Query Facet Mapping and its Applications in Streaming Services: the Netflix Case Study

Sudeep Das*
sdas@netflix.com
Netflix Inc.
Los Gatos, California, USA

Ivan Provalov*
iprovalov@netflix.com
Netflix Inc.
Los Gatos, California, USA

Vickie Zhang*
vzhang@netflix.com
Netflix Inc.
Los Gatos, California, USA

Weidong Zhang*
weidongz@netflix.com
Netflix Inc.
Los Gatos, California, USA

ABSTRACT

In an instant search setting such as Netflix Search where results are returned in response to every keystroke, determining how a partial query maps onto broad classes of relevant entities or *facets* — such as videos, talent, and genres — can facilitate a better understanding of the underlying objective of that query. Such a query-to-facet mapping system has a multitude of applications. It can help improve the quality of search results, drive meaningful result organization, and can be leveraged to establish trust by being transparent with Netflix members when they search for an entity that is not available on the service. By anticipating the relevant facets with each keystroke entry, the system can also better guide the experience within a search session. When aggregated across queries, the facets can reveal interesting patterns of member interest. A key challenge for building such a system is to judiciously balance lexical similarity with behavioral relevance. In this paper, we present a high level overview of a Query Facet Mapping system that we have developed at Netflix, describe its main components, provide evaluation results with real-world data, and outline several potential applications.

CCS CONCEPTS

• **Information systems** → **Users and interactive retrieval**; **Recommender systems**; *Retrieval models and ranking*.

KEYWORDS

search, intent detection, user intent

ACM Reference Format:
Sudeep Das, Ivan Provalov, Vickie Zhang, and Weidong Zhang. 2022. Query Facet Mapping and its Applications in Streaming Services: the Netflix Case Study. In *Proceedings of the 45th Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3536330

*All authors contributed equally to this research.

SIGIR '22, July 11–15, 2022, Madrid, Spain.

ACM ISBN 978-1-4503-8732-3/22/07.
https://doi.org/10.1145/3477495.3536330

1 INTRODUCTION

Search plays a key role in streaming media services by providing a mechanism for members to explore the vast repositories of content. Search helps members fetch an item on demand and discover new content [2, 4]. At Netflix, while the majority of content discovery (i.e. first time engagement) happens on the home page powered by recommendation systems, a sizable amount of discovery originates from Search. Although videos are the primary assets ranked by search, not all queries are lexically related to a video; e.g., we observe queries such as **`docume`** and **`bruce le`**. Neither necessarily matches a title; but intuitively it is likely that the former is about the genre *Documentaries*, while the latter is seeking titles featuring the actor *Bruce Lee*. Other queries (e.g. **`game of th`**) might be for videos that are not available to stream on Netflix at the time of the query (*Game of Thrones*). Most short queries, such as **`wes`** could fan out to multiple classes — the member could be searching for the title *West Side Story*, the genre *Western*, or the talent/director *Wes Anderson*. The Netflix search system currently indexes at least three classes of entities: Video (any stream-able asset), Talent (actors, directors, creators, etc.), and Collection (genre, language, etc.). Within each class, and given a country's licensing agreements, a subset of entities can be present (in-catalog or `IC`) or absent (out-of-catalog or `OOC`) from our catalog at a given point of time. This leads to a taxonomy of six classes which we formally call Query Facets: `IC-Video`, `OOC-Video`, `IC-Talent`, `OOC-Talent`, `IC-Collection`, and `OOC-Collection` (see, Fig. 1). By building a system that can map partial or full queries to broad categories of entities — videos (available and unavailable), talent, and collections — we can anticipate the overall objective of the query, and use that knowledge to power effective contextual experiences [5]. This was the motivation behind the Query Facet Mapping (QFM) system.

2 QUERY FACET MAPPING

As shown in Fig. 1, if lexical similarity were the only measure of relevance between a query and an entity, a query facet may end up with a long tail of entities that are not all behaviorally relevant. For example, someone typing in the query **`anim`** is much more likely to be looking for the genre *anime* than an obscure movie that has the fragment *anim* in its title. The behavioral relevance may also play out differently in different countries and languages [7]. For example, the query **`el`** in Spanish may be strongly related to titles starting with the article *el* (meaning *the* in Spanish, e.g. *El Chapo*)

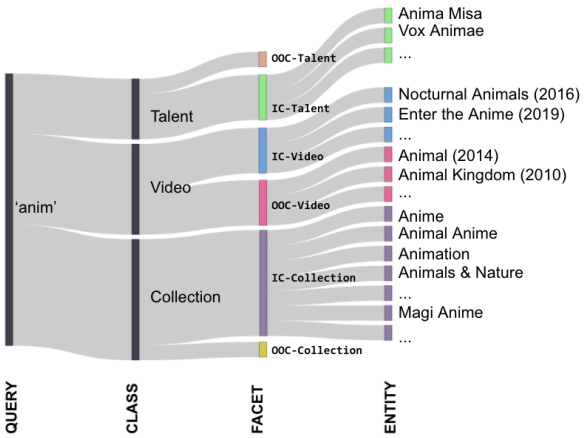

Figure 1: Query Facets: Synthetic illustration of how the query `anim` might map to the six predefined query facets. The leaf nodes show entities within each facet that are lexically related. Some facets may be empty, while some others, e.g. `IC-Collection` may have a long tail of lexically matching entities, not all of which may be relevant from the user engagement point of view. The QFM model combines lexical similarity with engagement signals to provide a probability estimate of how likely a query is to map to each of the facets, as well as a relevance score of each entity within each facet.

while the same query in English might be relevant to a different set of titles (e.g. *Elf*). As such, the relevance of entities to a query context: `(query, country, language)`, henceforth denoted by Q, cannot be purely determined by lexical similarity alone, and has to be readjusted based on member engagement signals. This is the key concept at the heart of the QFM model. In the following, we describe the how the engagement signal and lexical similarity are blended to produce the final QFM model

2.1 Member Engagement

For a given query context, we measure member engagement of each entity that is lexically similar to the query to determine its relevance to the query context. The Netflix Search UI is partitioned into two sections - the video gallery, and the suggestion section. (see Fig. 2). The video gallery section is the main section of the search result, where a grid of streamable videos are shown. In the suggestion section, entities like OOC videos, collections and talent are displayed as clickable links to allow for further exploration of the catalog. Because the videos in the video gallery section are ranked by a relevance model with relaxed matching constraints, they are not restricted only to videos that are lexically similar to the query. Therefore, these videos could either be lexically matching, such as *Scary Movie* for the query **`scary`**, or nonlexical, but thematically related, such as *The Conjuring*. Without any lexical requirement, Fig. 3 shows a synthetic example of how user engagement from queries that begin with **`anim`** might flow into each facet. Note that

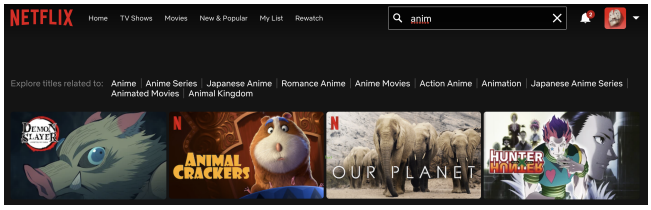

Figure 2: Netflix Search UI on the Web.

for the same query **`anim`**, there are entities that are included in Fig. 1, but not in Fig. 3. The reason is that there is a long tail of entities that are lexically similar to the query but are not engaged with by our members. While the entities that are engaged with by our members but are not lexically similar are not considered by the QFM system, they are useful for validating the performance of the system. A query with a high percentage of member engagement coming from entities that are not lexically similar should be less likely to match to the `IC-Video` facet, e.g., Fig. 3 shows query **`anim`** matches strongly to `IC-Collection` and the entity *Anime*.

In order to increase the accuracy as well as to make sure that the facet determination is smooth as a member continues to type more characters, a simple query expansion is implemented wherein engagement from a query is also attributed to the prefixes of that query. For example, the query **`anim`** is a prefix of the following queries: [**`anima`**, **`animal`**, **`anime`**], therefore, engagement from these queries would also be attributed back to the query **`anim`**. For a given query context Q, and an entity e, we apply these methods to define a member engagement based behavioral similarity score, $\mathcal{B}(Q, e)$, which is 0 when the entity e is not behaviorally relevant for the context Q, and can be interpreted as an engagement based popularity of the entity e given the query context Q.

2.2 Lexical Similarity

When it comes to providing a lexical score of a match between a query and a short title like a movie name, or a person name, common search engine scoring algorithms like TF/IDF [6] retrieval function may not be the best choice. In the majority of the cases, a query match is partial (e.g. for a show named *Orange is the New Black*, a query could be just **`ora`** or **`new b`**). In what follows, we discuss an alternative to the TF/IDF function for short title-like document match scoring, that we call the "N-gram[1] Query to Entity Match" score or NQEM. The NQEM score is a weighted combination of three measures of similarity: *percent match*, *arrangement*, and *quality*, which are described below.

2.2.1 Percent Match. While term frequency could be a good representation of the query term present in a full text document, for the short entity names an n-gram based scoring has to be based on the n-gram length if we want to get the percent of each term matching against a given entity. In addition, given some of the normalization modifications of the string, it is important to keep these modifications accounted for in the overall score. This was the motivation for using the *n-gram query percent match*, Π as a

[1]N-Gram is an output of generating the contiguous sequence of n characters out of each query or entity term.

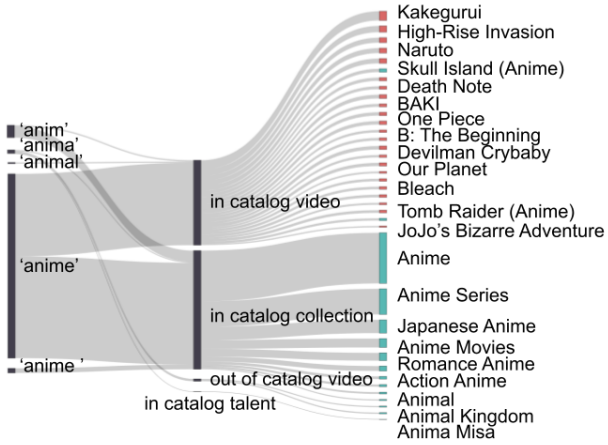

Figure 3: A synthetic example of how user engagement from queries beginning with `anim` is aggregated to the predefined query facets. Note `OOC-Collection` and `OOC-Talent` are not included here since there are no entities from these facets for members to engage with. The size of each node corresponds to the volume of user engagement. The leaf nodes show entities that users engaged with – those in green are entities that are lexically related to the query, while those in red are the entities that are not lexically related to the query.

entity	query	percent_match	startness	orderness	tightness	synonym	partial	mapped
Animated TV Shows	animated t	0.67	1	1	1	0	1	0
Animated TV Shows	animated television	0.73	1	1	1	1	0	0
犯罪ヒューマンドラマ	ハンザイ	0.20	1	1	1	0	0	1

Table 1: Main components of the N-gram Query to Entity Match (NQEM) score.

basis of the lexical score, defined as, $\Pi = \sum_{i=1}^{n} \lambda_n / D$, where λ_n is the adjusted n-gram length of the n-th matched query term and D is a document length (measured in characters).

2.2.2 Arrangement. When matching queries against short title-like documents, each character counts, and other attributes of the match become important, such as the arrangement of the match. Given a genre name, *Animated TV Shows*, and a query **`animated t`**, one way to describe the match is to list the arrangement attributes: "Startness" [2], "Tightness"[3], and "Orderness"[4]. These arrangement measures are then combined into a final score.

[2]Measures whether the match starts from the beginning of the string.
[3]Describes whether there is a gap between two matched terms measured in non-query matching terms.
[4]Describes whether the query term order matches the order of the entity terms.

2.2.3 Quality. In a similar way, we can describe the quality of each matching token. We can use the following: "Partial"[5], "Synonym"[6], and "Mapped" (ASCII Folding[7], Stemmed[8], or Phonetic Mapping[9]).

Table. 1 shows some examples of the scores described above. A "partial" flag for the first query "animated t" indicates that only a part of the word "TV" matched the entity "Animated TV Shows". A synonym flag for the second query "animated television" reveals that "television" matched the term "TV" as a synonym for the same entity. The Japanese query example contains the Kana characters mapping to the Kanji entity characters.

2.2.4 N-gram Query to Entity Match (NQEM) Score. The final score for lexical similarity between the query and the entity, namely, the N-gram Query Match (NQEM) score, is a weighted combination of the components - percent matching, arrangement and the quality - described above. Given a query context, Q we express the NQEM score of an entity e as $\mathcal{L}(Q, e)$ which is 0 when the entity e is not lexically matched to the query context Q.

2.3 The QFM model

With the member engagement based similarity $\mathcal{B}(Q, e)$ and NQEM based lexical similarity scores $\mathcal{L}(Q, e)$ between the query context Q and the entity e defined as in the previous sections, we proceed to building the QFM model as follows:

Step 1: Entity Level Relevance: In our simplest model, we apply a blending function ζ that combines the two similarity measures to produce a relevance score:

$$r_{Qe} = \zeta(\mathcal{B}(Q, e), \mathcal{L}(Q, e))$$

between the entity e and a context Q. The blending function ζ is in general nonlinear. For applications where ground truth is unavailable, we have found that a bivariate quadratic function with phenomenological weights performs well for a large majority of query contexts in our model evaluation task (see Section 3). The final score r_{Qe} can be loosely interpreted as the lexical similarity adjusted by the strength of the member engagement signal, such that an entity with a low lexical score can still have a high relevance score, if the the former is supported by strong member engagement and vice versa. We always normalize r_{Qe} so that it sums up to 1.0 across all matched entities.

Step 2: Confidence Score: For a broad query (e.g. **`horror`**) the relevance score r_{Qe} is typically spread over a large set of entities, whereas for a more specific query (e.g. **`squid ga`**) there is usually only a handful of entities with high relevance followed by a long tail of low relevance entities. To systematically discard low relevance entities, we convert the normalized relevance score into a confidence score c_{Qe}, defined as the cumulative sum of the normalized

[5]Indicates whether a matched query token completely/partially matches entity's token
[6]Contains a match used in the index to be completely replaceable with the original indexed term.
[7]Mapping of the characters from their original diacritical form to some universally agreed form (for example, the way most people would type a character if they had a limited keyboard choice). 'Á' => 'a'
[8]Modifying the word by removing or converting the suffixes (most of the time) to some normalized form. There are many forms of stemming, most common for English is plural to singular (movies=>movie).
[9]For example, converting from one writing system to another for Japanese Kanji to Katakana or Hiragana

relevance score over all entities with scores lower than that of the entity e. This way, when ordered by c_{Qe}, the most relevant entity always has a $c_{Qe} = 1.0$ and the score falls slowly for a broad query, and sharply for a narrow query. We then set a threshold $c_{Qe} > \alpha$ to discard the long tail associations (we set $\alpha = 0.98$ in practice).

Step 3: Facet Score: In the final step, using the fact that each entity e can belong to only one of the facets, we simply aggregate the entity level relevance score r_{Qe} within each facet to obtain the facet level score for a query context Q and facet f:

$$S_{Qf} = \sum_{e} r_{Qe} \delta_{ef}$$

where the Kronecker delta δ_{ef} is 1 when e belongs to facet f and zero otherwise. With this final step, we can not only quantify how likely a query is to belong to one of the six facets, but also identify the most relevant entities within each facet.

3 EVALUATION

To evaluate the performance of QFM, we compared its video facet detection ability against a corpus of 44000 queries annotated manually by subject matter experts (SME) in over 30 languages. First, we established that QFM has the desirable property of correctly predicting the SME-annotated facets with high accuracy for queries with high predicted facet score, S_{Qf}, and the accuracy degrades monotonically as S_{Qf} decreases (Fig. 4, left panel). For $S_{Qf} \geq 0.9$, QFM has an accuracy 95%. Next, we ran a comparison of the facet detection accuracy from a production service, called the "Query Processing Service" (QPS), against a new system that combines QPS with QFM. QPS provides Named Entity Recognition and parsing of queries based on a dictionary (gazetteer) approach and a shallow parser [3] which ranks candidates for each position in the query based on the query context. The output annotations of the service are used in handling the search, navigation and transport (play, pause, etc…) queries. QPS handles various language morphological variations, and some syntax variations (like word order in Japanese). However, without the benefit of the behavioral signal, QPS errs on the side of mostly recognizing fully matching entities, to maintain high precision. With the introduction of the QFM in QPS, the facet recognition rates went up significantly across several languages, as depicted by the relative accuracy lift in Fig. 4, right panel. As an illustration of how QFM helps, consider the English language query "what you"; the QFM enhanced QPS system correctly mapped it to the video facet entity *I Know What You Did Last Summer*, whereas QPS alone would have failed to make that connection. QFM fills in the gaps for QPS by helping recognize facets for partial entity strings by leveraging the behavioral signal.

4 POTENTIAL APPLICATIONS AND FUTURE WORK

The QFM model forms a basis for understanding of user intent in Search. Here we highlight some of its main potential applications.

Contextual Ranking of Search Results: Knowing how a query context Q maps to the facets scores S_{Qf} can be used as an additional contextual signal in a Search ranking model. This can help the model learn when the query is heavily relevant to a certain facet, and thereby help the model prioritize entities in that facet to be ranked higher. In many applications, simple heuristics, such as

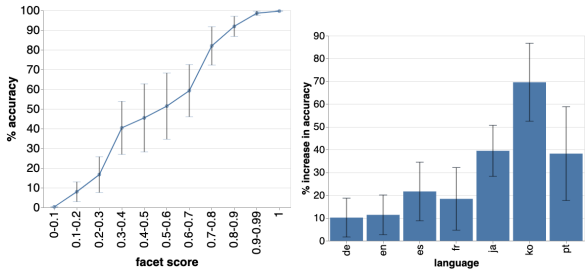

Figure 4: *Left*: Video facet detection accuracy of QFM as a function of facet score in SME annotated evaluation data. *Right*: Improvement in accuracy over the current production system by additionally leveraging QFM.

ranking the entities by the combined score: $r_{Qe} \times S_{Qf}$ in descending order can be very effective. In addition to searching for a specific title, members also explore our catalog through other facets such as talent (e.g., actors), and collections (e.g., genres or languages). By leveraging QFM, we can pivot the content into relevant groups to encourage further exploration and discovery along those facets.

Better Search Result organization: QFM allows the search results to be meaningfully structured by explicitly organizing results to better reflect user intents [1, 5], specifically when a query can have multiple intents. E.g., the intent behind the query **`irish`** could be to fetch the `IC-Video` *The Irishman*, but it could also be to explore the genre *Irish Movies*. In such cases, QFM can be used to identify the major facets, and guide the organization of the results into cohesive groups, where each group can have a descriptive label, thus reducing the cognitive load of parsing the results.

User Trust: When a member is searching for an unavailable video, the QFM model can be used to detect that intent even before the query is complete (e.g., **`game of t`** for *Game of Thrones*). We could message the member that the title is currently not available, thereby building trust, as well as meaningfully pivoting them to an exploration of similar titles that are available.

Outlier Detection: Since QFM is based on lexical and behavioral signals, we have been able to effectively use it in the detection and mitigation of outlier entities in Search results originating from human and algorithmic biases in our main relevance ranking model.

Future work: *ML based blending function*: While we have currently adopted a phenomenological model for the blending function ζ - one main avenue of future investigations is to leverage ML based techniques to learn this function in a (semi-)supervised setting. We are also pursuing how the QFM model could be incorporated into the main relevance ranking model within the context of a multitask learning setup. *Cold Starting new classes of entities*: As the classes of entities that Netflix Search indexes increase in scope, the QFM model needs to be able to cold start new facets. One idea could be to use a flat prior on the behavioral signal and let the model learn over time. We plan to discuss various methods for cold starting new facets, as well as unseen queries, as a part of future work.

ACKNOWLEDGMENTS

We are thankful to Jon Sanders, Yves Raimond, Sudarshan Lamkhede, Christoph Kofler, and Michael Galassi for their helpful feedback.

REFERENCES

[1] Hao Chen and Susan Dumais. 2000. Bringing Order to the Web: Automatically Categorizing Search Results. In *CHI 2000* (chi 2000 ed.). https://www.microsoft.com/en-us/research/publication/bringing-order-to-the-web-automatically-categorizing-search-results/

[2] Christine Hosey, Lara Vujović, Brian St. Thomas, Jean Garcia-Gathright, and Jennifer Thom. 2019. Just give me what I want: How people use and evaluate music search. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[3] Jurafsky and Manning. 2008. *Speech and Language Processing*. Prentice Hall.

[4] Sudarshan Lamkhede and Sudeep Das. 2019. Challenges in Search on Streaming Services: Netflix Case Study. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 1371–1374. https://doi.org/10.1145/3331184.3331440

[5] Sudarshan Dnyaneshwar Lamkhede and Christoph Kofler. 2021. *Recommendations and Results Organization in Netflix Search*. Association for Computing Machinery, New York, NY, USA, 577–579. https://doi.org/10.1145/3460231.3474602

[6] C.D. Manning, P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press. https://books.google.com/books?id=GNvtngEACAAJ

[7] Ivan Provalov. 2016. Global Languages Support at Netflix. https://medium.com/netflix-techblog/global-languages-support-at-netflix-testing-search-queries-ede40f7d93d3