Priberam at MESINESP Multi-label Classification of Medical Texts Task

Rúben Cardoso, Zita Marinho, Afonso Mendes, and Sebastião Miranda

Priberam Labs, Lisbon, Portugal
labs.priberam.com
{rac,zam,amm,ssm}@priberam.com

Abstract. Medical articles are a crucial tool to provide current state of the art treatments and diagnostics to medical professionals. However, existing public databases such as MEDLINE contain over 27 million articles, making the use of efficient search engines crucial in order to navigate and provide meaningful recommendations. Classifying these articles into broader medical topics can improve retrieval of related articles [1]. The set of medical labels considered for the MESINESP task is on the order of several thousands of labels (DeCS codes), which falls under the extreme multi-label classification problem [2]. The heterogeneous and highly hierarchical structure of medical topics makes the task of manually classifying articles extremely laborious and costly. It is, therefore, crucial to automate the process of classification. Typical machine learning algorithms become computationally demanding with such a large label set and achieving better recall becomes an unsolved problem.
This work presents Priberam's participation at the BioASQ task MESINESP. We address the large multi-label classification problem through the use of four different models: a Support Vector Machine (SVM) [3], the customised search engine Priberam Search [4], a BERT based classifier [5], and a SVM-rank ensemble [6] of all the previous models. Results show that all three individual models perform well and the best performance is achieved by their ensemble, granting Priberam the 6-*th* place in the present challenge and making it the 2-*nd* best team.

1 Introduction

A growing number of medical articles is published every year, with a current estimated rate of at least one new article every 26 seconds [7]. The large magnitude of both the documents and the assigned topics renders automatic classification algorithms a necessity in organising and providing relevant information. Search engines have a vital role in easing the burden of accessing this information efficiently, however, these usually rely on the manual indexing or tagging of articles, which is a slow and burdensome process [8].

The Mesinesp task consists in automatically indexing abstracts in Spanish from two well-known medical databases, IBECS and LILACS, with tags from a pool of 34118 hierarchically structured medical terms, the DeCS codes. This trilingual vocabulary (English, Portuguese and Spanish) serves as a unique vocabulary in indexing medical articles. It follows a tree structure that divides the codes into broader classes and more refined sub-classes respecting their conceptual and semantic relationships [9].

In this task, we tackle the *extreme multi-label (XML)* classification problem. Our goal is to predict for a given article the most relevant subset of labels from an extremely large label set (order of tens of thousands) using supervised training.[1] Typical multi-label classification techniques are not suitable for the XML setting, due to its large computational requirements: the large number of labels implies that both label and feature vectors are sparse and exist in high-dimensional spaces; and to address the sparsity of label occurrence, a large number of training instances is required. These factors make the application of such techniques highly demanding in terms of time and memory, increasing the requirements of computational resources.

The Mesinesp task is even more challenging due to two reasons: first, the articles' labels must be predicted only from the abstracts and titles; and second, all the articles to be classified are in Spanish, which prevents the use of additional resources available only for English, such as BioBERT [11] and ClinicalBERT [12].

This paper describes our participation at the BioASQ task Mesinesp. We explore the performance of a one-vs.-rest model based on Support Vector Machines (SVM) [3] as well as that of a proprietary search engine, Priberam Search [4], which relies on inverted indexes combined with a k-nearest neighbours classifier. Furthermore, we took advantage of BERT's contextualised embeddings [5] and tested three possible classifiers: a linear classifier; a label attention mechanism that leverages label semantics; and a recurrent model that predicts a sequence of labels according to their frequency. We propose the following contributions:

– Application of BERT's contextualised embeddings to the task of XML classification, including the exploration of linear, attention based and recurrent classifiers. To the best of our knowledge, this work is the first to apply a pretrained BERT model combined with a recurrent network to the XML classification task.
– Empirical comparison of a simple one-vs.-rest SVM approach with a more complex model combining a recurrent classifier and BERT embeddings.
– An ensemble of the previous individual methods using SVM-rank, which was capable of outperforming them.

---

[1] The task of multi-label classification differs from multi-class classification in that labels are not exclusive, which enables the assignment of several labels to the same article, making the problem even harder [10].

2　Related Work

Currently, there are two main approaches to XML: embedding based methods and tree based methods.

Embedding based methods deal with the problem of high dimensional feature and label vectors by projecting them onto a lower dimensional space [8,13]. During prediction, the compressed representation is projected back onto the space of high dimensional labels. This information bottleneck can often reduce noise and allow for a way of regularising the problem. Although very efficient and fast, this approach assumes however that the low-dimensional space is capable of encoding most of the original information. For real world problems, this assumption is often too restrictive and may result in decreased performance.

Tree based approaches intend to learn a hierarchy of features or labels from the training set [14,15]. Typically, a root node is initialised with the complete set of labels and its children nodes are recursively partitioned until all the leaf nodes contain a small number of labels. During prediction, each article is passed along the tree and the path towards its final leaf node defines the predicted set of labels. These methods tend to be slower than embedding based methods but achieve better performance. However, if a partitioning error is made near the top of the tree, its consequences are propagated to the lower levels.

Furthermore, other methods should be referred due to their simple approach capable of achieving competitive results. Among these, DiSMEC [10] should be highlighted because it follows a one-vs.-rest approach which simply learns a weight vector for each label. The multiplication of such weight vector with the data point feature vector yields a score that allows the classification of the label. Another simple approach consists of performing a set of random projections from the feature space towards a lower dimension space where, for each test data point, a k-nearest neighbours algorithm performs a weighted propagation of the neighbour's labels, based on their similarity [16].

We propose two new approaches which are substantially distinct from the ones discussed above. The first one uses a search engine based on inverted indexing and the second leverages BERT's contextualised embeddings combined with either a linear or recurrent layer.

3　XML Classification Models

We explore the performance of a one-vs.-rest SVM model in §3.1, and a customised search engine (Priberam Search) in §3.2. We further experiment with several classifiers leveraging BERT's contextualised embeddings in §3.3. In the end we aggregate the predictions of all of these individual models using a SVM-Rank algorithm in §3.4.

3.1　Support Vector Machine

Our first baseline consists of a simple Support Vector Machine (SVM) using a one-vs.-rest strategy. We train an independent SVM classifier for each possible

label. To reduce the burden of computation we only consider labels with frequency above a given threshold f_{min}. Each classifier weight $\boldsymbol{w} \in \mathbb{R}^d$ measures the importance assigned to each feature representation of a given article and is trained to optimise the max-margin loss of the support vectors and the hyper plane $\boldsymbol{x}_i \in \mathbb{R}^d$ [3]:

$$\min_{\boldsymbol{w}} \frac{1}{2} \boldsymbol{w} \boldsymbol{w}^T + C \sum_{i=1}^{l} \xi(\boldsymbol{w}; \boldsymbol{x}_i, y_i) \tag{1}$$

$$\text{s.t. } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + \boldsymbol{b} \geq 1 - \xi_i)$$

where $(\boldsymbol{x}_i, y_i)$ are the article-label pairs, C is the regularisation parameter, $\boldsymbol{b}$ is a bias term and ξ corresponds to a slack function used to penalise incorrectly classified points and $\boldsymbol{w}$ is the vector normal to the decision hyper-plane. We used the abstract's term frequency–inverse document frequency (tf-idf) as features to represent $\boldsymbol{x}_i$.

3.2 Priberam Search

The second model consists of a customised search engine, Priberam Search, based on inverted indexing and retrieval using the Okapi-BM25 algorithm [4]. It uses an additional k-nearest neighbours algorithm (k-NN) to obtain the set of k indexed articles closest to a query article in feature space. This similarity is based on the frequency of words, lemmas and root-words, as well as label semantics and synonyms. A score is given to each one of these articles and to each one of their labels and label synonyms, and a weighted sum of these scores yields the final score assigned to each label.

3.3 XML BERT Classifier

Language model pretraining has recently advanced the state of the art in several Natural Language Processing tasks, with the use of contextualised embeddings such as BERT, Bidirectional Encoder Representations from Transformers [5]. This model consists of 12 stacked transformer blocks and its pretraining is performed on a very large corpus following two tasks: next sentence prediction and masked language modelling. The nature of the pretraining tasks makes this model ideal for representing sentence information (given by the representation of the $[CLS]$ token added to the beginning of each sentence). After encoding a sentence with BERT, we apply different classifiers, and fine tune the model to minimise a multi-label classification loss:

$$\text{BCELoss}(\boldsymbol{x}_i; \boldsymbol{y}_i) = y_{i,j} \log \sigma(x_{i,j}) + (1 - y_{i,j}) \log(1 - \sigma(x_{i,j})), \tag{2}$$

where $y_{i,j}$ denotes the binary value of label j of article i, which is 1 if it is present and 0 otherwise, $x_{i,j}$ represents the label predictions (logits) of article i and label j, and σ is the sigmoid function.

3.3.1 In-domain transfer knowledge Additionally, we performed an extra step of pretraining. Starting from the original weights obtained from BERT pretrained in Spanish, we further pretrained the model with a task of masked language modelling on the corpus composed by all the articles in the training set. This extra step results in more meaningful contextualised representations for this medical corpus, whose domain specific language might differ from the original pretraining corpora.

After this, we tested three different classifiers: a linear classifier in §3.3.2, a linear classifier with label attention in §3.3.3 and a recurrent classifier in §3.3.4.

3.3.2 XML BERT Linear Classifier The first and simplest classifier consists of a linear layer which maps the sequence output (the 768 dimensional embedding - corresponding to the $[CLS]$ token) to the label space, composed by 33702 dimensions corresponding to all the labels found in the training set. Such architecture is represented in figure 1. We minimise binary cross-entropy using sigmoid activations to allow for multiple active labels per instance, see Eq. 2. This classifier is hereafter designated LINEAR.
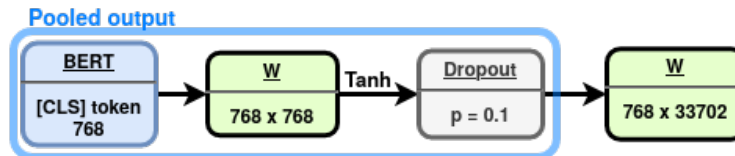

Fig. 1: XML BERT Linear Classifier: Flowchart representing BERT's pooled output (in blue) and the simple linear layer (W in green) used as XML classifier.

3.3.3 XML BERT With Label Attention For the second classifier, we assume a continuous representation with 768 dimensions for each label. We initialise label embeddings as the pooled output embeddings (corresponding to the $[CLS]$ token) of a BERT model whose inputs were the string descriptors and synonyms for each label. We consider a key-query-value attention mechanism [17], where the query corresponds to the pooled output of the abstract's contextualised representation and the keys and values correspond to the label embeddings. We further consider residual connections, and a final linear layer maps these results to the decision space of 33702 labels using a linear classifier, as shown in figure 2. Once again, we choose a binary cross-entropy loss (Eq.2). This classifier is hereafter designated LABEL ATTENTION.
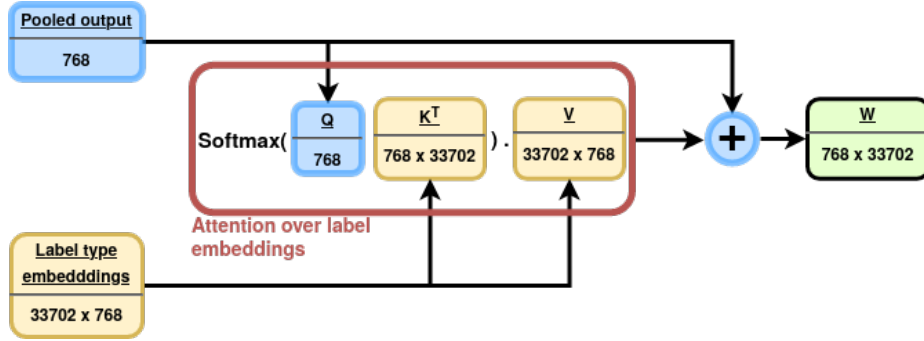
Fig. 2: XML BERT with Label Attention Classifier: Article's pooled output (blue) is followed by an extra step of attention over the label embeddings (red) which are finally mapped to a XML linear classifier over labels (green).

3.3.4   XML BERT With Gated Recurrent Unit In the last classifier, we predict the article's labels sequentially. Before the last linear classifier used to project the final representation onto the label space, we add a Gated Recurrent Unit (GRU) network [18] with 768 units that sequentially predicts each label according to label frequency. A flowchart of the architecture is shown in figure 3. This sequential prediction is performed until the prediction of the stopping label is reached.
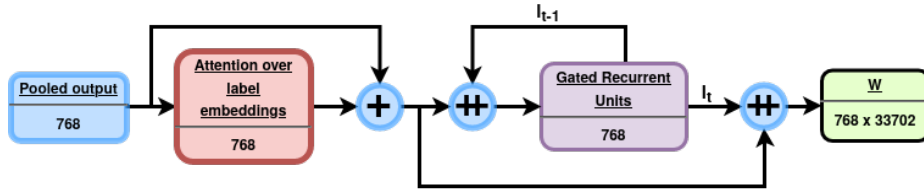

Fig. 3: XML BERT GRU Classifier: The GRU network precedes the linear layer and sequentially predicts the labels. The symbol ⧺ stands for vector concatenation and l_t is the label representation predicted by the GRU at time-step t.

We consider a binary cross-entropy loss with two different approaches. On the first approach, all labels are sequentially predicted and the loss is computed only after the stopping label is predicted, i.e., the loss value is independent of the order in which the labels are predicted. It only takes into account the final set. This loss is denominated Bag of Labels loss (BOLL) and it is given by:

$$\mathcal{L}_{BOLL} = \text{BCELoss}(\boldsymbol{x}_i; \boldsymbol{y}_i) \tag{3}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are the total set of predicted logits and gold labels for the current article i, correspondingly. The models trained with this loss are hereafter designated GRU BOLL.

The second approach uses an iterative loss which is computed at each step of the sequential prediction of labels. We compare each predicted label with the gold label, the loss is computed and added to a running loss value. In this case, the loss is denominated Iterative Label loss (ILL):

$$\mathcal{L}_{ILL} = \sum_{t \in T} \text{BCELoss}(\boldsymbol{x}_i^{(t)}; \boldsymbol{y}_i^{(t)}) \tag{4}$$

where T is the length of the label sequence, t denotes the time-steps taken by the GRU until the "stop label" is predicted, and $\boldsymbol{x}_i^{(t)}$ and $\boldsymbol{y}_i^{(t)}$ are the predicted logits and gold labels for time-step t and article i, respectively. Models trained with this loss are hereafter designated GRU ILL.

Although only one of the losses accounts directly for prediction order, this factor is always relevant because it affects the final set of predicted labels. This way, the model must be trained and tested assuming a specific label ordering. For this work, we used two orders: ascending and descending label frequency on the training set, designated GRU ASCEND and GRU DESCEND, respectively.

Additionally, we developed a masking system to force the sequential prediction of labels according to the chosen frequency order. This means that at each step the output label set is reduced to all labels whose frequency fall bellow or above the previous label, depending on the monotonically ascending or descending order, respectively. Models in which such masking is used are designated GRU W/ MASK.

3.4 Ensemble

Furthermore, we developed an ensemble model combining the results of the previously described SVM, Priberam Search and BERT with GRU models. This ensemble's main goal is to leverage the label scores yielded by these three individual models in order to make a more informed decision regarding the relevance of each label to the abstracts.

We chose an ensembling method based on a SVM-rank algorithm [6] whose features are the normalised scores yielded by the three individual models, as well as their pairwise product and full product. These scores are the distance to the hyper-plane in the SVM model, the k-nearest neighbours score for Priberam Search and the label probability for the BERT model.

An SVM-rank is a variant of the support vector machine algorithm used to solve ranking problems [19]. It essentially leverages pair-wise ranking methods to sort and score results based on their relevance for a specific query. This algorithm optimises an analogous loss to the one shown in Eq. 1. Such ensemble is hereafter designated SVM-RANK ENSEMBLE.

4 Experimental Setup

We consider the training set provided for the MESINESP competition containing 318658 articles with at least one DeCS code and an average of 8.12 codes per article. We trained the individual models with 95% of this data. The remaining 5% were used to train the SVM-rank algorithm. The provided smaller official development set, with 750 samples, was used to fine-tune the individual model's and ensemble's hyper-parameters, while the test set, with 500 samples, was used for reporting final results. These two sets were manually annotated by experts specifically for the MESINESP task.

4.1 Support Vector Machine

For the SVM model we chose to ignore all labels that appeared in less than 20 abstracts. With this cutoff, we decrease the output label set size to ≈ 9200. Additionally, we use a linear kernel to reduce computation time and avoid over-fitting, which is critical to train such a large number of classifiers. Regarding regularisation, we obtained the best performance using a regularisation parameter set to $C = 1.0$, and a squared hinge slack function whose penalty over the misclassified data points is computed with an ℓ_2 distance.

Furthermore, to enable more control over the classification boundary, after solving the optimisation problem we moved the decision hyper-plane along the direction of $\boldsymbol{w}$. We empirically determined that a distance of -0.3 from its original position resulted in the best $\mu F1$ score. This model was implemented using a scikit-learn.[2]

4.2 Priberam Search

To use the Priberam Search Engine, we first indexed the training set taking into account the abstract text, title, complete set of gold DeCS codes, and also their corresponding string descriptors along with some synonyms provided[3]. We tuned the number of neighbours $k = [10, 20, 30, 40, 50, 60, 70, 100, 200]$ in the development set for the k-NN algorithm and obtained the best results for $k = 40$. To decide whether or not a label should be assigned to an article, we fine-tuned a score threshold over the interval $[0.1, 0.5]$ using the official development set, obtaining a best performing value of 0.24. All labels with score above the threshold were picked as correct labels.

---

[2] scikit-learn.org

[3] https://temu.bsc.es/mesinesp/wp-content/uploads/2019/12/DeCS.2019.v5.tsv.zip

4.3 BERT

For all types of BERT classifiers, we used the Transformers and PyTorch Python packages [20, 21].

We initialised BERT's weights from its cased version pretrained on Spanish corpora, bert-base-spanish-wwm-cased[4].

We further performed a pretraining on the MESINESP dataset to obtain better in domain embeddings. For the pretraining and classification task, table 1 shows the training hyper-parameters.

For all the experiments with BERT, the complete set of DeCS codes was considered as the label set.

Hyper-parameter	Pretraining	Classification
Batch size	4	8
Learning rate	$5 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
Warmup steps	0	4000
Max seq lenght	512	512
Learning rate decay	-	linear
Dropout probability	0.1	0.1

Table 1: Training hyper-parameters used for BERT's pretraining and classification tasks.

4.4 Ensemble

Our ensemble model aggregates the prediction of all the individual contenders and produces a final predicted label set. To improve recall we lowered the thresholds set for each individual model until the value for which the average number of predicted labels per abstract was approximately double the average number of gold labels. This ensures that the SVM-rank algorithm was trained with a balanced set, resulting in a system in which the individual models have very high recall and the ensemble model is responsible for precision.

We trained the SVM-rank model with the 5% hold-out data of the training set. Furthermore, SVM-rank returns a score for each label in each abstract, making it necessary to define a threshold for classification. This threshold was fine-tuned over the interval $[-0.5, 0.5]$ using the official MESINESP development set, yielding a best performing cut-off score of -0.0233.

We also fine-tuned the regularisation parameter, C. We experimented the values $C = [0.01, 0.1, 0.5, 1, 5, 10]$ obtaining the best performance for $C = 0.1$. The current model was implemented using a Python wrapper for the dlib C++ toolkit [22].

---

[4] https://github.com/dccuchile/beto

5 Results

Table 2 shows the μ-precision, μ-recall and μ-F1 metrics for the best performing models described above, evaluated on both the official development and test sets.

The comparison between the scores obtained for the one-vs.-rest SVM and Priberam Search models shows that the SVM outperforms the k-NN based Priberam Search in terms of μF1, which is mostly due to its higher recall. Note that, although not ideal for multi-label problems, the one-vs.-rest strategy fro the SVM model was able to achieve a relatively good performance, even when the predicted label set was significantly reduced.

Model	Development set			Test set		
	μP	μR	μF1	μP	μR	μF1
SVM	0.4216	0.3740	0.3964	0.4183	**0.3789**	0.3976
Priberam Search	0.4471	0.3017	0.3603	0.4571	0.2700	0.3395
Bert-Gru Boll ascend	0.4130	**0.3823**	0.3971	0.4293	0.3314	0.3740
SVM-rank ensemble	**0.5056**	0.3456	**0.4105**	**0.5336**	0.3320	**0.4093**

Table 2: Micro precision (μP), micro recall (μR) and micro F1 (μF1) obtained with the 4 submitted models for both the development and test sets. For each metric, the best performing model is identified in bold.

Table 3 shows the performance of several classifiers used with BERT. Note that, for these models, in order to save time and computational resources some tests were stopped before achieving their maximum performance, allowing nonetheless comparison with other models.

We trained linear classifiers using the BERT model with pretraining on the MESINESP corpus for $660k$ steps ($\approx$ 19 epochs) and without such pretraining (marked with *). Results show that, even with an under-trained classifier, such pretraining is already advantageous. This pretraining was employed for all models combining BERT embeddings with a GRU classifier. The label-attentive Bert model (Gru Boll ascend) shows negligible impact on performance, when compared with the simple linear classifier (Linear).

We consider three varying architectures of the Bert-Gru model: Bag of Labels loss (Boll) or Iterative Label loss (Ill), ascending or descending label frequency, and usage or not of masking. Taking into account the best score achieved, the BOLL loss performs better than the ILL loss, even with a smaller number of training steps. For this BOLL loss, it is also evident that the ordering of labels with ascending frequency outperforms the opposite order, and that masking results in decreased performance.

On the other hand, for the ILL loss, masking improves the achieved score and the ordering of labels with descending frequency shows better results. The

best classifier for a BERT-based model is the GRU network trained with a Bag of Labels loss and with labels provided in ascending frequency order (Gru Boll ascend). This model was further trained for a total of 28 epochs resulting in a μF1=0.4918 on the 5% hold-out of the training set. It is important to notice the performance drop from the 5% hold-out data to the official development set. This drop is likely a result of the mismatch between the annotation methods used in the two sets, given that the development set was specifically manually annotated for this task.

Surprisingly, the BERT based model shows worse performance than the SVM on the test set. Despite their very similar μF1 scores for the development set, the BERT-GRU model suffered a considerable performance drop from the development to the test set due to a decrease in recall. This might indicate some over-fitting of hyper-parameters and a possible mismatch between these two expert annotated sets.

Additionally, as made explicit in table 2, the ensemble combining the results of the SVM, Priberam Search and the best performing BERT based classifier achieved the best performance on the development set, outperforming all the individual models.

BERT classifier	Training steps	μF1
Linear*	220k	0.4476
Linear	250k†	0.4504
Label attention*	700k	0.4460
Gru Boll ascend	80k	**0.4759**
Gru Boll descend	40k	0.4655
Gru Boll ascend w/ mask	100k†	0.4352
Gru Ill descend	240k†	0.4258
Gru Ill descend w/ mask	240k†	0.4526
Gru Ill ascend w/ mask	240k†	0.4459

Table 3: μF1 metric evaluated for the 5% hold-out of the training set. All models have been pretrained on the Mesinesp corpus, except for those duly marked. BOLL: Bag of Labels loss. ILL: Iterative Label loss. *: not pretrained on Mesinesp corpus. †: training stopped before maximum μF1 was reached.

Finally, table 4 shows additional classification metrics for each one of the submitted systems, as well as their rank within the Mesinesp task. The analysis of such results makes clear that for the three considered averages (Micro, Macro and per sample), the SVM model shows the best recall score. For most of the remaining metrics, the SVM-rank ensemble is able to leverage the capabilities of

the individual models and achieve considerable performance gains, particularly noticeable for the precision scores.

Metric	SVM	Priberam Search	BERT-GRU boll ascend	SVM-rank ensemble
μF1	0.3976 (7°)	0.3395 (13°)	0.3740 (9°)	**0.4093(6°)**
μP	0.4183 (17°)	0.4571 (10°)	0.4293 (15°)	**0.5336(6°)**
μR	**0.3789(6°)**	0.2700 (13°)	0.3314 (8°)	0.3320 (7°)
MaF1	**0.4183(8°)**	0.1776 (13°)	0.2009 (11°)	0.2115 (10°)
MaP	0.4602 (9°)	0.4971 (8°)	0.4277 (11°)	**0.5944(3°)**
MaR	**0.2609(8°)**	0.1742 (16°)	0.2002 (11°)	0.2024 (10°)
EbF1	0.3976 (7°)	0.3393 (13°)	0.3678 (9°)	**0.4031(6°)**
EbP	0.4451 (15°)	0.4582 (12°)	0.4477 (14°)	**0.5465(3°)**
EbR	**0.3904(6°)**	0.2824 (13°)	0.3463 (8°)	0.3452 (8°)

Table 4: Micro (μ), macro (Ma) and per sample (Eb) averages of the precision, recall and F1 scores, followed by score position within the Mesinesp task. For each metric, the best performing model is identified in bold.

6 Conclusions

This paper introduces three type of extreme multi label classifiers: an SVM, a k-NN based search engine and a series of BERT classifiers. Our one-vs.-rest SVM model shows the best performance on all recall metrics. We further provide an empirical comparison of different variants of multi-label BERT based classifiers, where the Gated Recurrent Unit network with the Bag of Labels loss shows the most promising results. This model yields slightly better results than the SVM model on the development set, however, due to a drop in recall, under-performs it on the test set. Finally, the SVM-rank ensemble is able to leverage the label scores yielded by the three individual models and combine them into a final ranking model with a precision gain on all metrics, capable of achieving the highest μF1 score (being the 6-th best model in the task).

7 Acknowledgements

References

1. Yi X, Allan J. A comparative study of utilizing topic models for information retrieval. European conference on information retrieval, pp. 29-41. Springer (2009).
2. Shen Y, Yu HF, Sanghavi S, Dhillon I. Extreme Multi-label Classification from Aggregated Labels. arXiv preprint arXiv:2004.00198 (2020).
3. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. LIBLINEAR: A library for large linear classification. Journal of machine learning research (2008).
4. Miranda S, Nogueira D, Mendes A, Vlachos A, Secker A, Garrett R, Mitchel J, Marinho Z. Automated Fact Checking in the News Room. In The World Wide Web Conference (2019).
5. Devlin J, Chang M, Lee K, Toutanova K. BERT: pretraining of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (2019).
6. Joachims T. Optimizing search engines using clickthrough data. InProceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (2002).
7. Garba, S., Ahmed, A., Mai, A., Makama, G. and Odigie, V. Proliferations of scientific medical journals: a burden or a blessing. Oman medical journal, 25(4), p.311 (2010).
8. Zhang W, Yan J, Wang X, Zha H. Deep extreme multi-label learning. Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (2018).
9. VHL Network Portal. Red.bvsalud.org. 2020. Decs. [online] Available at: http://red.bvsalud.org/decs/en/about-decs/ (Accessed 2 May 2020).
10. Babbar R, Schölkopf B. DiSMEC: Distributed Sparse Machines for Extreme Multi-label Classification. Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (2017).
11. Lee J, Yoon W, Kim S, Kim D, Kim S, So CH, Kang J. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics (2020 Feb).
12. Alsentzer E, Murphy JR, Boag W, Weng WH, Jin D, Naumann T, McDermott M. Publicly available clinical BERT embeddings. arXiv preprint arXiv:1904.03323 (2019).
13. Tai F, Lin HT. Multilabel classification with principal label space transformation. Neural Computation (2012).
14. Prabhu Y, Varma M. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (2014).
15. Agrawal R, Gupta A, Prabhu Y, Varma M. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. Proceedings of the 22nd international conference on World Wide Web (2013).
16. Verma Y. An Embarrassingly Simple Baseline for eXtreme Multi-label Prediction. arXiv preprint arXiv:1912.08140 (2019).
17. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Advances in neural information processing systems, pp. 5998-6008 (2017).
18. Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (2014).

19. Liu TY. Learning to rank for information retrieval. Springer Science & Business Media (2011).

20. Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, Cistac P, Rault T, Louf R, Funtowicz M, Brew J. HuggingFace's Transformers: State-of-the-art Natural Language Processing. ArXiv (2019).

21. Paszke A, Gross S, and Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems 32, p.8024–8035 (2019)

22. King DE. Dlib-ml: A machine learning toolkit. The Journal of Machine Learning Research (2009).