Text2Topic: Multi-Label Text Classification System for Efficient Topic Detection in User Generated Content with Zero-Shot Capabilities

Fengjun Wang[†], **Moran Beladev**[†], **Ofri Kleinfeld, Elina Frayerman, Tal Shachar, Eran Fainman, Karen Lastmann Assaraf, Sarai Mizrachi, Benjamin Wang**
Booking.com
{fengjun.wang, moran.beladev, ofri.kleinfeld, elina.frayerman, tal.shachar, eran.fainman, karen.lastmannassaraf, sarai.mizrachi, benjamin.wang }@booking.com
[†]These authors contributed equally to this work

Abstract

Multi-label text classification is a critical task in the industry. It helps to extract structured information from large amount of textual data. We propose Text to Topic (Text2Topic), which achieves high multi-label classification performance by employing a Bi-Encoder Transformer architecture that utilizes concatenation, subtraction, and multiplication of embeddings on both text and topic. Text2Topic also supports zero-shot predictions, produces domain-specific text embeddings, and enables production-scale batch-inference with high throughput. The final model achieves accurate and comprehensive results compared to state-of-the-art baselines, including large language models (LLMs).

In this study, a total of 239 topics are defined, and around 1.6 million text-topic pairs annotations (in which 200K are positive) are collected on approximately 120K texts from 3 main data sources on Booking.com. The data is collected with optimized smart sampling and partial labeling. The final Text2Topic model is deployed on a real-world stream processing platform, and it outperforms other models with 92.9% micro mAP, as well as a 75.8% macro mAP score. We summarize the modeling choices which are extensively tested through ablation studies, and share detailed in-production decision-making steps.

1 Introduction

In the digital age, large-scale online travel platforms (OTPs) face the challenge of effectively extracting valuable insights from massive volumes of textual data. Such an OTP can get hundreds of millions of customer reviews in one year, so structured insights are crucial for comprehending customer behavior and making data-driven decisions in order to improve the overall travel experience. One application example is to find the top facilities for each hotel, by extracting information from the positive reviews, which can lead to better accommodation recommendations. Similarly, understanding travel destination themes, such as romantic getaways, city trips, or family trips can enhance destination recommendations. In this study, we research use cases from Booking.com and define in total 239 valuable topics. Each topic is set with a topic name and a topic description, to better match the natural customer language and for optimal model training results. The main data source is user-generated content on Booking.com, including customer reviews and forum posts from hotel owners and travelers.

Developing an architecture that ensures high accuracy, scalability for a large number of topics, low cost and low latency on real-world inference is of utmost importance. Sentence-BERT (Reimers and Gurevych, 2019) extends BERT (Devlin et al., 2019) for sentence-level embeddings, achieving impressive performance on tasks like sentence similarity and semantic retrieval. Multilingual Universal Sentence Encoder for Semantic Retrieval (MUSE) (Yang et al., 2019), a multilingual extension of the Universal Sentence Encoder (Cer et al., 2018), enables cross-lingual semantic retrieval and provides multiple open-source models. Though there are also other state-of-the-art approaches, the two methods above are prevalent in real industry applications, due to the computational efficiency, high and robust in-domain performance by fine-tuning, zero-shot ability, and strengths in scalability.

Our proposed Text2Topic framework adopts a fine-tuning approach upon pre-trained language models. Specifically, we employ the bi-encoder transformer (Vaswani et al., 2017) architecture proposed by Sentence-BERT, which allows separate injection of the text and topic information, as Section 2 shows. This architecture not only enables the model to have zero-shot capabilities (handle new topics for inference) but also exhibits text embedding abilities. By leveraging the strengths of the pre-trained language model and incorporating topic-specific information, the model effectively

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 93–103

addresses the challenge of topic detection. The paper's contributions are summarized as follows:

- We propose a practical Text2Topic framework for the efficient extraction of topics from texts.
- We share the model development core findings, including multiple model architectures' comparison, training one universal model versus dedicated models per data source, outperforming against baselines (MUSE, GPT-3.5).
- We share efficient and practical dataset annotation strategies with smart model-based sampling, as described in Section 3.
- We provide zero-shot capability for unseen classes, which performs better than MUSE when the unseen class is in the travel domain.
- We detail the real-world use cases in Section 6, and deployment decisions in Section 7.

2 Architectures

In this study, we research 3 main architectures. For each text-topic (with topic description), we know one binary ground truth for this pair, and perform:

- Cross-encoder: the text and topic description are tokenized and concatenated as one input with `[SEP]` separator ("`TEXT[SEP]TOPIC`"), then passed into the transformer encoder and a classification head to derive logits, where Binary Cross Entropy (BCE) loss is applied.
- Bi-encoder Concatenation (Figure 1): we first generate a pair of embeddings (U, V) both as dimension d_{model}, where U is the topic description embedding and V is the text embedding. Then we feed E (the embedding concatenation, subtraction and multiplication) into 2 feedforward layers ($FFN_1 \in R^{d_E \times d_{model}}$, ReLU activation, and dropout, and then $FFN_2 \in R^{d_{model} \times 1}$), and finally apply BCE loss.
- Bi-encoder Cosine: similar to the Figure 1, but at step 4, instead of embedding combination, we apply cosine similarity directly on U and V, and then apply mean-squared-error loss as the objective function.

The bi-encoder architecture has 3 benefits over cross-encoder: 1) Low inference time-complexity: we pre-calculate and cache all topic embeddings, only embed each text once and repeat the text vector to score on all topics. Given N as the number of texts and T as the number of topics, to get $N \times T$ predictions, the bi-encoder needs $O(N + T)$ encoding operations, while it is $O(N \cdot T)$ for the cross-encoder. 2) In-house embedding: bi-encoder enables us to have the text part embeddings, that can be used as features for other tasks. 3) For the same base model, the bi-encoder allows longer text input since text and topic are embedded separately.

All the above architectures can easily extend to include new topics for training, and also have **zero-shot** possibility when the unseen topic is well-defined with a description. For all architectures, we experiment with three pooling strategies (Reimers and Gurevych, 2019): using the output of the `[CLS]`; computing the mean or max of output vectors on all tokens (mean-pooling, max-pooling).

3 In-house Dataset Construction

With hundreds of topics, the annotation becomes challenging: how to define, merge, and distinguish topics; how to decide annotation volume and candidate texts per topic and reduce cost. This section shows how we tackle them by smart model-based sampling and partial labeling.

3.1 Annotation Volume Estimation

We start with a proof-of-concept stage, where 43 topics are pre-defined and annotated by domain experts on 12K texts. With the data, we run multiple cross-encoder model training by increasing the number of positive annotations per topic in the training data. Figure 5 in the Appendix shows that for most topics, the mAP metrics saturate at 200 positive annotations, which reflects basic guidance.

3.2 Topic Definitions

In the end, we define 239 topics from user researches, covering broader topics such as trip types (romantic trip, city trip etc.), travel activities (surfing, hiking etc.), and specific user needs such as hotel facilities (garden, balcony etc.). Each topic has a name and a description which is refined with the help from LIME (see Section 5.5).

3.3 Smart Sampling and Partial Labeling

In our corpus, text is typically short and contains low number of topics, so we apply partial labeling instead of full annotation. The 239 topics are split

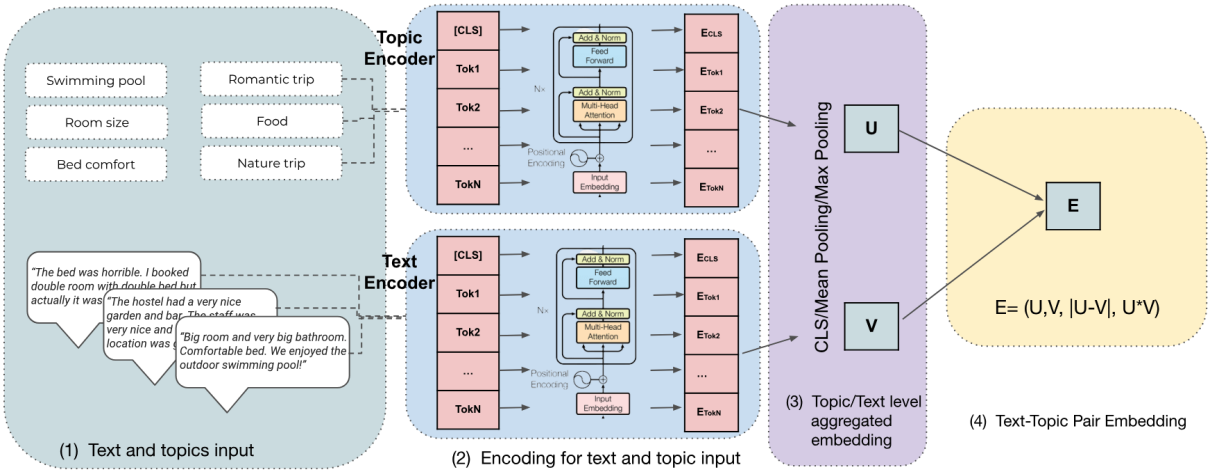

Figure 1: Bi-Encoder Concatenation. (1) Input Text-Topic pairs. (2) Encode each text and topic with transformer encoder (the two encoders share weights). (3) Aggregate each of the two text-topic token embeddings into one single vector to represent the topic (U) or the text (V) using CLS/mean-pooling/max-pooling. (4) Combine the two embeddings into one representation of the pair relationship, E. Then feed E into two feedforward layers to get logits output, where BCE loss is applied on. For inference, we broadcast topics embeddings to pair with text embeddings and score each pair.

into 38 multi-choice question groups (e.g., food topics are in one group). With the best 43-topic model (from the training as Section 3.1 shows), we apply it to predict on a large corpus, detecting 239 topics and generating text-topic scores. Notably, the prediction results for the unseen 196 topics are generated by zero-shot.

With the predictions, we perform smart text sampling: 1) Firstly, for each topic, we do probability-weighted sampling on the texts whose scores pass a threshold, and assign selected texts to the multi-choice group which contains that topic. Figure 4 in the Appendix shows an example for a text that passes the threshold for the "romantic trip" topic and was assigned with the group of topics that contains the "romantic trip" topic. 2) In addition, to avoid annotation bias, each selected text is also assigned to one random group (besides the already assigned relevant groups). For example, the text in Figure 4 is also assigned to another group randomly. 3) Besides the model-based text sampling, we also randomly sample some texts from corpus and randomly assign them to groups.

We use AWS SageMaker Ground Truth as the platform for the annotations collection, and leverage on some of the MuMIC (Wang et al., 2023) annotation pipelines and strategies (majority voting etc.). We recruit specialized annotators to form one auditing team, and multiple worker teams. After a knowledge transfer phase, we start the production phase where each task is done by 3 workers, and the labels are inferred by majority voting. The auditing team performs regular performance checks and we get > 95% annotation accuracy. Finally, we collected almost 1.6 million annotations at a low cost, with 200K of them being positive (which is 12.5%). These annotations are gathered from approximately 120K unique multilingual texts, including English, German, French, Russian, etc., from guests, travelers, and property owners, sourced from reviews, the travel community, and partner hub.

4 Experimental Setup

All experiments are performed on a computation instance equipped with 1 NVIDIA Tesla T4 Tensor Core GPU, 4 vCPU, and 16GB RAM. The experiments have the following general settings: bert-base-multilingual-cased model as the pre-trained base model; fine-tuning all layers; mixed-precision training; batch size of 12 text-topic pairs, with gradient accumulation steps as 8; weight decay (on all weights that are not gains or biases) with coefficient 0.01; AdamW optimizer (Loshchilov and Hutter, 2017); initial learning rate 1e-5 with a linear scheduler; allowing maximum 6 epochs with early stopping patience as 3 steps, and warm-up steps as 10K. We apply stratified sampling (on topic frequency) on the texts, and get training/validation/test sets, with a ratio of 70/15/15 respectively.

4.1 Evaluation Metrics

Given T topics, and N texts, the ground truth and the predictions can both be represented as a matrix with size $N \times T$ [1] . We use the below metrics as the main evaluation criterion (Sorower, 2010):

- Average Precision (AP) per class:

$$AP_j = \sum_{i=1}^{N} p_j(i)\Delta r_j(i) \quad (1)$$

  where p_j is the precision of class j, and r_j is the recall of class j. It is equivalent to the area under the precision-recall curve per class.

- macro Mean Average Precision

$$macro\, mAP = \frac{1}{T}\sum_{j=1}^{T} AP_j \quad (2)$$

  which is the unweighted average of AP on all classes, treating each class equally.

- weighted Mean Average Precision

$$weightedmAP = \frac{1}{\sum_{j=1}^{T} NP_j}\sum_{j=1}^{T} AP_j \cdot NP_j \quad (3)$$

  where NP_j is the number of positive samples of class j.

- micro Average Precision (global-based)

$$micro\, mAP = \sum_{i=1}^{N \cdot T} p(i)\Delta r(i) \quad (4)$$

4.2 Baselines

MUSE (Yang et al., 2019): Google provides multiple versions of MUSE models, and we use the "multilingual-large-3" one [2]. The model covers the languages we have in the dataset, is trained with multi-task learning on Transformer architecture, and is optimized for multi-word length text. Given a text, MUSE generates a 512-dimensional vector as the embedding. For each text-topic pair, we calculate the cosine similarity on text embedding and topic embedding as the model prediction score.
GPT-3.5: we choose the gpt-3.5-turbo-0301, which supports a maximum 4K token context length.

[1]With partial labeling, the matrix has null values, and we filter them out accordingly when do metrics calculation.

[2]https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3

method	micro mAP	macro mAP	weighted mAP
MUSE	37.9	41.2	60.4
Cross-encoder	**94.7**	**81.4**	**92.8**
Bi-encoder (cosine)	89.4	71.4	88.5
Bi-encoder (concat)	84.3	62.5	80.3
Bi-encoder (concat, sub)	91.4	72.6	89.5
Bi-encoder (concat, sub, mult)	**92.9**	**75.8**	**91.0**

Table 1: Performance comparison of Text2Topic, on the test set. All metrics are in %.

5 Experimentation Findings

5.1 Final Model Performance

Table 1 compares performance across multiple model architectures and MUSE baseline. We perform hyperparameter tuning on all methods, report the highest reachable performance and find all of them beat the MUSE baseline [3]. The cross-encoder outperforms all other architectures because it learns the topic-text relation attention layer by layer inside the transformer. Generally, the bi-encoder concatenation method is better than simple cosine similarity architecture, and the embedding subtraction and multiplication are both necessary.

It's worth mentioning that for all methods, the model training typically can saturate at around 2nd or 3rd epoch, which takes less than one day for one model training. The cross-encoder has the highest performance but with too high inference time complexity, so we choose the "bi-encoder (concat, sub, mult)" one for production and refer it as "bi-encoder concat" model in this paper.

5.2 Train One Model or Three Models?

As mentioned in Section 3.3, there are 3 data sources. In the dataset, customer reviews occupy more than 70% data, and have almost all 239 topics, while the other 2 sources both have less than 100 topics. Should we train one model on all data or 3 models per dataset? Table 2 shows the ablation results - under the same modeling set-up, we train and evaluate models on each source. Training on all datasets yields the best performance, we expect the model to learn patterns from all sources and gain better generalization ability. This decision also makes the model management easier.

[3]In Text2Topic hyperparamter tuning, we find the bi-encoder cosine architecture prefers mean-pooling, while bi-encoder concat models prefer [CLS] embedding.

	train on all data (all)			train on customer review (rev)			train on partner hub (ph)			train on travel community (tc)		
	macro mAP	micro mAP	weighted mAP	macro mAP	micro mAP	weighted mAP	macro mAP	micro mAP	weighted mAP	macro mAP	micro mAP	weighted mAP
all	**71.4**	**89.4**	**88.5**	66.1	87.0	86.6	33.5	23.1	53.4	32.6	34.9	55.4
rev	**71.7**	**90.3**	**89.4**	70.8	90.1	**89.4**	32.3	22.5	54.7	31.3	34	55.3
ph	**65.2**	73.2	72.8	28.2	17.5	35.1	58.7	**73.7**	**72.9**	27.4	21.1	39.2
tc	**57.9**	68.7	70.4	40.9	34.4	49.1	27.9	20.6	41.3	56.4	**72.7**	**72.0**

Table 2: Cross-dataset model training experimentation results, on test set, with bi-encoder cosine. We mark the highest scores as bold for each evaluation metric.

method	**macro mAP**	**weighted mAP**	**macro F1 score**
MUSE	41.2	60.4	46.4
Bi-encoder (cosine)	35.8	64.1	41.3
Bi-encoder (concat, sub, mult)	**46.8**	**71.1**	**51.1**

Table 3: Zero-shot overall test-set performance on all topics. We search the best F1 across all thresholds per topic, and then get macro averaged F1 across topics.

5.3 Zero-Shot Evaluation

We randomly split all topics into 5 groups, and then each time train 4 groups and evaluate the zero-shot ability on the remaining one group. Table 3 provides an overall performance comparison, and we see bi-encoder concat model performs the best. In the Appendix, Figure 6 and Figure 7 depict topic-level performance, where the bi-encoder concat has the best zero-shot ability in most topics, and Table 4 in the Appendix shows the aggregated performance on popular topics. In general, we can say that the Text2Topic keeps a balance between learning new capabilities and exposing existing capabilities.

5.4 Comparison with GPT-3.5

Considering the GPT-3.5 context length, we select 24 topics for the evaluation, covering 3 representative groups: food, trip types, and room conditions. With multiple prompt iterations, we find the few-shot prompting is necessary because it can regulate output format by showing examples. We finally get two best prompts: 8-shot and 38-shot. Both prompts include the 24 topic definition list with descriptions, and Chain-of-Thought (CoT) (Wei et al., 2023) rules: ask the model to quote each part of the text, infer topics, and then output a topic list. Besides topic description and CoT rules, the 8-shot prompt (around 1700 tokens) has 3 text examples, covering 8 positive annotations on 8 topics; while the 38-shot (around 2900 tokens) has 16 examples, covering 38 positive annotations on 24 topics.

Figure 2 shows that Text2Topic (our bi-encoder concat model) performs the best in almost all topics, and the 38-shot prompt slightly outperforms the 8-shot. This indicates that when already having clear topic descriptions in the prompt, adding more examples is not always powerful. In our case, Text2Topic is a better choice because of: 1) less dependency on non-open-source models, so that the model iteration and rate limits are under-control; 2) avoiding tedious prompt tuning procedures; 3) a lower cost and it's more eco-friendly: the Text2Topic model has less than 200M parameters (considering we cache the topics embedding during inference). If the GPT-3.5 has 175B parameters, it would be more than 800 times bigger. As described in Section 7.2, Text2Topic can reach 8000 text/min throughput, with a $7.5 cost predicting on 1M text, while GPT-3.5 would cost $6250 (assuming the prompt and text have 3.5K tokens, the output (CoT and topic list result) has 500 tokens) [4]. 4) better scalability for larger number of topics and less worrying about prompting length exceeding certain limitation. Though we can split topics into multiple groups/prompts and do multiple calls on GPT-3.5, it means a higher cost and the group split setting is difficult to optimize.

5.5 Model Interpretation

We use Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016) for model interpretation. LIME generates local explanations by perturbing individual text instances, approximating the model behavior by using a surrogate model that highlights the importance of words in the original text. LIME is effective in the error-analysis flow, and help topic description refinements at an early stage (see example in Figure 3 in the Appendix).

6 Real-World Applications

This section describes 3 main real-world use cases which employ Text2Topic. Besides, the Text2Topic

[4] OpenAI price on gpt-3.5-turbo-0301 is $0.0015 / 1K input tokens + $0.0020 / 1K output tokens, when writing this paper.

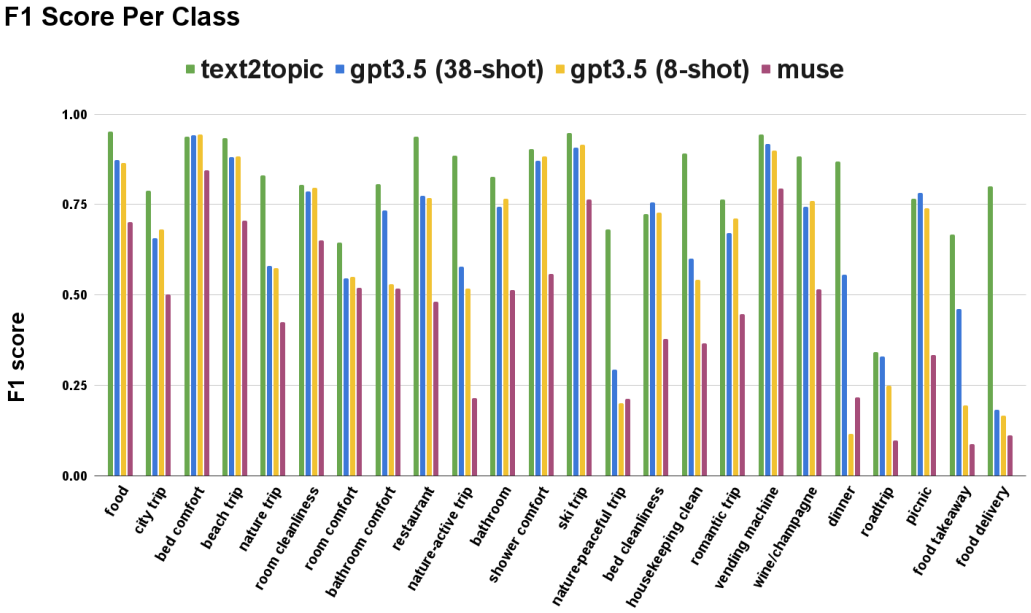

Figure 2: F1 score on each topic, on the test set. The x-axis represents the 24 topics, ordered by topic popularity (support-1). For Text2Topic and MUSE, we search the best F1 score across all thresholds for each topic.

training system is also re-used effectively for training other data sources like search queries.

It is important to note that use cases may require varying threshold settings. We use the F-beta score (Goutte and Gaussier, 2005) to determine the optimal threshold setting on the topic's probability score for a given use case. For recall-oriented scenarios, where minimizing false negatives is critical, we set beta > 1. Conversely, for precision-oriented cases, where reducing false positives is a priority, we set beta < 1. For each topic, to find the best threshold, we systematically vary its value by computing the threshold that yields the highest F-beta score, using the chosen beta value.

Property Recommendation: Reviews contain rich information that encapsulates the users' preferences towards different properties. Text2Topic turns them into structured features, which enhance the in-house property recommendation models' performance. With classification scores on reviews, we perform property-level score aggregations, to extract a variety of insightful attributes such as a property's relevancy for different themes (e.g., beach, spa/wellness). These attributes are integrated into the recommendation models to increase relevant inventory (e.g., number of beach properties is increased by 4% by leveraging Text2Topic); and to create novel and nuanced categories of recommendations (such as castle-type hotels, romantic getaways, etc.). Furthermore, the model provides a natural mechanism to serve explainable recommendations by linking them to relevant reviews.

Detect Property Type: With Text2Topic predictions on reviews, we are able to detect hidden properties categorizations, by analyzing relevant topic (guest house, farm stay, resort, chalet etc.) frequencies. For example, an Apartment property that is described as a Guest house, could be surfaced to users that are searching for a guest house. We detect over a million extra properties supply (774K more apartments, 25k more villas and 60k more cabins/chalets).

Fintech: Text2Topic training pipeline enables us to train a new model on Fintech data and topics, such as payments and questions about invoices and commissions. The model auto-classifies incoming messages from customers and correctly re-routes them to the right self-service solution, which increases the auto-reply success rate by 9% and reduces manual handling time.

7 Deployment

7.1 The Deployment Platform

The model is deployed and monitored on a stream processing platform based on Apache Flink (Katsifodimos and Schelter, 2016). It consumes real-time events from Kafka (Kreps et al., 2011) topics to

generate model-based predictions. It automatically scales up the number of model endpoints to better handle peak times and allows leveraging Apache Flink's asynchronous I/O operator to perform concurrent asynchronous HTTP calls to the model endpoint. The architectural design allows the platform to be also used for backfilling (scoring historical data with newly deployed model), by simulating events of historical data and pushing them to Kafka. The platform is designed to achieve high prediction throughput while keeping a low latency.

7.2 Model Serving and Batch Invocations

To maximize the hardware utilization (NVIDIA Tesla V100 GPUs for production), we combine batch model invocations with Flink's native asynchronous I/O support. For batch model invocations, we leverage Flink's windowing mechanism to implement the grouping of events that will be sent to the model together in a single API call. Events are accumulated to windows as soon as they become available for consumption from the source Kafka topic. The window is closed after a predefined time period (e.g. 3 seconds), or whenever the number of accumulated events reaches the desired batch size.

Aiming to minimize the cost per prediction, we start with grid searching the optimal batch size by performing stress tests using a single model endpoint. For each batch size we randomly sample batches of texts from the corpus, and then iterate over the batches sequentially and invoke the model. We observe that while increasing the batch size, the throughput (number of texts predictions per minute) first increases, and then starts decreasing as the number of available GPU cores exhausted. An optimal and memory-explosion safe batch size is 300. Then we run experiments to compare batch invocations against asynchronous I/O invocations. As Table 5 in the Appendix shows, using synchronous API calls with a batch size of 300 yields a cost of $7.5 for 1M predictions, while using 50 concurrent asynchronous I/O API calls without batching yields a cost of $15. So the former is selected. In addition, for backfilling, texts with similar lengths are grouped together and we apply dynamic padding to the longest element in each batch, which reduces computational overhead.

8 Conclusion

In this paper, we present Text2Topic, a flexible multi-label text classification system that is deployed at Booking.com, with high performance and supports multiple applications. We summarize lessons learnt from the end-to-end production journey, including practical annotation approaches, modeling choices, and production decisions, which are valuable references for the industry domain. We also compare the performance with LLM like GPT-3.5, and Text2Topic is a more feasible choice from multiple aspects. For future work, we can explore if parameter efficient fine-tuning techniques (e.g., LoRA (Hu et al., 2021), p-tuning (Liu et al., 2022)) on open-source LLMs could bring better performance, and how to better balance the model specialization and generalization power for zero-shot.

Acknowledgments

This work is supported by Booking.com. We would like to thank Satendra Kumar, Selena Wang, Michael Alo, and Guy Nadav for the paper review. We would also like to thank Ilya Gusev on contributing some GPT-3.5 prompting ideas.

References

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Cyril Goutte and Eric Gaussier. 2005. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European conference on information retrieval*, pages 345–359. Springer.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Asterios Katsifodimos and Sebastian Schelter. 2016. Apache flink: Stream analytics at scale. In *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, pages 193–193.

Jay Kreps, Neha Narkhede, Jun Rao, et al. 2011. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, volume 11, pages 1–7. Athens, Greece.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

Mohammad S Sorower. 2010. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18:1–25.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Fengjun Wang, Sarai Mizrachi, Moran Beladev, Guy Nadav, Gil Amsalem, Karen Lastmann Assaraf, and Hadas Harush Boker. 2023. Mumic–multimodal embedding for multi-label image classification with tempered sigmoid. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 15603–15611.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Multilingual universal sentence encoder for semantic retrieval.

A Appendix

method	macro mAP	weighted mAP	macro F1 score
MUSE	54.4	58.3	57.6
Bi-encoder (cosine)	47.4	62.4	51.6
Bi-encoder (concat, sub, mult)	**58.1**	**68.5**	**60.0**

Table 4: Zero-shot overall test-set performance on popular topics which have more than 50 positive annotations each in the test set. We search the best F1 across all thresholds per topic, and then get macro averaged F1 across topics.

Batch Size	#Concurrent Calls (Async I/O)	Throughput (#texts per minute)	$USD/1M Predictions
1	50	4,000	$15
300	3	7,200	$8.3
300	1	8,000	**$7.5**

Table 5: Comparing batch model invocations and Async I/O approach. Maximizing the batch size doubles the model invocation throughput and reduces the prediction costs by half. Combining too many asynchronous calls with a large batch size exhausted the GPU resources, which resulted in a reduced throughput compared to pure batch invocations.

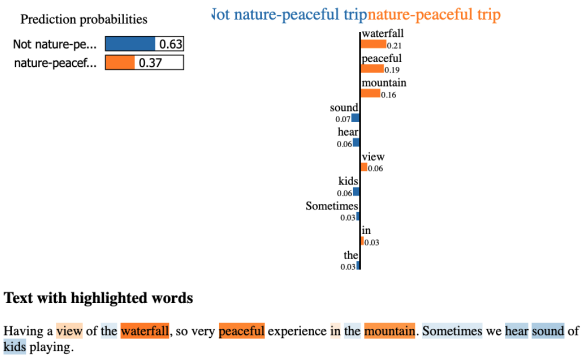

Figure 3: LIME explanation for a specific text-topic pair. We can see word-level importance in detecting "nature-peaceful trip" topic. Orange color indicates positive influence (words like: view, waterfall, peaceful, mountain) and Blue color indicates negative influence (words like: kids, sound). LIME explanation helps human to refine topic descriptions, for example by removing unclear wording, or adding more precise and concise wording, at the early stage of this project.

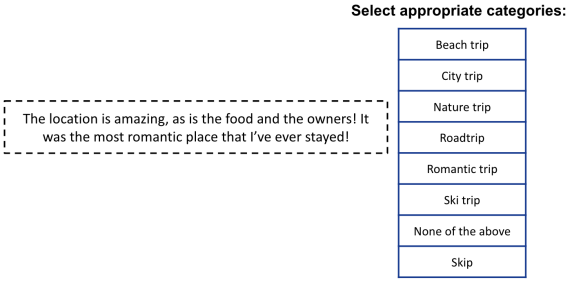

Figure 4: Example of the topics group assigned to a text in an annotation task. The annotator can select multiple topics.

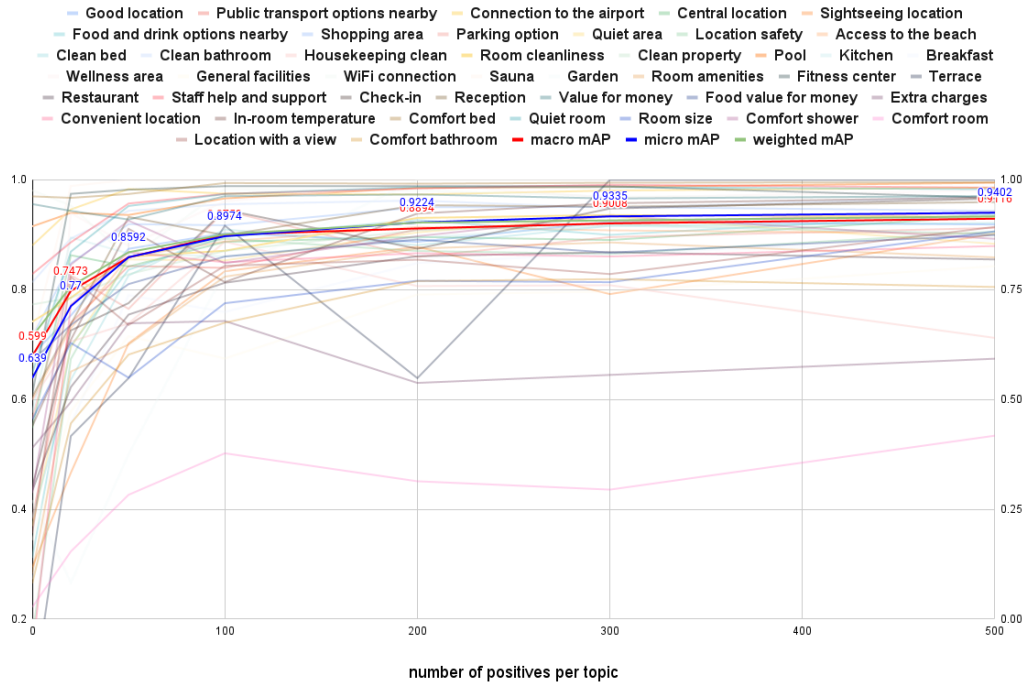

Figure 5: Performance tracking when sampling different number of positive annotations per topic for model training. This provides a general guide: for most topics, 200 number positive annotations is enough. However, for training hundreds of topics in one model, we might need more positive annotations per topic, so we also consider it when deciding on the annotation volume.

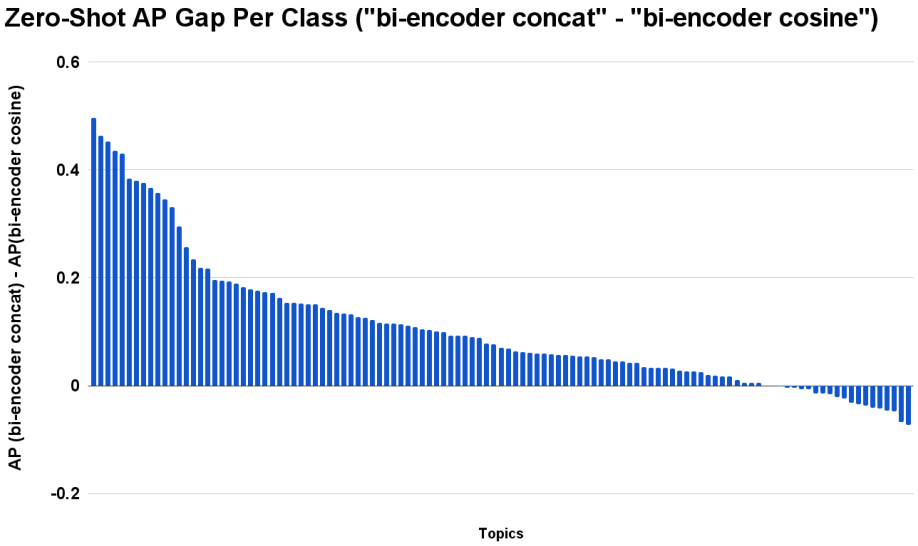

Figure 6: Zero-shot Average Precision score gap on each topic, on the test set. The x-axis represents the topics, ordered by the score gap. The y-axis shows the gap between the AP score of "bi-encoder concat" model and "bi-encoder cosine" model. This plot includes the popular topics which have more than 50 positive annotations each in the test set. We can see the cosine one is almost always worse than the concat one.

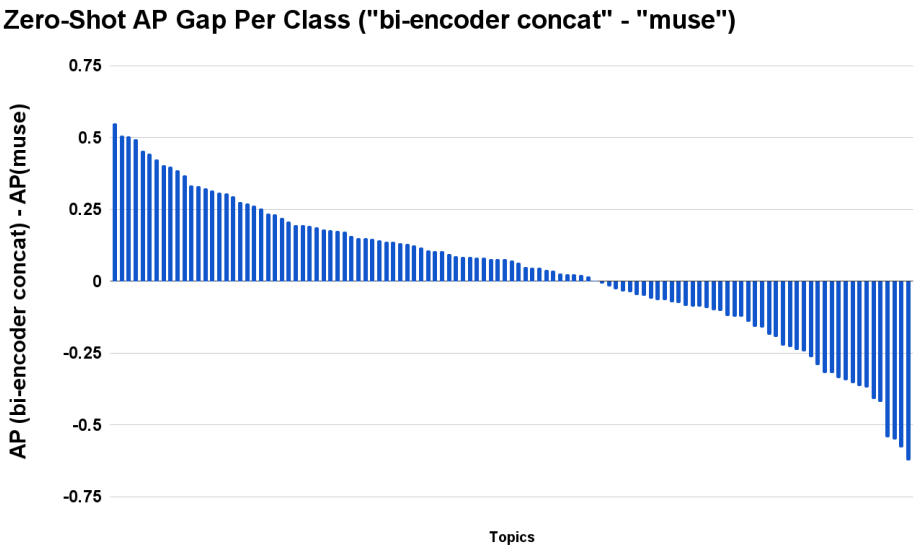

Figure 7: Zero-shot Average Precision score gap on each topic, on the test set. The x-axis represents the topics, ordered by the score gap. The y-axis shows the gap between the AP score of "bi-encoder concat" model and MUSE. This plot includes the popular topics which have more than 50 positive annotations each in the test set. We can see that when inspecting topic-level performance, the Text2Topic bi-encoder concat has stronger zero-shot ability than MUSE. MUSE is better at some topics, which we find are mainly facility specific topics (BBQ, towel, vending machine, stairs etc.).