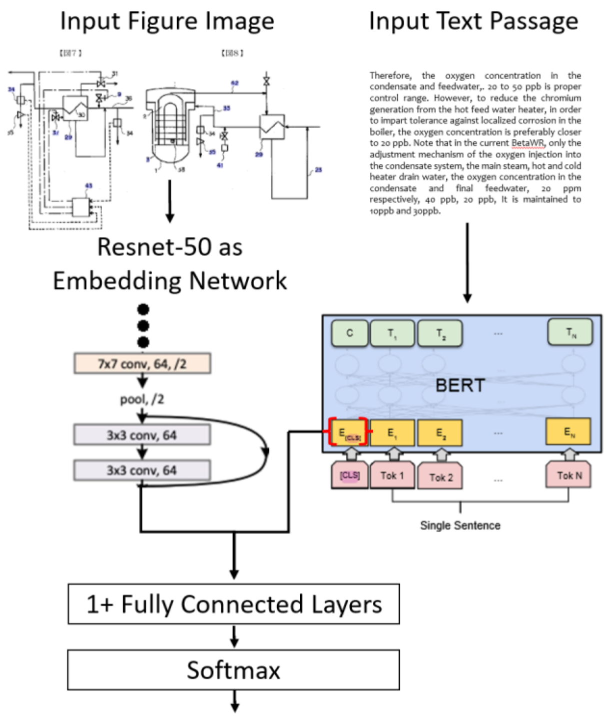
Multimodal Patent Document Classification

Ioannis MARIGGIS*, EPO Examiner in Pattern Recognition 1207
Hendrik STAPELBROEK*, Software Developer at BMW
Athanasios MARIGGIS, EPO Examiner in Protocol Security 1218
*equal contributions

1. Summary of the Approach

We used deep learning to classify the applications and leveraged open-source work to achieve this. Since both the figures and the text are vital information for classifying patent documents, we propose a multi-modal deep learning architecture that fuses features from figures and from text. In our experiments, our solution achieves an overall accuracy of 93.4% on classifying the Y02 CPC class for an unknown patent application (from the test-set) from the bulk data sets provided [link]. Note that accuracies close to 100% in classifying CPC labels will not be realistically possible because of the complex and in certain instances somewhat ambiguous boundaries between the CPC class definitions.

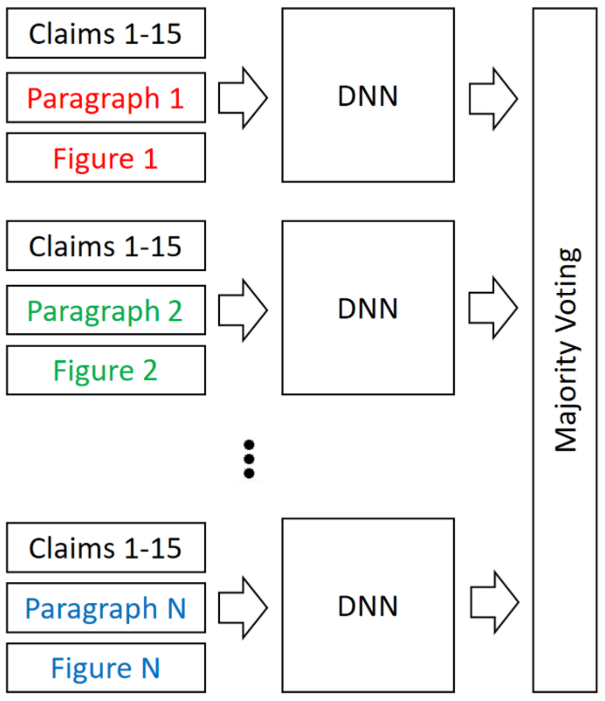
2. Classification



Therefore, the oxygen concentration in the condensate and feedwater,. 20 to 50 ppb is proper control range. However, to reduce the chromium generation from the hot feed water heater, in order to impart tolerance against localized corrosion in the boiler, the oxygen concentration is preferably closer to 20 ppb. Note that in the current BetaWR, only the adjustment mechanism of the oxygen injection into the condensate system, the main steam, hot and cold heater drain water, the oxygen concentration in the condensate and final feedwater, 20 ppm respectively, 40 ppb, 20 ppb, It is maintained to 10ppb and 30ppb.

Initially the text and the image data are processed separately to generate text and image feature embeddings, by utilising BERT and ResNet-50 trained with the BYOL method. The embeddings are then concatenated and passed through one or more fully connected layers followed by a softmax layer, yielding a probability for each of the target classes, which can either be fine grained CPC classes or a Yes/No decision on whether the file is classified as Y02 or one of the Y02 subclasses. The neural network model is evaluated on multiple text-image pairs and a soft majority voting determines the CPC classes for the whole document.

2.1. Overall Architecture



The overall architecture evaluates the deep neural network multiple times. Each pass creates a prediction based on up to 15 claims, text from one paragraph and a drawing that is referred to in that paragraph.

For a single pass of the neural network one paragraph is processed at a time. This is because different applications can have vastly different number of paragraphs and figure references within the paragraphs. Therefore, choosing a fixed size for the number of paragraphs to be processed, like it is the case for the claims, is too constraining and would be a speculative trade-off. Instead, the overall architecture relies on evaluating the neural network for every identified pair of paragraph passage and referred figure.

2.2. Claims

Since all claims are important for the application, we decided that a fixed size of 15 claims in the overall neural network makes sense. The typical patent application in the European phase usually has 15 or up to 15 claims.

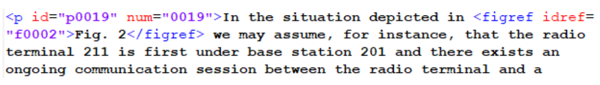
In case a patent application has less than 15 claims, zeros are inserted in the space of the BERT features. If the application has more than 15, then only the first 15 claims are used.

The text of the claims is truncated to the BERT fixed length of 512 tokens.

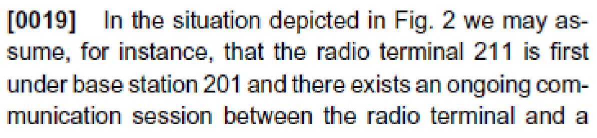
2.3. Paragraph and Figure

Paragraphs in the description of a patent application refer to figures. We utilise figrefs in the xml files to identify correspondences between paragraphs and figures.

The two screenshots below show figref and paragraph tags.



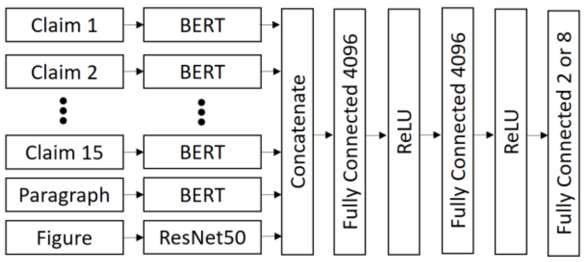
```
<p id="p0019" num="0019">In the situation depicted in <figref idref=
"f0002">Fig. 2</figref> we may assume, for instance, that the radio
terminal 211 is first under base station 201 and there exists an
ongoing communication session between the radio terminal and a
```



[0019] In the situation depicted in Fig. 2 we may assume, for instance, that the radio terminal 211 is first under base station 201 and there exists an ongoing communication session between the radio terminal and a

The chosen text passage that is passed to BERT cannot extend beyond the paragraph it is contained, i.e., a passage from paragraph [0015] cannot include text from either paragraph [0014] or [0016]. The 512 tokens are generated by "growing" the region from the identified figure reference, i.e. if the paragraph is long enough, the text within the figref would be in the middle of the 512 token sequence.

2.4. The Neural Network



The above figure illustrates a single pass of the neural network. The final fully connected layer's dimensionality varies depending on the specific classification task. The utilised BERT model was the pre-trained BERT for Patents [link] or a self-trained BERT-Base model.

In the case of the pretrained BERT, we could, due to hardware limitations, not further fine-tune the model on the data we downloaded from the EPO, because it is a BERT-Large model that would require more graphics memory than available on our GPUs. Note that finetuning could potentially lead to a performance boost.

In the case of the self-trained BERT, we used the BERT-Base model from the provided snapshot of the official repository and fine-tuned on claims and paragraphs from the files downloaded from the EPO. We used the official git repository's pretrain script to do so. We fine-tuned weights for a BERT-Base for claims and a BERT-Base for paragraphs separately.

The stacking of three fully connected layers has been inspired by the VGG-19 classification CNN.

2.5. BYOL and ResNet-50

We used BYOL to train a ResNet-50 that is used as the image embedding network.

There are multiply reason why we consider BYOL a suitable choice:

1. BYOL is the open source software of a very recent yet often cited paper with 2374 citations on the day of this writing.
2. It will be class agnostic. It will learn how to extract embeddings from the figures, irrespectively of the class of the document. This also circumvents the problem of under- and overrepresented classes.
3. BYOL does not require negative samples (i.e. a dog classifier would need images of dogs and images where there are no dogs) and since there is no clear answer to what type of image is clearly not a patent figure for a certain class, it is a very suitable self-supervised learning algorithm for this challenge.

In view of these points, this would make the algorithm suitable for training it on the entire set of recent patent literature for a hypothetical use-case, where all CPC classes constitute the output domain.

BYOL can be found on GitHub [link]

2.6. BERT

Bert is also a very recent approach that has 52695 citations. It is the most implemented paper on paperswithcode [link]. It is ranked very high in a multitude of NLP (Natural Language Processing) tasks and is the 2nd [link] best performing model on the DBpedia dataset. Since it generates feature embeddings of sequences of words it is suitable to be combined with feature embeddings from images, thus enabling implementing a multimodal downstream classification task.

BERT can be found on GitHub [link]

2.7. Fusion of Features

The BERT model is not suitable for usage with large pieces of texts and a patent application can have a variable number of figures. Moreover, certain passages in the description make references to certain figures. Intuitively, as a patent examiner, one paragraph or even a few sentences by themselves, possibly together with one good figure are enough to classify a patent application relatively accurately. Multiple such passage-figure pairs are even better.

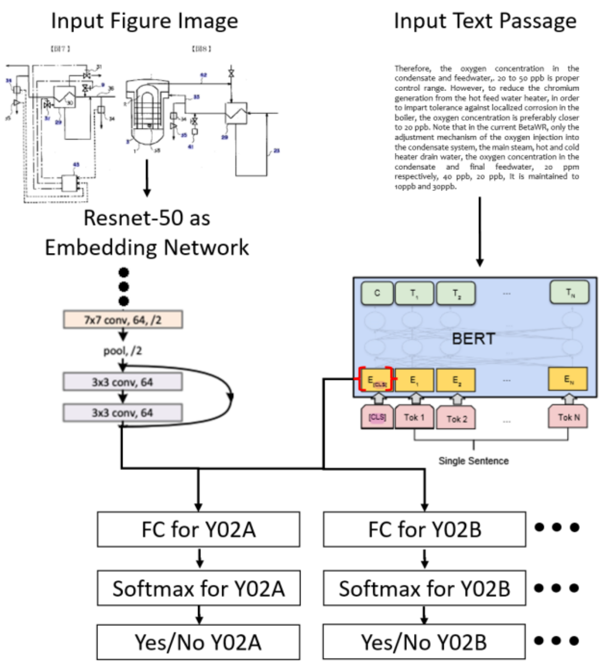
Therefore, each evaluation of the overall neural network is based on the feature embeddings of a reasonably long piece of text and a single figure. The whole document classification is subsequently based on multiple such neural network evaluations. Note that BERT has a limited sequence length, its official GitHub [link] repository suggests this to be up to 512 words.

In our implementation, since claims are the most important parts of a patent application, and in particular a B1 publication, and since most files in the European phase contain up to 15 claims, the first 15 claims are evaluated/used for every batch of data that is processed by a single pass of the overall neural network.

2.8. Soft Majority Voting

We used soft majority voting, which is the common technique of averaging the probabilities from all the input samples, for estimating the combined class probabilities. The final output label is the one scoring the largest probability.

2.9. Binary Classification



In our experiments, we also implemented and evaluated the described neural network architecture for the cases of a multi-class predictor as well as a binary classification task. The binary task, which predicts a yes/no result on a patent application belonging to one of the green plastics classes seemed in our results to outperform the multi-label classification task.

In the binary classification, when applied to multiple classes, the backbone of the network that consists of the BERT and ResNet models is trained and evaluated once. Merely the fully connected layers are trained separately for every target class. This has the benefit that a patent application can be classified directly with more than one class label, without having to resort to defining thresholds for the probability scores for cases in which more than one CPC class is to be classified.

The above figure illustrates this configuration; note that the Yes/No outputs can have respective probabilities that can be used in the subsequent majority voting step.

3. Data

We used the data provided by the competition account [link] for the publications of 2022. We only considered the B1 publications in order to avoid duplicates.

3.1. Green Plastics

We had a look at the CPC class definitions, while considering the amount of patent applications from the available data. Too fine-grained CPC classes would have too few files to train successfully. We identified following classes to be relevant to green plastics but also more broadly to technologies relevant for a greener future, either directly or indirectly:

- Y02A, containing 292 documents
- Y02B, containing 482 documents
- Y02C, containing 28 documents
- Y02D, containing 604 documents
- Y02E, containing 1605 documents
- Y02P, containing 653 documents
- Y02T, containing 1300 documents
- Y02W, containing 94 documents

We decided to broaden the scope from green plastics to more technologies related to a greener future, because we felt that the number of files in the accessible xml files would have been too small otherwise. These are already rather small document counts, with all these Y02 classes containing in total 5058 patent application documents.

3.2. Remarks on Data Quality and Predictive Performance

In order to gain some qualitative insights into the predictive capacity of our algorithm with the data that we used to train it, we had a closer look to some of the files in the test set, where the ground truth and the predicted labels were different.

From these few examples we can infer that our architecture would perform better if we trained it on better (and more) data.

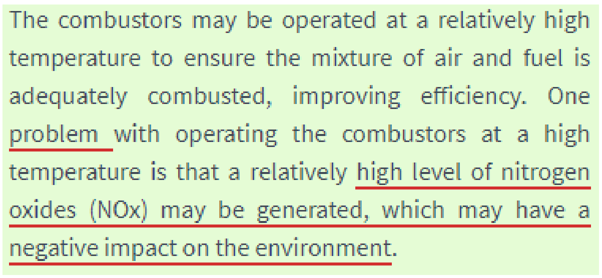
3.2.1. EP18834309

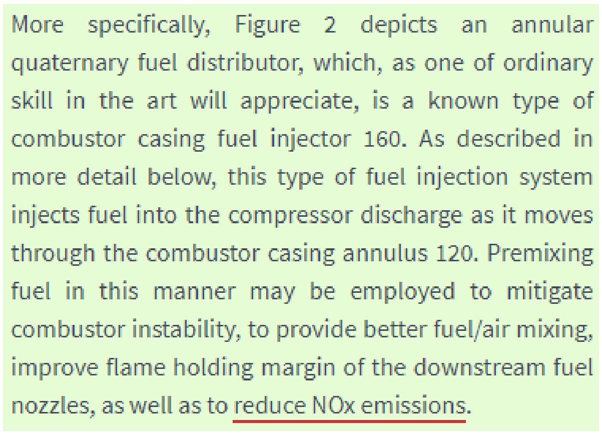
During a manual review of a small number of predicted vs ground truth labels, we noticed a discrepancy in CPC labels in the XML files vs the CPC classes in Ansera*. In the example of EP18834309, the XML file contains a CPC classification of Y02B, which is not present in the classes that can be seen in Ansera. In this example, the ground truth label based on the XML file was “Y02”, while our neural network predicted “not Y02”.

*Working with Ansera and the relevant parts in this report document have been done and seen only by the EPO internal team members.

3.2.2. EP12171869

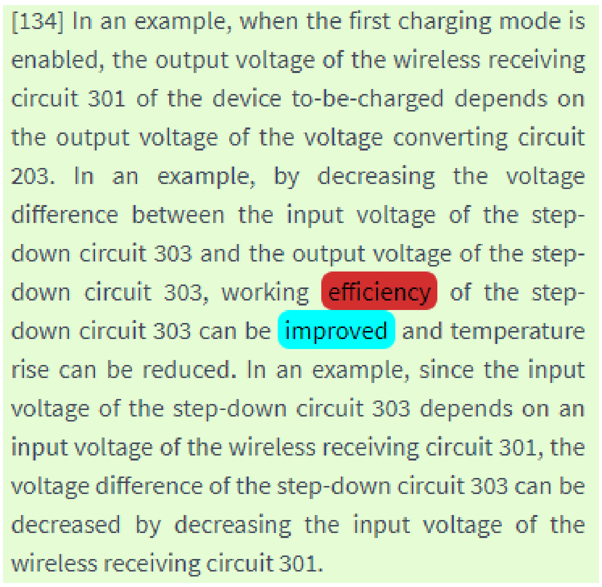
In this specific example, our binary neural network predicted that the file should have a “Y02” tag, while the ground truth label was “not Y02”. Upon a closer look at the description of the patent application, it seems that the Y02T tag is an appropriate CPC classification for this document (see the figures, which highlight some relevant passages from the description).


The combustors may be operated at a relatively high temperature to ensure the mixture of air and fuel is adequately combusted, improving efficiency. One problem with operating the combustors at a high temperature is that a relatively high level of nitrogen oxides (NOx) may be generated, which may have a negative impact on the environment.


More specifically, Figure 2 depicts an annular quaternary fuel distributor, which, as one of ordinary skill in the art will appreciate, is a known type of combustor casing fuel injector 160. As described in more detail below, this type of fuel injection system injects fuel into the compressor discharge as it moves through the combustor casing annulus 120. Premixing fuel in this manner may be employed to mitigate combustor instability, to provide better fuel/air mixing, improve flame holding margin of the downstream fuel nozzles, as well as to reduce NOx emissions.

3.2.3. EP18781213

Similar to the previous example, for this file our binary neural network predicted a Y02 class for this document, while the ground truth label was “not Y02”. This invention seems to be relevant to the Y02 and Y02D classes.


[134] In an example, when the first charging mode is enabled, the output voltage of the wireless receiving circuit 301 of the device to-be-charged depends on the output voltage of the voltage converting circuit 203. In an example, by decreasing the voltage difference between the input voltage of the step-down circuit 303 and the output voltage of the step-down circuit 303, working efficiency of the step-down circuit 303 can be improved and temperature rise can be reduced. In an example, since the input voltage of the step-down circuit 303 depends on an input voltage of the wireless receiving circuit 301, the voltage difference of the step-down circuit 303 can be decreased by decreasing the input voltage of the wireless receiving circuit 301.

4. Evaluation

We separated a testing set of 10% of all data for either of the classification tasks, which was not used for training. This fixed size split choice made it simpler to implement and avoid spending time at better train-test-set split approaches such as the bootstrap method.

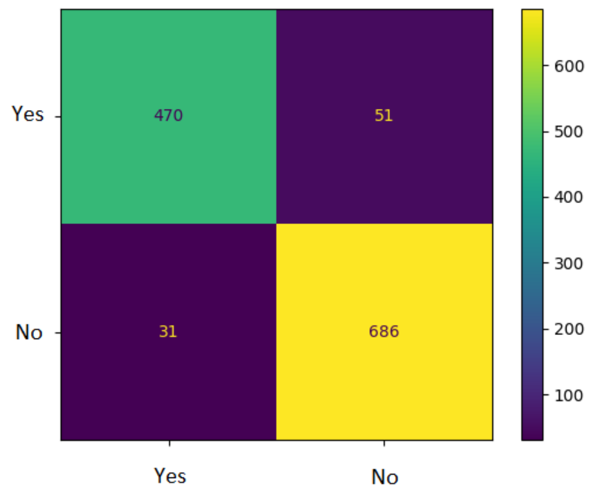
The figures below show confusion matrices, where the numbers represent counts for unique patent application documents. On all the confusion matrix figures, the ground truth labels are on the y-axis and the predictions on the x-axis.

4.1. Binary Classification

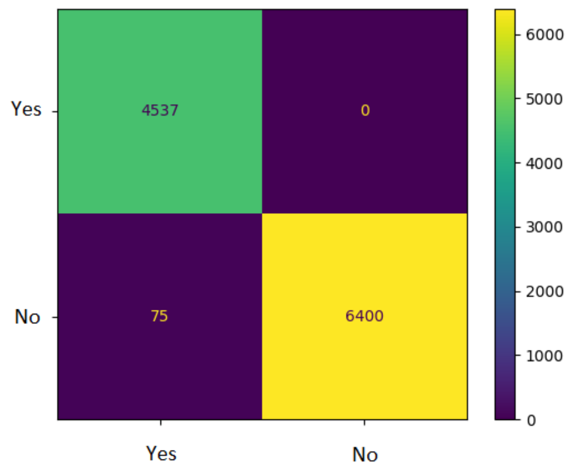
Here we evaluated a yes/no classification as to whether a patent application belongs to one of the specified classes in Y02. The training set contained 11012 unique patent applications, 4537 of which from the selected Y02 classes and the rest randomly selected from other CPC classes.

On the testing set of 1238 separate files our evaluation achieved an overall accuracy of 93.4%, with a true positive rate of 90.2% and a true negative rate of 95.7%.

Confusion matrix on the test set:



Confusion matrix on the training set:



Evaluating the neural network on the training set yields an overall accuracy of 99.3%. The confusion matrix on the training shows that some of the “not Y02” ground truth samples have been predicted as “yes Y02”. This might be related to the examples in the data quality section above.

	Number of train set files	Number of test set files
Yes Y02	4537	521
Not Y02	6475	717

Results shown for weights obtained after 17 epochs of training the fully connected layers.

4.2. Multi-Label Classification

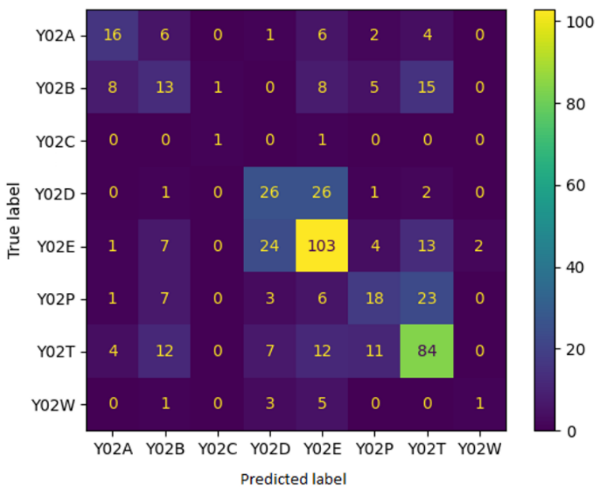
Here we evaluated a multi-label classification as to whether a patent application belongs to one of the specified sub-classes within Y02. All documents evaluated contained one of the subclasses of Y02. The output of the soft majority voting was a single class label. Note, that about 15% of the patent document evaluations were on duplicate entries, as several files had more than one CPC class from the Y02 subclasses assigned to them. Because we didn’t compensate for this in the evaluation code, it implies 85% is the maximum possible achievable score in this evaluation.

Both the test and train sets of the multi-class classification had much fewer samples than the binary classification evaluation:

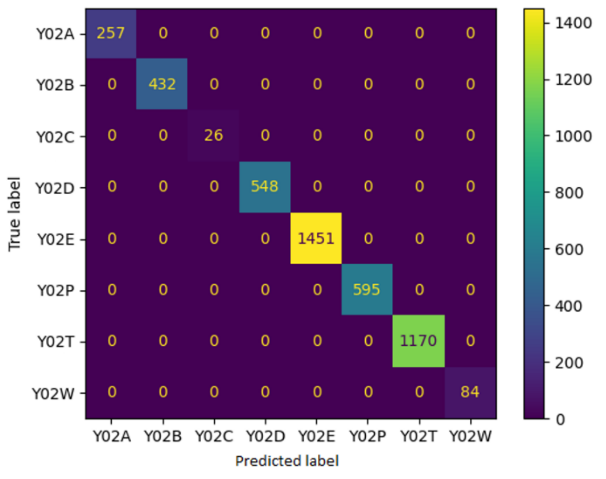
	Number of train set files	Number of test set files
Y02A	257	35
Y02B	432	50
Y02C	26	2
Y02D	548	56
Y02E	1451	154
Y02P	595	58
Y02T	1170	130
Y02W	84	10

Because of this, only the classification results of Y02E and Y02T are somewhat meaningful, with 66.9% and 64.6%, and with respect to the 85% maximum score consideration, 78.7% and 76.0% accuracies for multi-label classification within the Y02 subclasses.

Confusion matrix on the test set:



Confusion matrix on the training set:



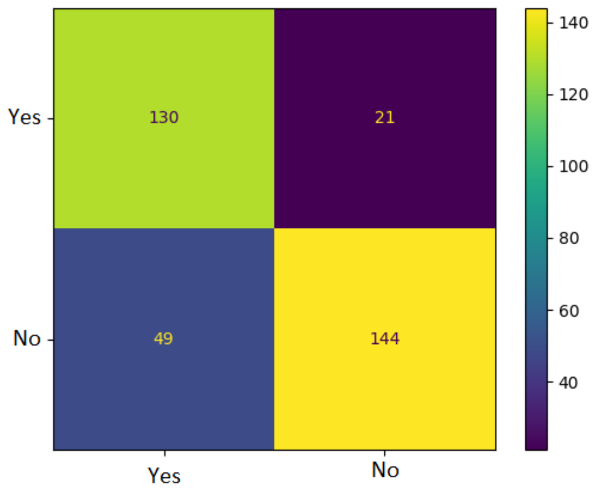
The neural network seems to be overfitting on the training set, and dropout layers seem to not help. With more data and a more even distribution of samples among the classes, much better results are expected.

Results shown for weights obtained after 80 epochs of training the fully connected layers.

4.3. Binary Multi-Label Classification

We got better results when we used the binary classifier on the Y02E subset, to differentiate Y02E vs the other classes within Y02. Here we trained the neural network with data from Y02E for the “Yes” case, and data from the other Y02 subclasses for the “No” case. The results obtained show a 79.7% overall accuracy.

Confusion matrix on the test set:



5. Similarity Search

Since up until the fully connected layer we are essentially dealing with feature embeddings, one possible alternative to classification is to use these embeddings to find similar documents.

5.1. Simple Similarity

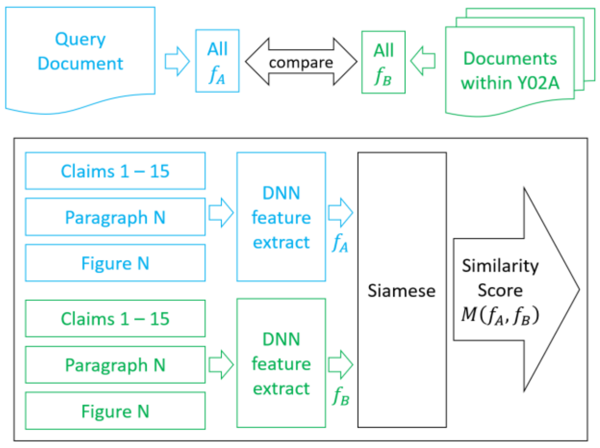
We tried to test the potential of a similarity search based on a simple Euclidean distance and/or cosine similarity metric. Already here we can see that simple metrics yield relatively good findings. In this initial experiment, rigorous search was performed on all files with all files, where for every feature f_A from a file A the closest feature f_B from a file B was taken and the sum of feature distance metrics of all these pairs constituted the similarity metric of the files A and B, where M is either the Euclidean distance or the cosine similarity:

$$Similarity(A, B) = \sum_{\text{all } f_A \text{ of } A} \min_{\text{all } f_B \text{ of } B} \{M(f_A, f_B)\}$$

The evaluation table below shows the performance of choosing the Euclidean and/or the cosine metric to try to find the top-N closest documents. The percentage shows how often the top-N documents contained a document with the correct classification. A more fine-grained analysis can be found in our Google-Drive text file [link].

	top-1	top-5	top-10	top-50
Y02A	57 %	93 %	98 %	100 %
Y02B	52 %	87 %	97 %	100 %
Y02C	66 %	77 %	80 %	89 %
Y02D	55 %	75 %	86 %	100 %
Y02E	82 %	100 %	100 %	100 %
Y02P	70 %	99 %	100 %	100 %
Y02T	66 %	98 %	100 %	100 %
Y02W	59 %	80 %	88 %	96 %

5.2. Deep Similarity

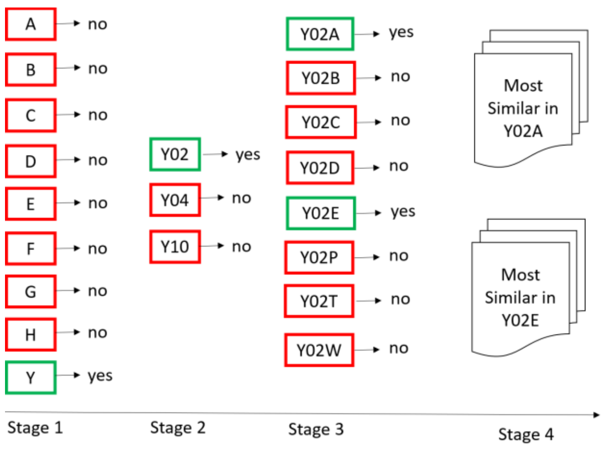


The results with the simple similarity metrics indicate that the features extracted by the neural network can be suitable for a similarity search too. For such a task, a better way that we expect to perform better would be to utilise an architecture that is trained specifically for similarity tasks. One simple example would be to train a Siamese neural network to compare features from two documents and output a similarity score that can be used to rank similar documents in a certain CPC sub-class to an incoming query document.

In the above figure depicting an example implementation for finding similar documents in Y02A, the operations in blue would be performed once per query document, while the operations in green would be performed once during each database entry and the features only looked-up in query-time. Thus, the calculation heavy parts of the deep similarity algorithm would be kept to a bare minimum and with a lightweight Siamese network such a deep similarity-based data query would be scalable to large document databases.

Given the low amount of potential training data in fine grained CPC classes close to the leaves of the CPC tree, deep similarity might perhaps be a better alternative to finding CPC class tags for an incoming document compared to predictions by training a multi-label or many binary classifiers on small training sets.

6. Transferability



Since the CPC is not a simple classification task but is rather a class tree and since our results suggest that binary classification yields better accuracies, it might be conceivable that a system that is really deployed on an EPO-wide scale would combine features from the classification and the similarity search examples shown above.

The figure illustrates an example flow of binary classifications in increasingly deeper levels of the CPC tree followed by similarity searches within sufficiently small document-sets.

One possible algorithm could start at the highest level and either classify via a binary or a multi-class model the classes on that level. For the "Yes" classified classes, it can then go one level deeper in the CPC tree and a model with fine-tuned weights can take over and classify subclasses iteratively.

The higher up in the classification tree we are, the better one can expect the accuracy of the binary or multi-label classification to be. Since the number of files contained within the sub-tree becomes smaller with every iteration, at some point, instead of classifying via an end-to-end network, an approach that either classifies by deep similarity comparisons or that aids a binary/multi-label classifier by deep similarity comparisons could be the better performing option.

7. Hardware

All experiments and training were done on a local machine, with an RTX 3070 GPU. Unfortunately, this meant that BERT-Large models could not be fine-tuned due to memory constraints.

8. Open-Source Usage Remark

There is a lot of successful work in academia, which is open source and generally it is preferable to use as much of the successful open source as possible compared to designing custom solutions. In our proposal, the customization is kept to a minimum and is only present where necessary to be able to use the different input branches.

9. Code Usage Remarks

The code we uploaded in GitHub is complete. We also uploaded all relevant data for the various tasks in our Google Drive [link].

To quickly reproduce the reported values, we provide a "plug and play" tar file for the active data environment [link]. Note that paths are mostly hardcoded, so in order for the code to work with the "plug and play" data without modifications, you need to install Ubuntu 20.04, with a user with the username "im1011", and follow the README dependency installation instructions. Note that in this case the GitHub repository should be cloned under ~/DEV and the data extracted under ~/DATA.

For the quick "plug and play" reproduction, the ResNet and BERT features have been precalculated and only the evaluation script that runs the fully connected layers needs to be executed. The relevant neural network weights can be found as following:

- Binary Y02 vs non-Y02 [link]
- Multi-Class labelling within Y02 [link]
- Binary Y02E vs non-Y02E within Y02 [link]

For training BERT and ResNet via BYOL, download the accordingly named data files within the Google Drive folder and see the accordingly named Python and C++ files and README entries. The weights for BERT and ResNet can be used to reproduce the computation of the respective features and are also in the Google Drive folder in accordingly named files and subfolders.

We can happily assist in installing and running the code on your setup.

Supplementary Material after the EPO Competition

Further Training and OpenAI Embeddings

OpenAI and especially ChatGPT have recently received a lot of attention. The company has made an embedding generation API publicly available, which allows for generating textual embeddings for query strings [link]. We made experiments after the EPO competition to test compatibility and benchmark the performance of our algorithm when incorporating OpenAI embeddings.

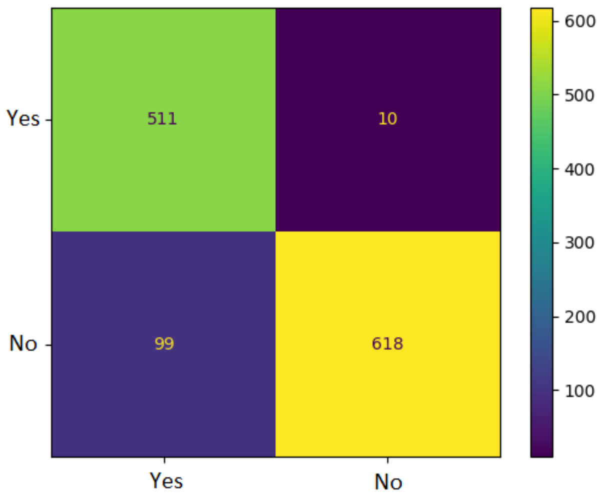
At the time of writing, OpenAI recommends using the “ada-002” model for generating textual embeddings for downstream tasks. Therefore, we used the “ada-002” model to generate embedding vectors for the same text strings as the embedding vectors generated with the BERT model.

We present two additional experiments. In the first we replaced the BERT embeddings with the OpenAI embeddings and in the second we concatenated the OpenAI embeddings to the BERT embeddings to create longer, combined, embedding vectors. Then based on these we trained the fully connected layers to classify for the binary classification task.

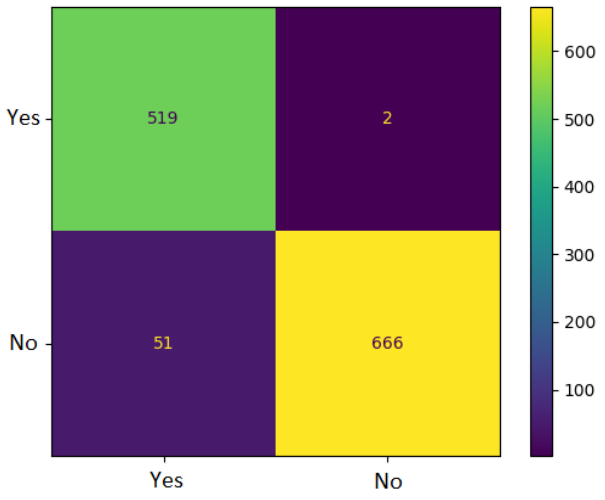
We ran the training for more epochs, since the dimensionality of the OpenAI embeddings and the OpenAI-BERT combined embeddings was larger than the dimensionality of the original BERT embeddings. We also trained the algorithm based on the original BERT embeddings again for this larger number of epochs for comparison, which showed an increase in accuracy. The table below shows overall accuracies for these experiments. Our algorithm seems to perform better with BERT.

BERT only	97.3 %
OpenAI and BERT	95.7 %
OpenAI only	91.2 %

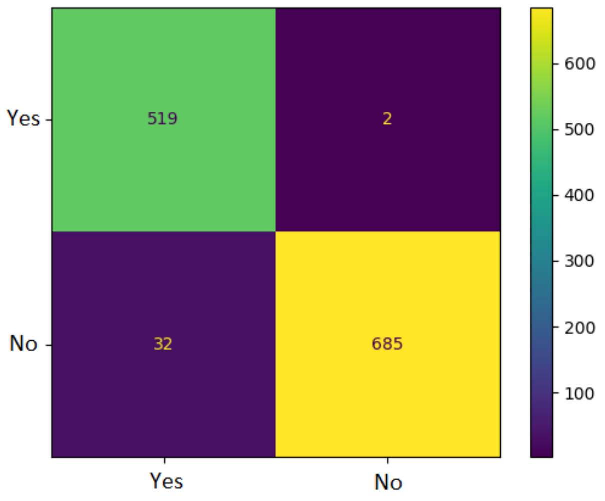
Confusion matrix when using only OpenAI embeddings:



Confusion matrix when using the merged OpenAI and BERT embeddings:



Confusion matrix when using only BERT embeddings:


Yes
No
Yes
No