Supplementary Materials for "On the Influence of Hyperparameters in Tree-Based Linear Methods for Extreme Multi-Label Text Classification"

Kuan-Ting Chen
National Taiwan University
Taipei, Taiwan
andyngewcz2d@gmail.com

Hung-Chih Chiang
National Taiwan University
Taipei, Taiwan
r13922189@ntu.edu.tw

Chih-Jen Lin
National Taiwan University
Taipei, Taiwan
Mohamed bin Zayed University of Artificial Intelligence
Abu Dhabi, United Arab Emirates
cjlin@csie.ntu.edu.tw

A Illustration of the Beam Search Process

Now we illustrate the prediction process with beam width $B = 2$. Based on Figure 1, Figure I adds red arrows to indicate the paths selected by beam search. During prediction, the score of each node is computed based on the prediction scores along the path from the root to that node. At depth 1, the model selects nodes A and C based on their scores. At depth 2, the model considers node A and the children of node C, namely nodes D, E, and F. Since the children of node A are already leaf labels, we do not further expand node A and keep all labels under node A for final evaluation. Based on the scores of nodes A, D, E, and F, the model selects nodes A and D. Afterward, since nodes A and D contain only leaf labels, the beam search terminates. Therefore, the model evaluates labels 1, 2, 3, 6, and 7, then ranks them based on their scores to make predictions.

B Hyperparameter Settings and Analysis Setup

To conduct our analysis, we adopt LibMultiLabel,[1] an end-to-end XMTC library that supports configuration and training/prediction of tree-based linear models from raw text to final evaluation.

Table I summarizes the hyperparameters considered in our analysis, along with their search ranges and default values taken from LibMultiLabel. Note that the number of clusters K controls how the label set is recursively partitioned. Therefore, when designing the search space for K, we should take the number of labels L into account.

However, prior works such as [4, 9, 10] directly use a fixed value of K throughout their experiments and may not fully reveal the influence of K. For example, with $L = 1{,}000$ and $K = 1{,}000$, the label set is divided into 1,000 clusters of size one, and the tree stops growing after the first split. In contrast, with $L = 100{,}000$ and the same K, each cluster contains around 100 labels on average, so the tree continues to grow as each cluster still has multiple labels. As a result, using the same K may actually reflect different behaviors when L varies. To alleviate this issue, we decide to use a dynamic setting for K based on L. We note that [4, 5] consider

$$K = \left\lceil L^{1/d_{\max}} \right\rceil \tag{I}$$

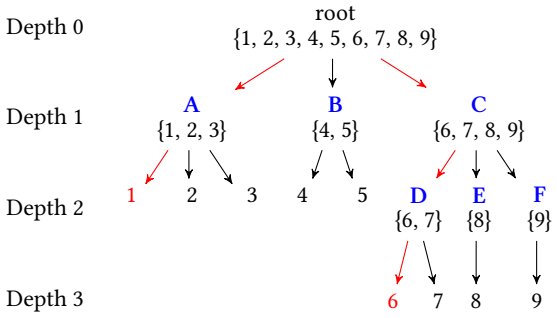


Figure I: Illustration of beam search with beam width $B = 2$ based on Figure 1. The red arrows indicate the paths selected by beam search.

Table I: Hyperparameters in tree-based linear methods and their corresponding search spaces for sensitivity analysis. For tokenizer, Porter and Snowball refer to NLTK stemmers, while NLTK and SpaCy denote lemmatizers.

Hyperparameters	Our additional search space	Default
TF-IDF feature generation		
strip_accents	{'ascii', 'unicode'}	None
tokenizer	{NLTK, SpaCy, Porter, Snowball}	None
stop_words	{'english'}	None
ngram_range	{(1, 2), (1, 3), (1, 4), (1, 5)}	(1, 1)
max_features[2]	$\{2^i \times 10^4 \mid i = 0, 1, 2, 3, 4, 5\}$	320,000
min_df	{1, 2, 5, 10}	1
Training		
C	$\{2^{-2}, 2^{-1}, \ldots, 2^5\}$	1
K	{Equation (II) $\mid \alpha = 2^{-2}, 2^{-1}, \ldots, 2^5$}	100
$d_{\max}$	$\{2, 3, \ldots, 9\}$	10
Prediction		
B	$\{2^2, 2^3, \ldots, 2^5\}$	10

[1] https://www.csie.ntu.edu.tw/~cjlin/libmultilabel/
[2] In scikit-learn, the default is None, but this leads to an excessively large vocabulary with a large ngram_range. Therefore, we use a relatively large value as our default.

when analyzing the effect of depth. Building on this idea, we adopt a more flexible formulation:

$$K = \left\lceil \alpha \cdot L^{1/d_{\max}} \right\rceil, \tag{II}$$

where α is a scaling factor that helps adjust K to represent smaller or larger clusters.

For stemming, we use the NLTK [1] implementations of the Porter [7] and Snowball [8] algorithms, while for lemmatization, we use implementations from NLTK and SpaCy [2].

For each dataset, we use the provided training and test sets. We run each configuration three times on the training set with different seeds, evaluate the test scores from these runs, and average them to obtain the performance for that configuration.

C Variance of P@1, P@3, and P@5 for Different Hyperparameters

We present the P@1, P@3, and P@5 results for four datasets corresponding to Table 1 in Tables II, III, and IV, respectively.

Similar to Table 1, Tables II, III and IV also indicate that the cost parameter C, the number of clusters K, and the ngram_range play crucial roles in model performance. In addition, we observe that the beam width B has a greater impact on the results for P@5 compared with P@1 and P@3, which we further examine in Section D.

D Full Results for Sensitivity Analysis

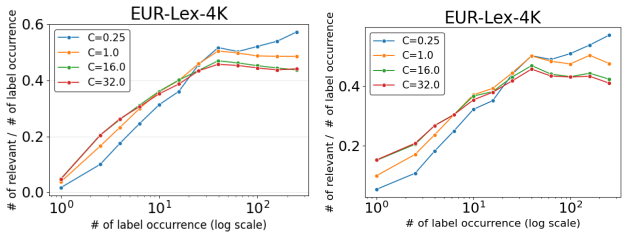
The sensitivity analysis results for the hyperparameters are presented in Figures V, VI, VII, VIII, IX, X, XI, XII, XIII, and XIV. Each figure shows the results for the four datasets evaluated under the three metrics P@1, P@3, and P@5. For the most influential hyperparameters, as discussed in the main text, Figure XI shows that the cost parameter C has a strong impact on model performance, and each loss function performs best in a different range. Figure XII further shows that the number of clusters K exhibits a U-shaped effect. This trend holds across all metrics.

For ngram_range in Figure VIII, the most notable performance improvement occurs when moving from (1,1) to (1,2), where bi-gram features are introduced. However, bi-grams can also hurt model performance under certain settings. Moreover, when tri-grams or higher-order n-grams are used, the improvement becomes marginal or even turns negative.

For stop_words in Figure VII, which filter out common words, this step generally improves model performance. Since all datasets in our experiments are in English, we adopt the 'english' setting to remove stop words.

Since we note that the beam width B also affects P@5 in Table IV, we check the beam width B in Figure XIV. The major influence appears at smaller values ($B = 4, 8$), where the performance drops noticeably, whereas for larger B, the effect becomes negligible. In this case, using the default value $B = 10$ is sufficient.

For the remaining hyperparameters, we find that their effects are at most dataset-dependent or relatively minor. In these cases, the default values already work well.



(a) OVR **(b) Tree-based**

Figure II: Performance of OVR and tree-based models with different C. The y-axis shows label-level precision, detailed in the figure description in Section E, while the x-axis indicates the number of label occurrences in the test set on a log scale. Each point represents a group of labels with similar numbers of occurrences, and the y-axis values are averaged across the labels in each group.

E How C Affects Model Performance

Besides examining how C affects model performance, we further analyze the predictive behavior of C. In a binary problem, the number of positive and negative instances in the training set may affect the choice of C for achieving good performance. This issue becomes more challenging in multi-label classification, where different binary problems with different positive and negative instances are involved. For OVR, in each binary problem, the number of positive instances is the same as the number of training instances associated with the label, for which we call the label occurrences. Therefore, for both OVR and tree-based linear methods, subsequently we design experiments to evaluate how C affects prediction performance across varying numbers of label occurrences in the test set.

Figure II shows the performance of OVR and tree-based models under different values of C, evaluated with respect to the number of label occurrences. For each label, we collect the instances associated with the label in the test set and retrieve the top three predictions for those instances. We regard an instance as relevant to the given label when the given label appears within the instance's top three predictions. We then normalize the number of relevant instances by the number of label occurrences to obtain the precision of each label, which reflects label-level prediction performance. In this way, the figure shows, for both OVR and tree-based settings, how label-level precision varies with the number of label occurrences. The results by using both OVR and tree-based models reveal a clear pattern: smaller C works well for frequent labels but struggles with rare ones, while larger C helps rare labels but may hurt frequent ones. This trade-off suggests that using different C values may be necessary for binary classifiers under different levels of positive/negative imbalance.

F Comparison of Different Approaches to C Tuning

Recall from Section 4.2.1 that when tuning the cost parameter C, we cannot consider all C hyperparameters across all classifiers and tune them together with other hyperparameters. Therefore, we only have two options:

Table II: Variance of P@1 for different hyperparameters. The highest result is bold-faced, the second highest is underlined, and the third highest is double underlined. P@1 values are scaled to $[0, 100]$ for variance calculation.

Hyperparameters	EUR-Lex-4K			EUR-Lex-57K			Mimic-iii-8K			AmazonCat-13K		
	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}
TF-IDF feature generation												
strip_accents	0.0976	0.0387	0.0213	0.0106	0.0034	0.0064	0.0106	0.0290	0.0001	0.0002	0.0051	0.0004
tokenizer	0.0154	0.0294	0.0109	0.0694	0.1066	0.2709	0.1810	0.1504	0.0200	0.0062	0.0150	0.0128
stop_words	0.0335	0.0197	0.0037	0.0000	0.2982	0.4780	0.1425	0.6020	0.9714	0.0055	0.0146	0.0022
ngram_range	0.3162	0.8519	0.0587	0.0844	0.4442	0.1369	0.2594	0.5355	0.4018	0.0178	0.0899	0.0108
max_features	0.0080	0.0530	0.0310	0.0134	0.0486	0.0207	0.0493	0.2255	0.0075	0.6733	1.0487	0.6887
min_df	0.0193	0.0518	0.0157	0.0287	0.0184	0.0080	0.0356	0.0464	0.0041	0.0028	0.0093	0.0003
Training												
C	**2.6719**	**6.7639**	**46.1668**	**1.2484**	**2.5919**	**19.6449**	**7.2883**	**8.1813**	**12.8834**	**4.5672**	**2.7713**	**0.7852**
K	0.4683	0.9021	0.7830	0.1335	0.2969	0.3212	0.3406	3.0790	0.3361	0.0068	0.0417	0.0183
$d_{\max}$	0.0011	0.0099	0.0004	0.0013	0.0071	0.0002	0.0325	0.0310	0.0019	0.0000	0.0003	0.0000
Prediction												
B	0.0011	0.0642	0.0001	0.0008	0.0203	0.0006	0.0468	0.0736	0.0502	0.0001	0.0017	0.0000

Table III: Variance of P@3 for different hyperparameters. The highest result is bold-faced, the second highest is underlined, and the third highest is double underlined. P@3 values are scaled to $[0, 100]$ for variance calculation.

Hyperparameters	EUR-Lex-4K			EUR-Lex-57K			Mimic-iii-8K			AmazonCat-13K		
	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}
TF-IDF feature generation												
strip_accents	0.0027	0.0392	0.0304	0.0028	0.0008	0.0028	0.0019	0.0012	0.0001	0.0003	0.0005	0.0004
tokenizer	0.0051	0.0181	0.0037	0.0283	0.0946	0.1102	0.0193	0.0572	0.0161	0.0102	0.0159	0.0143
stop_words	0.0003	0.0012	0.0028	0.0556	0.5532	0.8766	0.0031	0.3206	0.6499	0.0082	0.0003	0.0000
ngram_range	0.7303	1.2533	0.0144	0.3557	0.5011	0.2914	0.1677	0.4667	0.8419	0.0219	0.0328	0.0155
max_features	0.0023	0.0222	0.0106	0.0176	0.0302	0.0114	0.0187	0.0199	0.0039	1.5710	1.8942	1.1454
min_df	0.0078	0.0433	0.0066	0.0055	0.0097	0.0225	0.0032	0.0013	0.0018	0.0078	0.0051	0.0002
Training												
C	**1.4908**	**10.0579**	**53.8410**	**0.8239**	**7.2829**	**29.9562**	**5.5934**	**9.3842**	**14.8875**	**3.5628**	**2.0693**	**3.3609**
K	0.4271	0.8218	0.5404	0.1294	0.1931	0.1911	0.4039	2.7993	0.3190	0.0147	0.0159	0.0555
$d_{\max}$	0.0005	0.0010	0.0001	0.0002	0.0004	0.0000	0.0061	0.0296	0.0006	0.0000	0.0000	0.0000
Prediction												
B	0.0013	0.1370	0.0155	0.0032	0.0447	0.0368	0.4665	0.4301	0.5027	0.0000	0.0005	0.0001

(1) Independently optimize C for each binary classifier and collect them to form the final model. For example, for each binary classifier, we conduct a validation procedure to select the corresponding best C.
(2) Use a single shared C for all binary classifiers and optimize the overall performance.

For option 1, since the goal is to optimize each binary classifier separately, we apply three-fold cross-validation to each classifier within the tree-based methods and select the value of C that yields the highest precision. For option 2, we use a single shared C for all binary classifiers and directly optimize the overall metrics P@1, P@3, and P@5. In this case, we apply three-fold cross-validation to select the value of C that maximizes each metric, respectively. We run three random seeds for both options and report the averaged overall P@1, P@3, and P@5. The search range of C is $\{2^{-10}, 2^{-9}, \ldots, 2^{10}\}$.

Table V summarizes the results for the two options. We first note that, in XMTC with tree-based linear methods, using the same C can lead to good results. However, unlike in the multi-class setting, using a single shared C (i.e., option 2) consistently outperforms tuning C locally for each binary classifier (i.e., option 1). The reason may be that, in the tree-based setting, the limitation of option 1 becomes more pronounced due to the many interconnected binary problems across nodes, where tuning C separately for each classifier can easily lead to overfitting and thus degrade overall performance. In addition, local tuning only benefits the individual binary problems and does not ensure better global performance. In contrast, a shared C yields more stable and reliable results across P@1, P@3,

Table IV: Variance of P@5 for different hyperparameters. The highest result is bold-faced, the second highest is underlined, and the third highest is double underlined. P@5 values are scaled to $[0, 100]$ for variance calculation.

Hyperparameters	EUR-Lex-4K			EUR-Lex-57K			Mimic-iii-8K			AmazonCat-13K		
	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}
TF-IDF feature generation												
strip_accents	0.0041	0.0415	0.0022	0.0030	0.0020	0.0002	0.0003	0.0017	0.0000	0.0003	0.0002	0.0005
tokenizer	0.0029	0.0042	0.0023	0.0249	0.0729	0.0619	0.0058	0.0771	0.0177	0.0081	0.0156	0.0079
stop_words	0.0073	0.0001	0.0016	0.0356	0.4491	0.8697	0.0250	0.3174	0.9797	0.0047	0.0027	0.0014
ngram_range	0.4937	1.0080	0.0154	0.2938	0.5504	0.2865	0.4358	0.4302	1.4196	0.0162	0.0202	0.0244
max_features	0.0080	0.0136	0.0136	0.0176	0.0290	0.0065	0.0187	0.0233	0.0025	1.4263	**1.7895**	0.9454
min_df	0.0039	0.0026	0.0351	0.0025	0.0102	0.0082	0.0006	0.0050	0.0086	0.0067	0.0039	0.0001
Training												
C	**1.2866**	**10.2700**	**42.6434**	**0.7508**	**7.9807**	**26.5287**	**6.7930**	**9.7192**	**16.6639**	**2.4224**	1.3870	**4.2701**
K	0.2860	0.5135	0.3692	0.0644	0.1841	0.1324	0.3456	2.4061	0.4135	0.0126	0.0117	0.0601
$d_{\max}$	0.0007	0.0003	0.0001	0.0001	0.0003	0.0000	0.0021	0.0118	0.0010	0.0000	0.0000	0.0000
Prediction												
B	0.1396	0.1508	0.1497	0.1919	0.1014	0.2137	2.3346	2.6652	2.1463	0.0420	0.0246	0.0307

Table V: P@1, P@3, and P@5 of tree-based model for the two options of tuning C: C tuned locally for each binary classifier (option 1) and a single shared C for all binary classifiers (option 2), evaluated on ξ_{L2}, ξ_{L1} and ξ_{LR}.

	P@1			P@3			P@5		
	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}	ξ_{L2}	ξ_{L1}	ξ_{LR}
EUR-Lex-4K									
Individual C (option 1)	73.90	74.90	72.53	61.73	60.53	59.82	51.48	48.47	49.44
Shared C (option 2)	**81.99**	**80.95**	**81.32**	**69.08**	**68.33**	**68.45**	**57.95**	**56.63**	**57.20**
EUR-Lex-57K									
Individual C (option 1)	82.76	79.44	80.87	70.88	66.02	67.88	58.82	52.93	55.92
Shared C (option 2)	**91.16**	**89.79**	**91.23**	**81.31**	**80.54**	**80.69**	**68.64**	**67.21**	**68.20**
Mimic-iii-8K									
Individual C (option 1)	74.09	74.89	72.48	65.94	66.73	63.36	59.04	59.08	56.50
Shared C (option 2)	**84.32**	**82.20**	**84.84**	**77.60**	**77.35**	**77.37**	**71.51**	**71.58**	**70.85**

and P@5. Therefore, for tree-based linear methods, we recommend using a single C for all classifiers.

G Verifying the Relationship Between K and Evaluated Labels in Real-world Settings

In this section, we study the relationship between K and the number of evaluated labels in real-world settings. We first describe how we estimate the expected number of evaluated labels in Appendix G.1, and then discuss how the number of clusters influences the number of evaluated labels in Appendix G.2.

G.1 Estimating the Expected Number of Evaluated Labels in Practice

To examine the effect of K in a real-world setting, we compute the number of evaluated labels under different K using data from public industry applications. Unlike the balanced tree example in Figure 5, where all clusters within each level have the same size, we often

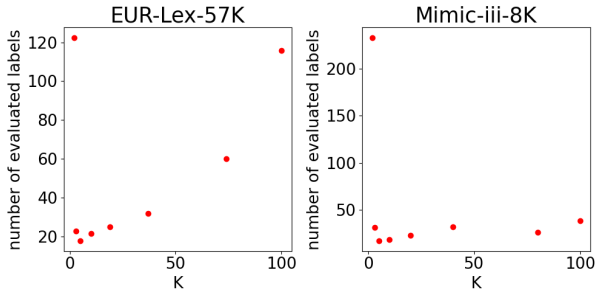


Figure III: Expected number of evaluated labels on Mimic-iii-8K and EUR-Lex-57K with beam width $B = 10$. We simulate the beam search process 10,000 times and report the averaged results. Each red dot represents the same K value as in Figure 2.

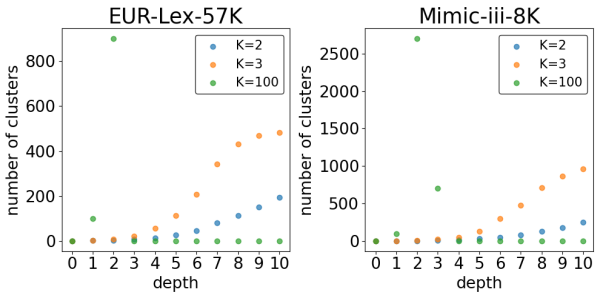


Figure IV: Number of clusters at each depth on Mimic-iii-8K and EUR-Lex-57K.

obtain unbalanced trees in practice. When we partition a label set into K parts, the resulting clusters may contain different numbers of labels. Therefore, the number of evaluated labels depends on which clusters are selected during beam search. To account for this variation, we estimate the expected number of evaluated labels through simulation.[3] During beam search, we randomly keep B clusters at each level and count the labels evaluated in this process.[4] We repeat the procedure 10,000 times and take the average to obtain the expected number of evaluated labels. After 10,000 repetitions, the result stabilizes, as running the procedure additional times produces nearly identical outcomes. Therefore, we report the results based on 10,000 runs in Figure III. Interestingly, the trend in Figure III is similar to the patterns in Figure 2. The values are higher when $K = 2$ or $K = 100$, while intermediate values, especially $K = 3$, drop sharply.

G.2 How Number of Clusters Affects the Number of Evaluated Labels

Next, we check the number of clusters at each depth. Note that this number depends only on the tree structure and is independent of the beam width B. We will explain how the number of clusters relates to the number of evaluated labels. We focus on the cases of $K = 2$, $K = 3$, and $K = 100$, and present the results in Figure IV. By our settings, the number of clusters increases with depth. Clearly, $K = 3$ follows exponential growth with a base of three and thus increases more rapidly than the base-two growth in $K = 2$. For $K = 100$, the number of clusters grows even faster, with exponential growth based on 100. However, most clusters reach leaf labels at very shallow depths. Therefore, at deeper levels, $K = 100$ actually has fewer clusters than $K = 3$. As a result, we can link the number of clusters at each depth to the number of evaluated labels. When comparing $K = 2$ with $K = 3$, the $K = 2$ tree has fewer clusters at each depth, so beam search removes fewer clusters. Consequently, the $K = 2$ setting results in more labels being evaluated at the last level. For $K = 3$ and $K = 100$, although beam search removes more clusters for the $K = 100$ tree at shallower depths, the pruning proceeds for only a few levels. In contrast, beam search for the $K = 3$ tree continues removing labels at each level, leaving fewer labels at the final level. Therefore, the $K = 100$ setting ultimately retains more labels for evaluation.

[3] Besides simulation, we can alternatively compute the number of evaluated labels directly on a validation set.

[4] As described in Section A, if a cluster (i.e., a node in the tree) contains only leaf labels, we stop expanding the cluster and keep all labels within the cluster for evaluation.

H Datasets

We download the raw text data from the LIBSVM repository,[5] except for Mimic-III-8K. For Mimic-III-8K, we use the data from [3] and follow the setup in [6] to create the training and test sets. Dataset sources used in our experiments are provided below.

- EUR-Lex-4K: training data[6] and test data.[7]
- EURLex-57K: training data,[8] validation data,[9] and test data.[10] We merge the original training and validation data to create a new training dataset for three-fold cross-validation.
- Mimic-iii-8K: The raw data is provided from [3], and we follow [6] to generate training and test data. We do not redistribute the processed data due to restrictions on sharing the original dataset.
- AmazonCat-13K: training set[11] and test set.[12]
- Wiki10-31K: training data[13] and test data.[14]
- Amazon-670K: training data[15] and test data.[16]

References

[1] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly.

[2] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python. doi:10.5281/zenodo.1212303

[3] Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data* 3, 1 (2016), 1–9. doi:10.1038/sdata.2016.35

[4] Sujay Khandagale, Han Xiao, and Rohit Babbar. 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning* 109 (2020), 2099–2119. doi:10.1007/s10994-020-05888-2

[5] He-Zhe Lin, Cheng-Hung Liu, and Chih-Jen Lin. 2024. Exploring Space Efficiency in a Tree-based Linear Model for Extreme Multi-label Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).* 16245–16260. https://www.csie.ntu.edu.tw/~cjlin/papers/multilabel_tree_model_size/multilabel_tree_model_size.pdf

[6] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. 2018. Explainable prediction of medical codes from clinical text. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).* 1101–1111. doi:10.18653/v1/N18-1100

[7] Martin Porter. 1997. An algorithm for suffix stripping. In *Readings in information retrieval.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 313–316.

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

[6] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/eurlex_raw_texts_train.txt.bz2

[7] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/eurlex_raw_texts_test.txt.bz2

[8] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/eurlex57k_raw_texts_train.txt.bz2

[9] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/eurlex57k_raw_texts_val.txt.bz2

[10] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/eurlex57k_raw_texts_test.txt.bz2

[11] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/AmazonCat-13K_raw_texts_train.txt.bz2

[12] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/AmazonCat-13K_raw_texts_test.txt.bz2

[13] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/wiki10_31k_raw_texts_train.txt.bz2

[14] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/wiki10_31k_raw_texts_test.txt.bz2

[15] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/Amazon-670K_raw_texts_train.txt.bz2

[16] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel/Amazon-670K_raw_texts_test.txt.bz2

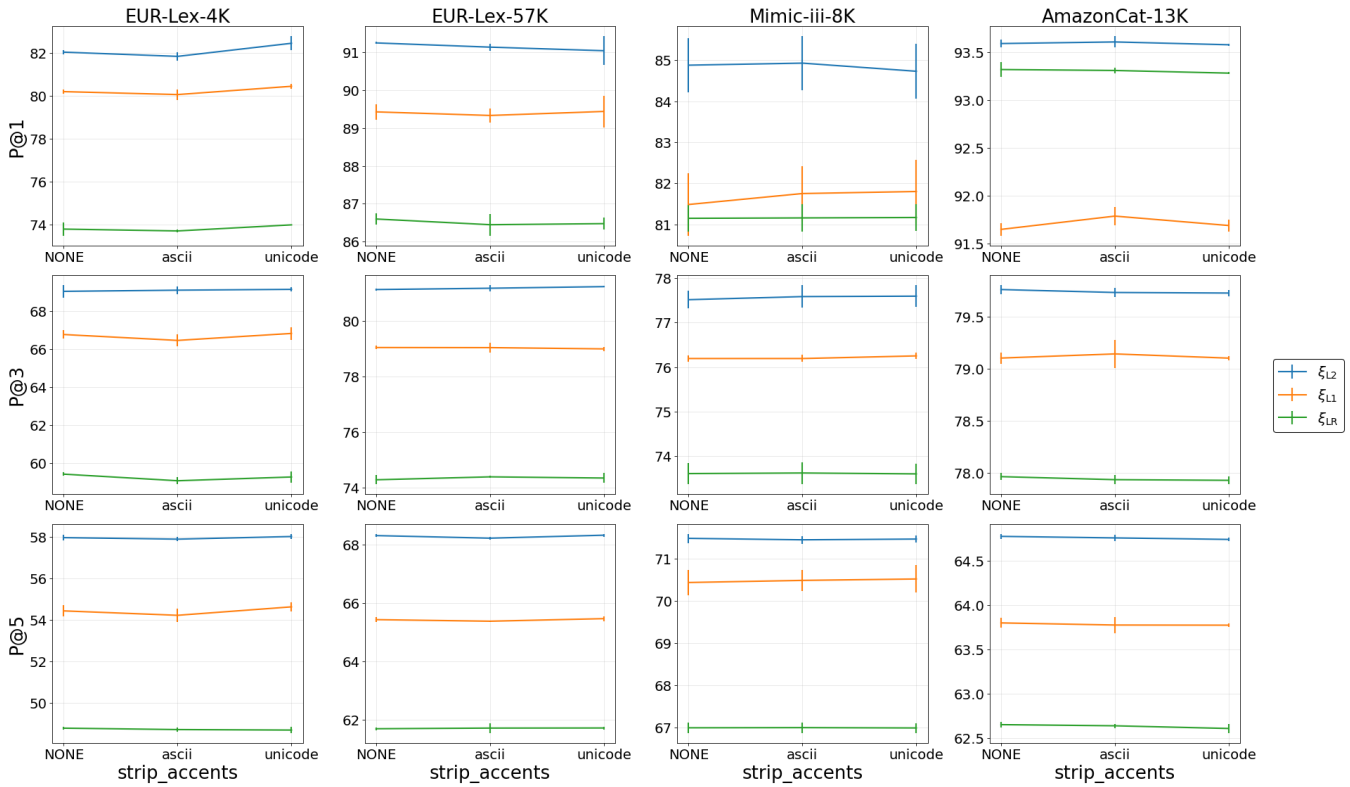


Figure V: Sensitivity analysis of strip_accents over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

[8] Martin F. Porter. 2001. Snowball: A language for stemming algorithms. https://tartarus.org/martin/PorterStemmer/

[9] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising. In *Proceedings of the 2018 World Wide Web Conference (WWW)*. 993–1002.

[10] Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S. Dhillon. 2022. PECOS: Prediction for Enormous and Correlated Output Spaces. *Journal of Machine Learning Research* 23, 98 (2022), 1–32.

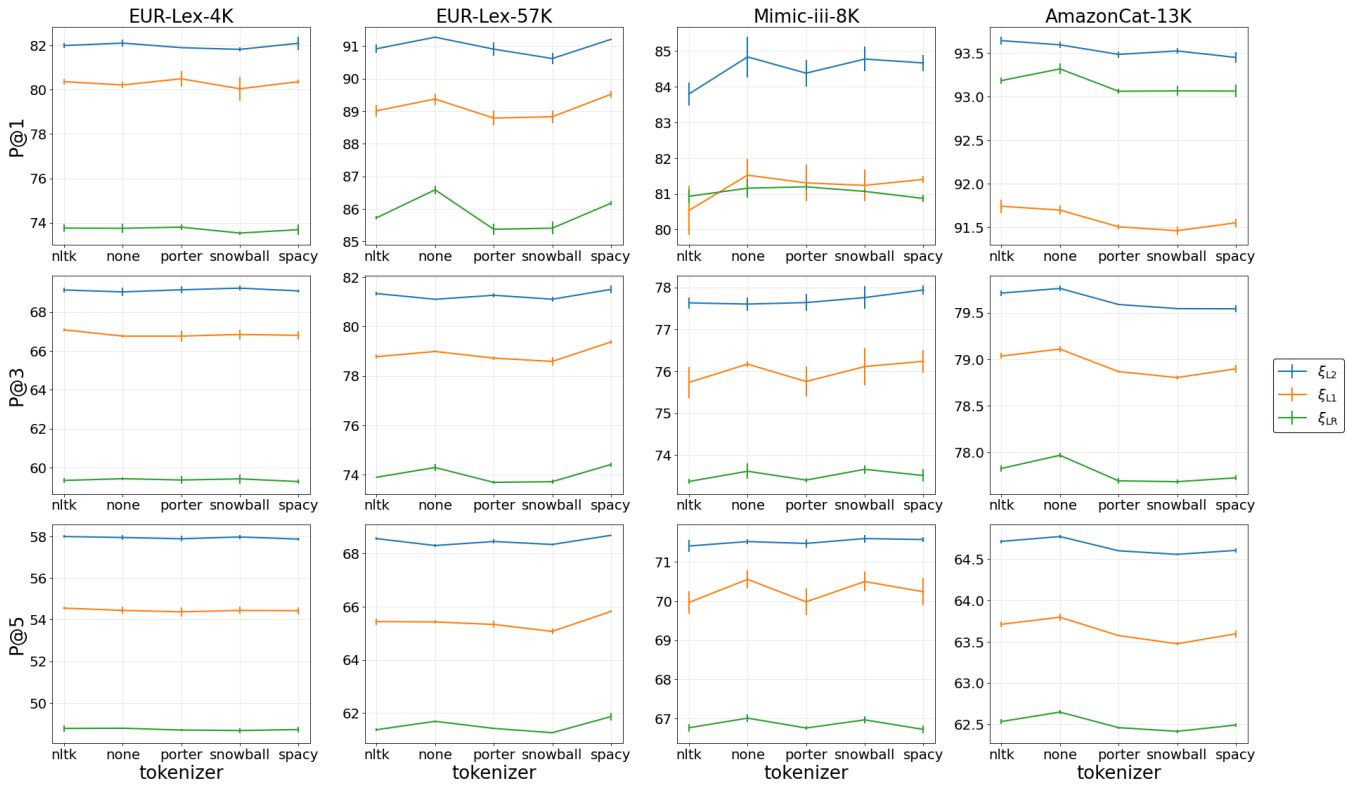


Figure VI: Sensitivity analysis of tokenizer over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

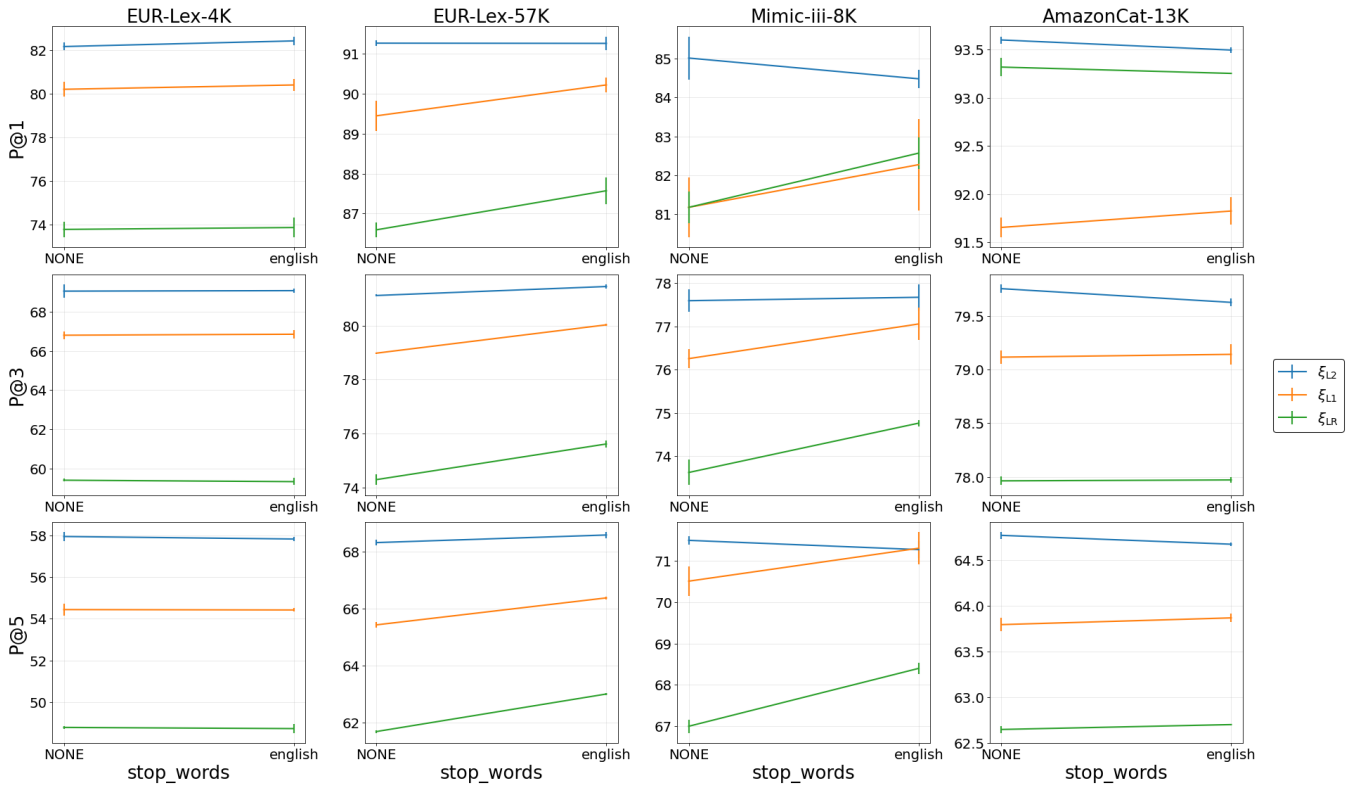


Figure VII: Sensitivity analysis of stop_words over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

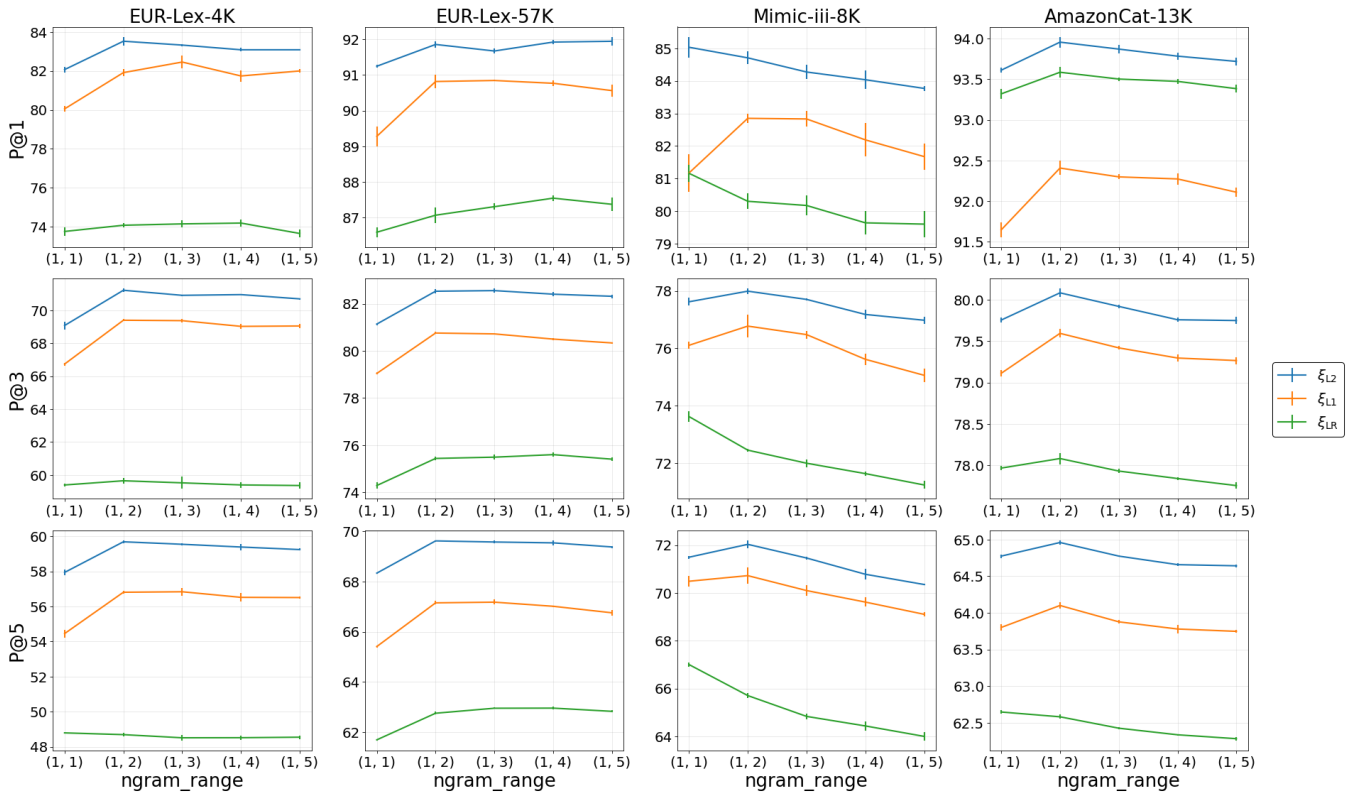


Figure VIII: Sensitivity analysis of ngram_range over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

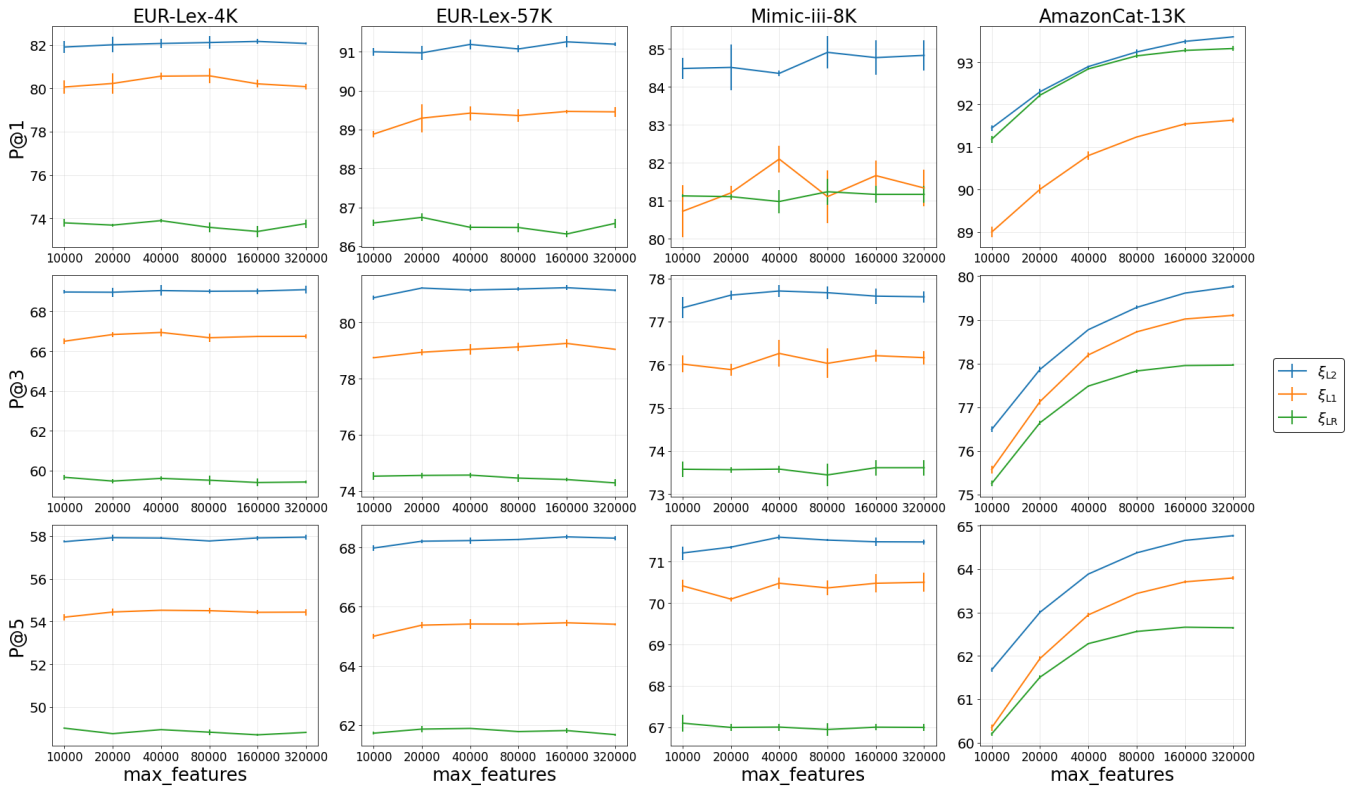


Figure IX: Sensitivity analysis of max_features over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

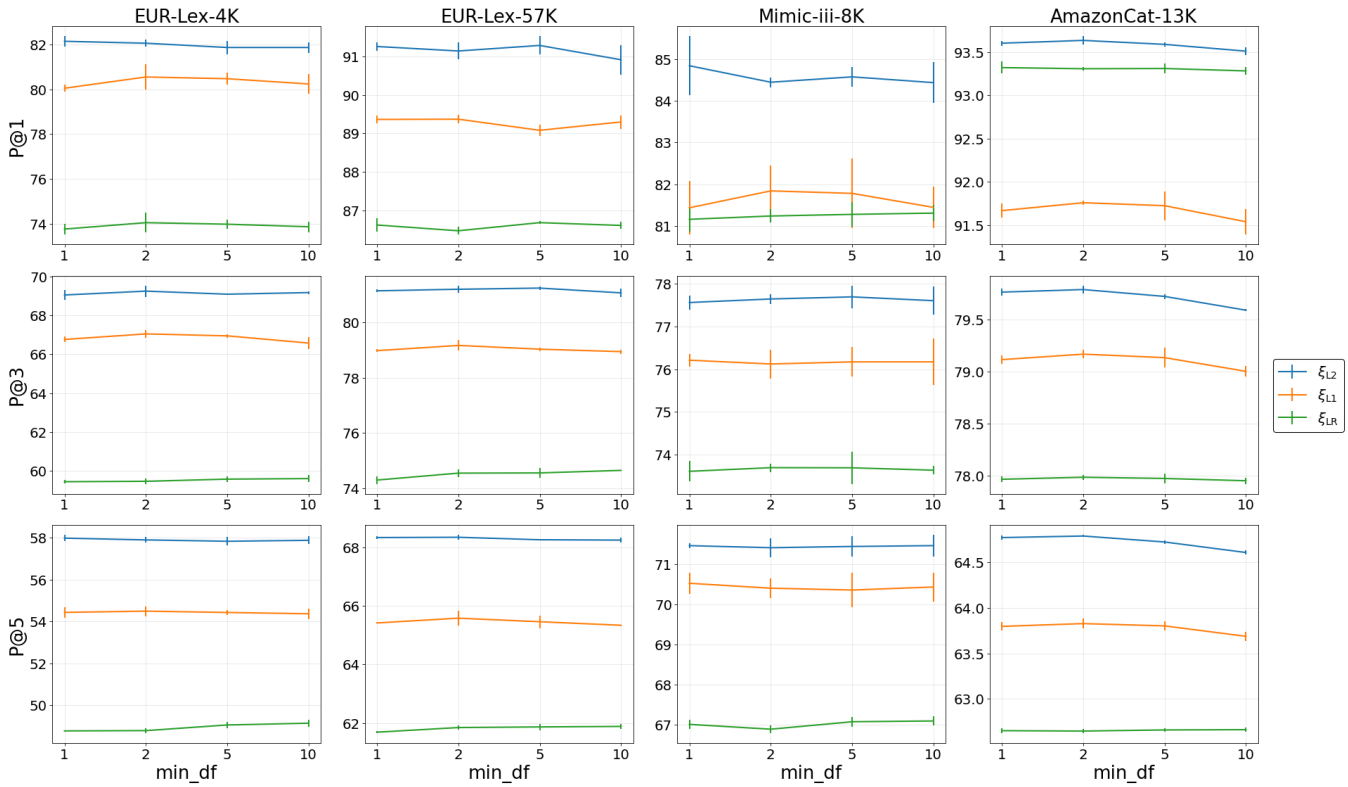


Figure X: Sensitivity analysis of min_df over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

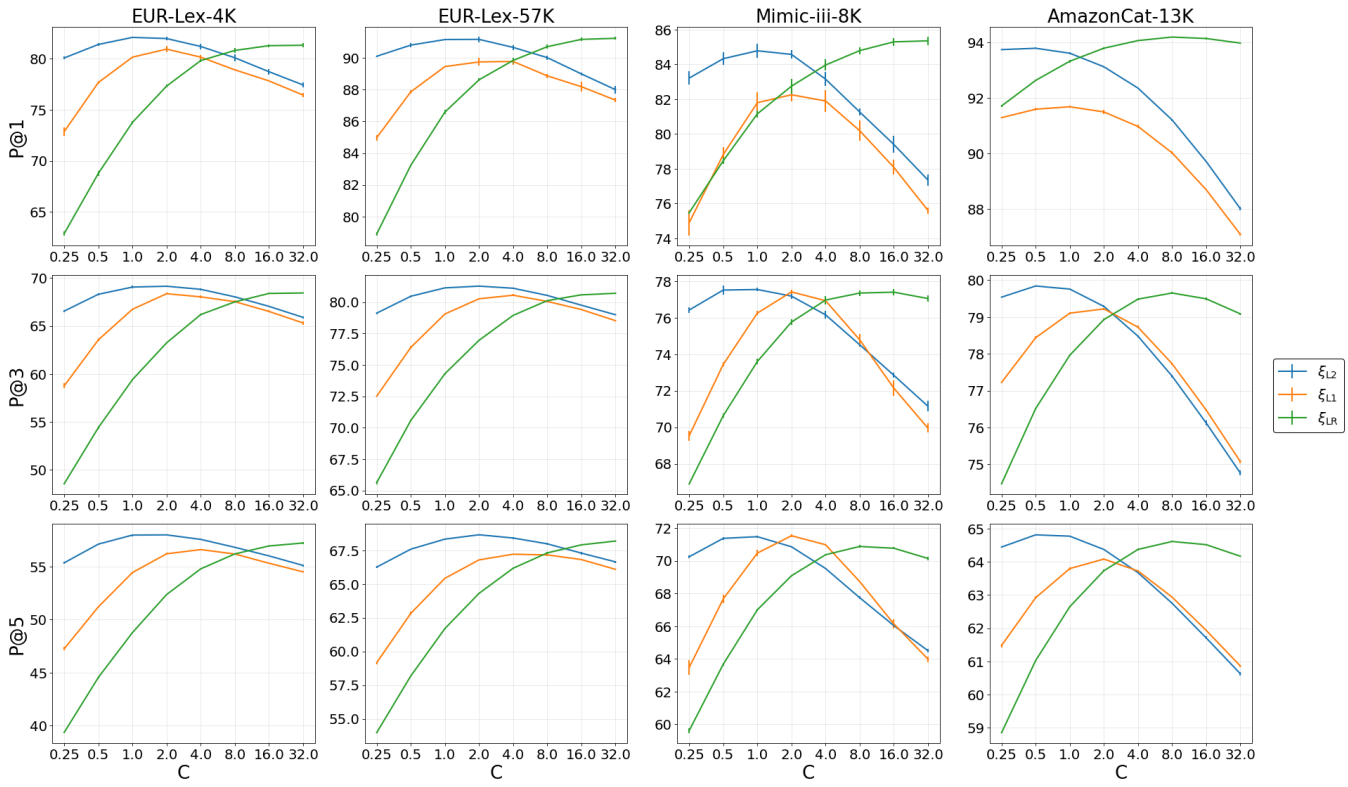


Figure XI: Sensitivity analysis of C over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

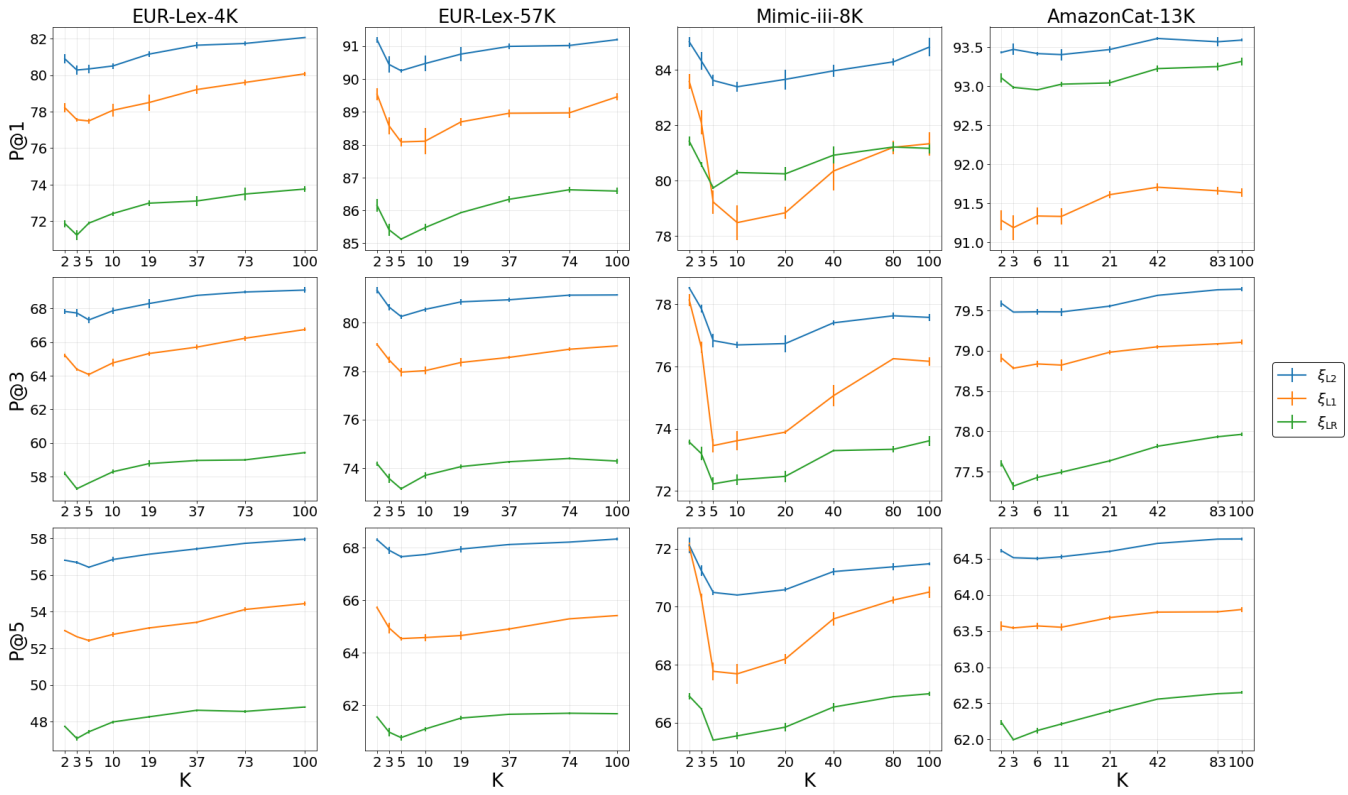


Figure XII: Sensitivity analysis of K over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

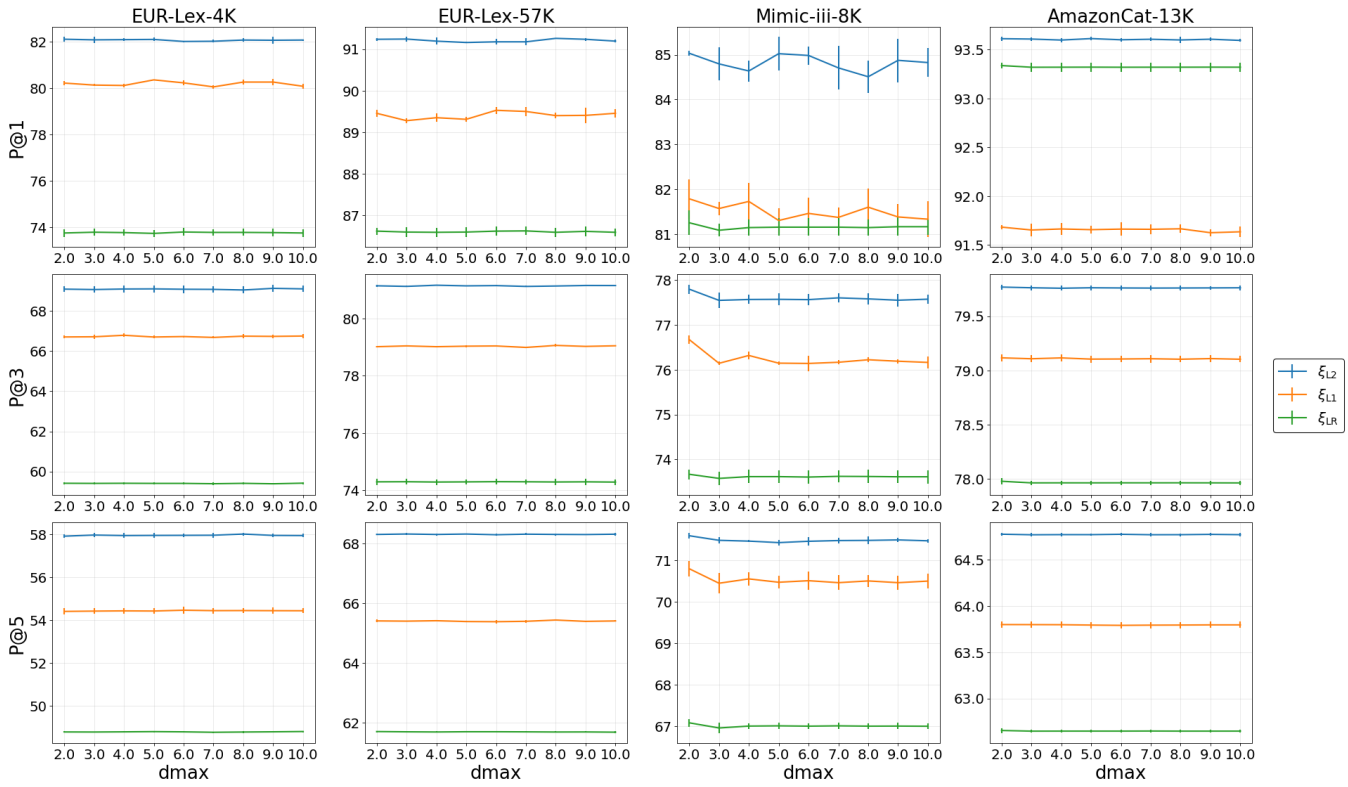


Figure XIII: Sensitivity analysis of $d_{\max}$ over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.

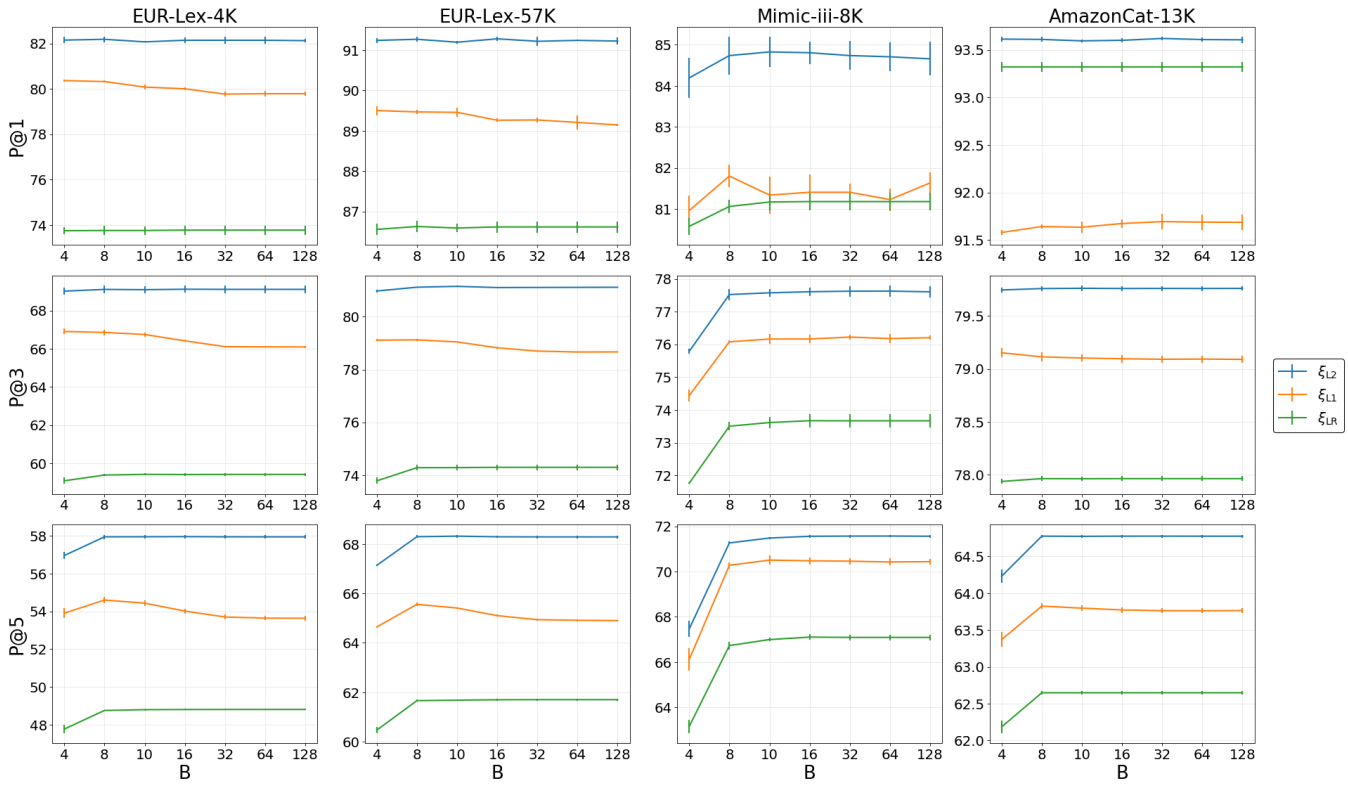


Figure XIV: Sensitivity analysis of B over EUR-Lex-4K, EUR-Lex-57K, Mimic-iii-8K, and AmazonCat-13K on P@1, P@3, and P@5. We average the results over three runs with different random seeds and plot the 95% confidence intervals as error bars.