TREC-ToT: Endicott and UNC Notebook Paper

Henry Feild
Computer Science Department
Endicott College
hfeild@endicott.edu

Jaime Arguello
School of Information and Library Science
University of North Carolina at Chapel Hill
jarguello@unc.edu

ABSTRACT

Tip-of-the-tongue (ToT) known-item retrieval involves retrieving a previously encountered item for which the searcher is unable to reliably recall an identifier. The TREC 2023 ToT track focused on an ad-hoc retrieval task in the movie identification domain. The Endicott and UNC team submitted four runs to the track. Our baseline run used BM25, while our three experimental runs used a "boosted" version of BM25 that weighed query-terms differently. All ToT queries used in the track had sentence-level annotations based on the topics and language phenomena found in the sentence. Our three experimental runs weighed query-terms depending on the sentence-level categories associated with the sentence from which each query-term originated. One experimental run weighed query-terms using gold-standard sentence-level categories. The other two used predicted categories. Across all metrics considered, our three experimental runs outperformed our baseline run by a statistically significant margin. Differences between experimental runs were not statistically significant across metrics. Our results suggest that sentence-level categories were predicted with sufficient accuracy to inform the re-weighing of query-terms to improve retrieval performance.

1 INTRODUCTION

The Endicott and UNC group submitted four runs to the TREC 2023 ToT Track. All runs used Lucene version 9.6.0. Our baseline run (endicott_unc_baseline) used BM25. Our three experimental runs leveraged the sentence-level annotations described in Arguello et al. [1]. These sentence-level annotations describe the topics and language phenomena associated with each sentence in the query. For example, some topical categories describe aspects of the movie referrenced in the sentence (e.g., the plot, a scene, a character, etc.). Other topical categories describe aspects of the context in which the movie was seen (e.g., time, location, medium, etc.) Non-topical categories describe language phenomena present in the sentence. For example, some non-topical categories describe whether the sentence mentions uncertainty, draws comparisons, or mentions the searcher's emotional response to the movie.

Our three experimental runs are motivated by the hypothesis that certain query-terms are more important than others based on the categories associated with the sentence from which they originated. For example, perhaps query-terms from sentences about the plot of the movie are more important for retrieval than query-terms from sentences about a scene in the movie. After all, the document corpus consisted of Wikipedia pages, which tend to describe the movie's plot and not specific scenes in the movie. Our three experimental runs leveraged Lucene's ability to process weighted queries.

Our first experimental run (endicott_unc_boost_oracle) leveraged *gold-standard* sentence-level annotations. Categories were grouped into an up-boost list and a down-boost list. Query-terms from sentences associated with at least one category in the up-boost list were given a weight of 1.0. Conversely, query-terms from sentences associated with only categories in the down-boost list were down-boosted by some factor in the range $[0, 1]$.

Our second experimental run (endicott_unc_boost_pred) leveraged *predicted* sentence-level annotations. Sentence-level categories were predicted using a series of weighted $k-$nearest neighbor (KNN) classifiers (one per category). Query-terms were weighted following the same approach described above. That is, query-terms from sentences associated with only categories in the down-boost list were down-boosted by some factor in the range $[0, 1]$.

Our third experimental run (endicott_unc_boost_conf) also used predicted sentence-level annotations. Query-terms from sentences associated with only categories in the down-boost list were given a weight of 0.0. That is, those query-terms were entirely ignored. Conversely, query-terms from sentences associated with at least one category in the up-boost list were up-boosted by some factor in the range $[0, 1]$. This factor was proportional to the highest prediction confidence value associated with categories in the up-boost list.

2 PRELIMINARIES

All of the runs we report on were retrieved using Lucene version 9.6.0 over an index that included the *text* field of each Wikipedia page in the ToT corpus. We used Lucene's *StandardAnalyzer*,[1] which performs case folding and stop-word removal.

All queries were downcased and runs of non-alphanumeric characters were replaced with a space. We used Lucene's *BM25Similarity*[2] scorer with $k = 0.8$ and $b = 1$ based on the parameter sweep results reported by the TREC ToT organizers during their benchmarking.[3] Three of our four runs rely on query-clause boosting. Lucene boosts by multiplying document scores for each query-term by the given boosting factor[4] (see Equation 2). All queries included the topic title with a boost of 1.0. We did not *explicitly* specify boosts of 1.0 as that is the default boost value used by Lucene.

3 ALGORITHMS

3.1 Overview

Our group submitted four runs to the TREC 2023 ToT track. Our baseline run (endicott_unc_baseline) used BM25. Our three experimental runs used a "boosted" version of BM25 that weighs query-terms differently. Query-terms were assigned weights based

[1]https://lucene.apache.org/core/9_6_0/core/org/apache/lucene/analysis/standard/StandardAnalyzer.html
[2]https://lucene.apache.org/core/9_6_0/core/org/apache/lucene/search/similarities/BM25Similarity.html
[3]https://github.com/TREC-ToT/bench/
[4]https://lucene.apache.org/core/9_6_0/core/org/apache/lucene/search/package-summary.html#package.description

on the sentence-level categories associated with the sentence from which the query-term originated. One of our three experimental runs (endicott_unc_boost_oracle) used gold-standard sentence-level categories. Our other two experimental runs used *predicted* sentence-level categories. As described in Section 3.2, sentence-level categories were predicted using a weighted k-nearest neighbor approach. One of our experimental runs (endicott_unc_boost_pred) used predicted categories in a binary fashion. The other experimental run (endicott_unc_boost_conf) used predicted categories by also considering the classifier's confidence value.

3.2 Predicting Sentence-Level Annotations

All train, development, and test queries used in the TREC 2023 ToT Track included sentence-level annotations based on the qualitative analysis reported in Arguello et al. [1]. Sentences were classified along 39 categories based on the topics and language phenomena found in the sentence. Appendix A describes these sentence-level categories. Categories 1-24 belong to the general category "movie" (i.e., the sentence describes something about the movie itself) and categories 25-31 belong to the general category "context" (i.e., the sentence describes something about the context in which the movie was seen). Categories were designed to be non-mutually exclusive. That is, a sentence can be associated with zero, one, or more than one category. Our three experimental runs leveraged sentence-level categories. One run (i.e., endicott_unc_boost_oracle) leveraged ground-truth categories and the other two runs leveraged predicted categories. In this section, we describe how sentences were classified into categories.

Sentences were classified into categories using 39 independent, binary classifiers (one per category). For each category, we used a weighted k-nearest neighbor (KNN) approach. Sentence were represented using embeddings provided by the Open AI API[5] and semantic similarity between sentences was measured using cosine similarity.

For a given sentence $\mathcal{S}_{\text{test}}$ and category C, our goal was to output a confidence value in the range [-1,+1]. Values closer to +1 indicate greater confidence that C applies to $\mathcal{S}_{\text{test}}$ and values closer to -1 indicate greater confidence that C does not apply to $\mathcal{S}_{\text{test}}$. Confidence values were generated using the following formula:

$$\sum_{\mathcal{S}_i \in \text{KNN}(\mathcal{S}_{\text{test}})} \text{CAT}(\mathcal{S}_i, C) \times \frac{\text{SIM}(\mathcal{S}_{\text{test}}, \mathcal{S}_i)}{\sum_{\mathcal{S}_j \in \text{KNN}(\mathcal{S}_{\text{test}})} \text{SIM}(\mathcal{S}_{\text{test}}, \mathcal{S}_j)}, \quad (1)$$

where $\text{KNN}(\mathcal{S}_{\text{test}})$ denotes the k-nearest neighbors for sentence $\mathcal{S}_{\text{test}}$ in the training data, $\text{CAT}(\mathcal{S}, C)$ indicates whether sentence $\mathcal{S}$ belongs to category C (i.e., -1 or +1), and $\text{SIM}(\mathcal{S}_{\text{test}}, \mathcal{S})$ denotes the cosine similarity between $\mathcal{S}_{\text{test}}$ and $\mathcal{S}$.

Sentences in the test set were classified by combining all sentences in the training and development sets as a single training set. For each category, we tuned parameter k using 10-fold cross-validation and optimized for F1 classification performance.

We were curious about classification performance across categories. To this end, we conducted experiments by classifying all sentences in the development set using sentences in the training set. For each category, we tuned parameter k using 10-fold cross-validation and optimized for F1 classification performance. Table 1 shows performance in terms of precision, recall, and F1. The second and third columns show the frequency of each category in sentences from the training and development set, respectively. As expected, performance was higher for categories that are topical and common.

[5]https://platform.openai.com/docs/guides/embeddings/

3.3 Baseline

Our baseline run (endicott_unc_baseline) used BM25 and included the topic title and all sentences following the processing described in Section 2. Specifically, we used the implementation of BM25 provided with Lucene 9.6.0. Our experimental runs (Sections 3.4-3.6) used a "boosted" version of BM25 in which query-terms were weighted differently depending on the categories associated with the sentence from where the query-term originated. This boosted version of BM25 scores documents according to:

$$\text{BM25}(Q, D) = \sum_{i=1}^{n} \mathcal{B}(q_i) \cdot \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)},$$

where

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right).$$

The multiplier $\mathcal{B}(q_i)$ denotes the amount of boosting associated with query-term q_i. Our baseline run always set $\mathcal{B}(q_i) = 1$.

3.4 Boosting with Oracle Annotations

For our first experimental run (endicott_unc_boost_oracle), we used the gold-standard, oracle sentence-level categories to determine how to boost the terms from each sentence in the final query along with the topic title. Given an up-boost list of categories, terms from sentences associated with at least one category in the up-boost list were given the default boost of 1.0 and terms from other sentences were down-boosted. To determine the up-boost list of categories and the down-boost value, we performed a two-dimensional parameter sweep.

As a first pass, we considered down-boost values between 0.0 and 1.0 in increments of 0.1. For each down-boost value, we used a simple greedy approach to select which categories to include in the up-boost list. First, we considered the NDCG@1000 performance of each category in isolation (i.e., it was the only category in the up-boost list for that run). Then, we considered three additional runs where the up-boost list consisted of the top-two, top-three, and top-four best-performing individual categories. On the union of the training and development data, we found that a down-boost value of 0.0 had the best performance. Therefore, we narrowed our sweep to consider down-boost values between 0.0 and 0.1 in increments of 0.01. On this sweep, we found that 0.0 was still the best performing down-boost value.

Across all down-boost values, we found NDCG@1000 performance to be highest when including only one or two categories in the up-boost list. Ultimately, by optimizing for NDCG@1000 performance on the union of the training and development set, our

Table 1: Classification performance across sentence-level categories.

	training set freq.	dev set freq.	precision	recall	F1
movie	944	896	0.934	0.984	0.959
character	594	560	0.865	0.918	0.891
social	125	115	0.808	0.878	0.842
temporal	112	102	0.768	0.843	0.804
scene	502	407	0.719	0.887	0.794
category	307	297	0.772	0.798	0.785
context	129	126	0.775	0.794	0.784
production_visual	50	58	0.760	0.655	0.704
origin_language	43	35	0.628	0.771	0.692
hedging	294	389	0.752	0.568	0.647
physical_medium	29	51	0.828	0.471	0.600
object	255	299	0.639	0.545	0.588
location_type	126	199	0.675	0.427	0.523
release_date	34	68	0.765	0.382	0.510
genre_traditional_tone	25	54	0.680	0.315	0.430
search	16	19	0.375	0.316	0.343
genre_audience	8	5	0.250	0.400	0.308
location_specific	11	23	0.455	0.217	0.294
music_compare	4	3	0.250	0.333	0.286
plot	44	133	0.545	0.180	0.271
opinion	17	19	0.235	0.211	0.222
production_audio	6	3	0.167	0.333	0.222
origin_movie	3	23	0.667	0.087	0.154
timeframe_singular	6	10	0.167	0.100	0.125
comparison_relative	35	27	0.086	0.111	0.097
situational_witness	16	6	0.063	0.167	0.091
emotion	0	3	0.000	0.000	0.000
music_specific	0	1	0.000	0.000	0.000
person_fictional	2	8	0.000	0.000	0.000
negation	1	15	0.000	0.000	0.000
person_real	2	16	0.000	0.000	0.000
production_camera_angle	9	11	0.000	0.000	0.000
origin_actor	4	5	0.000	0.000	0.000
quote	7	26	0.000	0.000	0.000
timeframe_plural	1	7	0.000	0.000	0.000
physical_user_location	0	5	0.000	0.000	0.000
situational_count	0	1	0.000	0.000	0.000
cross_media	5	12	0.000	0.000	0.000
situational_evidence	0	2	0.000	0.000	0.000

endicott_unc_boost_oracle run used an up-boost value of 1.0, a down-boost value of 0.0, and an up-boost list that included two categories: *character* and *scene*. As in all our other runs, every query included the topic title with a default up-boost value of 1.0.

The following is an example of the Lucene query generated for Topic 162. Only three sentences were associated with categories in the up-boost list (i.e., *character* and *scene*) and were thus included in the final query (see Table 2). Giving a sentence a boost value of 1.0 is equivalent to not specifying a boost value at all. Thus, the query simply includes the topic title and three up-boost sentences without any up-boost values specified. All other sentences were excluded.

Topic 162 Lucene query: `a group of teens kids maybe did some kind of experiment and brought these tv characters to life a group of teens kids maybe did some kind of experiment and brought these tv characters to life and then were killed by them it was an old movie like 80s if not before i can only remember two of the characters they brought to life one was a purple monster thing and the other was like some sort of doctor i remember this one scene when one of the kids were killed because one of the characters made him laugh to death`

Sentence	*scene* Gold	*character* Gold	Conf	Pred class	Scaled conf
"a group of teens(kids maybe) did some kind of experiment and brought these tv characters to life and then were killed by them it was an old movie like 80s if not before."	T	T	-0.22333	0	0
"i saw it a while back."	F	F	-0.86156	0	0
"i can only remember two of the characters they brought to life, one was a" purple monster thing and the other was like some sort of doctor."	T	T	0.44558	1	0.45558
"i remember this one scene when one of the kids were killed because one" of the characters made him laugh to death."	T	T	0.11019	1	0.12019
"the film was a gore kind of film to."	F	F	-0.75090	0	0
"it was a terrible movie but one of those random ones youd find on ion or scifi or spike tv on an off day lol."	F	F	-0.83503	0	0
"please help me figure it out"	F	F	-0.84598	0	0

Table 2: The gold-standard annotation (Gold), prediction confidence (Conf), binary prediction classification (Pred class), and scaled prediction confidence (Scaled conf) for each sentence in Topic 162 titled "a group of teens(kids maybe) did some kind of experiment and brought these tv characters to life" from the test set for the *scene* and *character* categories (*scene* was only used in the oracle run, so we've left out the prediction-based columns). A threshold of 0 was used in calculating the classification and scaled confidence for the *character* category.

3.5 Boosting with Predicted Annotations

Our second experimental run (endicott_unc_boost_pred) is very similar to our endicott_unc_boost_oracle run. However, our endicott_unc_boost_pred run used predicted (versus gold-standard) sentence-level categories. Sentence-level categories were predicted as described in Section 3.2. For each category, we used a weighted k-nearest neighbor (KNN) classifier to output confidence values in the range $[-1, +1]$. Turning these confidence values into binary predictions requires setting a threshold (e.g., 0.0). Rather than using a default threshold of 0.0, we used category-specific thresholds by optimizing for F1 classification performance on sentences in the union of the training and development set. For example, Table 2 shows the prediction confidence value of each sentence in Topic 162 with respect to the *character* category and the corresponding binary classification decision.

Once the predicted categories were established, we used the same procedure described in Section 3.4 to determine which categories to include in the up-boost list and to determine the down-boost value. Our two-dimensional parameter sweep found very similar results. Ultimately, by optimizing for NDCG@1000 performance on the union of the training and development set, our endicott_unc_boost_pred run used an up-boost value of 1.0, a down-boost value of 0.0, and an up-boost list that only included the *character* category. Similar to our endicott_unc_boost_oracle run, we found that excluding sentences without a category in the up-boost list performed better than down-boosting them by some value greater than 0.0. As in all our other runs, every query included the topic title with a default up-boost value of 1.0.

The following is an example of the Lucene query generated for Topic 162. As shown in Table 2, the first sentence belongs to the *character* category. However, the prediction confidence value was not above the predefined threshold. Therefore, while this sentence was included in our endicott_unc_boost_oracle run, it was excluded in our endicott_unc_boost_pred run.

Topic 162 Lucene query: `a group of teens kids maybe did some kind of experiment and brought these tv characters to life i can only remember two of the characters they brought to life one was a purple monster thing and the other was like some sort of doctor i remember this one scene when one of the kids were killed because one of the characters made him laugh to death`

3.6 Weighted Boosting with Predicted Annotations

The run file that we submitted for our final experimental run (endicott_unc_boost_conf) was incorrect. It was actually the same as the run file that we submitted for our baseline run (endicott_unc_baseline). In this section, we describe what this run should have been. In Section 4, we report on results for the correct version of our endicott_unc_boost_conf run.

This experimental run was similar to the previous with one major difference. As in the previous approach, sentences without a category in the up-boost list were assigned a down-boost value of 0.0 (i.e., were effectively removed from the query). However, sentences with a predicted category in the up-boost list were not assigned an up-boost value of 1.0. Instead, the up-boost value was proportional to the prediction confidence value for the most confident up-boost category.

The approach proceeded as follows. First, sentences were assigned to categories in a weighted fashion. Let $\text{KNN}(\mathcal{S}, \text{C})$ denote the confidence value that sentence $\mathcal{S}$ belongs to category C (Equation 1). Additionally, let $\mathcal{T}_C$ denote the category-specific threshold used in our endicott_unc_boost_pred approach to make binary classification decisions with respect to C. If $\text{KNN}(\mathcal{S}, \text{C}) < \mathcal{T}_C$, then sentence $\mathcal{S}$ was assigned a weight of 0.0 with respect to category C. Otherwise, the weighted membership between $\mathcal{S}$ and C was set according to:

$$\frac{\text{KNN}(\mathcal{S}, \text{C}) - \mathcal{T}_C}{1.0 - \mathcal{T}_C} + 0.01 \quad (2)$$

This is a form of min-max scaling. It measures the extent to which $\text{KNN}(\mathcal{S}, \text{C})$ exceeds $\mathcal{T}_C$.

Finally, sentences with a predicted category in the up-boost list were up-boosted as follows. If a sentence had one predicted category in the up-boost list, this sentence was up-boosted according to its weighted membership to that category. Conversely, if a sentence had multiple predicted categories in the up-boost list, this sentence was up-boosted according to the *maximum* weighted membership across those categories. The 0.01 in Equation 2 was included to prevent assigning sentences with memberships at the margin (i.e., $\text{KNN}(\mathcal{S}, \text{C}) = \mathcal{T}_C$) an up-boost of 0.0.

As with our two previous approaches, we used the same greedy approach to select which categories to include in the up-boost list. As with the previous run, including only the *character* category in the up-boost list resulted in the best performance on the combined training and development set. As in all our other runs, every query included the topic title with a default up-boost value of 1.0.

The following is an example of the Lucene query generated for Topic 162. As shown in Table 2, based on the prediction confidence value with respect to the *character* category, the third sentence is up-boosted by 0.45558, which is less than half the up-boost value using oracle categories or binary predictions. The fourth sentence has an even lower up-boost value of 0.12019.

Topic 162 Lucene query: `a group of teens kids maybe did some kind of experiment and brought these tv characters to life (i can only remember two of the characters they brought to life one was a purple monster thing and the other was like some sort of doctor) ^ 0.45558 (i remember this one scene when one of the kids were killed because one of the characters made him laugh to death) ^ 0.12018999999999999`

4 RESULTS

Table 3 shows results from our four runs in terms of NDCG@10, NDCG@1000, mean reciprocal rank (MRR), and recall@1000. We used a two-tailed Fischer's randomization test [2] to test for statistically significant differences between all pairs of runs across metrics. Across all metrics, our three experimental runs outperformed our baseline run (BM25) by a statistically significant margin. All differences between our three experimental runs were not statistically significant.

Our results show four main trends. First, by comparing between our three experimental runs (i.e., endicott_unc_boost_oracle, endicott_unc_boost_pred, and endicott_unc_boost_conf) with our baseline run (i.e., endicott_unc_baseline), we can see that ignoring query-terms from sentences associated with specific categories improves retrieval performance for ToT queries. Second, this improvement is more pronounced for metrics that focus on the top results. This can be seen by comparing the percent improvement over our baseline approach across metrics. Percent improvement was higher for NDCG@10 (i.e., 36-45%) than NDCG@1000 (i.e., 29-34%). Similarly, percent improvement was higher for MRR (i.e., 37-44%) than recall@1000 (i.e., 16-25%). Third, our KNN classifier

Table 3: Results for our four runs. Percentage values correspond to percent improvement over our baseline run (BM25). Symbol ▲ denotes statistically significant improvements of our baseline run (BM25). No other differences between runs were statistically significant across metrics.

NDCG@10	
endicott_unc_baseline	0.0749
endicott_unc_boost_oracle	0.1018 (36%)▲
endicott_unc_boost_pred	0.1089 (45%)▲
endicott_unc_boost_conf	0.1033 (38%)▲

NDCG@1000	
endicott_unc_baseline	0.1116
endicott_unc_boost_oracle	0.1439 (29%)▲
endicott_unc_boost_pred	0.1516 (36%)▲
endicott_unc_boost_conf	0.1492 (34%)▲

MRR	
endicott_unc_baseline	0.0663
endicott_unc_boost_oracle	0.0907 (37%)▲
endicott_unc_boost_pred	0.0954 (44%)▲
endicott_unc_boost_conf	0.0925 (40%)▲

recall@1000	
endicott_unc_baseline	0.3667
endicott_unc_boost_oracle	0.4267 (16%)▲
endicott_unc_boost_pred	0.4533 (24%)▲
endicott_unc_boost_conf	0.4600 (25%)▲

was able to successfully classify sentences in order inform which query-terms should be ignored in order to improve retrieval performance. Across all metrics, retrieval performance using predicted categories (i.e., endicott_unc_boost_[pred|conf]) was statistically indistinguishable from the approach that used gold-standard categories (i.e., endicott_unc_boost_oracle). Finally, there is much room for improvement. Based on recall@1000, our approaches were able to rank the correct answer in the top-1000 results for fewer than half of all test queries.

We were also curious to see whether are runs performed well for the same or different queries. Table 4 show Pearson correlation values between pairs of runs across metrics. All correlation values were high and statistically significant ($p < .05$). This trend suggests that our runs performed well on the same queries.

5 CONCLUSION

We described four submissions to TREC 2023 ToT Track: a baseline run (endicott_unc_baseline) and three experimental runs that boosted associated with specific categories (endicott_unc_boost_oracle, endicott_unc_boost_pred, endicott_unc_boost_conf). When incorporating sentence-level category annotations, we found a statistically significant increase in all the measures used (NDCG@10, NDCG@1000, MRR, and recall@1000), with particularly substantial improvements for NDCG@10, NDCG@1000, and MRR.

Our results suggest that discarding sentences associated with certain categories removes query-terms that diverge the relevant

Table 4: Pearson correlation value between pairs of runs. All correlation values are statistically significant ($p < .05$)

NDCG@10

	endicott_unc_baseline	endicott_unc_boost_oracle	endicott_unc_boost_pred	endicott_unc_boost_conf
endicott_unc_baseline	1.00	0.87	0.80	0.79
endicott_unc_boost_oracle	0.87	1.00	0.89	0.88
endicott_unc_boost_pred	0.80	0.89	1.00	0.91
endicott_unc_boost_conf	0.79	0.88	0.91	1.00

NDCG@1000

	endicott_unc_baseline	endicott_unc_boost_oracle	endicott_unc_boost_pred	endicott_unc_boost_conf
endicott_unc_baseline	1.00	0.89	0.86	0.84
endicott_unc_boost_oracle	0.89	1.00	0.93	0.92
endicott_unc_boost_pred	0.86	0.93	1.00	0.93
endicott_unc_boost_conf	0.84	0.92	0.93	1.00

MRR

	endicott_unc_baseline	endicott_unc_boost_oracle	endicott_unc_boost_pred	endicott_unc_boost_conf
endicott_unc_baseline	1.00	0.87	0.83	0.83
endicott_unc_boost_oracle	0.87	1.00	0.90	0.92
endicott_unc_boost_pred	0.83	0.90	1.00	0.91
endicott_unc_boost_conf	0.83	0.92	0.91	1.00

recall@1000

	endicott_unc_baseline	endicott_unc_boost_oracle	endicott_unc_boost_pred	endicott_unc_boost_conf
endicott_unc_baseline	1.00	0.74	0.72	0.63
endicott_unc_boost_oracle	0.74	1.00	0.89	0.77
endicott_unc_boost_pred	0.72	0.89	1.00	0.83
endicott_unc_boost_conf	0.63	0.77	0.83	1.00

document and thus improves precision. Improvements were less pronounced in terms of recall@1000. This suggests that further improvements may require introducing additional terms that are not present in the query. As a future direction, we are considering combining sentence-level category information with a query expansion approach.

REFERENCES

[1] Jaime Arguello, Adam Ferguson, Emery Fine, Bhaskar Mitra, Hamed Zamani, and Fernando Diaz. 2021. Tip of the Tongue Known-Item Retrieval: A Case Study in Movie Identification. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval (CHIIR '21)*. Association for Computing Machinery, New York, NY, USA, 5–14. https://doi.org/10.1145/3406522.3446021

[2] Mark D. Smucker, James Allan, and Ben Carterette. 2007. A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management (CIKM '07)*. Association for Computing Machinery, New York, NY, USA, 623–632. https://doi.org/10.1145/1321440.1321528

A SENTENCE-LEVEL CATEGORIES

The following list describes the 37 sentence-level categories developed by Arguello et al. [1].

(1) category: describes the movie's category (e.g., movie, tv movie, miniseries, etc.)
(2) character: describes a character in the movie.
(3) genre_audience: describes the movie's target audience (e.g., for kids).
(4) genre_traditional_tone: describes the movie's genre or tone (e.g., romantic comedy).
(5) location_specific: describes a specific location in the movie (e.g., the boy lives with his mom in Arizona).
(6) location_type: describes a type of location in the movie (e.g., a European castle).
(7) music_compare: describes the movie's soundtrack (e.g., lots of electronic music).
(8) music_specific: describes a song in the movie (e.g., the main character sings "Looking for the Heart of Saturday Night").
(9) negation: uses negation to describe aspects of the movie in negative terms (e.g., not scary, but a bit weird).
(10) object: describes a tangible object in the movie (e.g., they're in a car that almost crashes into a beast).
(11) origin_actor: describes the nationality or ethnicity of actors/actresses in the movie.
(12) origin_language: describes languages spoken in the movie.
(13) origin_movie: describes the movie's region of origin.
(14) person_fictional: references a fictional character (e.g., the main character looks like Indiana Jones).
(15) person_real: references a real person (e.g., the main character looks like Harrison Ford).
(16) plot: describes the movie's plot.
(17) production_audio: describes characteristics of the audio (e.g., badly dubbed).

(18) production_camera_angle: describes camera movements (e.g., the camera suddenly cuts to the monster under the bed)
(19) production_visual: describes the movie's visual production (e.g., black and white).
(20) quote: describes a quote from the movie.
(21) release_date: describes the movie's release date.
(22) scene: describes a scene from the movie.
(23) timeframe_plural: describes the passage of time in the movie (e.g., decades later, the house is believed to be haunted).
(24) timeframe_singular: describes a time period in the movie (e.g., set in the 1920's).
(25) cross_media: describes exposure to the movie through other media (e.g., trailer, DVD cover, poster, etc.)
(26) physical_medium: describes the physical medium through which the movie was seen (e.g., on late-night TV).
(27) physical_user_location: describes the physical location in which the movie was seen (e.g., I watched it in film class).
(28) situational_count: describes the number of times the movie was seen (e.g., I watched the series once a week).
(29) situational_evidence: describes evidence used to recall contextual information (e.g., I watched it around 2006 because I watched it alongside Hard Candy).
(30) situational_witness: describes other people who watched the movie (e.g., with my 6-year old nephew).
(31) temporal: describes when the movie was seen (e.g., I rented it in the early 2000's).
(32) prevous_search: the sentence describes previous attempts to re-find the movie.
(33) opinion: the sentence describes an opinion about some aspect of the movie.
(34) emotion: the sentence describes an emotional response to the movie.
(35) hedging: the sentence includes mentions of uncertainty (e.g., I think it was released in the early 2000's).
(36) social: the sentence includes a social nicety (e.g., thanks in advance!).
(37) comparison_relative: the sentence describes something in relative terms (e.g., the movie stars someone who looks like Brad Pitt) versus absolute terms (e.g., the movie stars Brad Pitt).