”sparsemax” (Martins and Ramon, 2016) or ”probabilistic label trees” (Wydmuch et al., 2018), but due to the computational complexity of these approaches, we did not apply these to the current problem.

4.2 Learning label co-occurrences

The literature on MLC often establishes a link between label dependence and loss minimization, indicating that the predictive accuracy of a label can be improved by using information about the other labels (Dembczynski et al., 2012). For CPC labeling, the model could learn co-occurences of labels and then use this information as a feature (e.g. cosine similarity of predicted labels) in an ensemble model. However, when analyzing the label distribution, we could not identify a strong marginal dependency between CPC labels, which is why we did not follow this approach and leave this for future research.

4.3 Network architecture

For language representation, we apply two models: GloVe (Global Vectors for Word Representation, see Pennington et al., 2014) and BERT (Bidirectional Encoder Representations from Transformers, see Devlin et al., 2018). GloVe is a log-bilinear regression model for the unsupervised learning of word representations based on word co-occurrences. We use the GloVe 6B 100-dimensional pre-trained word embeddings. As opposed to its predecessors such as word2vec or GLoVe, BERT generates dynamic contextual word representations. We use the BERT pre-trained 768-dimensional model. Both models are configured to a maximum sentence length of 250, because this is the cutoff point of the 99-percentile sentence length of abstracts in the dataset.

We create two different models: (1) GloVe + CNN with 23.5M parameters, and (2) BERT + CNN with 115.9M parameters. Both models follow a similar architecture: at the input layer, the word embeddings are passed into a CNN with four parallel convolutional layers, each with 512 filters of kernel size {2, 3, 4, 5}. We apply global max pooling for each convolution. The concatenated representations are fed into a dense layer with 512 hidden units, which are regularized with batch normalization and dropout. The final output layer gives the sigmoid activations for each of the more than 600 CPC labels.

4.4 Model training

For model optimization we used *adam*. For the weighted binary cross-entropy function, we set the hyperparameter α to 2.

Because the complexity of training a model with BERT word embeddings is significantly higher than for GloVe — training takes roughly 60 times longer — we have defined a subset of the USPTO 2M with only 0.3M entries and 624 labels (USPTO 0.3M). We use this subset to provide GloVe and BERT metrics for an equal number of training epochs.

We randomly split into train and test sets with a test size of 20%. The models are trained on 20 epochs each, with the exception of BERT+CNN on the USPTO 2M dataset, where - due to time constraints - we reduced training to 10 epochs. On the USPTO 2M dataset, loss for the GloVe+CNN and BERT+CNN models decreased steadily even when reaching the final epoch, with no indication of overfitting. We assume that metrics would further improve if trained longer. On the USPTO 0.3M dataset however, training stagnated around epoch 15 and we started to see indications of overfitting for both models.

5. Model diagnostics

We notice that the models did not cope well with the sparse label set, even when training the models with a priority on recall by using weighted binary cross-entropy loss: The average number of CPC subclass labels per patent predicted by the model is only 1.04 (maximum: 5 labels), while the test set has 1.34 labels per patent on average. 20% of predictions actually had no labels at all.

By looking at the metrics achieved by CPC section (see table 3 in appendix B), we see that the models perform better if given a sufficient number of training examples: F1 scores on the test set ranged from 0.65 for CPC labels in frequent sections and decreased down to 0.36 for infrequent sections.

6. Empirical results

6.1 Results analysis

In table 1 we show an overview of the different model configurations and their respective performance metrics. In our best configuration GloVe+CNN with weighted binary cross-entropy loss, we achieve an F1 score of 59.2 on the USPTO 2M dataset. While BERT+CNN achieved the highest precision (76.1%), it did not exceed the F1 score achieved by the GloVe-based model. We speculate that with more training epochs, BERT would eventually surpass GloVe.

Model	**Loss function**[1]	**Corpus**	**Labels**	**Precision** (%)	**Recall** (%)	**F1**
GloVe+CNN	BCE	USPTO 2M	632	74.6	45.1	56.2
GloVe+CNN	Weighted BCE	USPTO 2M	632	62.4	**56.3**	**59.2**
BERT+CNN	BCE	USPTO 2M	632	**76.1**	40.1	52.5
BERT+CNN	Weighted BCE	USPTO 2M	632	63.9	52.6	57.8
GloVe+CNN	BCE	USPTO 0.3M	624	73.3	41.1	52.7
GloVe+CNN	Weighted BCE	USPTO 0.3M	624	67.3	50.6	56.2
BERT+CNN	BCE	USPTO 0.3M	624	**74.4**	41.4	53.2
BERT+CNN	Weighted BCE	USPTO 0.3M	624	58.9	**54.8**	**56.8**

Table 1: Models and performance metrics

When working with an almost seven times smaller dataset (USPTO 0.3M), we observe an up to 4 percentage point drop in performance metrics. Here, models using BERT word embeddings scored highest, showing slight superiority over GloVe. However, differences were not as pronounced as expected, considering the computational resources required for BERT.

On both datasets, the application of the weighted binary cross-entropy loss function lead to a clear improvement of the F1 score, at the cost of slightly lower precision.

The highest predicted label had a accuracy of 96.3% (3433 out of 3564).

6.2 Benchmark results

As a reference, we include a benchmark comparison on the metrics used in *DeepPatent* (Shaobo et al., 2018) and *PatentBERT* (Lee and Hsiang, 2019). These studies have reported ranked retrieval metrics such as "top-k" precision, recall and F1 for k=1 and k=5. In our view, since CPC labels are not ranked, the usefulness of ranked retrieval metrics for patent labeling is limited (see appendix C for a critical discussion on ranked retrieval metrics).

We show results for top-1 precision and top-5 recall in table 2. All our algorithms achieved similar top-k performance. On the USPTO 2M dataset, our GloVe+CNN slightly surpassed DeepPatent (7,3%-points lower precision-1 but 9.2%-points higher recall-5). Even when extrapolating our own BERT results with more training epochs, we were not able to exceed the baseline given by PatentBERT (Shaobo et al., 2018, does not mention any model

1. BCE = binary cross-entropy

architecture details, limiting our ability to reproduce the results). On the small USPTO 0.3 dataset, BERT+CNN performed better than GloVe+CNN model, but the differences where smaller than expected, given the high training effort for BERT.

Model	**Loss function**[1]	**Corpus**	**Labels**	**Precision** (top-1, %)	**Recall** (top-5, %)
DeepPatent	BCE	USPTO 2M	632	73.9	74.0
PatentBERT	BCE	USPTO 2M	632	**80.6**	**86.1**
GloVe+CNN	BCE	USPTO 2M	632	**66.6**	83.2
GloVe+CNN	Weighted BCE	USPTO 2M	632	66.1	**83.3**
BERT+CNN	BCE	USPTO 2M	632	63.9	80.3
BERT+CNN	Weighted BCE	USPTO 2M	632	63.9	80.7
GloVe+CNN	BCE	USPTO 0.3M	624	67.6	77.5
GloVe+CNN	Weighted BCE	USPTO 0.3M	624	67.3	**77.8**
BERT+CNN	BCE	USPTO 0.3M	624	**68.6**	77.7
BERT+CNN	Weighted BCE	USPTO 0.3M	624	68.1	77.4

Table 2: Benchmark comparison

7. Conclusion

Significance of work: provide a new baseline for GloVe embeddings applied to patent data (finding: works well, compares with word2vec). Provide new(?) BERT baselines - surprisingly small performamce difference to GloVe - it seems that for classification tasks both embeddings work well. Mention that GloVe is much more computationally efficient than BERT - are the few percentage points improvement worth the effort? Show sensitivity of patent NLP (w.r.t. dataset size). Propose new cost function for MLC to improve recall over sparse label sets. Provide explanations on why NLP models perform good/bad. Discussion on patent classifier for practical application incl. suitable metrices (e.g. top-k discussion).

For future research: normalization and weighting of the multi-label distribution, learning label co-occurrences, ...

References

D. De Clerc et al. Multi-label classification and interactive NLP-based visualization of electric vehicle patent data. *World Patent Information*, 58, 2019.

K. Dembczynski et al. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45, 2012.

J. Devlin et al. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, May 24, 2018.

EPO and USPTO. *Cooperative Patent Classification - Overview and Scheme.* EPO, https://www.cooperativepatentclassification.org, 2015.

J.-S. Lee and J. Hsiang. PatentBERT: Patent classification with fine-tuning a pre-trained BERT model. *arXiv:1906.02124*, May 14, 2019.

T.-Y. Lin et al. Focal loss for dense object detection. *arXiv:1708.02002v2*, 7 Feb, 2018.

A. Martins and F. Ramon. From Softmax to Sparsemax: A sparse model of attention and multi-label classification. *Proceedings of the 33 rd International Conference on Machine Learning*, 48, 2016.

J. Pennington et al. GloVe: Global vectors for word representation. *https://nlp.stanford.edu/projects/glove/*, 2014.

L. Shaobo et al. DeepPatent: Patent classification with convolutional neural networks and word embedding. *Scientometrics*, 117(2):721–744, 2018.

S. Verberne et al. Patent classification experiments with the linguistic classification system LCS. *CLEF notebook papers*, 2010.

WIPO. *World Intellectual Property Indicators 2019.* WIPO, Geneva, 2019.

M. Wydmuch et al. A no-regret generalization of hierarchical softmax to extreme multi-label classification. *arXiv:1810.11671*, 2018.

Appendix A - Objective functions

Binary cross-entropy loss

The binary cross entropy loss function H is defined as:

$$H(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log(\hat{y_i}) + (1 - y_i) * \log(1 - \hat{y_i})$$

where y is the label and $\hat{y}$ is the predicted probability for a given set of examples.

Custom objective functions

We first implemented a variation of the binary cross-entropy loss function which optimizes solely the true positives (see listing 1).

Listing 1: Masked binary cross-entropy objective function

```
from tensorflow.keras import backend as K
def masked_binary_cross_entropy(y_true, y_pred):
  masked_bce = y_true * -K.log(y_pred + K.epsilon())
  return K.sum(masked_bce)/K.sum(y_true)
```

However, because this cost function only "costed" the positive labels and did not have an "opinion" on the negative labels, it tended to push all labels towards 1, even though the vast majority were true negatives. As a result, we discarded this objective function.

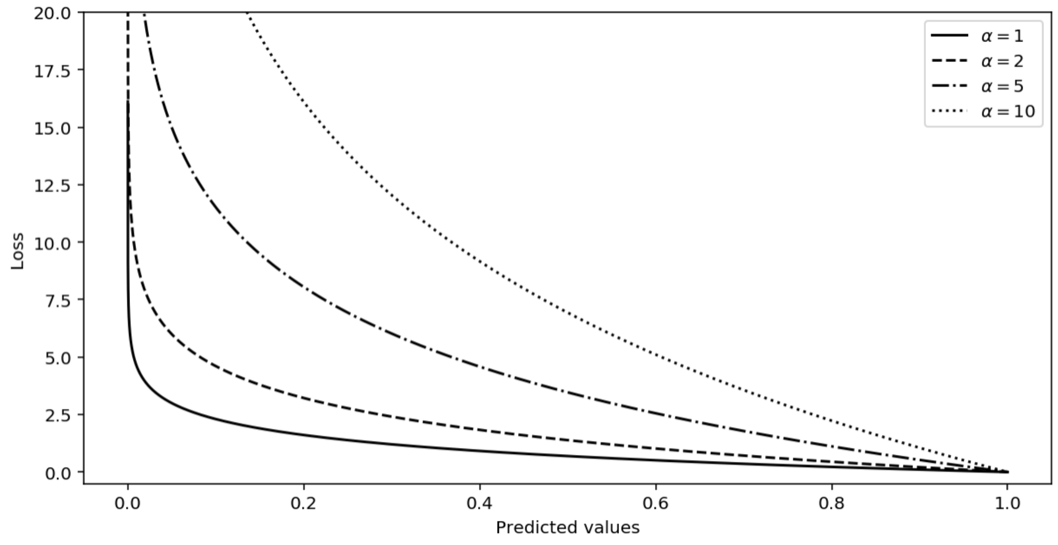
We then moved on to implement a custom loss function that applies a weighting matrix to the binary cross-entropy output (see listing 2). The weighting matrix increases the loss for false negative examples (predicted "0" but labeled "1") by multiplication with the hyper-parameter α, while the loss for false positive examples (predicted "1" but labeled "0") is kept constant.

Listing 2: Weighted binary cross-entropy objective function

```
from tensorflow.keras import backend as K
def weighted_binary_cross_entropy(y_true, y_pred):
    alpha = 2. # weighting factor
    # 1 * alpha if y_true is 1, and 1 if y_true is 0
    weights = (y_true * alpha) + (1. - y_true)
    bce = K.binary_crossentropy(y_true, y_pred)
    return K.mean(bce * weights)
```

Figure 2 shows the loss distribution for predicted values $\hat{y}$ if the true label $y = 1$, for different parameters of α. Cross-entropy loss increases as the predicted probability diverges from the actual label. We can also see that higher values of α increase the gradient false negative predictions. This pushes more predictions toward 1, thereby improving recall at the cost of slightly lower precision. A grid search revealed that for our dataset, best results are achieved for $\alpha = 2$.

Figure 2: Weighted binary cross-entropy loss for $y = 1$



Appendix B - Example observations

CPC Label by Section Performance

Table 3 below shows how our system performed by CPC section for the test data. Generally, higher number patents for a section resulted in higher scores.

Section	**Recall** (%)	**Precision** (%)	**F1**	**Number of Patents**
A	0.59	0.71	0.65	58,400
B	0.31	0.61	0.41	57,740
C	0.43	0.58	0.50	41,138
D	0.25	0.64	0.36	2,438
E	0.32	0.69	0.43	9,519
F	0.38	0.58	0.46	27,173
G	0.56	0.69	0.62	142,986
H	0.55	0.62	0.58	119,198

Table 3: Performance by CPC label sections

Results for specific label - A01H

Description for A01H

NEW PLANTS OR PROCESSES FOR OBTAINING THEM; PLANT REPRODUCTION BY TISSUE CULTURE TECHNIQUES

This subclass covers all aspects related to new plants, including disease resistance, cold resistance and growth speed. In this subclass, angiosperms, i.e. flowering plants, are classified in group A01H 6/00 according to their botanic taxonomy and in group A01H 5/00 according to their plant parts, where disclosed.

Description for C12N (commonly associated with A01H):

MICROORGANISMS OR ENZYMES; COMPOSITIONS THEREOF (biocides, pest repellants or attractants, or plant growth regulators, containing microorganisms, viruses, microbial fungi, enzymes, fermentates or substances produced by or extracted from microorganisms or animal material A01N 63/00; food compositions A21, A23; medicinal preparations A61K; chemical aspects of, or use of materials for, bandages, dressings, absorbent pads or surgical articles A61L; fertilisers C05); PROPAGATING, PRESERVING OR MAINTAINING MICROORGANISMS (preservation of living parts of humans or animals A01N 1/02); MUTATION OR GENETIC ENGINEERING; CULTURE MEDIA (microbiological testing media C12Q)

Example abstract for ground truth [A01H], prediction [A01H]:

a new and distinct hybrid of artichoke named ps h1860 characterized by its numerous bud numbers fleshiness of bracts fleshiness of hearts and uniformity of head shapes ability to bolt in warm summer conditions reduced vernalization requirements allowing spring summer planting and fall production

Example abstract for ground truth [C12P, C07K, A01H], prediction [C07K, C12N]:

the present invention relates to the fields of molecular biology and plant biology specifically the invention is directed to the methods for expressing spider silk proteins in plants and the synthesis and purification of spider silk proteins therefrom

Table 4 below shows for A01H the common CPC codes that are associated with it both for ground truth (y_test) and the predictions (y_pred). For the predictions for A01H, the overall results are shown along with the breakdowns for A01H correctly and incorrectly predicted.

Appendix C - Critical discussion of ranked retrieval metrics

We see fundamental flaws associated with the top-k metric, when applied to datasets with unranked labels: If the average number of labels (in our case 1.3) is greater than the value of "k" (e.g. k=1), this will negatively impact the top-k precision values (because always only k predictions will be evaluated) and the top-k recall will face an artificial upper bound (because there are not enough slots to hold the predictions). And consequently, if the average number of labels (1.3) is lower than the value of "k" (e.g. k=5), precision will be drawn down because it will always consider the top-k predictions as positives, even if they are below the activation threshold, but recall will benefit from the extended k slots.

From y-test for A01H with associated labels (count)									
A01H	C12N	C07H	C07K	A23L	C12P	C12Q	A01N	A01G	A23K
3564	1573	220	154	50	41	29	21	15	10
From y-pred for A01H predicted and other labels predicted (count)									
A01H	C12N	C07H	C12P	A61K	C07K	A01N	C12Q	A01G	G01N
3678	1888	259	64	8	6	6	3	3	2
From y-pred for A01H predicted correctly and other labels predicted (count)									
A01H	C12N	C07H	C12P	A61K	A01N	C07K	C12Q	G01N	B32B
3433	1662	177	32	6	2	2	2	1	0
From y-pred for A01H predicted incorrectly and other labels predicted (count)									
A01H	C12N	C07H	C12P	C07K	A61K	C12Q	A01N	A01C	A01G
131	80	31	25	18	12	7	6	3	2

Table 4: Counts number of A01H labels and associated pairings