Benchmarking XMLC Algorithms with OAXMLC

Christophe Broillet, Philippe Cudré-Mauroux, Julien Audiffren

As the main use-case of OAXMLC is XMLC, we provide below benchmark of a number of important baselines using this dataset. We start below by introducing the experimental setup used for benchmarking XMLC techniques on OAXMLC, followed by the presentation of the results. All experiments were run on a Linux machine equipped with a NVIDIA Tesla V100 SXM2 32 GB VRAM and an Intel(R) Xeon(R) Gold 6142 with 32 cores @ 2.4 GHz CPU with 768 GB of RAM.

1 Experimental Setup

1.1 Baselines

We benchmarked all the most commonly used XMLC algorithms, which are detailed below. The methods were used with implementation as close as possible to the original work, with minor changes when required. Unless specified otherwise, algorithm-specific parameters were set using the recommendation of the original papers. A learning rate of $5 \cdot 10^{-5}$ was used for all experiments.

1. XML-CNN [6], one of the first deep learning-based XMLC algorithm. XML-CNN uses a convolutional neural network architecture (CNN) [5] with dynamic max pooling, i.e. a pooling scheme that retains multiple features. We kept the original architecture: a three-layers CNN with 1D convolutional filters of window sizes 2, 4 and 8, keeping 128 features each. The bottleneck layer dimension is set to 512 and the dropout to 0.5.

2. MATCH [14], a transformer-based [12] method that takes advantage of the metadata of the document and the label hierarchy. It adds regularization terms to the objective function based on the taxonomy on the one hand, and includes metadata, such as the author or the venue of the document, in the document embedding representation on the other hand. We also kept the original architecture, that is a three-layers transformer encoder with 2 attention heads per layer, and 8 classification tokens.

3. AttentionXML [13], a tree-based approach building a shallow, wide label probabilistic tree. It uses bi-directional LSTMs in combination with attention mechanisms and fully connected layers. The original codebase is taken from the official implementation of the authors.

4. LightXML [2], a method that applies a generative adversarial network based on clusters of labels. The generator produces positive and negative labels while the discriminator tries to distinguish them. As the taxonomies of OAXMLC are *small*, i.e., $\leq 30'000$ labels according the original paper, the clusters comprise one single label each. For the architecture, we used the same backbone as in the MATCH baseline, i.e. a three-layers transformer encoder.

5. FastXML [10], a non-deep learning, tree-based method leveraging node-partitioning of document space. It learns the partition using the normalized discounted cumulative gain optimization objective. We took the PfastreXML [1] variant, and we used 50 trees, a maximum leaf size of 200 data points, the maximum labels per leaf is 200 and the hinge loss objective with L2-regularization.

6. Parabel [11], another non-deep learning method, that partitions the label space in a balanced hierarchical tree. It learns the partitions based on the maximum a posteriori estimate. This approach yielded a lower complexity in training and inference compared to the other state-of-the-art method at that time. We used 1 tree, a maximum leaf size of 200 labels, and a search width of 200 for the topics taxonomy, and 10 for the concepts, due to scaling issues.

7. CascadeXML [3], a transformer-based model that leverages hierarchical clustering to extract intermediate label representation at each level of the clustering. For this method we kept the balanced 2-means clustering on two levels as in the original paper. We used a transformer encoder architecture with 4 layers.

8. HECTOR [7], a transformer based model that predicts paths directly on the hierarchical taxonomy. While the model was built for XMLCompletion, it is also a powerful XMLC method, as discussed by the authors [7].

1.2 Data pre-processing

Before evaluating the different XMLC methods on the OAXMLC dataset, we first preprocessed the data as follows. First, and in line with [7], we recursively removed the leaves in the taxonomy that had fewer than 50 documents. Second, the abstracts and label descriptions were processed using the SentencePiece tokenizer [4], and were encoded using a vocabulary of size 10'000. When needed, we used the `GloVe.840B.300d` pre-trained word embeddings [9]. We then randomly split the data in a training/validation/testing set, representing respectively 70%, 15% and 15% of the total number of documents.

1.3 Metrics

In order to get a complete evaluation, we measured multiple metrics on different subsets of labels. The included metrics measured both classification performance and ranking performance. For classification metrics, we measured precision, recall and F1-score, with both micro and macro-average. The threshold discriminating positive and negative classifications from probabilities was inferred on the validation set as the average threshold maximizing the (micro) F1-score. Table 1 shows the thresholds for each combination of method and taxonomy. For ranking performance, we measured the Precision at k ($P@k$

Table 1: **Classification thresholds.**

	Topics	**Concepts**
XML-CNN	0.309	0.316
MATCH	0.328	0.361
AttentionXML	0.342	0.360
LightXML	0.314	0.296
FastXML	0.316	0.348
Parabel	0.336	0.466
CascadeXML	0.305	0.520
HECTOR	0.179	0.061

for short) and the Normalized Discounted Cumulative Gain at k ($NDCG@k$, or $N@k$) for different values of k between 1 and 10, depending on each experiment. These metrics were defined as follows. For any $1 \leq i \leq N$ (the number of possible labels), let

$$y_i = \begin{cases} 1 & \text{if the i-th most confident prediction of the algorithm is correct} \\ 0 & \text{otherwise} \end{cases}$$

With these notations, $P@k$ and $N@k$ are defined as:

$$P@k = \frac{1}{k} \sum_{n=1}^{k} y_n,$$
$$N@k = \frac{\sum_{n=1}^{k} \frac{y_n}{\log(n+1)}}{\sum_{n=1}^{\min(k,k_y)} \frac{1}{\log(n+1)}},$$

where k_y denotes the number of labels of a document. In other words, $P@k$ represents the average number of correctly predicted labels among the first k most confident prediction, while $NDCG@k$ provides a smoother measurement of the quality of the ranking by assigning lower weights to failed predictions in the tail of the ranking. Importantly, $\forall k \geq 1$, $P@k$ and $N@k$ are only computed on documents with at least k labels.

1.4 Experimental Design

To benchmark the impact of the taxonomies on XMLC algorithm, we conducted three types of experiments on OAXMLC, each one focusing on a different property of the taxonomies. This approach results in a broad view of each model performance, allowing for a deeper analysis and benchmarking. Those experiments are:

1. **Global experiment.** This experiment builds on the most commonly used XMLC evaluation, by measuring the performance of each method on predicting the labels for each document on both taxonomies (Concepts and Topics). More precisely, we used all labels, excluding the root (i.e., *computer science*, which is trivial to predict), and the labels appearing on the first level of the taxonomy since (i) those are the most common and simple labels to predict, allowing the experimental results to focus on challenging labels and (ii) state-of-the-art methods such as HECTOR uses these labels as prefix for the path prediction, and thus their inclusion would skew the results in favor of these methods. The goal of the global experiment is to compare the overall performance of each XMLC method on both taxonomies.

2. **Level-based experiment.** This experiment focuses on the performance of each method on the labels within the same level (i.e., depth in the hierarchical taxonomy). This evaluation is performed for each level from the second to the last in both taxonomies. This experiment informs how an algorithm is performing as target labels become more specific, and thus rarer and more difficult to predict.

3. **Sub-taxonomy experiment.** This evaluation emphasizes the impact of the width of a taxonomy (i.e., the average number of children in the hierarchical label organization). To achieve this, we selected two sub-taxonomies for both concepts and topics, that were characterized by a smaller (resp. larger) average width. As a result, we selected the sub-taxonomies starting from the label `Human-Computer Interaction` (label `1709`) for the narrow sub-taxonomy of the topics (resp. starting from `Information Systems` (label `1710`) for the wide sub-taxonomy). For the concepts, these sub-taxonomies were respectively starting from `Computer engineering` (label `C113775141`) and from `World Wide Web` (label `C136764020`).

2 Results

Table 2: **Training and inference times for each XMLC method**. This table details the approximate training and inference time, in hours, of each algorithm for both the topics and the concepts taxonomies.

	Topics		Concepts	
Method	**Tr. time**	**Inf. time**	**Tr. time**	**Inf. time**
AttentionXML	22	< 1	112	4
HECTOR	128	11	262	72
MATCH	10	1	17	3
XML-CNN	5	< 1	15	1
CascadeXML	12	2	49	3
FastXML	12	1	13	15
LightXML	10	< 1	48	2
Parabel	38	3	113	83

The training and the inference times of each method on the global experiment can be found in Table 2. We observe that each method is slower on the concepts taxonomy, due to the fact that it is 10 times larger than the topics taxonomy. Interestingly, FastXML is almost on a par in the training time between the Topics and the Concepts taxonomies, since it is the single method among the baselines that has a training time complexity independent of the number of labels, due to the partitioning of the document space. Conversely, methods such as AttentionXML see their training time increase almost five times. Finally, while HECTOR is the slowest method in terms of training time, it is important to note that its training time only doubles for the Concepts taxonomy, which is ten times larger. Moreover, its larger inference time compared to the other methods can be explained by the fact that this approach builds and keeps multiple candidate paths in the taxonomy for each document.

2.1 Global Evaluation

Table 3 displays the results of the global experiment on both taxonomies. Overall, AttentionXML and MATCH achieve better performance than the others for most metrics, with the advantage of AttentionXML increasing on the larger taxonomy. Interestingly, HECTOR outperforms the other baselines on the precision@1 metric for Topics, but not for Concepts. However, this advantage quickly fades away for $k > 1$, with precision@5 and precision@10 getting significantly worse than AttentionXML and MATCH. This may be due to the fact that HECTOR is specifically trained on path completion within sub-taxonomies, hence is better at predicting the most probable label, especially in a smaller taxonomy (like Topics). These observations particularly emphasizes the usefulness of OAXMLC, as it highlights the impact of the taxonomy on each method. XML-CNN performs in general slightly worse than the other deep learning algorithms, in particular with respect to AttentionXML and MATCH. This difference may reflect the fact that there is no attention mechanism in its architecture. The attention block architecture element has already been proven to be particularly effective for dealing with textual data [8]. Interestingly, CascadeXML achieves performance very close to AttentionXML and MATCH on the Topics Taxonomy, but is significantly worse than them on the Concepts Taxonomy. This is possibly due to the choice of the balanced 2-means clustering, which may be counter-productive on larger taxonomies with more relations between the labels. Finally, FastXML and Parabel got significantly worse metrics than the other methods, illustrating the advantage of recent deep learning-based methods for XMLC.

Table 3: **Global experiment.** This table displays the results of each method on both taxonomies for the global experiment. P@k (resp. N@k) denotes the precision @k (resp. the NDCG @k), while μ (resp. M) denotes the micro average (resp. the macro average). The best result for each metric and taxonomy is written in bold.

	Method	$P@1$	$P@5$	$P@10$	$N@5$	$N@10$	μP	μRec	$\mu F1$	MP	$MRec$	$MF1$
Topics	Att.XML	0.751	**0.750**	**0.772**	**0.795**	**0.799**	0.667	0.621	0.643	0.594	0.490	0.537
	HECTOR	**0.765**	0.658	0.708	0.710	0.745	0.446	0.577	0.503	0.384	**0.523**	0.443
	MATCH	0.763	0.745	0.727	0.791	0.756	**0.668**	**0.639**	**0.653**	**0.615**	0.496	**0.549**
	XML-CNN	0.692	0.652	0.681	0.709	0.716	0.604	0.551	0.577	0.482	0.361	0.413
	FastXML	0.390	0.366	0.374	0.405	0.402	0.433	0.251	0.317	0.323	0.080	0.128
	Casc.XML	0.750	0.724	0.759	0.775	0.791	0.648	0.620	0.634	0.553	0.451	0.497
	LightXML	0.748	0.723	0.761	0.774	0.791	0.651	0.614	0.632	0.569	0.452	0.504
	Parabel	0.439	0.454	0.483	0.506	0.520	0.360	0.264	0.304	0.277	0.190	0.225
Concepts	Att.XML	**0.889**	**0.850**	**0.830**	**0.881**	**0.873**	0.711	**0.695**	**0.703**	**0.542**	0.394	0.456
	HECTOR	0.814	0.670	0.645	0.719	0.705	0.515	0.495	0.505	0.285	0.229	0.254
	MATCH	0.886	0.847	**0.830**	0.878	**0.873**	0.713	0.691	0.702	0.524	**0.415**	**0.463**
	XML-CNN	0.829	0.737	0.720	0.785	0.783	0.617	0.552	0.582	0.370	0.234	0.287
	FastXML	0.513	0.490	0.540	0.531	0.599	0.562	0.242	0.338	0.190	0.032	0.055
	Casc.XML	0.863	0.789	0.757	0.831	0.816	**0.777**	0.504	0.612	0.408	0.181	0.251
	LightXML	0.872	0.820	0.796	0.855	0.846	0.679	0.644	0.661	0.454	0.324	0.378
	Parabel	0.554	0.508	0.529	0.553	0.594	0.444	0.311	0.366	0.272	0.088	0.133

2.2 Per-level Evaluation

Table 4 shows the results of the per-level experiment on both taxonomies on level 2 and 3. Table 5 shows the results for the concepts taxonomy on the per-level experiment, from level 2 to 5. Interestingly, most metrics increased for level 3 of the Topics taxonomy compared to level 2, with the notable exception of Precision@1 for HECTOR. This may seem surprising, as labels deeper in the taxonomy are more specific and should thus be harder to predict correctly. However, this can be explained by the specific property of the Topic taxonomy, where the third level is narrower than the second. Indeed, this phenomenon does not occur for the Concept taxonomy, which is significantly broader. This observation emphasizes the importance of studying the impact of taxonomical properties on XMLC methods, which can be achieved using OAXMLC. HECTOR performance, while close to the best on level 2 across all metrics, significantly worsen on level 3 for both taxonomies. This behavior may be explained by the fact that HECTOR is originally designed as a label completion algorithm, and its optimal performance displayed in [7] on the deeper lever of the taxonomy is conditioned to being provided with the correct label prefix. As this was not the case in our XMLC experiments, this may have led to sub-par performance. Finally, and similarly to the global experiment, FastXML and Parabel are performing worse than the other deep learning algorithms, in both ranking and predictive metrics.

2.3 Sub-taxonomies Evaluation

Table 6 displays the results of the sub-taxonomy experiments, for both taxonomies. Overall, and as expected, all XMLC algorithms achieve higher performance on the *narrow* sub-taxonomy than on the *wide* taxonomy. We observe in particular a strong performance decrease for the ranking metrics (e.g.,10 % for P@1 on both taxonomy for most methods), with the difference being larger for FastXML and Parabel, which can go up to 15% for the $P@3$ for example. Importantly, the decrease differs between concept and topics. For instance, P@3 varies significantly less for the topics sub-taxonomies (e.g., 2% for AttentionXML) compared to the concept sub-taxonomies (13%). These results highlight the fact that XMLC algorithm performance is not homogeneous within a taxonomy, and that some sub-

Table 4: **Per-level experiment.** This table displays the results of each method on both taxonomies for the first two levels. P@k (resp. N@k) denotes the precision @k (resp. the NDCG @k), while μ (resp. M) denotes the micro average (resp. the macro average). The best result for each metric and taxonomy is written in bold.

	Level	Method	P@1	P@5	N@5	μP	μRec	$\mu F1$	MP	$MRec$	$MF1$
Topics	2	Att.XML	0.751	0.752	0.794	0.666	0.620	0.642	0.613	0.509	0.556
		HECTOR	**0.765**	**0.756**	**0.797**	**0.706**	0.571	0.631	**0.631**	0.481	0.546
		MATCH	0.764	0.743	0.783	0.666	**0.641**	**0.653**	0.623	**0.536**	**0.576**
		XML-CNN	0.692	0.652	0.703	0.604	0.556	0.579	0.530	0.420	0.468
		FastXML	0.390	0.378	0.411	0.432	0.266	0.329	0.424	0.147	0.218
		Casc.XML	0.750	0.734	0.779	0.647	0.623	0.635	0.595	0.506	0.547
		LightXML	0.749	0.729	0.775	0.649	0.616	0.632	0.592	0.502	0.543
		Parabel	0.439	0.469	0.515	0.367	0.265	0.308	0.308	0.201	0.243
	3	Att.XML	**0.883**	**0.850**	**0.902**	0.681	0.634	**0.657**	0.575	0.471	0.518
		HECTOR	0.621	0.800	0.850	0.081	**0.660**	0.144	0.128	**0.566**	0.208
		MATCH	0.879	**0.850**	0.894	**0.707**	0.604	0.651	**0.607**	0.454	**0.520**
		XML-CNN	0.771	0.700	0.760	0.615	0.488	0.544	0.431	0.300	0.354
		FastXML	0.417	0.650	0.727	0.643	0.036	0.067	0.219	0.010	0.020
		Casc.XML	0.831	0.650	0.733	0.664	0.578	0.618	0.511	0.393	0.444
		LightXML	0.846	0.800	0.855	0.678	0.580	0.625	0.546	0.401	0.462
		Parabel	0.632	0.750	0.767	0.279	0.250	0.264	0.244	0.179	0.207
Concepts	2	Att.XML	**0.889**	**0.834**	**0.872**	0.730	**0.729**	**0.729**	**0.544**	0.395	**0.458**
		HECTOR	0.814	0.751	0.791	0.680	0.560	0.614	0.349	0.252	0.293
		MATCH	0.888	0.832	0.870	0.740	0.717	0.728	0.527	**0.398**	0.454
		XML-CNN	0.828	0.696	0.755	0.640	0.573	0.605	0.366	0.210	0.267
		FastXML	0.513	0.455	0.506	0.555	0.251	0.346	0.159	0.022	0.039
		Casc.XML	0.864	0.760	0.813	**0.797**	0.543	0.646	0.388	0.185	0.251
		LightXML	0.872	0.796	0.841	0.707	0.676	0.691	0.464	0.320	0.378
		Parabel	0.544	0.469	0.524	0.417	0.333	0.371	0.245	0.080	0.120
	3	Att.XML	**0.825**	**0.834**	**0.869**	0.697	0.661	**0.679**	**0.546**	0.400	0.461
		HECTOR	0.574	0.621	0.655	0.355	0.446	0.395	0.265	0.260	0.263
		MATCH	0.821	0.832	0.868	0.686	**0.666**	0.676	0.529	**0.419**	**0.467**
		XML-CNN	0.715	0.730	0.779	0.597	0.531	0.562	0.368	0.237	0.288
		FastXML	0.453	0.579	0.623	0.588	0.226	0.327	0.181	0.028	0.049
		Casc.XML	0.737	0.746	0.797	**0.757**	0.464	0.575	0.411	0.181	0.252
		LightXML	0.792	0.799	0.841	0.653	0.612	0.632	0.456	0.326	0.380
		Parabel	0.489	0.548	0.606	0.528	0.281	0.367	0.269	0.088	0.133

taxonomies can be significantly more challenging for some methods. Interestingly, CascadeXML, performs better for the wide sub-taxonomy of Concept than for the narrow one. This may be due to the choice of the balanced 2-means clustering, as this structure appears to be more efficient for more complex sub-taxonomies. It may overfit on the simpler sub-taxonomies, by incorrectly clustering together too many labels.

References

[1] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. "Extreme Multi-label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 935–944.

[2] Ting Jiang et al. "LightXML: Transformer with Dynamic Negative Sampling for High-Performance Extreme Multi-label Text Classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.9 (9 May 18, 2021), pp. 7987–7994. ISSN: 2374-3468. DOI: 10.1609/aaai.v35i9.16974. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16974 (visited on 04/08/2025).

[3] Siddhant Kharbanda et al. "CascadeXML: Rethinking Transformers for End-to-end Multi-resolution Training in Extreme Multi-label Classification". In: *Advances in Neural Information Processing Systems* 35 (Dec. 6, 2022), pp. 2074–2087. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/0e0157ce5ea15831072be4744cbd5334-Abstract-Conference.html (visited on 04/08/2025).

[4] Taku Kudo and John Richardson. "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing". In: *arXiv preprint arXiv:1808.06226* (2018).

[5] Yann LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541.

[6] Jingzhou Liu et al. "Deep Learning for Extreme Multi-label Text Classification". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. ACM, 2017, pp. 115–124.

[7] Natalia Ostapuk et al. "Follow the Path: Hierarchy-Aware Extreme Multi-Label Completion for Semantic Text Tagging". In: *Proceedings of the ACM Web Conference 2024*. WWW '24. New York, NY, USA: Association for Computing Machinery, May 13, 2024, pp. 2094–2105. ISBN: 979-8-4007-0171-9. DOI: 10.1145/3589334.3645558. URL: https://dl.acm.org/doi/10.1145/3589334.3645558 (visited on 04/07/2025).

[8] Narendra Patwardhan, Stefano Marrone, and Carlo Sansone. "Transformers in the real world: A survey on nlp applications". In: *Information* 14.4 (2023), p. 242.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: https://aclanthology.org/D14-1162.pdf (visited on 04/08/2025).

[10] Yashoteja Prabhu and Manik Varma. "FastXML: A Fast, Accurate and Stable Tree-Classifier for Extreme Multi-Label Learning". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, NY, USA: Association for Computing Machinery, Aug. 24, 2014, pp. 263–272. ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623651. URL: https://dl.acm.org/doi/10.1145/2623330.2623651 (visited on 04/08/2025).

[11] Yashoteja Prabhu et al. "Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising". In: *Proceedings of the 2018 World Wide Web Conference*. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 10, 2018, pp. 993–1002. ISBN: 978-1-4503-5639-8. DOI: 10.1145/3178876.3185998. URL: https://dl.acm.org/doi/10.1145/3178876.3185998 (visited on 04/07/2025).

[12] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 04/08/2025).

[13] Ronghui You et al. "AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019, pp. 5812–5822.

[14] Yu Zhang et al. "MATCH: Metadata-Aware Text Classification in A Large Hierarchy". In: *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2021, pp. 3246–3257.

Table 5: **Per-level experiment on the concepts taxonomy.**

Level	Method	P@1	P@5	N@5	μP	μRec	$\mu F1$	MP	$MRec$	$MF1$
2	AttentionXML	**0.889**	**0.834**	**0.872**	0.730	**0.729**	**0.729**	**0.544**	0.395	**0.458**
	HECTOR	0.814	0.751	0.791	0.680	0.560	0.614	0.349	0.252	0.293
	MATCH	0.888	0.832	0.870	0.740	0.717	0.728	0.527	**0.398**	0.454
	XML-CNN	0.828	0.696	0.755	0.640	0.573	0.605	0.366	0.210	0.267
	FastXML	0.513	0.455	0.506	0.555	0.251	0.346	0.159	0.022	0.039
	CascadeXML	0.864	0.760	0.813	**0.797**	0.543	0.646	0.388	0.185	0.251
	LightXML	0.872	0.796	0.841	0.707	0.676	0.691	0.464	0.320	0.378
	Parabel	0.544	0.469	0.524	0.417	0.333	0.371	0.245	0.080	0.120
3	AttentionXML	**0.825**	**0.834**	**0.869**	0.697	0.661	**0.679**	**0.546**	0.400	0.461
	HECTOR	0.574	0.621	0.655	0.355	0.446	0.395	0.265	0.260	0.263
	MATCH	0.821	0.832	0.868	0.686	**0.666**	0.676	0.529	**0.419**	**0.467**
	XML-CNN	0.715	0.730	0.779	0.597	0.531	0.562	0.368	0.237	0.288
	FastXML	0.453	0.579	0.623	0.588	0.226	0.327	0.181	0.028	0.049
	CascadeXML	0.737	0.746	0.797	**0.757**	0.464	0.575	0.411	0.181	0.252
	LightXML	0.792	0.799	0.841	0.653	0.612	0.632	0.456	0.326	0.380
	Parabel	0.489	0.548	0.606	0.528	0.281	0.367	0.269	0.088	0.133
4	AttentionXML	**0.813**	0.900	0.915	0.639	0.600	0.619	**0.551**	0.410	0.471
	HECTOR	0.537	0.656	0.670	0.209	0.254	0.230	0.258	0.231	0.244
	MATCH	0.811	**0.907**	**0.921**	0.624	**0.615**	**0.620**	0.526	**0.444**	**0.482**
	XML-CNN	0.705	0.854	0.873	0.534	0.505	0.519	0.387	0.283	0.327
	FastXML	0.504	0.783	0.801	0.553	0.247	0.341	0.256	0.055	0.090
	CascadeXML	0.681	0.835	0.861	**0.696**	0.399	0.507	0.448	0.199	0.275
	LightXML	0.776	0.882	0.900	0.585	0.559	0.571	0.457	0.350	0.396
	Parabel	0.487	0.668	0.702	0.530	0.267	0.355	0.327	0.110	0.165
5	AttentionXML	0.776	0.836	0.870	0.584	0.497	0.537	**0.520**	0.358	0.424
	HECTOR	0.552	0.555	0.626	0.127	0.119	0.123	0.193	0.092	0.125
	MATCH	**0.780**	**0.844**	**0.875**	0.558	**0.535**	**0.547**	0.502	**0.416**	**0.455**
	XML-CNN	0.669	0.782	0.818	0.483	0.384	0.428	0.366	0.238	0.288
	FastXML	0.483	0.726	0.764	0.555	0.142	0.226	0.219	0.044	0.073
	CascadeXML	0.562	0.743	0.790	**0.618**	0.271	0.376	0.415	0.152	0.222
	LightXML	0.741	0.819	0.854	0.520	0.436	0.475	0.418	0.304	0.352
	Parabel	0.480	0.657	0.702	0.447	0.176	0.252	0.287	0.084	0.130

Table 6: **Sub-taxonomy experiment.** This table displays the results of each method on both taxonomies for each of the two selected sub-taxonomies. P@k (resp. N@k) denotes the precision @k (resp. the NDCG @k), while μ (resp. M) denotes the micro average (resp. the macro average). The best results are written in bold.

	Sub-Tax.	Method	P@1	P@3	N@3	μP	μRec	$\mu F1$	MP	$MRec$	$MF1$
Topics	Narrow	Att.XML	0.837	**0.816**	**0.850**	0.802	0.581	0.674	0.677	0.451	0.542
		HECTOR	0.837	0.705	0.749	0.494	**0.601**	0.542	0.404	**0.554**	0.467
		MATCH	**0.851**	0.813	0.845	**0.815**	0.592	**0.686**	**0.746**	0.476	**0.581**
		XML-CNN	0.788	0.765	0.800	0.744	0.510	0.605	0.617	0.337	0.436
		FastXML	0.597	0.628	0.670	0.749	0.185	0.296	0.336	0.065	0.110
		Casc.XML	0.828	0.792	0.829	0.792	0.586	0.674	0.655	0.420	0.512
		LightXML	0.831	0.784	0.820	0.782	0.576	0.664	0.680	0.416	0.516
		Parabel	0.649	0.687	0.725	0.693	0.249	0.367	0.584	0.177	0.271
	Wide	Att.XML	0.758	**0.793**	**0.815**	0.757	0.572	0.651	0.654	**0.436**	0.523
		HECTOR	0.733	0.708	0.738	0.478	0.516	0.496	0.472	0.430	0.450
		MATCH	**0.774**	0.789	0.810	**0.779**	**0.581**	**0.666**	**0.690**	0.433	**0.532**
		XML-CNN	0.687	0.681	0.708	0.699	0.479	0.569	0.543	0.300	0.387
		FastXML	0.474	0.448	0.472	0.621	0.222	0.327	0.405	0.064	0.110
		Casc.XML	0.746	0.764	0.791	0.748	0.559	0.640	0.623	0.400	0.487
		LightXML	0.748	0.759	0.785	0.746	0.555	0.636	0.632	0.405	0.493
		Parabel	0.512	0.537	0.568	0.523	0.292	0.374	0.400	0.179	0.247
Concepts	Narrow	Att.XML	0.988	**0.990**	**0.992**	0.942	0.480	0.636	**0.927**	0.419	0.577
		HECTOR	0.833	0.911	0.915	0.671	**0.623**	0.646	0.528	0.464	0.494
		MATCH	**0.991**	0.980	0.984	0.944	0.571	**0.712**	0.925	**0.509**	**0.657**
		XML-CNN	0.973	0.953	0.960	0.913	0.353	0.509	0.772	0.273	0.403
		FastXML	0.908	0.901	0.914	**1.000**	0.021	0.041	0.444	0.008	0.016
		Casc.XML	0.669	0.672	0.733	0.921	0.337	0.494	0.678	0.232	0.346
		LightXML	0.984	0.973	0.979	0.906	0.495	0.640	0.875	0.396	0.545
		Parabel	0.485	0.501	0.575	0.932	0.133	0.233	0.787	0.115	0.200
	Wide	Att.XML	**0.880**	**0.860**	**0.883**	0.801	0.652	**0.719**	**0.640**	0.394	0.487
		HECTOR	0.800	0.710	0.749	0.611	0.465	0.528	0.331	0.227	0.269
		MATCH	0.875	0.857	0.879	0.793	**0.654**	0.717	0.622	**0.415**	**0.498**
		XML-CNN	0.752	0.756	0.789	0.696	0.524	0.598	0.432	0.247	0.314
		FastXML	0.522	0.609	0.645	0.703	0.224	0.340	0.223	0.041	0.070
		Casc.XML	0.779	0.759	0.796	**0.850**	0.459	0.596	0.465	0.184	0.264
		LightXML	0.844	0.827	0.855	0.744	0.613	0.672	0.525	0.328	0.404
		Parabel	0.494	0.548	0.589	0.659	0.279	0.392	0.392	0.097	0.155