Proceedings of the LREC2020 Industry Track , pages 26–30
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

Hybrid Tagger – An Industry-driven Solution for Extreme Multi-label Text Classification

Kristiina Vaik, Marit Asula, Raul Sirel
TEXTA OÜ
kristiina@texta.ee, marit@texta.ee, raul@texta.ee

Abstract

This paper presents an industry-driven solution for extreme multi-label classification with a massive label collection. The proposed approach incorporates a large number of binary classification models with label pre-filtering and employs methods and technologies shown to be applicable in industrial scenarios where high-end computational hardware is limited. The system is evaluated on an Estonian newspaper article dataset which contains almost 2000 unique labels and has shown to perform over 80 times faster than applying all the binary models of the entire label set without negative impact on prediction scores.

Keywords: text classification, extreme multi-label classification, data processing workflows

1. Introduction

The interest of automating certain tasks in the news industry and other sectors dealing with large volumes of textual data has been growing rapidly. Today these tasks often rely on human's judgement and manual assignment which consumes a lot of resources. However, the market has shown a higher demand on using already existing data and NLP technologies efficiently. One of such tasks is extreme multi-label text classification, i.e automatically assigning a list of most relevant labels based on the content of the text from a large label set. In the industry multi-label text classification can be used for many different applications, such as describing the subject of a news article by assigning a general topic (e.g. *politics*, *history*, *sports*, etc.) or add specific keywords (e.g. *NATO*, *Donald Trump*, etc.) based on the relevant content. Such scenarios produce an increasingly large number of labels to predict making it a challenge for extreme multi-label classification settings to synchronously handle massive label sets.

We propose an industry-driven solution, *Hybrid Tagger* (HT), for extreme multi-label classification combining supervised and unsupervised text processing methods which have been proven to be applicable in the industrial settings. HT is part of TEXTA Toolkit[1] – an open-source framework for building and executing machine learning pipelines and analysing textual content. HT categorizes each given text in real-time with the most relevant labels from a massive label collection. The development of the HT was motivated by the need to perform classification with a high volume of labels and the lack of existing out-of-box solutions. Another issue in industrial scenarios is the limited availability of high-end computational hardware because clients often require running production-grade applications using their own infrastructure. Hence HT was designed to work with fairly limited computational power regarding today's standards. HT is currently used by two Estonian media corporations to label newspaper articles with topics & keywords and the National Library of Estonia to label books, dissertations, periodicals, etc.

The paper is further structured as follows: Section 2. gives a brief overview of the existing research on this topic, Section 3. provides an overview of the workflow used in Hybrid Tagger, Section 4. describes the results of applying the Hybrid Tagger on a dataset of newspaper articles, and Section 5. concludes the paper.

[1] https://docs.texta.ee/

2. Related Research

Multi-class classification is a type of problem where an input is assigned with a single label from a finite set of labels. However, in real-life scenarios, the content of the text is semantically far more variable, which means it most probably contains many different topics as opposed to a binary topic distribution. This limitation leads to the *multi-labelled classification* problem in which a subset of labels is assigned to the input object from a finite set of labels (Tsoumakas and Katakis, 2009; Godbole and Sarawagi, 2004).

A common approach to solve the multi-label classification problem is the *problem transformation*, specifically the *binary relevance method* (Tsoumakas and Katakis, 2009; Godbole and Sarawagi, 2004; Zhang et al., 2017a) in which the multi-label problem will be split into binary classification subtasks. After splitting the task into binary subtasks, these binary classifiers will be converted into the multi-label representation meaning that n binary classifiers are trained whereas each classifier gives a prediction of 0 or 1, i.e giving a corresponding label from a finite set of labels. Binary relevance method has often been overlooked because of the assumption that it ignores the correlations between labels, meaning that in real life it will likely give too many or too few labels. However, as Read et al. (2009) point out that if the multi-labelled datasets grow in size, methods taking label correlations into account struggle with the exponential growth of the possible correlations.

On the industry level scalability in text processing is a very important factor, although the number of labels to predict is often disregarded. Existing extreme multi-label classification papers use tree-based methods (Agrawal et al., 2013; Weston et al., 2013; Prabhu and Varma, 2014) or reduce the dimensions of the original label matrix, e.g. Bi and Kwok (2013) reduced the number of labels by doing random sampling; Zhou et al. (2012) have proposed a method called

compressed labelling which compresses the original label matrix in order to reduce the number of binary classifiers. Both of these methods can be generalized as two-stage approaches depending on complex matrix computations to find a subset of most probable labels (stage one) and to apply the corresponding binary classifiers (stage two). Zhang et al. (2017b) have applied deep learning by establishing a non-linear embedding for both feature and label spaces and combine it with a label graph, which is built from label space (two nodes share an edge if corresponding labels co-occur frequently enough). However, none of the referenced works is applicable in the industry because of the computational, infrastructural, and (labelled) resource limitations.

3. Workflow of Hybrid Tagger

Hybrid Tagger incorporates a high number of binary classification models combined with unsupervised label pre-filtering in order to achieve real-time predictions for thousands of labels. HT uses traditional classification algorithms because neural network classifiers require more training data per label to provide adequate results.

The supervised part of HT has been developed by using scikit-learn (Pedregosa et al., 2011), primarily logistic regression or SVM algorithms. The unsupervised pre-filtering is achieved by using Elasticsearch[2] engine's document retrieval features where all texts used for training and validating the classification models are indexed. Elasticsearch is used because of its stable and scalable platform for retrieving documents (for training) and performing document similarity queries for the label pre-filtering. In contrast to existing extreme multi-label classification solutions, Elasticsearch allows to disregard complex matrix computations and instead offers a fairly transparent way to filter labels based on the document similarity they have been assigned to. In this section, the workflow pipeline of HT will be described in detail.

3.1. Preprocessing

The preprocessing pipeline consists of tokenization, lemmatization, part of speech (POS) tagging, and named entity recognition (NER). This is done by using TEXTA Multilingual Processor (MLP) which uses NLTK (Bird et al., 2009) with Stanford models and EstNLTK (Orasmaa et al., 2016). Preprocessing pipeline can be omitted, but it depends on the language and domain HT is applied to. For example, morphologically less inflective languages do not necessarily require to be lemmatized to provide adequate features, whilst for other languages (e.g. Estonian) it is crucial. It decreases the size of the vocabulary, thus model size and computation time. NER can be used for speeding up the prediction process.

3.2. Training

Binary classifiers used in HT can be trained on any text segment, e.g. title, content, author, etc. In our experimental setup models are usually trained using lemmatized content and also optional POS tags. For vocabulary reduction stop words are removed from all texts prior to training.

[2] `https://www.elastic.co/elasticsearch`

Training data is selected according to the pre-existing labelling. For each label, all existing *positive* examples (texts annotated with the specific label) and the same amount of randomly selected *negative* examples (texts not annotated with the specific label) are retrieved. Examples for all labels are then randomly split into training and validation sets (default 80-20).

In real-life scenarios, the training process may result in thousands of classification models which have a significant memory imprint when combined. To combat this problem, HashingVectorizer (Pedregosa et al., 2011) is used to vectorize the training data. It has significantly smaller memory imprint than other vectorizers supported by scikit-learn (e.g. commonly used TfIdfVectorizer) as it uses the hashing trick to find the token string name to feature integer index mapping.

For training models 5-fold cross-validation and grid search is used to find the most optimal parameters for the C-value; minimum and maximum length of *n*-grams used in the model; and whether to use words (and word-based *n*-grams) or word-bound character-based *n*-grams as features. The best model from grid search is then validated using the validation set. For each model, a confusion matrix with precision, recall, and F1 score are computed.

3.3. Prediction

The training phase's aftermath is a large volume of binary models making it infeasible to apply them for label prediction in a sequential manner. This limitation is handled by devising an approach in which the number of executable binary models are limited by finding a subset of models which most likely will provide an accurate label prediction. This is done by finding n (default n=10) similar texts from the training data indexed in Elasticsearch and finding m (default m=10) most frequent labels assigned to the texts. Retrieving similar texts is done by using an Elasticsearch *more like this* query[3] which calculates top k words with the highest TF-IDF score per text and afterwards performs a disjunctive query using the pre-existing labels to match similar texts.

In general, the prediction pipeline is as follows:

- preprocess the input text (optional);
- find n similar texts indexed in Elasticsearch;
- find top m labels attached to the texts found in the previous step;
- apply named entity recognition on the input to identify l entity-related labels and remove these binary models from the list of m models which will be used later for label prediction;
- retrieve all models for each of m-l top labels;
- apply models, retrieve and combine the list of predicted labels classified by the binary models and the entity-related labels, and output the results.

[3] `https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html`

In the presented pipeline both Elasticsearch-based label pre-filtering and NER-based labelling are used to reduce the number of binary classification models applied to each text. We acknowledge that the effectiveness of NER greatly depends on the specific domain and label set, nevertheless, in our use cases, it has proven to be an effective method to reduce the number binary models required.

4. Evaluation

The purpose of the evaluation is to show that Hybrid Tagger performs significantly faster than the baseline model without negative impact on prediction scores.
First, we provide a brief theoretical overview of how the applicability of Hybrid Tagger depends on the available computation power and number of labels present in the dataset. Then we evaluate Hybrid Tagger's performance on a real multi-label dataset.

4.1. Hybrid Tagger's Applicability

Let n_t be the number of classifiers used for prediction, n_c be the number of cores used for computation, μ_t the average prediction time of one binary classifier and t_{mlt_s} the time of Elasticsearch's *more like this* query with number of similar documents set to s. The approximate simplified formula for time consumption of tagging one document is thus:

$$t_{tag} = \left\lceil \frac{n_t}{n_c} \right\rceil \cdot \mu_t + t_{mlt_s} \quad (1)$$

However, it is important to note that *more like this* query does not have a significant impact on time consumption in most practical scenarios as s is usually set in the range between 10 and 100 resulting in query time under 1 second (except for datasets containing very long texts, e.g. books). Let n be the total number of labels in the dataset and m be the number of top labels used for prediction by Hybrid Tagger. The gain in time efficiency (i.e. how many times faster are HT predictions compared to applying all binary classifiers) depending on number of available cores and size of the label set, can be calculated by the following formula:

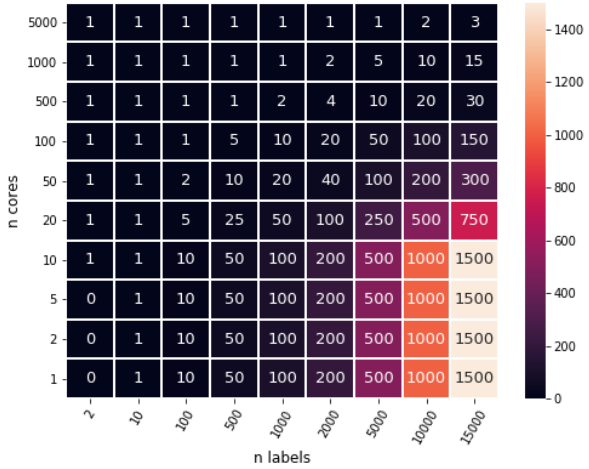
$$HT_{gain} = \left(\left\lceil \frac{n}{n_c} \right\rceil \div \left\lceil \frac{m}{n_c} \right\rceil \right) - t_{mlt_s} \quad (2)$$

Figure 1 illustrates how the gain in time efficiency with HT changes based on the number of available cores and the number of labels with m set to 10 and t_{mlt_s} set to 0 for simplification. We can see that the gain in time with HT grows as the label set increases while the number of available cores stays relatively low. For example, applying HT with original label set size of 10 000 on a machine with 5 available cores is 1000× faster than applying all binary classifiers.

4.2. Case Study: Õhtuleht Newspaper Dataset

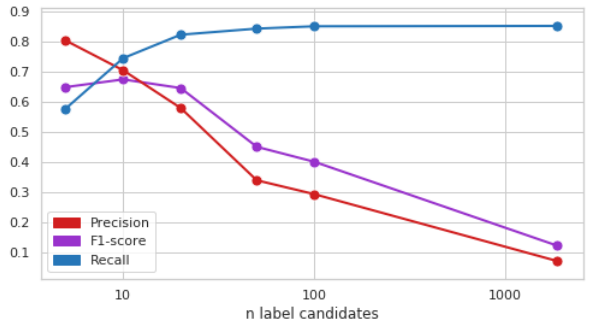
Õhtuleht dataset contains newspaper articles spanning from years 2013 to 2019, covering a wide range of topics (news, sports, entertainment, crime, etc). The dataset is not publicly available because of legal limitations. It contains 102

Figure 1: HT gain in time efficiency, if $m = 10$



450 documents with 1978 unique labels. The average number of labels per document is 4.4.
For evaluating HT, Õhtuleht dataset is split into train and test set (100 000 and 2450, respectively), and trained 1870 binary classifiers on lemmatized articles' content using logistic regression as the predictor function. The minimum number of examples per each label was set to 50, resulting in disqualifying 108 labels with a smaller number of examples. For choosing the best parameter configuration for HT, we measured average precision, recall and f1-score on the test data with (*n label candidates*, *n similar texts*) set to (5,10), (10,10), (20,10), (50,30) and (100,30). Figure 2 illustrates these results by showing how the prediction scores change when the size of the label candidate set increases. We see that precision starts decreasing after 10 candidate labels while recall stabilizes after 100. The best trade-off between precision and recall is obtained at 10 resulting in the highest f1-score.

Figure 2: Label candidates set size effect on prediction scores



To further evaluate the results, we measured average prediction time, precision, recall, f1-score and the number of predicted labels on the test data by applying:

1. Baseline model (BL) consisting of all binary classifiers;
2. Hybrid Tagger with the best detected parameter configuration (*n label candidates = 10*, *n similar texts =*

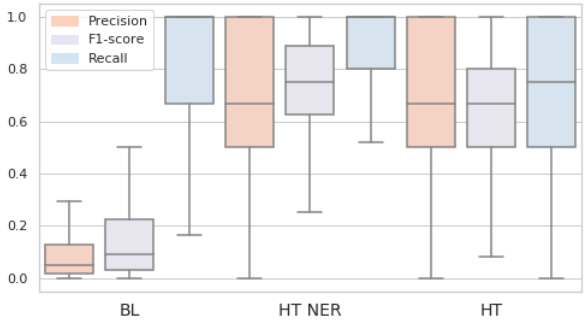
10) with NER enabled (HT NER) and NER disabled (HT).

Table 1: Comparing Baseline with Hybrid Tagger

	BL	HT NER	HT
n taggers	1870	10	10
n similar texts	nan	10	10
n cores	24	24	24
NER enabled	no	yes	no
Time (s)	82.34	1.01	1.01
Precision	0.07	0.70	0.71
Recall	0.85	0.92	0.75
F1-score	0.12	0.76	0.67
n predicted labels	128.87	6.21	4.89

Table 1 gives an overview of Hybrid Tagger's performance compared with the baseline model (BL). Figure 3 visualizes the prediction scores' distribution of the same models, while the times' distribution can be seen on Figure 4 and the distribution of predicted labels per one document on Figure 5.

Figure 3: Prediction precision, f1 and recall scores



To verify that our experimental results are in accordance with theory, we can calculate the theoretical gain in time with our HT configuration by inserting the chosen parameter combination into formula (2) and compare it with the measured result. Thus, for theoretical gain in time we get: $HT_{gain} = \left(\left\lceil \frac{1870}{24} \right\rceil \div \left\lceil \frac{10}{24} \right\rceil\right) = 78\times$.

BL model's actual average prediction time is 82.34 seconds while HT labels one document on average with 1.01 seconds resulting in an actual gain of $81.5\times$. The minor difference in magnitude can be caused by practical reasons, e.g. small variations of prediction times of individual taggers (the slower models are always included in the BL label set, while might not be in HT label candidates). Nonetheless, our experimental results prove that HT performs significantly faster than the BL model.

Furthermore, BL's f1-score is only 0.12 as a result of extreme over labelling causing a very low precision of 0.07. We can see from Figure 5 that the number of labels predicted with BL is close to 100, while the actual number of labels seldom passes 10. As HT limits the number of label candidates, it does not suffer from the same problem and precision remains 0.7 both with and without NER (see Figure 3). HT's recall without NER is 0.75 being slightly lower than BL's average of 0.85. However, HT with NER obtains even better recall than the BL model with an average score of 0.92. It is still important to keep in mind that the applicability of NER is dataset-specific as using entity-related labels is fairly common with newspaper articles and enabling NER helps to combat the problem of low number of training examples of such labels, but might make the model prone to false positives in some other domains.

Figure 4: Prediction times in seconds

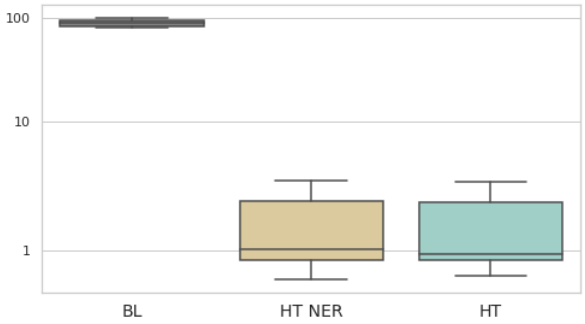
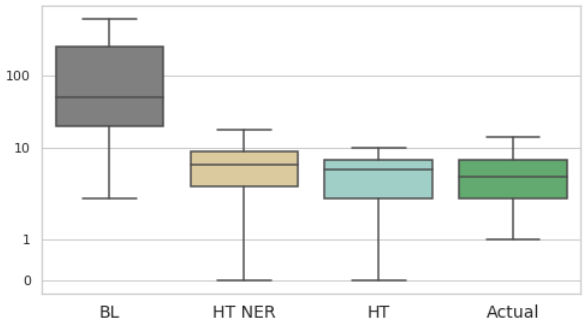


Figure 5: Number of predicted labels per document



5. Conclusion

This paper presented an industry-driven solution, Hybrid Tagger, for extreme multi-label classification with a large volume of unique labels. The proposed system incorporates a high number of binary classification models coupled with unsupervised label pre-filtering and named entity recognition to achieve real-time predictions with thousands of labels. As the development of Hybrid Tagger was industry-driven, it does not employ state-of-art methods but rather relies on methods and technologies proven to work on an industrial scale.

The evaluation of Hybrid Tagger on Õhtuleht newspaper dataset shows that Hybrid Tagger helps to significantly improve both the prediction times and precision scores in comparison to executing all binary classification models of the label set.

Hybrid Tagger is currently used by two Estonian newspaper corporations and Estonian National Library to label their content. Since the workflow of HT is language independent, the next step is to apply the solution for industrial projects in other languages.

6. Acknowledgements

The work described in this paper has been supported by the language technology research and development program "Estonian Language Technology 2018–2027" of the Ministry of Education and Research under grant EKTR3, by European Union's Horizon 2020 research and innovation programme under grant agreement No 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media) and Enterprise Estonia project No. EU48684, Research Project No. 1.11 (Deep neural models and cross-lingual embeddings in TEXTA Toolkit).

7. Bibliographical References

Agrawal, R., Gupta, A., Prabhu, Y., and Varma, M. (2013). Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. pages 13–24, 05.

Bi, W. and Kwok, J. (2013). Efficient multi-label classification with many labels. *ICML*, pages 405–413, 01.

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In Honghua Dai, et al., editors, *Advances in Knowledge Discovery and Data Mining*, pages 22–30, Berlin, Heidelberg. Springer Berlin Heidelberg.

Orasmaa, S., Petmanson, T., Tkachenko, A., Laur, S., and Kaalep, H.-J. (2016). Estnltk - nlp toolkit for estonian. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Prabhu, Y. and Varma, M. (2014). Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009). Classifier chains for multi-label classification. In Wray Buntine, et al., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 254–269, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tsoumakas, G. and Katakis, I. (2009). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 09.

Weston, J., Makadia, A., and Yee, H. (2013). Label partitioning for sublinear ranking. *30th International Conference on Machine Learning, ICML 2013*, pages 840–848, 01.

Zhang, M.-L., Li, Y.-K., Liu, X.-Y., and Geng, X. (2017a). Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12, 11.

Zhang, W., Wang, L., Yan, J., Wang, X., and Zha, H. (2017b). Deep extreme multi-label learning. 04.

Zhou, T., Tao, D., and Wu, X. (2012). Compressed labeling on distilled labelsets for multi-label learning. *Machine Learning*, 88, 07.