Automatic ICD Code Classification with Label Description Attention Mechanism

Kathryn Chapman[a,b], Günter Neumann[a,b]

[a]DFKI GmbH, Campus D3 2, 66123 Saarbrücken, Germany
[b]Saarland University, Saarbrücken, Germany

Abstract

We present our submission for the CANTEMIST (CANcer TExt Mining Shared Task – tumor named entity recognition) 2020 task [1]. We participated in track 3, which focuses on automatic eCIE-O-3.1 codes assignment (English version: ICD-O-3), an extreme multi-label classification (XMLC) problem. We developed a model which utilizes a BERT-like encoder and word-level attention mechanism between input clinical cases and textual descriptions of the labels. We found that our model predicted a wider variety of codes across the test set than our baseline, thereby capturing more low-resource labels. Our final submission achieved a mean average precision (MAP) of 39.4% and F1-micro of 26.8%.

Keywords

ICD, Extreme Multilabel Classification, Electronic Health Records, Cancer Text Mining

1. Introduction

Automatic International Classification of Diseases (ICD) code assignment is a classification problem which involves taking as input an electronic health record (EHR) and outputting a set of ICD codes (labels). Unlike multi-class classification, where only one label is assigned from three or more classes, ICD code assignment is a multi-label classification problem, meaning that there can be any number of labels from a set assigned to an instance. Furthermore, automatic ICD code assignment constitutes an *extreme multi label classification* (XMLC) problem as there are 140,000+ ICD-10 codes, a number which regularly increases due to new diagnoses and procedures in the medical field. In XMLC, a small subset of labels from a very large label space are assigned to an instance. A challenge specific to XMLC is known as the 'tail end problem', which means that there are a few labels assigned to many instances, while the vast majority of labels - the 'tail end' - in the labels space are assigned to very few instances [2]. As such, it is important in XMLC to ensure that models are not achieving high scores by simply assigning only the frequent labels, as this collapses the XMLC problem into a basic multi-label classification one. In other words, if a label space has a dimensionality of 1000, and a classifier performs well according to a given metric but only ever assigns the top 20 labels, this is not a good XMLC classifier. Therefore, the major challenge in XMLC is to capture these rare or unseen

Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020)
EMAIL: kathryn.chapman@dfki.de (K. Chapman); guenter.neumann@dfki.de (G. Neumann)
URL: https://github.com/kathrynchapman/CANTEMIST2020 (K. Chapman)
ORCID:

Split	# Docs	# Unique Codes	# Unseen Codes
Train	501	493	-
Dev1	249	338	208
Dev2	250	334	203
Test	300	386	164 (train), 107 (+devs)

Table 1
Number of documents, unique codes, and unseen codes per data split. '# Unseen codes' refers to codes which are not present in training data split.

labels, and develop a classifier which can detect and assign them. Lastly, in the medical field, ICD codes are assigned to an EHR in descending order of importance, as these codes are used for billing purposes. To our knowledge, this code ranking has not been a major focus in past automatic ICD code assignment research, which has mostly sought to develop systems which can correctly predict the codes as an unordered set. However, this code ranking aspect will need to be addressed if these automatic classifiers are expected to be used in a real-world application. In the CANTEMIST 2020 task, track 3, participants were expected to submit model predictions in this ranked order. The submissions were evaluated using the mean average precision (MAP) metric. In this metric, if all predictions for a given instance are correct, their ordering does not matter. However, an incorrect prediction occurring high on a ranked list or before other correct predictions causes the MAP to decrease.

2. Related Work

Until recently, research on automatic ICD code assignment has focused mostly on classic machine learning architectures. Many researchers have exploited external ICD code information, such as [3], where they incorporated the ICD code heirarchy in their SVM model which yielded an F1 increase from 27.6% to 39.5%. With the recent boom in deep learning, research has shifted from classic machine learning models to neural architectures. [4] evaluated several neural network based classifiers in assigning ICD-10 codes to non-technical summaries of animal experiments. They found that a BERT-based classifier outperformed the other models, achieving an F1-micro score of 80.82%. Other recent work has focused on incorporating external resources to aid in automatic ICD code assignment. [5] developed the Convolutional Attention for Multi-Label classification (CAML) model, which used the ICD code textual descriptions to identify spans of text within a document which triggered the assignment of a given code. This model architecture increased performance on the 'low-resource' codes, those which have few positive training instances. [6] developed the Label-Specific Attention Network (LSAN), a general-purpose multi-label text classification architecture which utilizes label vectors generated from textual label descriptions and computes attentions scores between each label vector and each word in a document. [7] created a framework which generates pseudo features from the ICD code textual descriptions, and achieved an increase in F1 score from nearly 0 to 20.91% for the codes not seen during training, referred to as 'zero-shot codes'.

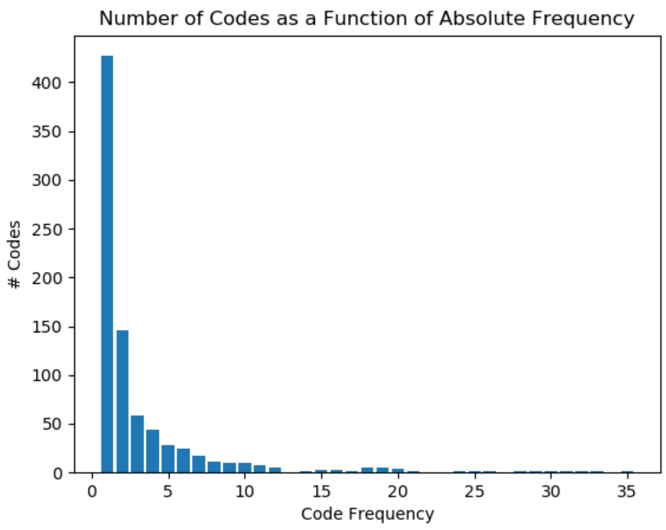


Figure 1: Number of codes as a function of absolute frequency. For example, 427 codes occur with 1 document, 146 codes occur with 2 documents, and so on. Truncated on x-axis to top 30 absolute frequencies.

3. Data

The data for the CANTEMIST 2020 - Coding Track consists of a training set of 501 documents, two development sets consisting of 249 and 250 documents, and a test set of 300 documents. Across the training and development splits, the average number of whitespace-split tokens per document is 738. Each document is assigned one or more eCIE-O-3.1 (English version: ICD-O-3) codes, with an average of 5.4 codes per document across train, dev, and test data. Across all datasets, there are 850 unique eCIE-O-3.1 codes, many of which are present only in the test split (see Table 1). Additionally, the two most frequent codes are assigned to the majority of the documents, followed by a rapid decrease in code frequency (Table 2). This demonstrates the aforementioned 'tail end problem' in XMLC, and is further illustrated in Figure 1, which shows that across the training, dev, and test data, 427 of the 850 unique codes are only assigned to a single document, 146 are only assigned to two documents, etc. This means that over half of all codes present in the entirety of the data set only have a single positive training instance, and 67% of the 850 codes have only 1-2 positive training instances, etc. Meanwhile, the most frequent code, 8000/6, is assigned to 1000+ documents across all data splits. This constitutes an extreme label imbalance.

Code	Train + Devs	Test
8000/6	89%	83%
8000/1	76%	74%
8000/3	40%	42%
8140/3	22%	21%
8010/3	14%	9%
8140/6	13%	9%
8500/3	10%	7%
8001/1	9%	7%
8001/3	8%	7%

Table 2
Top 9 codes and percentage of positive instances across the train (including dev) and test splits

3.1. Additional Data

Additional to the data provided by the task organizers, we downloaded textual descriptions corresponding to each eCIE-O-3.1 code[1]. Figure 2 shows how the numerical composition of each code is informative, as the first four digits specify a cell type, the fifth digit a behavior (benign, malignant, etc), and the sixth digit a grade (well differentiated, poorly differentiated, etc). A small number of 6-digit codes present in the provided datasets lacked descriptions, however. This means that we obtained text descriptions for all codes which specify cell type and behavior, but were missing some for a few codes which specify cell type, behavior, and grade. We therefore generated our own descriptions by taking the description corresponding to the first five digits of that code (cell type + behavior), and simply appending text which corresponds to the final digit, or the grade. There are only four possible digits to specify the grade, and therefore only four grade descriptions[2].

4. Methods

4.1. Models

Baseline Our baseline model utilized a BERT encoder [8] followed by a classification layer. The encoder passes the pooled output of the input sequence to the classification layer, which outputs a binary matrix representing the presence or absence of a given code.
Label Description Attention The authors in [6] generated a vector for each label from their corresponding textual descriptions and calculated attention scores between each label vector and word in a document. Building off of this, we developed a label description attention component of our model which calculates attention scores at the word level. Our label attention model (Appendix A) employs a BERT or BERT-like encoder which it applies both to the input text from the documents, as well as the textual descriptions of the labels (described in 3.1). This model utilizes the sequence output, rather than pooled output, of both of these input texts,

[1]https://eciemaps.mscbs.gob.es/ecieMaps/documentation/documentation.html, version 3.0, last edited 01.01.2018, accessed 28.07.2020
[2]http://www.sld.cu/galerias/pdf/sitios/dne/vol1_morfologia_tumores.pdf

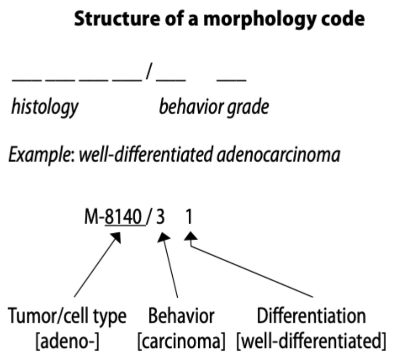

Structure of a morphology code
___ ___ ___ ___ / ___ ___
histology
behavior grade
Example: well-differentiated adenocarcinoma
M-8140 / 3 1
Tumor/cell type [adeno-]
Behavior [carcinoma]
Differentiation [well-differentiated]

Figure 2: Structure of the ICD-O-3 codes. [9]

calculating attention scores between each word in each input document, and each word in each label description. To the best of our knowledge, this is the first time word-level attention has been applied between textual label descriptions and document text in multi label classification.

4.2. Document Batching

A drawback of BERT and BERT-like models is that they can only accept input up to a pre-defined maximum sequence length, due to memory constraints regarding the way they calculate self-attention. Additionally, this maximum sequence length applies to the number of tokens *after* the subword tokenization is applied, meaning that a sequence may have fewer than 510 whitespace tokens, but many more when tokenized according to the BERT-like models' sub-word tokenization scheme. As the average number of whitespace tokens in the CANTEMIST data sets is around ~768 per document, we felt it important to utilize as much of each document as possible. Therefore, we developed a model component we have designated as 'document-batching'. This component fits each document into varying-size batches, such that each batch is only one document, and uses a 'sliding window' so that there is an overlap between sequences in a batch in order to preserve context. We then perform max-pooling over the batch dimension of the logits, and pass the resulting vector to our loss function, or to make predictions.

5. Experiments

All experiments were conducted using the provided train data split as our training data, and the dev1 as our evaluation data. We combined the baseline model (simple encoder plus classification layer) and the label attention model with the three BERT and BERT-like models in 5.1, as well as with/without the document batching component.

5.1. BERT Models

We utilized the publicly available HuggingFace[3] implementations of the following models [10]. All experiments employ the following hyperparameters unless specified otherwise: document maximum sequence length 200, label maximum sequence length 15, 45 epochs, per-gpu batch size 8, and the remaining parameters default. All experiments were run across 4 GeForce GTX TITAN X GPUs.

Multilingual BERT We used the pre-trained BERT-Base-Multilingual-Cased encoder (ML BERT) in our experiments.

XLM-RoBERTa [11] introduced XLM-RoBERTa (XLM-R), which is nearly identical to the RoBERTa [12] architecture except it is extended to a multi-lingual setting. XLM-R has outperformed multi-lingual BERT on various benchmark tasks, thereby motivating our decision to explore this model. We used a XLMR-base-cased model.

Further Pretrained XLM-R We additionally further pretrained an XLM-R-base-cased model (FP XLM-R) using the masked language modeling objective on all of the text from the training, dev, test, and background data releases. Next, we fine-tuned the model exactly as we did with the BERT and 'out of the box' XLM-R models.

5.2. Document Batching

We ran experiments both with and without the document batching component. When we included it, we set the maximum per-gpu batch size to 10 and used an overlap of size 50. This means that each next sequence in a batch contained the last 50 tokens of the previous sequence in the batch.

5.3. Code Ordering

As the task organizers expected a ranked list of code predictions, we simply ordered the labels by their confidence scores.

5.4. Loss Functions

5.4.1. Binary Cross Entropy

We used the PyTorch loss function `BCEWithLogitsLoss`. We observed however that this loss function was resulting in very high precision and very low recall. We investigated the predictions and found that the classifier was predicting only a few codes from the entirety of the label space, and achieving what appeared to be a competitive performance. For example, using this loss function with the baseline classifier plus document batching resulted in a MAP of 0.331, F1 of 0.491, precision of 0.855, and recall of 0.343. Despite the deceiving metric scores, this classifier was assigning the two most frequent codes to every document. We observed this across several experiments and also with the label attention model. In ICD code classification, recall is arguably slightly more important than precision, as the precision can be very high if the classifier simply assigns the most frequent codes to every document. However, detecting

[3]https://github.com/huggingface/transformers, version 2.11.0

a)

Model	MAP	F1	P	R	Avg # Labels/Doc
Label Attn	44%	**28%**	**20.9%**	57.4%	**19**
Baseline	**51.7%**	14.4%	8%	**76.1%**	48.1

b)

Document Batching	MAP	F1	P	R	Avg # Labels/Doc
✓	48.3%	**21.3%**	**14.3%**	66.9%	**31.28**
✗	**48.9%**	18.4%	12%	**70.2%**	41.7

c)

BERT	MAP	F1	P	R	Avg # Labels/Doc
ML BERT	44.6%	25%	**18.8%**	60.4%	**27.6**
XLM-R	**51.7%**	14.9%	8.3%	**75.8%**	46
FP XLM-R	51%	**17.2%**	10%	73.2%	40.6

Table 3
Average MAP, F1, Precision, and Recall scores of the (a) label attention component vs. no label attention, (b) document batching vs. no document batching, and (c) the different BERT-like models from our pre-submission experiments.

the 'rare' codes is the real challenge in ICD classification. Therefore, we opted for a modified `BCEWithLogitsLoss` loss function.

5.4.2. Balanced Binary Cross Entropy

Building off of PyTorch's `BCEWithLogitsLoss`, we created a loss function which calculates positive class weights based on the ratio of negative classes per example in a batch. For example, if there are 5 negative labels for every 1 positive label, each positive label for that example receives a weight of 5. This appropriately addressed the problems with the basic `BCEWithLogitsLoss` loss function, in that we saw a substantial increase in recall, albeit at the expense of the precision.

5.5. Results and Discussion

Label Attention vs Baseline Table 3 shows that while the baseline achieves on average a higher MAP score, the precision and recall are far less balanced and many more labels are predicted per document. When looking into these labels, the baseline predicts only on average (across all experiments) 71.8 unique labels across all predictions on the dev1 set, whereas the label attention model predicts on average 81.75 unique labels across the dev1 set. We therefore feel that even if the MAP metric is lower, the label attention model produces a more varied set of predictions. Additionally, in all of the twelve experiments conducted (six of which included the baseline), there was only one time that the baseline classifier did not assign the same codes to every single document. Conversely, we observed that the label attention model only assigned all of the same codes to every document once. However, the label attention model plus ML BERT only assigned 32 and 30 unique codes across the entire dev1 set with and without document

	# Codes Removed											
	None				Top 2				Top 9			
Model	MAP	F1	P	R	MAP	F1	P	R	MAP	F1	P	R
Label Attn	50.2	**24%**	**15%**	60%	23.3	**13%**	**8%**	42%	6.2	**6%**	**3%**	17%
Baseline	**51.6**	16%	9%	**65%**	**25.1**	9%	5%	**50%**	**7.7**	5%	2%	**29%**

Table 4
Pre-submission model performances when n most frequent codes are removed.

batching respectively, but even then, the models assigned only 6.9 and 7.98 on average to each document. This, we feel, is much better than assigning 45 and 52 unique labels, but assigning all of them to every document, as we observed with the baseline classifier with ML BERT with/without document batching respectively. Lastly, we evaluated model performance when removing the top 2 most frequent labels, as they occur in over half of the documents, and the top 9 most frequent codes (see table 4). Both models were trained on the training set, using document batching and evaluated on dev1. The label attention model outperforms the baseline with respect to F1 and precision, but fails regarding MAP and recall.
Document Batching The average MAP scores with respect to document batching are nearly identical across the experiments, but the F1 are higher when this is used due to a slightly more balanced P/R. While we expected to see more of a performance boost with the document batching, but its inclusion does help to make up for the precision decrease from our loss function.
BERT & Friends There does not seem to be a clear winner regarding the type of BERT-like encoder, as shown in table 3c. However, we explored the number of unique codes assigned during eval on the dev1 set, and found that the FP XLM-R is the clear winner when it comes to variety of codes assigned. On average, the FP XLM-R model predicted 126.8 unique codes, as opposed to 39.8 and 46 for the ML BERT and XLM-R models respectively. This means that more rare codes were predicted, which is what we hoped to see.

Finally, we feel that models which can predict on average fewer labels per document and still be competitive are learning more than the models which predict more labels per document on average. While recall is perhaps slightly more important than precision here, models which continuously predict 50 labels per document are not necessarily achieving high recall by catching the 'rare' labels; if the models are predicting 50 codes per document and only predicting 52 unique codes, this is not a very well-informed model. Therefore, high recall is only beneficial if the average number of labels predicted per document is substantially lower than the number of unique labels predicted. Otherwise, the model is repeatedly assigning most or all of the top most frequent codes, and presumably failing to capture the rarer codes.

5.6. Submission and Performance

Our submitted model predictions were generated from a further pretrained XLM-R base model, trained for 45 epochs on a concatenation of the training and dev1 data, with a document maximum sequence length of 200, a label maximum sequence length of 11, document batching, and the balanced binary cross entropy loss function. Since we trained the model on the concatena-

Model, Doc MSL, Label MSL	MAP	F1	P	R	Avg #Codes/Doc	# Unique Codes
Final Submission, 200, 11	39.4	**26.8**	**18.2**	51	**15**	**178**
Label Attention, 150, 15	**48.2**	9.9	5.3	**67.2**	70.5	76
No Label Attention, 200, -	48	12.4	6.8	64.6	52	52

Table 5
Comparison of model performances between the submitted model and other identical models, differing only where specified. 'MSL' refers to 'maximum sequence length'.

tion of the training and dev1 data, 130 of the 386 labels present in the test data were unseen during training, leaving 256 unique labels in the test data which our model had been trained on. Table 5 shows our model performance according to various metrics.

5.7. Post-Submission Experiments

When developing our model, we found that setting document maximum sequence length to 200, label maximum sequence length to 15, while using the document batching component with a maximum batch size of 10 yielded promising results. However, when training the final model on the concatenation of the training and dev1 splits, this meant that our label space increased from 493 to 623. As all of the label descriptions are re-encoded in every forward pass, this caused memory issues and we had to decrease the label maximum sequence length and 11 in order to fit all label descriptions into the memory. To probe what effect this had on the model's performance on the test set, we trained another model with maximum sequence lengths of 150 (document) and 15 (label). The resulting model achieved an MAP of 0.48, which surpasses our submitted model's MAP but performs worse in F1 micro and precision. Additionally, this model only predicts 76 unique codes, and assigns on average 70.5 codes per document. Recall that across all data splits, the average number of codes per document is 5.4.

Additionally, the post-submission experimental model's top codes predictions almost entirely lack variation across all 300 test documents. When treating model predictions as an unordered set, the top 15 predictions are identical across all test documents (with only two distinct orderings). Similarly with the top 25 codes, we see only 2 distinct code set predictions and 9 distinct orderings. In summation, this model appears to perform better than our submission by the MAP metric alone, but if the top 15 (or 25) code predictions across all documents are nearly identical, it seems that the model is over-assigning frequent codes, and perhaps the lower-ranking predictions are more informative, as we observed much greater variation across these 'bottom' codes predictions.

We also trained a model identical to our submitted model, just without the label attention component. Table 5 shows that without the label attention component (essentially, our baseline model plus document batching), the model predicts the same 52 codes for all documents, in exactly the same order, despite achieving a higher MAP score.

6. Conclusion

Automatic ICD code classification is not only challenging, but evaluating the quality of such a classifier is also a daunting task. This is because models can trick the evaluation metrics, while in the end producing useless predictions. We feel that it is important to not only evaluate a model by it performance metric, but to also explore the predictions: How many unique labels are predicted? Are only the most frequent labels predicted? What is the ratio of total unique labels predicted to the average number of labels assigned to a document? These are important questions to ask if the ultimate goal is to develop a system which can function in real-world applications.

Our label description attention model did not always win in our pre-submission experiments, but it did consistently produce more varied predictions, a larger set of unique labels, and fewer codes per document, all while maintaining high recall and a competitive F1 score. Our motivation to use our chosen model in the final submission as opposed to another which produces a higher MAP score, is that we sought to increase performance on the low-resource codes without assigning too many codes to each input document, thereby maintaining precision. A drawback of our model is its computational expense: our current architecture requires re-encoding all label descriptions in every forward pass, as the encoder parameters are updated during training. Future work could focus on finding a more efficient way to embed the label descriptions, as our architecture would be extremely computationally expensive in a larger label space. We planned to use higher capacity models in our experiments and submission, but ran into memory issues; this could also make for interesting future work, as such large label spaces could benefit from this increase in capacity. Furthermore, exploiting the ICD code hierarchy, which is encoded in the composition of the codes themselves (figure 2), and incorporating this into the label description embeddings could make for interesting future work.

Acknowledgments

Special thanks to Saadullah Amin and the DeepLee[4] team for guidance and helpful discussion. This work was partially funded by European Union's Horizon 2020 research and innovation program under grant agreement No. 777107, through the project Precise4Q[5].

References

[1] A. Miranda-Escalada, E. Farré, M. Krallinger, Named entity recognition, concept normalization and clinical coding: Overview of the cantemist track for cancer text mining in spanish, corpus, guidelines, methods and results, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020), CEUR Workshop Proceedings, 2020.

[2] R. Babbar, B. Schölkopf, Data scarcity, robustness and extreme multi-label classification, Machine Learning (2019) 1–23.

[4]https://www.dfki.de/en/web/research/projects-and-publications/projects-overview/project/deeplee/

[5]https://www.dfki.de/en/web/research/projects-and-publications/projects-overview/project/precise4q/

[3] A. J. Perotte, R. Pivovarov, K. Natarajan, N. Weiskopf, F. D. Wood, N. Elhadad, Diagnosis code assignment: models and evaluation metrics, in: JAMIA, 2014.

[4] S. Amin, G. Neumann, K. Dunfield, A. Vechkaeva, K. Chapman, M. K. Wixted, Mlt-dfki at clef ehealth 2019: Multi-label classification of icd-10 codes with bert, in: CLEF, 2019.

[5] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, J. Eisenstein, Explainable prediction of medical codes from clinical text, 2018, pp. 1101–1111. doi:10.18653/v1/N18-1100.

[6] X. Huang, B. Chen, L. Xiao, L. Jing, Label-aware document representation via hybrid attention for extreme multi-label text classification, 2019. arXiv:1905.10070.

[7] C. Song, S. Zhang, N. Sadoughi, P. Xie, E. Xing, Generalized zero-shot icd coding, 2019. arXiv:1909.13154.

[8] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. arXiv:1810.04805.

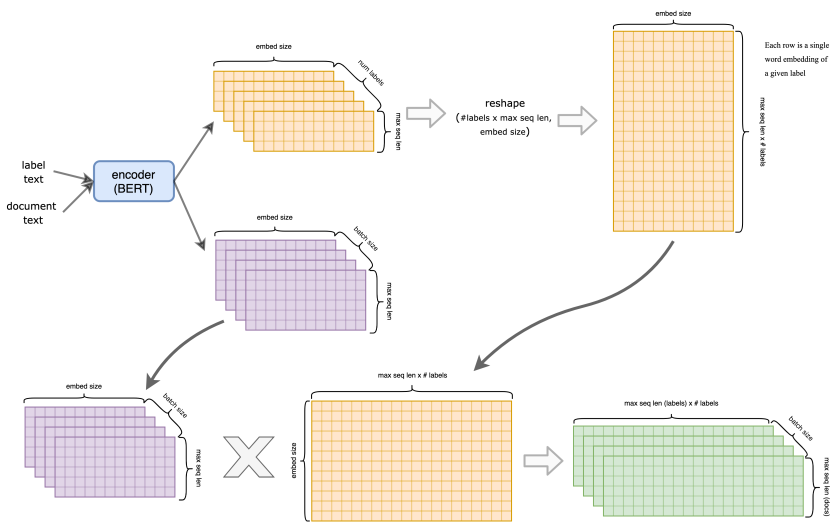
[9] W. H. Organization, et al., International classification of diseases for oncology (icd-o)–3rd edition, 1st revision (2013).

[10] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Brew, Huggingface's transformers: State-of-the-art natural language processing, ArXiv abs/1910.03771 (2019).

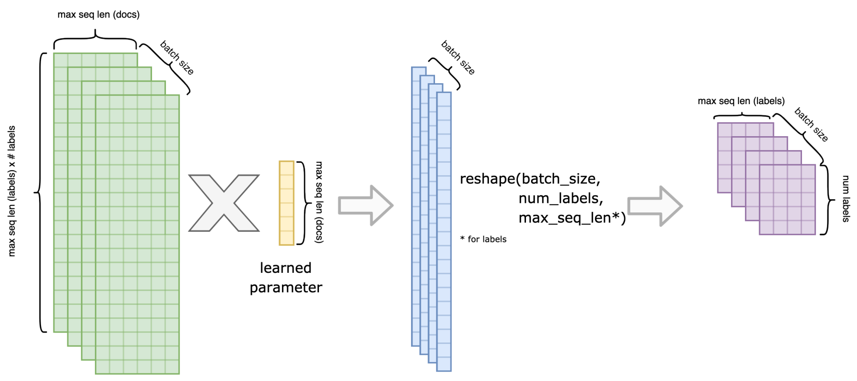
[11] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised cross-lingual representation learning at scale, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (2020). URL: http://dx.doi.org/10.18653/v1/2020.acl-main.747. doi:10.18653/v1/2020.acl-main.747.

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. arXiv:1907.11692.

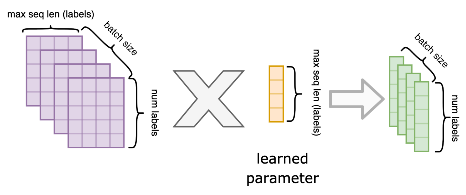
A. Label Description Attention Model



The resulting matrices are one per input document in the batch. Each row correspond to a word in the document, and each column corresponds to a word in a label description. If the maximum sequence length for label descriptions is 3, then three-column 'chunks' correspond to a label, each column in the chunk representing a sub-word token.



Now, we have batch_size number of vectors, each representing a document.


max seq len (labels)
batch size
num labels
max seq len (labels)
learned parameter
batch size
num labels