Entity-Centric Query Refinement

David Wadden[1*] DWADDEN@CS.WASHINGTON.EDU
Nikita Gupta[2] GNIKITA@GOOGLE.COM
Kenton Lee[2] KENTONL@GOOGLE.COM
Kristina Toutanova[2] KRISTOUT@GOOGLE.COM
[1]*Paul G. Allen School of Computer Science & Engineering, University of Washington*
[2]*Google Inc.*

Abstract

We introduce the task of entity-centric query refinement. Given an input query whose answer is a (potentially large) collection of entities, the task output is a small set of query *refinements* meant to assist the user in efficient domain exploration and entity discovery. We propose a method to create a training dataset for this task. For a given input query, we use an existing knowledge base taxonomy as a source of candidate query refinements, and choose a final set of refinements from among these candidates using a search procedure designed to partition the set of entities answering the input query. We demonstrate that our approach identifies refinement sets which human annotators judge to be interesting, comprehensive, and non-redundant. In addition, we find that a text generation model trained on our newly-constructed dataset is able to offer refinements for novel queries not covered by an existing taxonomy. Our code and data are available at `https://github.com/google-research/language/tree/master/language/qresp`.

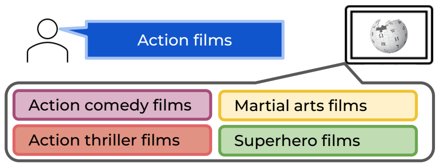
1. Introduction

During interactive search, the system user may issue queries that are under-specified, ambiguous, or open-ended. For instance, a user interested in finding a new movie might search for "Action films", or a computer vision researcher interested in learning more about NLP might search for "Pretrained NLP models". These forms of interaction are examples of *exploratory search* [Marchionini, 2006, White and Roth, 2009].

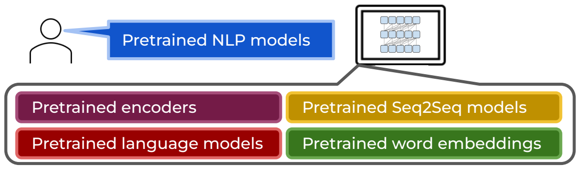
We focus specifically on queries whose answer is a list of entities, known as *list-intent* queries. For example, "Rush Hour" is one of the thousands of answers to the query "Action films". List-intent queries are common, comprising 10% of all web searches [Chakrabarti et al., 2020]. However, simply displaying the answer to such a query (e.g. a list of all action films) is more likely to cause confusion than to satisfy the user's information needs. Instead, systems for *query refinement* – also known as *query recommendation* or *query suggestion* [Sordoni et al., 2015, Baeza-Yates et al., 2004] – can assist users by offering a set of followup queries that clarify and focus the user's search, progressively drilling down on the topics and entities of most interest (Fig. 1).

With this motivation in mind, we propose the task of *entity-centric query refinement*. The task input is a list-intent query. The task output is a collection of k *query refinements*, referred to as a *refinement set*. The refinement set should provide a reasonably comprehensive overview of the set of entities answering the input query. For instance, Fig. 1b shows an example of $k = 4$ refinements that could familiarize the system user with some common types of pretrained NLP models, and point the user in interesting new search directions.

[*]. Work primarily completed while interning at Google.



(a) A query paired with a refinement set consisting of $k = 4$ query subtypes available in an existing knowledge base.


Pretrained NLP models
Pretrained encoders
Pretrained Seq2Seq models
Pretrained language models
Pretrained word embeddings

(b) Refinements for a query unlikely to be covered by existing taxonomies, generated by a text-to-text model.

Figure 1: Examples of the entity-centric query refinement task. We propose a method to select high-quality refinement sets for queries covered by an existing taxonomy (Fig. 1a), and use these refinement sets to train a model which can generate refinements for queries unlikely to be covered by any taxonomy (Fig. 1b).

To our knowledge, no datasets are available which provide instances of list-intent queries paired with refinement sets satisfying our task goals. Therefore, we propose a method to create query / refinement set pairs which can be used to train a model for this task. We leverage the YAGO3 [Mahdisoltani et al., 2015] knowledge base, using YAGO entity types as training queries. YAGO types are based on the Wikipedia category system, which provides a rich, crowdsourced taxonomy of real-world entity types. Given a YAGO type, we consider all subtypes in the taxonomy as potential refinements (Fig. 1a shows an example). We propose a method **Q**uery **R**efinement via **E**ntity **S**pace **P**artitioning (`QRESP`), which selects as refinements the k subtypes which provide the most comprehensive and non-redundant summary of the entities answering the input query. In head-to-head comparisons, we find that human annotators prefer refinement sets chosen using our proposed method over refinement sets consisting of k randomly-chosen subtypes of the input query.

We use the resulting dataset to train a T5 model [Raffel et al., 2020] capable of generating a refinement set for any input query. Since no evaluation dataset is available, we perform comparisons on both in-domain queries (held-out categories from the YAGO taxonomy) as well as out-of domain queries selected from Natural Questions [Kwiatkowski et al., 2019] and the TREC 2009 Million Query Track [Carterette et al., 2009]. We find that the outputs of a model trained on `QRESP` refinement sets are preferred over the outputs of a model trained on random query subtypes, suggesting that the properties captured by `QRESP` are generalizable to new queries. However, we also find that our models sometimes predict off-topic or irrelevant refinements, particularly on queries from Natural Questions and TREC. This points toward the need for future work to improve the reliability of refinement systems under domain shift, and to develop automated metrics of refinement quality which can be used to speed up the model development process.

In summary, our contributions are threefold: (1) We introduce the task of entity-centric query refinement, and outline key desiderata and evaluation criteria for this task. (2) We propose `QRESP`, which optimizes an entity-centric cost function to select refinement sets for queries covered by an existing knowledge base. The resulting refinement sets can be used both for entity exploration within the knowledge base, and as instances to train a refinement set generation model. (3) We develop a baseline model trained on instances selected by `QRESP` to generate refinement sets for queries unseen during training, and identify important areas for future work on this task based on analysis of our system outputs.

2. Task definition

2.1 Entity-centric query refinement

We aim to generate refinements which facilitate exploration and discovery for list-intent queries, helping the user drill down on entities of interest. Formally, the task input is a query q whose answer is a list of entities. Equivalently, the query q specifies an entity type. We refer to the list of entities answering q as the *answers* to q, or $\mathcal{A}(q)$. The task output is a collection of k *refinements*[1] $\mathcal{R}(q) = \{q'_1, \dots, q'_k\}$. We refer to $\mathcal{R}(q)$ as a *refinement set*, or "RS". Each refinement q'_i should itself be a list-intent query. In addition, each answer to q'_i should be among the answers to q: $\mathcal{A}(q'_i) \subseteq \mathcal{A}(q)$. In other words, each q'_i should specify a subtype of q. For instance, every movie that is an answer to the refinement "martial arts films" is also an answer to the input query "action films".

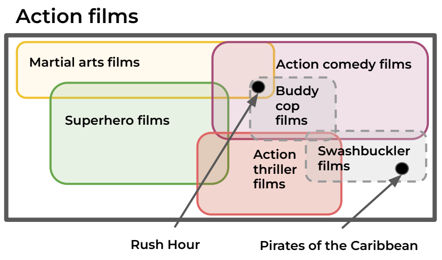


Figure 2: An entity-space view of a refinement set for the query "Action films". Rectangles indicate refinements, and black circles indicate entities. The four rectangles with solid borders correspond to the refinements in Fig. 1. The rectangles with dashed borders show two subgenres that were not included in the refinement set, since they are redundant / do not cover many films. The four selected refinements approximately partition the answer space.

2.2 Desiderata for refinement sets

The query refinement task is inherently open-ended, and there may be many reasonable RSs (refinement sets) for any given query. Inspired by prior work on faceted search interfaces for the Wikipedia category taxonomy [Li et al., 2010], we conceptualize refinement generation from the standpoint of *entity discovery*: given an input query q, can we design a refinement system such that any entity $e^* \in \mathcal{A}(q)$ is discoverable after a few rounds of system interaction? From this standpoint, the best-possible refinement set would partition the entities $\mathcal{A}(q)$ into k disjoint, equally-sized subsets, such that each answer $e_j \in \mathcal{A}(q)$ is an answer to exactly one refinement $q'_i \in \mathcal{R}(q)$. This would ensure that any entity in $\mathcal{A}(q)$ is discoverable after at most $\log_k(|\mathcal{A}(q)|)$ refinements, since the refinements would induce a k-ary search tree over the entities answering q (see Appendix A.1). We refer to such a refinement set as *ideal*.

In practice, it will almost never be possible to generate an ideal RS, since each refinement must specify a semantic category expressed in natural language. Fig. 2 provides an example showing how the refinements for the query "action films" from Fig. 1 provide a good approximation to an ideal RS. We formalize this notion in §3.

Query: Action films

Refinement	Fluent	Relevant
Martial arts films	✓	✓
Romance films	✓	✗
Martially flims arts of	✗	✗

Table 1: **Stage 1 evaluation** screens out individual refinements which are not fluent and relevant.

2.3 Evaluation criteria

Efficient entity discovery provides motivation for our task formulation, but does not admit a simple evaluation strategy. Therefore, to assess the usefulness of proposed refinement sets,

1. In this work, we provide k as a model input; dynamically choosing k based on the query represents an important future research direction.

Query: Action films				
	Refinement set	Compre-hensive	Inter-esting	Non-redundant
1	Action comedies, Action thrillers, Martial arts films, Spy films	✓	✓	✓
2	Action films set in {Asia, North America, Africa, Europe}	✓	✗	✓
3	Karate films, Kung Fu films, Boxing films, Films with boxing	✗	✓	✗

Table 2: **Stage 2 evaluation** assesses the overall quality of refinement sets. The notation "{Asia, North America, . . . }" means "Action films set in Asia, Action films set in North America, . . . ". Row (2) is comprehensive since many action films take place on one of the listed continents, but is not interesting since many different kinds of queries can be categorized by continent. Row (3) is redundant and not comprehensive since it only covers martial arts movies, and repeats "boxing films". Human evaluation makes comparisons between two refinement sets, rather than binary ✓/ ✗ decisions for a single set; we show binary decisions for illustration.

we conduct A / B tests where annotators compare two competing refinement sets $\mathcal{R}_{\mathrm{A}}(q)$ and $\mathcal{R}_{\mathrm{B}}(q)$ on a number of attributes. The evaluation includes two stages.

Stage 1: Validity of individual refinements First, the annotator confirms that the individual refinements making up $\mathcal{R}_{\mathrm{A}}(q)$ and $\mathcal{R}_{\mathrm{B}}(q)$ conform to the task definition, requiring:

1. **Fluency**: Each refinement must be fluent and grammatical.
2. **Relevance**: Each refinement must specify a subtype of q. Non-fluent refinements are automatically judged as not relevant.

Table 1 provides some examples of queries that pass and fail these requirements. If fewer than half of the refinements from either RS satisfy the criteria, annotation stops here. In our experiments (§5), we find that virtually all refinements are fluent, and the Stage 1 screen serves in practice to filter out irrelevant refinements.

Stage 2: Overall refinement set quality If the majority of refinements in both $\mathcal{R}_{\mathrm{A}}(q)$ and $\mathcal{R}_{\mathrm{B}}(q)$ are judged valid, the annotator compares the two RSs, based on four attributes:

1. **Comprehensiveness**: Does $\mathcal{R}(q)$ provide a good overview of the entities answering the query q?
2. **Interestingness**: Do the refinements in $\mathcal{R}(q)$ provide new information about the different kinds of entities answering q, or are they generic and uninteresting?
3. **Non-redundancy**: Does each refinement in $\mathcal{R}(q)$ specify a unique entity type, or are some of them redundant?
4. **Overall usefulness**: Overall, how useful are the refinements $\mathcal{R}(q)$ for learning more about the entities answering q?

Table 2 provides examples. For each attribute, the annotator selects whether $\mathcal{R}_{\mathrm{A}}$ is better, $\mathcal{R}_{\mathrm{B}}$ is better, or whether the two RSs are equally good. Details of the annotation process are provided in §4.2. The full annotation guide is included in Appendix G.

3. Refinement set selection

To build a baseline system for RS generation, we proceed as follows: (1) Leverage an existing knowledge base to create a collection of query / RS pairs that are close to ideal, and (2) Train a text-to-text model on this collection, with the goal of generalizing to queries not covered by a taxonomy. We describe (1) in this section and §4, and describe (2) in §5.

Source taxonomy We use the YAGO3 taxonomy [Suchanek et al., 2007, Mahdisoltani et al., 2015] (referred to simply as YAGO) as our source to construct a training dataset. The YAGO entity type system is adopted from the Wikipedia category system, a crowdsourced polyhierarchy used to organize Wikipedia pages. We use types in the YAGO taxonomy as input queries q. Given a type q, we consider all sub-types of q in the taxonomy as refinement *candidates*, denoted $\mathcal{C}(q)$. From this collection of K candidates, we aim to select k sub-types to form our refinement set $\mathcal{R}(q)$. We use the entities in the YAGO knowledge base that are instances of type q as the answers to q, or $\mathcal{A}(q)$. As a concrete example, in Fig. 2, "Action films" is the query q, the shaded rectangles correspond to members of the candidate set $\mathcal{C}(q)$, and the black points are two answers in $\mathcal{A}(q)$.

Even when a list of K candidate subtypes is available, selecting the best k refinements is challenging because (1) K may be large; for example, there are 68 subtypes of the type "Action films", (2) Many of the candidate subtypes may be too specific to be part of a useful summary (e.g. "Tomb Raider films" is a subtype of "Action films"; Appendix B.2 provides additional examples), and (3) subtypes may be redundant, overlapping, or generic.

`QRESP` for refinement selection To select an RS of k refinements from among the K candidate subtypes associated with a given input query, we propose `QRESP`: **Q**uery **R**efinement via **E**ntity **S**pace **P**artitioning. We define a cost function which selects RSs according to the desiderata described in §2.2, rewarding RSs whose answers approximately partition the answers to the original query. Then, we minimize this cost function over all refinement sets.

Let e_j denote a particular entity in $\mathcal{A}(q)$, and let $n = |\mathcal{A}(q)|$. Given a pool of candidate refinements $\mathcal{C}(q)$, let $\mathfrak{R}(q)$ indicate the collection of all size-k subsets of $\mathcal{C}(q)$. Given a possible refinement set $\mathcal{R}(q) \in \mathfrak{R}(q)$, and a refinement $q'_i \in \mathcal{R}(q)$, let $a_{ij} = \mathbb{1}\left[e_j \in \mathcal{A}(q'_i)\right]$. Define $c_j = \sum_{i=1}^{k}(a_{ij})$, the number of refinements in $\mathcal{R}(q)$ for which entity j is an answer. Let $n_i = \sum_{j=1}^{n}(a_{ij}) = |\mathcal{A}(q'_i)|$, the number of entities that answer refinement i. Then we define a cost function $\mathcal{S}$ measuring the quality of a given $\mathcal{R}(q)$, and minimize over $\mathfrak{R}(q)$:

$$\mathcal{S}(\mathcal{R}(q)) = \left[\sum_{j=1}^{n} |c_j - 1|\right] - \min_{i\in\{1,\ldots,k\}} n_i \quad (1) \qquad\qquad \mathcal{R}^*(q) = \operatorname*{arg\,min}_{\mathfrak{R}(q)} \mathcal{S}(\mathcal{R}(q)). \quad (2)$$

The first term in Eq. 1 is minimized when each answer to q is an answer to exactly one of the q'_i, encouraging comprehensiveness and non-redundancy. The second term encourages the smallest refinement to have as many answers as possible. Combined with the first term, this rewards the selection of refinements which all have a similar number of answers.

Eq. 2 selects the refinement set $R^*(q)$ minimizing Eq. 1. In Appendix A.2, we provide a proof that the RS selected by `QRESP` is the *best achievable* in the following sense: the scoring function $\mathcal{S}(\mathcal{R}(q))$ achieves its global minimum if and only if $\mathcal{R}(q)$ is *ideal* as defined in §2.2. Thus, selecting refinements according to `QRESP` yields the RS that is closest to ideal, given the subcategories $\mathcal{C}(q)$ available in YAGO. We use integer linear programming to perform the combinatorial optimization over $\mathfrak{R}(q)$ in Eq. 2; see Appendix A.3 for details.

Query: Action films	
$\mathcal{D}_{\texttt{QRESP}}$	{Action comedy, Action thriller, Martial arts, Science fiction action, Spy} films
$\mathcal{D}_{\text{Random}}$	{1910s, 1920s, Australian, Brazilian, Nigerian} action films
$\mathcal{D}_{\text{Random-F}}$	{Action comedy, Action thriller, Animated action, The purge} films, Action films based on actual events.

Table 3: Refinement sets from $\mathcal{D}_{\texttt{QRESP}}$, $\mathcal{D}_{\text{Random}}$, and $\mathcal{D}_{\text{Random-F}}$. $\mathcal{D}_{\text{Random}}$ includes many refinements based on a time period or a country of origin, which does not provide interesting new information about the topic. $\mathcal{D}_{\text{Random-F}}$ is overly specific (e.g. "The Purge films"), and thus does not provide as good an overview as $\mathcal{D}_{\texttt{QRESP}}$.

4. A dataset for entity-centric query refinement

We apply `QRESP` to YAGO to create a dataset for entity-centric query refinement, and conduct A / B tests to assess whether our approach aligns with human judgments. We set the number of refinements to $k = 5$ for all experiments; this is large enough to enable diverse refinement sets, but not so large as to be overwhelming for users.

4.1 Dataset creation

We apply `QRESP` to select RSs from the YAGO taxonomy. We use YAGO types with at least 50 answer entities as our training queries, as it may be difficult to select non-trivial refinements for types with only a few answers. Given a type q with candidate subtypes $\mathcal{C}(q)$, we use rule-based filters to remove candidates that differ from q only by the addition of a date, location, or a gender (e.g. "singers" $\rightarrow$ "female singers"), as these tend to lead to generic refinements (see Appendix B.5 for the full list of rules). We refer to the filtered collection as $\mathcal{C}_{\text{Filter}}(q)$. Queries with fewer than k subtypes post-filtering are removed. For the remaining queries, we apply `QRESP` to the candidates $\mathcal{C}_{\text{Filter}}(q)$ to select refinements $\mathcal{R}_{\texttt{QRESP}}(q)$. We call the resulting dataset $\mathcal{D}_{\texttt{QRESP}} = (q_i, \mathcal{R}_{\texttt{QRESP}}(q_i))_{i=1}^{N}$. We also construct two ablation datasets for comparison. For $\mathcal{D}_{\text{Random}}$, we select refinements by randomly choosing k subtypes from $\mathcal{C}(q)$, without filtering. For $\mathcal{D}_{\text{Random-F}}$, we choose k random subtypes from $\mathcal{C}_{\text{Filter}}(q)$. Table 3 and Appendix B.2 provide examples. In Appendix B.6, we present results confirming that RSs selected using `QRESP` consistently achieve lower costs as measured by Eq. 1, compared to randomly-chosen RSs. Our training dataset consists of $8,958$ query / RS instances for $\mathcal{D}_{\texttt{QRESP}}$ and $\mathcal{D}_{\text{Random-F}}$, and $17,598$ instances for $\mathcal{D}_{\text{Random}}$.

We hold out 282 query / RS instances to be used for model development in §5. Development queries and their subtypes are excluded from the train set. We select dev set queries by randomly sampling YAGO types with at least 15 subtypes, each of which must have at least 200 answer entities. We use categories with many subtypes and answers because these are the categories for which query refinement has the most potential to facilitate user exploration and discovery. From our development set, we manually select a collection of 105 examples to be used for human evaluation in A / B tests. We select a diverse set of interesting, open-ended queries across a variety of domains, intended to mirror the types of queries likely to be seen in real-world use. Appendix B includes a discussion of all data selection and preprocessing choices, as well as additional statistics on YAGO. Appendix F includes the full list of queries used for human evaluation.

		A = `QRESP` vs. **B** = Random			**A** = `QRESP` vs. **B** = Random-F		
	$N = 105$	**A**	Neutral	**B**	**A**	Neutral	**B**
Stage 1	Fluent + Relevant	**98%**	-	97%	100%	-	100%
Stage 2	Comprehensive	**88%****	10%	2%	**75%****	15%	10%
	Interesting	**71%****	26%	3%	**64%****	24%	12%
	Non-redundant	-	-	-	-	-	-
	Overall	**86%****	10%	4%	**73%****	17%	10%

Table 4: A / B tests comparing refinement sets from $\mathcal{D}_{\texttt{QRESP}}$ against $\mathcal{D}_{\text{Random}}$ (left) and $\mathcal{D}_{\text{Random-F}}$ (right). N indicates the number of annotated instances. For Stage 1 evaluation, we report the percentage of refinement sets from each system that passed the Stage 1 screen. For Stage 2 evaluation, we report the percentage of queries for which annotators preferred refinements from System A vs. System B. The "non-redundant" evaluation was added after these tests were performed, and is left blank. ** indicates significance at $p < 0.001$.

4.2 Human evaluation

We compare refinements from $\mathcal{R}_{\texttt{QRESP}}$ with refinements chosen randomly, $\mathcal{R}_{\text{Random}}$. In addition, we compare $\mathcal{R}_{\texttt{QRESP}}$ to $\mathcal{R}_{\text{Random-F}}$, to confirm that the improved ratings are due to selection using `QRESP`, and not simply the filtering out of uninteresting candidates. Details on the annotation process are included in Appendix C.

A / B test results are shown in Table 4. We perform statistical significance tests for all human comparisons; see Appendix D for details on the statistical procedures used. More than 97% of RSs from all approaches pass the Stage 1 screen for relevance and fluency. This is expected, since the refinements for training queries come from the YAGO taxonomy. In Stage 2, RSs selected by `QRESP` are preferred by annotators 86% of the time over random RSs, and 73% of the time over random RSs post-filtering. $\mathcal{R}_{\texttt{QRESP}}$ is preferred more frequently for comprehensiveness than for interestingness, likely because Eq. 1 explicitly rewards comprehensiveness. Overall, the results indicate that `QRESP` captures human intuitions about refinement quality.

5. Refinement generation

Having established that `QRESP` selects high-quality refinement sets for queries covered by a knowledge base, we finetune a pretrained language model to generate refinements for queries not found in the KB. We experiment with two evaluation datasets: held-out categories from the YAGO taxonomy, and a collection of list-intent queries selected from Natural Questions [Kwiatkowski et al., 2019] and the TREC 2009 Million Query Track [Carterette et al., 2009].

5.1 Model

We use T5-3B [Raffel et al., 2020] as our base model. Given a dataset $\mathcal{D}$ consisting of training pairs $(q_i, \mathcal{R}(q_i))_{i=1}^{N}$, we provide q as the input for T5 and train it to generate $\mathcal{R}(q)$. We format $\mathcal{R}(q)$ by concatenating the individual refinements in alphabetical order, separated by a sentinel token. At prediction time, we provide q as input and greedily decode (greedy outperformed sampling and beam search on automated metrics). We train models on $\mathcal{D}_{\texttt{QRESP}}$, $\mathcal{D}_{\text{Random-F}}$, and $\mathcal{D}_{\text{Random}}$; we call these $\mathcal{M}_{\texttt{QRESP}}$, $\mathcal{M}_{\text{Random-F}}$, and $\mathcal{M}_{\text{Random}}$, respectively. We train with a batch size of 32 for 8000 steps, using the default T5 optimizer.

Model	Sequence		Set			Perplexity
	BLEU	ROUGE-L	P	R	F1	
$\mathcal{M}_{\texttt{QRESP}}$	**67.1**	**69.4**	**24.8**	**24.3**	**24.6**	**2.01**
$\mathcal{M}_{\text{Separate}}$	66.9	68.2	19.6	19.2	19.4	2.35
$\mathcal{M}_{\text{Random-F}}$	61.5	66.0	15.1	14.3	14.7	2.11
$\mathcal{M}_{\text{Random}}$	57.6	63.8	6.8	6.7	6.8	2.19

Table 5: Automated evaluation of generation models, using $\mathcal{D}_{\texttt{QRESP}}$ as "silver" evaluation targets. Evaluations are categorized into Sequence-based, Set-based, and Perplexity-based. $\mathcal{M}_{\texttt{QRESP}}$ outperforms models trained on randomly-chosen refinements, or trained to generate refinements separately rather than as a single sequence.

As an additional ablation, we train a model $\mathcal{M}_{\text{Separate}}$ on $\mathcal{D}_{\texttt{QRESP}}$, which predicts a single refinement q'_i at a time, rather than predicting a full refinement set $\mathcal{R}(q)$. At prediction time, given an input q, we sample 5 separate predictions from $\mathcal{M}_{\text{Separate}}$ and concatenate to form $\mathcal{R}(q)$; a similar approach is used by MacAvaney et al. [2021] to generate possible query intents for search result diversification.

5.2 Automated evaluations

The results of §4 indicate that human annotators prefer refinements from $\mathcal{D}_{\texttt{QRESP}}$ over those from $\mathcal{D}_{\text{Random}}$ and $\mathcal{D}_{\text{Random-F}}$. Therefore, for our automated evaluations, we treat RSs from the $\mathcal{D}_{\texttt{QRESP}}$ dev set as "silver" data against which to evaluate the predictions of our trained models. We evaluate using the following metrics:

1. **Sequence-based metrics**: We treat a generated RS as a text sequence with no additional structure, and compare it to the corresponding silver RS using BLEU and ROUGE-L (ROUGE-1 and -2 showed the same trend).
2. **Set-based metrics**: We treat the generated RS as a set of k refinements, and evaluate the precision, recall, and F1 relative to the set of silver refinements.
3. **Perplexity**: We evaluate the perplexity of the silver RSs, formatted as a sequence, under each generation model.

The results are shown in Table 5. $\mathcal{M}_{\texttt{QRESP}}$ performs best on all metrics. Training on randomly-chosen candidate subtypes ($\mathcal{M}_{\text{Random-F}}$ and $\mathcal{M}_{\text{Random}}$) decreases performance, particularly as measured by F1. We also observe that sequential refinement set generation by $\mathcal{M}_{\texttt{QRESP}}$ outperforms separate prediction of individual refinements by $\mathcal{M}_{\text{Separate}}$. We examine the reasons for this improvement in §5.3.

5.3 Human evaluation

We conduct A / B tests on two datasets. First, we evaluate on the (in-domain) human evaluation set of YAGO types described in §4.1. Second, to measure the ability to generalize to *out-of-domain* queries, we evaluate on 93 real-world list-intent queries selected by one of the paper authors from the Natural Questions dataset and the TREC 2009 Million Query Track (referred to as NQ+TREC). As with YAGO, we aimed to select exploratory, list-intent queries on a variety of topics. Appendix F includes the full list of NQ+TREC queries, and Appendix E.1 includes a discussion of our query selection criteria.

Query: Physicians	
$\mathcal{M}_{\texttt{QRESP}}$	Alternative medicine physicians, cardiologists, dermatologists, ophthalmologists, psychiatrists
$\mathcal{M}_{\text{Random}}$	{Canadian, German, Norwegian} dermatologists, Pediatricians, Women physicians
$\mathcal{M}_{\text{Random-F}}$	Fictional physicians, Oncologists, Psychiatric physicians, Radiologists, surgeons
$\mathcal{M}_{\text{Separate}}$	{Atheist, Baritone, Fictional, Military} physicians, Neurologists

Table 6: Refinements of $\mathcal{M}_{\texttt{QRESP}}$ and three ablations on a YAGO evaluation query. The $\mathcal{M}_{\texttt{QRESP}}$ suggestions cover 5 common types of physicians. In contrast, some ablation refinements are generic (Canadian dermatologists) or idiosyncratic (Fictional physicians).

		A = QRESP vs. **B** = Random			**A** = QRESP vs. **B** = Separate		
	$N = 105$	**A**	Neutral	**B**	**A**	Neutral	**B**
Stage 1	Fluent + Relevant	93%	-	**100%***	93%	-	**95%**
Stage 2	Comprehensive	**77%****	12%	11%	**47%**	20%	33%
	Interesting	**70%****	22%	8%	**28%**	45%	27%
	Non-redundant	**22%**	66%	12%	**42%****	45%	14%
	Overall	**76%****	17%	7%	**46%***	26%	28%

Table 7: Results of A / B tests on the YAGO human evaluation set. $\mathcal{M}_{\texttt{QRESP}}$ is preferred over both ablations. * and ** indicate significance at $p < 0.05$ and $p < 0.001$, respectively.

Results on YAGO We compare refinements from $\mathcal{M}_{\texttt{QRESP}}$ against $\mathcal{M}_{\text{Random}}$, $\mathcal{M}_{\text{Random-F}}$ and $\mathcal{M}_{\text{Separate}}$. Table 6 shows the predictions of each system on a single query. Table 7 shows the results of A / B tests comparing $\mathcal{M}_{\texttt{QRESP}}$ against $\mathcal{M}_{\text{Random}}$ and $\mathcal{M}_{\text{Separate}}$; results for $\mathcal{M}_{\text{Random-F}}$ are similar to $\mathcal{M}_{\text{Random}}$ and are included in Appendix E.2.

	A = QRESP vs. **B** = Random		
$N = 93$	**A**	Neutral	**B**
Fluent + Relevant	56%	-	**81%****
Comprehensive	**45%**	30%	26%
Interesting	**40%**	34%	26%
Non-redundant	23%	51%	**26%**
Overall	**43%**	28%	30%

Table 8: Human evaluations on NQ+TREC. $\mathcal{M}_{\texttt{QRESP}}$ refinements tend to be more interesting and comprehensive, provided that they are relevant. ** indicates $p < 0.001$.

Compared to $\mathcal{M}_{\text{Random}}$, refinements from $\mathcal{M}_{\texttt{QRESP}}$ are much more likely to be comprehensive and interesting. On the other hand, the models are similar in terms of non-redundancy, since $\mathcal{M}_{\text{Random}}$ tends to offer overly-specific refinements which provide a poor summary but do not overlap. $\mathcal{M}_{\texttt{QRESP}}$ and $\mathcal{M}_{\text{Separate}}$ have nearly identical interestingness. However, $\mathcal{M}_{\texttt{QRESP}}$ enjoys a large advantage in non-redundancy, since it can condition each new refinement on earlier refinements in the RS.

Results on NQ+TREC We compare predictions from $\mathcal{M}_{\texttt{QRESP}}$ vs. $\mathcal{M}_{\text{Random}}$. Interestingly, only 56% of RSs generated by $\mathcal{M}_{\texttt{QRESP}}$ pass the Stage 1 screen, compared to 81% for $\mathcal{M}_{\text{Random}}$. However, the RSs from $\mathcal{M}_{\texttt{QRESP}}$ that pass the screen are preferred over RSs from $\mathcal{M}_{\text{Random}}$ in terms of comprehensiveness, interestingness, and overall quality (Table 8). While the differences do not reach the threshold of statistical significance, this trend

Query: Functions of government	
$\mathcal{M}_{\texttt{QRESP}}$	{Agricultural, Defense, Education, Finance, Foreign} functions of the government
$\mathcal{M}_{\text{Random}}$	Functions of the {Canadian, Yukon, Quebec, Northern Mariana Islands, United States} government

(a) $\mathcal{M}_{\texttt{QRESP}}$ offers a list of five interesting, diverse government functions. $\mathcal{M}_{\text{Random}}$ provides a generic list of five locations, including two Canadian provinces.

Query: Popular YouTube Channels	
$\mathcal{M}_{\texttt{QRESP}}$	{Celebrity YouTube, Music video, Religious YouTube, Religious television, Religious video} channels
$\mathcal{M}_{\text{Random}}$	{American, Australian, British, Japanese, Pakistani} popular YouTube channels

(b) Three $\mathcal{M}_{\texttt{QRESP}}$ refinements were judged irrelevant since they don't mention YouTube specifically. $\mathcal{M}_{\text{Random}}$ makes five generic, but relevant, refinements by once against listing five locations.

Table 9: Predictions on two queries from NQ+TREC. $\mathcal{M}_{\texttt{QRESP}}$ provides high-quality refinements for the first query, but is slightly off-topic for the second one.

suggests that the two models behave differently on out-of-domain queries. $\mathcal{M}_{\texttt{QRESP}}$ refinements can be interesting and informative (Table 9a), but sometimes go off-topic (Table 9b). $\mathcal{M}_{\text{Random}}$ tends to make "safe" refinements that are generic but relevant.

6. Related work

Query refinement *Search results clustering* [Carpineto et al., 2009] offers refinements by retrieving a collection of documents in response to an input query, clustering them, and assigning an informative name to each cluster. Names can be assigned based on word count statistics [Zamir and Etzioni, 1999, Osinski and Weiss, 2005], or generated using Seq2Seq models [Medlar et al., 2021]. *Faceted search* [Tunkelang, 2009, Hearst, 2006] also retrieves documents in response to the user's search, but organizes them according to a faceted concept hierarchy, which can be used as a source of refinements. The hierarchy can be created by the system designers [Yee et al., 2003], adapted from an existing taxonomy [Li et al., 2010, Arenas et al., 2016], or constructed automatically [Stoica et al., 2007].

Like our work, these approaches output a collection of query refinements, aided by an existing taxonomy in the case of faceted search. Unlike these approaches, we optimize an entity-centric objective function to select a set of refinements which provides a comprehensive summary of the input query, rather than organizing a collection of retrieved documents. In addition, unlike faceted search, our proposed baseline can provide refinements for an arbitrary input query which may not be covered by an existing taxonomy.

Systems have also been trained on *search query logs* to predict likely followup queries given a user search history [Sordoni et al., 2015, Dehghani et al., 2017, Boldi et al., 2008]. Our modeling goals differs from this setting in that (1) we do not assume access to query logs, which are often proprietary, and (2) we aim to output a set of refinements with a specific entity structure, rather than reproducing the search behavior of users.

Under-specified queries Researchers in NLP and IR have developed systems to resolve ambiguity [Min et al., 2020], incorporate dialogue context [Elgohary et al., 2019, Anantha et al., 2021], and ask questions [Aliannejadi et al., 2019, Sekulic et al., 2021, Zamani et al., 2020] to clarify user intent in response to ambiguous or multi-faceted input queries. As an alternative to clarifying user intent directly, *search results diversification* [Santos et al., 2015] predicts possible user intents with the aid of taxonomies [Agrawal et al., 2009], search logs [Santos et al., 2010], or recently transformer language models [MacAvaney et al., 2021], and then selects a collection of documents which comprehensively and non-redundantly addresses all predicted user intents [Carbonell and Goldstein-Stewart, 1998].

In general, these approaches were developed to distinguish between a handful of possible query intents, while our goal is to facilitate exploration and entity discovery for open-ended list-intent queries. Like search results diversification, we also aim for comprehensiveness and non-redundancy. However, we measure these quantities directly over sets of entities, rather than over collections of documents, whose similarity to each other and to the input query must be approximated using a metric like cosine distance.

7. Conclusion and future work

In this work, we proposed the task of entity-centric query refinement. We developed QRESP to select high-quality refinement sets which partition the set of entities answering the input query, and showed that our methodology has good agreement with human judgments. We then demonstrated that a text-to-text model trained on QRESP-selected data was able to generate refinement sets for queries not found in an existing knowledge base.

Our findings point toward two key open challenges for entity-centric query refinement. First, in §5.3, we found that $\mathcal{M}_{\texttt{QRESP}}$ can sometimes generate off-topic or irrelevant refinements under domain shift. This points to a need for domain adaptation techniques which do not require supervised query / refinement set pairs. A second challenge is the development of high-quality automated evaluation metrics to speed the model development process. While we experimented with a number of evaluation metrics and found that they correlated with human judgments of refinement quality (§5.2), the metrics we examined perform comparisons to a single reference refinement set. Given the open-endedness of the task, these approaches may not adequately reward refinements which a human would judge as high-quality, but which do not closely match the reference.

The QRESP framework has the potential to facilitate progress on both challenges. In this work, we used the QRESP scoring function (Eq. 1) to select refinement sets from a pool of candidates for which gold answer entities could be found in a knowledge base. In the future, this same scoring function could be used to evaluate refinement sets proposed by a generation model, using an entity-centric QA model [Ling et al., 2020, Févry et al., 2020] to predict the list of entities answering each refinement. This QRESP-QA score would serve as a flexible automated metric to compare the performance of refinement generation models.

Further, the availability of a flexible automated metric for refinement set quality could be leveraged to improve out-of-domain generalization. For instance, the QRESP-QA score could be used as a reward signal to be optimized via reinforcement learning; this approach could steer the model away from generating irrelevant refinements. Similarly, QRESP-QA could be used at inference time to re-rank a list of generated candidate refinements, filtering out irrelevant or off-topic candidates.

In summary, we believe that the entity-centric query refinement task presents a number of exciting research avenues for researchers. We hope that our dataset, modeling baselines, and analysis will motivate future work on this challenging and relevant task.

Acknowledgments

Thanks to Livio Baldini Soares for help with the RELIC code, and to Andrew McCallum, Ming-Wei Chang, Rob Logan, Sam Oates, Taylor Curtis, and the Google Research Language and Google Search teams for helpful discussions and feedback. Thanks also to Donald Metzler and John Blitzer for comments on a draft of this work.

References

Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *WSDM*, 2009.

Mohammad Aliannejadi, Hamed Zamani, Fabio A. Crestani, and W. Bruce Croft. Asking Clarifying Questions in Open-Domain Information-Seeking Conversations. In *SIGIR*, 2019.

R. Anantha, Svitlana Vakulenko, Zhucheng Tu, S. Longpre, Stephen G. Pulman, and Srinivas Chappidi. Open-Domain Question Answering Goes Conversational via Question Rewriting. In *NAACL*, 2021.

Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Sarunas Marciuska, and Dmitriy Zheleznyakov. Faceted search over RDF-based knowledge graphs. *Journal of Web Semantics*, 2016.

Ricardo Baeza-Yates, Carlos A. Hurtado, and Marcelo Mendoza. Query Recommendation Using Query Logs in Search Engines. In *EDBT Workshops*, 2004.

Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021. URL http://www.optimization-online.org/DB_HTML/2021/12/8728.html.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, A. Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *CIKM*, 2008.

Jaime G. Carbonell and Jade Goldstein-Stewart. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

Claudio Carpineto, Stanislaw Osinski, Giovanni Romano, and Dawid Weiss. A survey of Web clustering engines. *ACM Computing Surveys*, 2009.

Ben Carterette, Virgiliu Pavlu, Hui Fang, and Evangelos Kanoulas. Million Query Track 2009 Overview. In *TREC*, 2009.

Kaushik Chakrabarti, Zhimin Chen, Siamak Shakeri, Guihong Cao, and Surajit Chaudhuri. TableQnA: Answering List Intent Queries With Web Tables. *ArXiv*, abs/2001.04828, 2020.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. 2009.

Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. Learning to Attend, Copy, and Generate for Session-Based Query Suggestion. In *CIKM*, 2017.

Ahmed Elgohary, Denis Peskov, and Jordan L. Boyd-Graber. Can You Unpack That? Learning to Rewrite Questions-in-Context. In *EMNLP*, 2019.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. Entities as Experts: Sparse Memory Access with Entity Supervision. In *EMNLP*, 2020.

Marti A. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 2006.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc V. Le, and Slav Petrov. Natural Questions: A Benchmark for Question Answering Research. *TACL*, 2019.

J Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 1977.

Chengkai Li, Ning Yan, Senjuti Basu Roy, Lekhendro Lisham, and Gautam Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *WWW*, 2010.

Jeffrey Ling, Nicholas FitzGerald, Zifei Shan, Livio Baldini Soares, Thibault Févry, David Weiss, and T. Kwiatkowski. Learning Cross-Context Entity Representations from Text. *ArXiv*, abs/2001.03765, 2020.

Sean MacAvaney, Craig MacDonald, Roderick Murray-Smith, and Iadh Ounis. IntenT5: Search Result Diversification using Causal Language Models. *ArXiv*, abs/2108.04026, 2021.

Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *CIDR*, 2015.

Gary Marchionini. Exploratory Search: From Finding to Understanding. *Communications of the ACM*, 2006.

Alan Medlar, Jing Li, and Dorota Glowacka. Query Suggestions as Summarization in Exploratory Search. In *CHIIR*, 2021.

George A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 1995.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering Ambiguous Open-domain Questions. In *EMNLP*, 2020.

Stanislaw Osinski and Dawid Weiss. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 2005.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR*, 2020.

Rodrygo L. T. Santos, Craig MacDonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *WWW*, 2010.

Rodrygo L. T. Santos, Craig MacDonald, and Iadh Ounis. Search Result Diversification. *Foundations and Trends in Information Retrieval*, 2015.

Ivan Sekulic, Mohammad Aliannejadi, and Fabio A. Crestani. Towards Facet-Driven Generation of Clarifying Questions for Conversational Search. In *SIGIR*, 2021.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jianyun Nie. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *CIKM*, 2015.

Emilia Stoica, Marti A. Hearst, and Megan Richardson. Automating Creation of Hierarchical Faceted Metadata Structures. In *NAACL*, 2007.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.

Daniel Tunkelang. *Faceted Search.* 2009.

Ryen W. White and Resa A. Roth. Exploratory Search: Beyond the Query-Response Paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2009.

Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti A. Hearst. Faceted metadata for image search and browsing. In *CHI*, 2003.

Hamed Zamani, Susan T. Dumais, Nick Craswell, Paul N. Bennett, and Gord Lueck. Generating Clarifying Questions for Information Retrieval. In *WWW*, 2020.

Oren Zamir and Oren Etzioni. Grouper: A Dynamic Clustering Interface to Web Search Results. *Computer Networks*, 1999.

A. Additional details on `QRESP`

A.1 `QRESP` and entity discovery

In §2.2 we mentioned that, when interacting with a (hypothetical) system which outputs k refinements whose answers evenly partition the answers to the input query q, any entity answering q is discoverable after at most $\log_k(|\mathcal{A}(q)|)$ rounds of system interaction. To see why, suppose we aim to discover a target entity $e^* \in \mathcal{A}(q)$. When given q as input, this ideal system would output k refinements, each with $|\mathcal{A}(q)|/k$ answers, and exactly one of which has e^* as an answer. We would then input this refinement q' as our new query, and the system would output k new refinements, each with $|\mathcal{A}(q)|/k^2$ answers, exactly one of which has e^* as an answer. We would use this refinement q'' as our new input query, and repeat the process until we arrive at a refinement that includes only e^*. This process induces a $k - ary$ tree over the set of all entities answering the original query q, with depth $\log_k(|\mathcal{A}(q)|)$. In other words, any entity answering q can be discovered after $\log_k(|\mathcal{A}(q)|)$ system interactions.

This type of system interaction can be viewed as a form of Huffman coding, where each entity $e_j \in \mathcal{A}(q)$ is represented by a k-ary prefix code indicating the sequence of refinements used to discover the entity. Assuming that all entities in $\mathcal{A}(q)$ are equally likely to be the target entity e^*, choosing refinements which partition the entity space into equal-sized subsets induces an optimal prefix code [Cormen et al., 2009, Chapter 16.3]. Future work could extend this framework to model entity popularity, assigning shorter codes (i.e. shorter refinement sequences) to more frequently-searched entities.

A.2 Best achievable refinements

We provide a proof that the scoring function $\mathcal{S}(\mathcal{R}(q))$ defined in Eq. 1 achieves its global minimum if and only if $\mathcal{R}(q)$ is ideal – i.e. all refinements in $\mathcal{R}(q)$ have the same number of answers, and every entity answering q answers exactly one $q'_i \in \mathcal{R}(q)$.

For brevity, we denote an RS $\mathcal{R}(q)$ simply as $\mathcal{R}$. Assume $n \triangleq |\mathcal{A}(q)|$ is divisible by k; this eliminates some edge cases but does not change the main idea. Re-write Eq. 1 by defining $t_1 \triangleq \sum_j |c_j - 1|$ and $t_2 \triangleq \min_i n_i$, so $\mathcal{S}(\mathcal{R}) = t1 - t2$. We claim that $\mathcal{R}$ is ideal as defined in §2.2 if and only if $t_1 = 0$ *and* $t_2 = n/k$. As proof, the forward direction follows from the definition. For the reverse direction, $t_1 = 0$ means that each e_j answers exactly one q'_i. Combined with the fact that the smallest $\mathcal{A}(q'_i)$ has n/k answers, this implies that *all* q'_i have n/k answers, thus $\mathcal{R}$ is ideal. It follows that $\mathcal{S}(\mathcal{R}) = -n/k$ if $\mathcal{R}$ is ideal.

Now assume that $\mathcal{R}$ is some RS for which $\mathcal{S}(\mathcal{R}) \leq -n/k$. Since t_1 is lower-bounded by 0, it must be that $t_2 \geq n/k$; equivalently, $t_2 = n/k + \delta$ for some integer $\delta \geq 0$. This necessitates that $|\mathcal{A}(q'_i)| \geq n/k + \delta$ for all i, which implies $t_1 \geq \delta k$. Then $\mathcal{S}(\mathcal{R}) = t_1 + t_2 \geq \delta k - (n/k + \delta) = (k-1)\delta - n/k$. If $\delta > 0$, then $\mathcal{S}(\mathcal{R}) > -n/k$, contradicting our assumption. On the other hand, if $\delta = 0$, then we have $t_1 = 0$ and $t_2 = n/k$, which (from our earlier claim) is only possible if $\mathcal{R}$ is ideal. Thus, $\mathcal{S}(\mathcal{R})$ achieves its minimum if and only if $\mathcal{R}$ is ideal.

A.3 Optimization

We describe how to convert the optimization problem Eq. 2 into an integer linear program. We use the same notation as in §3, with one change: instead of using i to index refinements in $\mathcal{R}(q)$, we use it to index refinement *candidates* $q'_i \in \mathcal{C}(q)$. Let $x_i = \mathbb{1}[q'_i \in \mathcal{R}(q)]$, $a_{ij} = \mathbb{1}[e_j \in \mathcal{A}(q'_i)]$, and $c_j = \sum_i x_i a_{ij}$. Let $n_i = |A(q'_i)|$ and $n_{max} = \max_i |A(q'_i)|$. Then Eq. 2 can be re-written as:

$$\begin{aligned}
\min_{x_i, c_j, y_j, \xi} \quad & \left[\sum_{j=1}^{n} y_j\right] - \xi \\
\text{s.t.} \quad & c_j - 1 \leq y_j \text{ and } 1 - c_j \leq y_j \qquad \forall j \\
& c_j = \sum_i x_i a_{ij} \qquad \forall j \\
& k = \sum_i x_i \\
& \xi \leq (1 - x_i) n_{max} + x_i n_i \qquad \forall i
\end{aligned}$$

We use SCIP to perform the optimization [Bestuzheva et al., 2021].

B. Dataset construction

B.1 YAGO source data

The versions of the YAGO dataset used in this work can be downloaded from `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads`, under the heading `Download YAGO themes` and in the colored box labeled `TAXONOMY`. For the taxonomy (type hierarchy), we used `yagoTaxonomy`. For the list of entities that are instances of each type in the hierarchy, we used `yagoTransitiveType`. Entity type transitive closure is enforced; if e_j is an instance of q, then it is also an instance of all ancestors of q.

The full YAGO taxonomy is constructed by merging the WordNet taxonomy [Miller, 1995] with the Wikipedia category hierarchy, using heuristics to label Wikipedia categories as subclasses of WordNet types. This merging process, while generally very effective, may be noisy or inaccurate in some cases. Since the Wikipedia category system already provides a rich, crowd-sourced taxonomy of real-world entity types, we remove the WordNet types for our experiments and focus on the Wikipedia category hierarchy.

B.2 YAGO3 examples

Table 10 shows the YAGO category $q =$ "Action films", paired together with a sample of its sub-types $\mathcal{C}(q)$ and answers $\mathcal{A}(q)$. Many of the subtypes are generic or overly-specific and would not be suitable refinements. Table 3 in §4.1 showed RSs from $\mathcal{D}_{\texttt{QRESP}}$, $\mathcal{D}_{\text{Random}}$, and $\mathcal{D}_{\text{Random-F}}$ for a single query. In Table 11, we include examples for three additional queries.

B.3 YAGO preprocessing steps

As described in §4.1, we perform a number of preprocessing steps before running `QRESP` to select RSs from YAGO.

1. Use YAGO types with ≥ 50 answer entities as training queries (discarding other types): This is done because the task we propose is intended to help users in situations where

Query q	Action films
Candidates $\mathcal{C}(q)$	1900s action films, 1910s action films, . . . , Action adventure films, Action comedy films, . . . , British action films, Chinese action films, . . . , The Purge films, Tomb Raider films
Answers $\mathcal{A}(q)$	Bad Boys, John Wick, Rock Balboa, Foxy Brown, Rush Hour, Pirates of the Caribbean, Mad Max, . . .

Table 10: An example of candidate refinements (i.e. sub-types) $\mathcal{C}(q)$ and answers (i.e. instances) $\mathcal{A}(q)$ for the example query "Action films". Some sub-types are specific to the domain (e.g. "Action adventure films"), while others generically modify the query by adding a time period or country or origin.

simply reading the answers to an input query would be overwhelming. For a query with a handful of answer entities, offering refinements is unlikely to be helpful.

2. Filter out queries with fewer than $k = 5$ subtypes: The majority of YAGO types have only a single subtype; Figure 3 shows the distribution of subtypes per parent type. Unfortunately, these queries cannot be sensibly used as training data, since our goal is to generate refinement sets including $k = 5$ refinements. Fortunately, there is still enough data in the "tail" of this distribution to support a reasonably-sized training dataset.
3. Filter out refinements which differ from the input query by the addition of generic modifiers (dates / locations / genders): This is done to focus on refinements which reveal some kind of interesting structure specific to the query in question. Given "action films" as an input, providing a list of action movie subgenres (e.g. "martial arts films", "action comedy films", etc.) reveals new information specific to the domain of the query, while "action films from 1993" or "action films set in Great Britain" does not. The full list of filtering rules is included in Appendix B.5.

The full YAGO dataset includes 187K Wikipedia types. Preprocessing leaves 8,958 queries for $\mathcal{D}_{\texttt{QRESP}}$ and $\mathcal{D}_{\text{Random-F}}$, and 17,598 for $\mathcal{D}_{\text{Random}}$ ($\mathcal{D}_{\text{Random}}$ has more queries because there are some types with more than 5 subtypes in total, but fewer than 5 that pass pass the date / location / gender filters described above). The preprocessing steps are performed to ensure that the RSs in the final dataset are high-quality, suitable for model training, and represent realistic queries for which query refinement is a meaningful task.

B.4 YAGO dev and evaluation set selection

Our goal for our dev and test set is to select YAGO types that provide a realistic approximation of list-intent queries likely to be entered by users during search; these are also the types of queries where refinement selection approaches like `QRESP` have the potential to improve over a simpler approach. As such, for the dev set, we select YAGO types with a large number of subcategories (at least 15), each of which must have at least 200 answers. We require at least 15 candidate subtypes since `QRESP` (and methods like it) only has the potential to improve over random selection if there are a reasonable number of candidate categories to choose from. Similarly, we require 200 answers per candidate because trivial subtypes with only a handful of answers are clearly not suitable refinements (e.g. "Pirates of the Caribbean films" is not a good refinement for "Action films"); the refinement se-

Query: Academic journals	
$\mathcal{D}_{\texttt{QRESP}}$	Healthcare journals, Humanities journals, John Wiley & Sons academic journals, SAGE Publications academic journals, Scientific journals
$\mathcal{D}_{\text{Random}}$	Academic journals associated with non-profit organizations, Croatian-language journals, Latin-language journals, NRC Research Press academic journals, Ubiquity Press academic journals
$\mathcal{D}_{\text{Random-F}}$	Berghahn Books academic journals, English-language journals, Multidisciplinary academic journals, Spanish-language journals, World Scientific academic journals
Query: Islands	
$\mathcal{D}_{\texttt{QRESP}}$	Disputed islands, New islands, River islands, Uninhabited islands, Volcanic islands
$\mathcal{D}_{\text{Random}}$	Barrier islands, Islands of Europe, Islands of Sierra Leone, Mediterranean islands, Wikipedia categories named after islands
$\mathcal{D}_{\text{Random-F}}$	Barrier islands, Coral islands, Former islands, Private islands, Volcanic islands
Query: Musicians	
$\mathcal{D}_{\texttt{QRESP}}$	Composers, Electronic musicians, Percussionists, Singers, Woodwind musicians
$\mathcal{D}_{\text{Random}}$	Australian musicians, Compost Records artists, Good Vibe Recordings artists, Inner Ear artists, Wax Trax! Records artists
$\mathcal{D}_{\text{Random-F}}$	Black River Entertainment artists, DJM Records artists, Indianola Records artists, Pony Canyon artists, Ruthless Records artists

Table 11: Additional example refinements from $\mathcal{D}_{\texttt{QRESP}}$, $\mathcal{D}_{\text{Random}}$, and $\mathcal{D}_{\text{Random-F}}$.

lection task is much more challenging when it requires selecting among a large number of non-trivial potential refinements.

For our human evaluation set, we manually selected queries from the dev set which we judged to be interesting and realistic. We also attempted to choose queries from a variety of subject areas, including sports, music, science, politics, entertainment, and technology; see Appendix F for the full list.

B.5 YAGO Filtering rules

We use the following heuristics to obtain $\mathcal{C}_{\text{Filter}}(q)$ from $\mathcal{C}(q)$. For each q'_i in $\mathcal{C}(q)$, we compare q to q'_i. We remove q'_i from $\mathcal{C}_{\text{Filter}}(q)$ if it differed from q by:

- The addition of an entity tagged by Spacy as `DATE, GEP, NORP`, or `LOC`. (e.g. "Politicians" → "American Politicians").
- The addition of a phrase matching one of the following regular expressions:
  - `[0-9]{1,2}(st|th)(-| )century`
  - `1[0-9]{3}[^0-9]`
  - `[0-9]{3}[^0-9]`

  This filters out refinements like "Politicians" → 19^{th} century politicians.
- The addition of one of the following: `male, female, men, women` (e.g. "Politicians" → "Male politicians").

B.6 Checks on `QRESP` optimization

In Fig. 4, we plot the costs $\mathcal{S}(\mathcal{R}(q))$ of the RSs in $\mathcal{D}_{\texttt{QRESP}}$ versus those in $\mathcal{D}_{\text{Random-F}}$. The results confirm that QRESP is able to identify RSs that have substantially lower cost than choosing randomly from $\mathcal{C}_{\text{Filter}}(s)$.

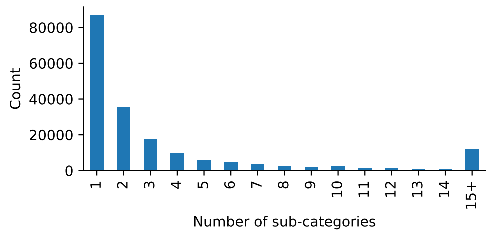


Figure 3: YAGO3 category distribution. The x-axis indicates the number of sub-categories associated with a given YAGO3 category, and the heigh indicates the number of categories with that many subcategories. For instance, roughly 80K YAGO3 categories have exactly one subcategory.

C. Annotation process

Annotations were collected using the Amazon Mechanical Turk platform.

C.1 Annotators

Mechanical Turk crowd workers were required to have Masters qualifications, and also required to pass a qualification quiz testing comprehension and good performance on the task. Roughly 25 workers took the qualification quiz; we selected the top 5 to perform annotations.

We maintained contact with annotators via email, fielding questions and discussing challenging cases. Annotators were paid per annotation (or HIT); we chose the HIT rate to target an hourly wage of between \$15 / hour and \$18 / hour.

C.2 Annotation procedure and annotator agreement

Each example was annotated by two crowd workers. For the Stage 1 evaluation (see §2.3), a refinement is considered fluent if both annotators agree that it is fluent, and similarly for relevance. For Stage 2 evaluation, if one annotator is neutral while the other prefers choice A, we mark choice A as preferred overall. If one annotator prefers A while the other prefers B, we mark the example as neutral.

Table 12 shows measures of inter-annotator agreement as measured by Cohen's κ, as well as the percentage of refinements that passed Stage 1 quality filters. The κ values range between 0.4 and 0.6, often considered to indicate "moderate" agreement [Landis and Koch, 1977].

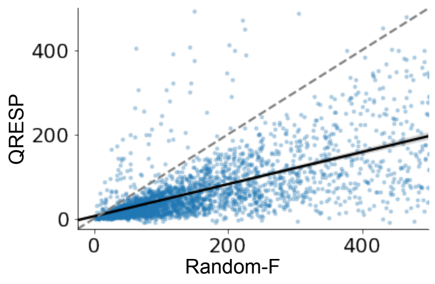


Figure 4: Costs of refinement sets $\mathcal{R}_{\text{Random-F}}$ and $\mathcal{R}_{\texttt{QRESP}}$. Each point represents a single query q. The x-axis indicates the cost $\mathcal{S}(\mathcal{R}_{\text{Random-F}}(q))$ computed by Eq. 1, and the y-axis indicates the cost $\mathcal{S}(\mathcal{R}_{\texttt{QRESP}}(q))$. The solid black line is the least-squares fit. In general, $\mathcal{R}_{\texttt{QRESP}}$ refinements achieve much lower cost compared to $\mathcal{R}_{\text{Random-F}}$. Points above the diagonal indicate "search errors", where the ILP solver reached its time budget before finding a lost-cost solution.

D. Statistical testing

For the human evaluations reported in §4.2 and §5.3, we perform statistical tests to determine

	Criterion	Cohen's κ	% passed
Stage 1	Fluency	0.37	99.1
	Relevance	0.61	90.6
Stage 2	Comprehensiveness	0.53	-
	Interestingness	0.46	-
	Non-redundancy	0.39	-
	Overall	0.52	-

Table 12: Measures of annotation quality. For Stage 1, "% passed" shows the percentage of refinements passing initial quality filters. The κ value for "Fluency" is deceptively low, and is a result of the heavily imbalanced label distribution: > 99% of refinements passed the fluency screen.

whether system A (`QRESP`) is preferred by annotators over system B at a rate statistically different from chance.

For Stage 1, we test the null hypothesis that refinements from systems A and B pass the Stage 1 screen at the same rate. We construct a contingency table where the first column contains counts for the number of refinements from system A that pass and fail the screen, respectively. The second column contains similar counts for system B. We then perform Fisher's exact test[2] on the resulting contingency table.

For Stage 2, we perform a separate test for each of the four attributes. In particular, we perform a binomial test[3] to test the null hypothesis that a rater is equally likely to prefer A as to prefer B, against the alternative that a rater is more likely to prefer A. For simplicity, we excluded "neutral" cases where the rater could not decide between A and B.

E. Refinement generation

E.1 Selection of queries for NQ+TREC

In the interest of evaluating our models on real-world list-intent queries, we selected a handful of evaluation queries from Natural Questions and from the TREC 2009 Million Query track. For Natural Questions, we filtered down to questions with answers found in a table or list, and then manually selected a collection of queries from this list. We aimed to select list-intent queries with a large number of potential answer entities, across a variety of domains. For TREC, we similarly filtered down to queries from the `List` query class, and then manually selected queries for diversity and interest. The TREC data can be downloaded from `https://trec.nist.gov/data/million.query09.html`.

The full list of TREC and NQ evaluation queries is included in Appendix F.

E.2 Additional human evaluations

In §5.3, we compare the results of $\mathcal{M}_{\texttt{QRESP}}$ against $\mathcal{M}_{\text{Random}}$ and $\mathcal{M}_{\text{Separate}}$ on queries from YAGO. We also performed A / B tests for $\mathcal{M}_{\text{Random-F}}$; the results are generally similar to what we found comparing against $\mathcal{M}_{\text{Random}}$ and are shown in Table 7. $\mathcal{M}_{\texttt{QRESP}}$ shows a smaller advantage in interestingness over $\mathcal{M}_{\text{Random-F}}$ compared to $\mathcal{M}_{\text{Random}}$; this likely

2. `docs.scipy.org/doc/scipy/reference/generated/scipy.stats.fisher_exact.html`
3. `docs.scipy.org/doc/scipy/reference/generated/scipy.stats.binomtest.html`

occurs since $\mathcal{M}_{\text{Random-F}}$ is not trained on generic refinements which simply add locations or dates.

	A = `QRESP` vs. **B** = Random-F		
$N = 107$	**A**	Neutral	**B**
Fluent + Relevant	**94%**	-	92%
Comprehensive	**59%****	21%	20%
Interesting	**39%**	35%	26%
Non-redundant	**23%**	62%	15%
Overall	**59%****	23%	18%

Table 13: Results of A / B tests on the YAGO human evaluation set, comparing $\mathcal{M}_{\texttt{QRESP}}$ against $\mathcal{M}_{\text{Random-F}}$. $\mathcal{M}_{\texttt{QRESP}}$ is more comprehensive. ** indicates $p < 0.001$.

F. Evaluation queries

The full list of human evaluation queries for YAGO and NQ+TREC is shown in Table 14.

G. Annotation guide

The annotation UI is shown in Fig. 5. The instructions summary is show in Fig. 6. Detailed instructions are shown in Fig. 7.

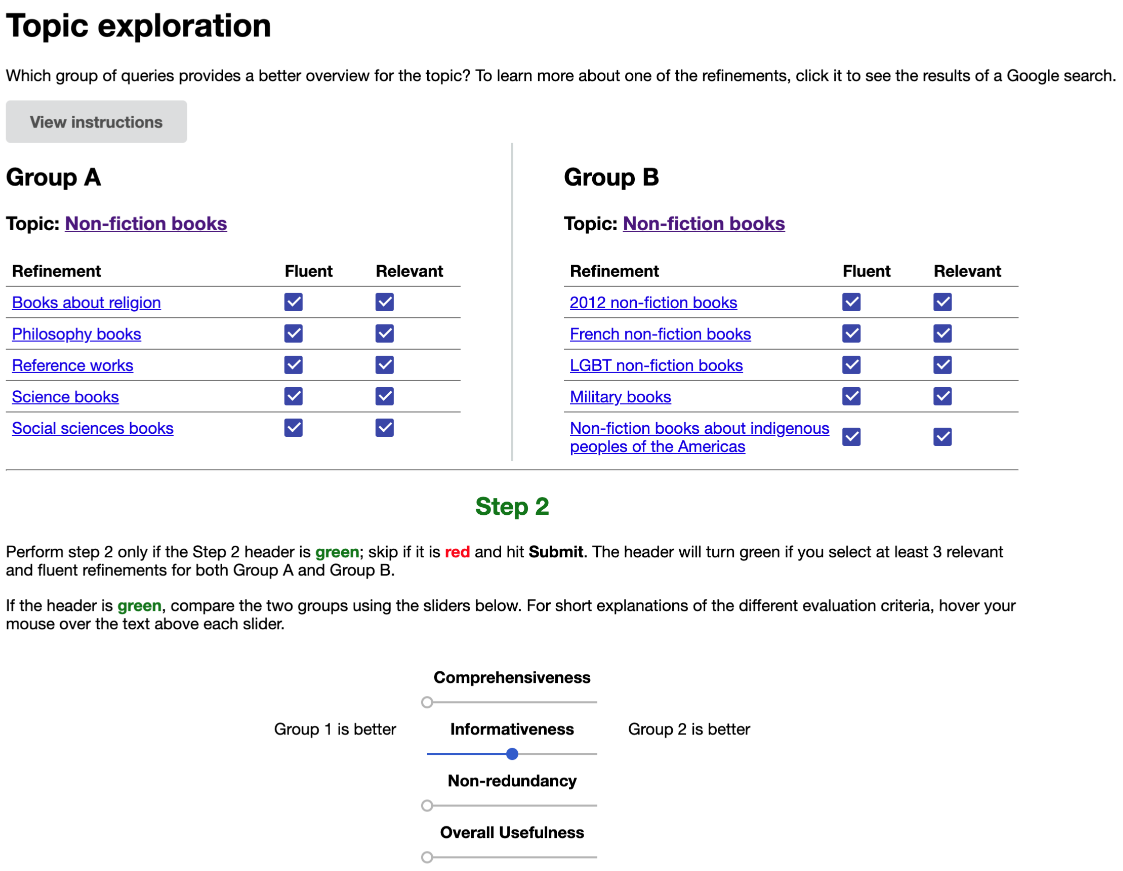
Academic journals, Action films, Activists, Albums, American activists, American artists, American military officers, American songs, Animals, Aviators, Baseball players, Battles, Battles of the Middle Ages, Birds, Boxers, Brazilian people, Bridges, British musicians, Buildings and structures in England, Canadian people, Charitable organizations, Chinese people, Christian saints, Classical musicians, Comics characters, Companies based in Tokyo, Companies of the United States, Concertos, Dance music albums, Dictionaries, Electronic albums, Energy companies, Engineers, Engines, European films, Films, Finnish writers, Firearms, Foods, French novels, German films, History books, History museums, Indian films, Indian writers, Insect genera, Islands, Italian painters, Japanese songs, Lakes, Languages, Locomotives, Magazines, Manufacturing companies, Martial arts people, Media executives, Medical journals, Minerals, Mobile phones, Music award winners, Musicians, Newspapers, Non-fiction books, Novels, Operas, Organisations based in India, Organisations based in Singapore, Organizations based in the United States, Painters, Pakistani films, People associated with crime, People associated with religion, People from New York City, People in finance, Philosophers, Philosophical works, Physicians, Plays, Poets, Political organizations, Political parties, Proteins, Racing drivers, Radio stations, Researchers, Rock songs, Schools in London, Scientists, Ships, Singers, Social scientists, Songs, Sports competitions, Sports events, Sports leagues, Sports venues, Swimmers, Tools, Typefaces, United States federal judges, Vehicles, Video games, Weapons, Websites, Writers

(a) YAGO queries

19th-century artists, Active volcanoes in the Philippines, African wool producers, Apple products, Architects in New Jersey, Arguments for the existence of God, Astronauts who stepped on the moon, BBC science news, BMW car models, Backward compatible games for XBox One, Baseball players featured on postage stamps, Battles of the revolutionary war, Best selling artists of all time, Billboard hot 100 number-one singles, Books in the New Testament, Branches of medicine, Bright stars in the sky, Cast of the movie "Now you see me", Causes of the French Revolution, Census regions in the United States, Chief ministers of Indian states, Cities and towns in Northern California, Cities that have held the Olympic Games, Cities with high murder rates, Communist countries during the Cold War, Countries where US citizens can travel without a visa, Countries with French-speaking people, Democratic countries, Disney Pixar movies, Disney princesses, Division 2 colleges in the midwest, Earthquakes, Foods brought to the New World from Europe, Forbes list of largest companies in the world, Functions of all the body systems, Functions of the government, Games for super nintendo classic, Gods and goddesses of the world, Government monopolies in the United States, Greatest NBA players of all time, Hall of fame football players, Hotels near downtown Houston, Independent power producers in South Africa, Indian spices, James Bond movies, Jesuit universities in the United States, John D Rockefeller's philanthropic projects, Languages spoken in India, Large charities, Largest cities in the world, Major exports of the United States, Malayalam movies, Most densely populated areas in the world, Most frequently used words in English, Most popular video games, Most spoken languages in the world, Movies Robert de Niro played in, National monuments in the United States, Natural air pollutants, Natural resources, Nobel Prize winners, Oil and gas companies in Kuwait, Oscar winners, PS4 games, Participants at the Battle of Wounded Knee, Places where carbon is stored on earth, Players who have a receiving touchdown in a superbowl, Political parties in India, Popular YouTube channels, Private medical colleges in Sindh, Public high schools in Brooklyn New York, Public sector mutual funds in India, Renewable energy companies, Rock and Roll Hall of Fame artists, Roles of local government in the Philippines, Romantic anime shows in English dub, Rulers of England, Satellites launched by India, Schools that offer architecture in Nigeria, Songs in West Side Story, Songs with California in the title, Sources of US oil, States in Nigeria, Stocks in the Dow Jones industrial average, Supreme court justices, Time Magazine person of the year winners, Trees with heart-shaped leaves, US presidents, Universities and colleges in Australia, Walt Disney films, World stock exchanges, Wrestling promotions in the United States, XBox 360 games

(b) NQ+TREC queries

Table 14: Full list of queries used for human evaluation.



Topic exploration

Which group of queries provides a better overview for the topic? To learn more about one of the refinements, click it to see the results of a Google search.

Group A

Topic: Non-fiction books

Refinement	Fluent	Relevant
Books about religion	✓	✓
Philosophy books	✓	✓
Reference works	✓	✓
Science books	✓	✓
Social sciences books	✓	✓

Group B

Topic: Non-fiction books

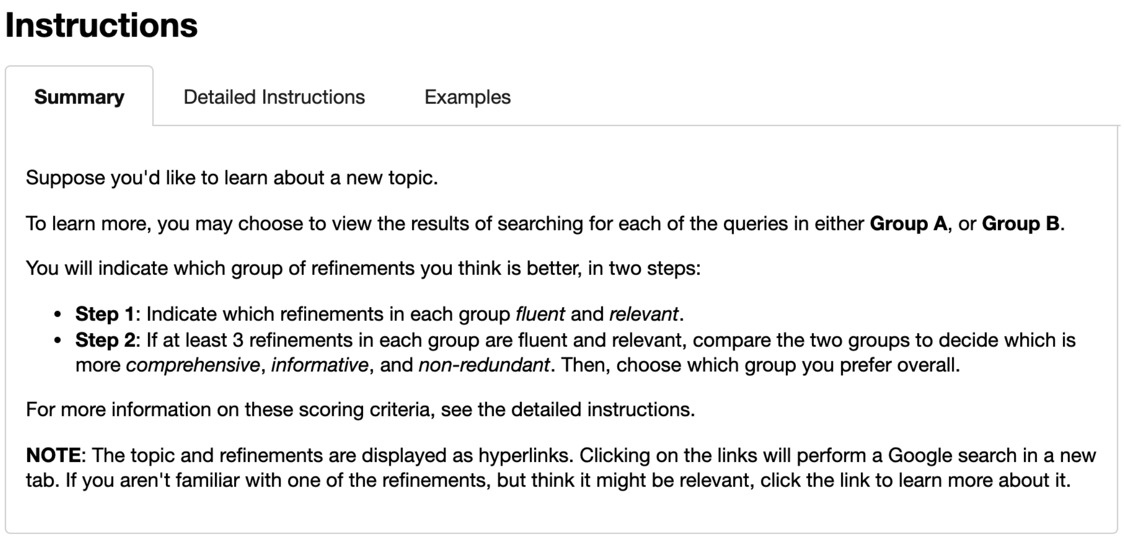
Refinement	Fluent	Relevant
2012 non-fiction books	✓	✓
French non-fiction books	✓	✓
LGBT non-fiction books	✓	✓
Military books	✓	✓
Non-fiction books about indigenous peoples of the Americas	✓	✓

Step 2

Perform step 2 only if the Step 2 header is **green**; skip if it is **red** and hit **Submit**. The header will turn green if you select at least 3 relevant and fluent refinements for both Group A and Group B.

If the header is **green**, compare the two groups using the sliders below. For short explanations of the different evaluation criteria, hover your mouse over the text above each slider.

Figure 5: Annotation UI and summary of annotation instructions.



Instructions

Suppose you'd like to learn about a new topic.

To learn more, you may choose to view the results of searching for each of the queries in either **Group A**, or **Group B**.

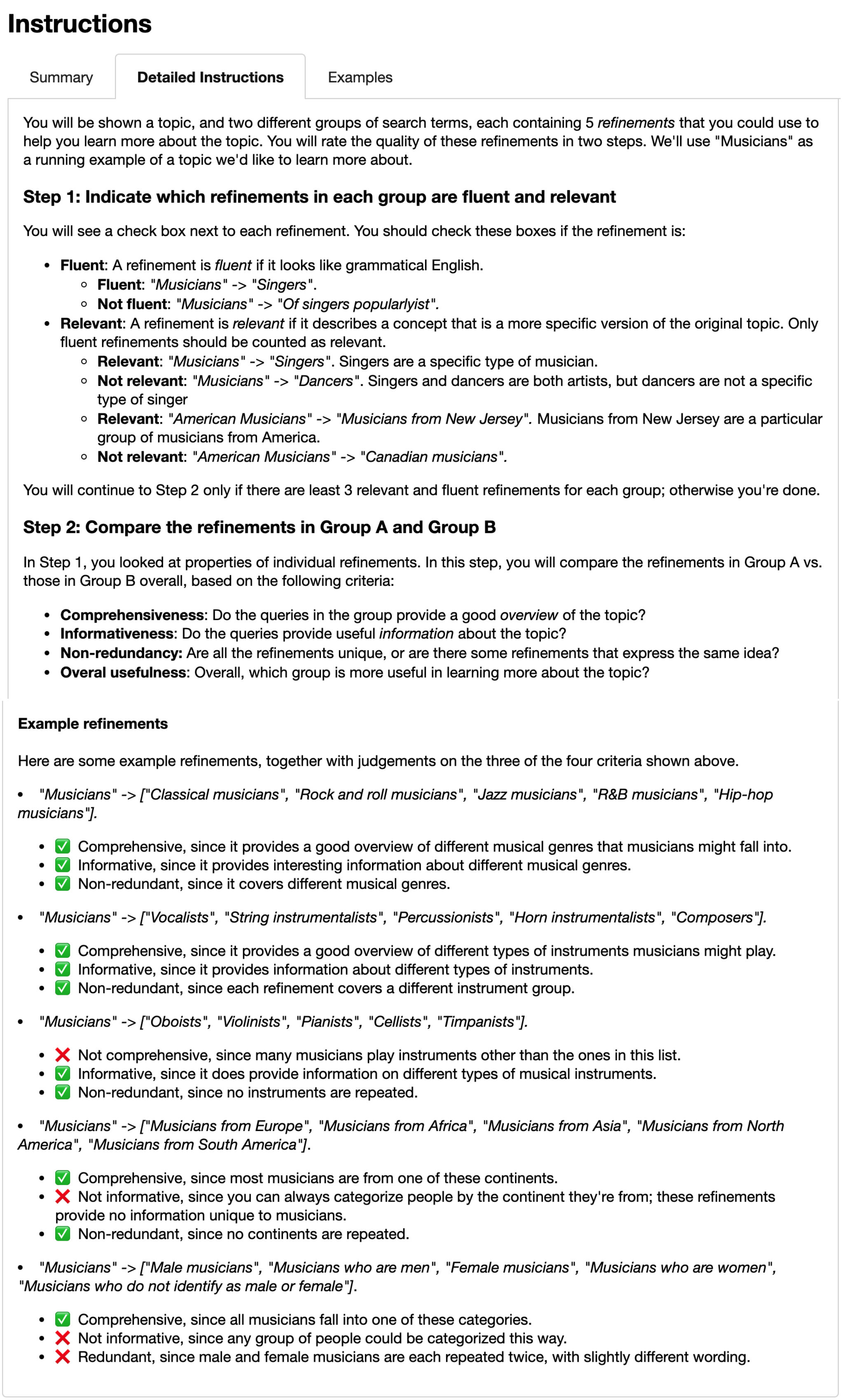
You will indicate which group of refinements you think is better, in two steps:

- **Step 1**: Indicate which refinements in each group *fluent* and *relevant*.
- **Step 2**: If at least 3 refinements in each group are fluent and relevant, compare the two groups to decide which is more *comprehensive*, *informative*, and *non-redundant*. Then, choose which group you prefer overall.

For more information on these scoring criteria, see the detailed instructions.

NOTE: The topic and refinements are displayed as hyperlinks. Clicking on the links will perform a Google search in a new tab. If you aren't familiar with one of the refinements, but think it might be relevant, click the link to learn more about it.

Figure 6: Annotation UI and summary of annotation instructions.



Instructions

Summary | **Detailed Instructions** | Examples

You will be shown a topic, and two different groups of search terms, each containing 5 *refinements* that you could use to help you learn more about the topic. You will rate the quality of these refinements in two steps. We'll use "Musicians" as a running example of a topic we'd like to learn more about.

Step 1: Indicate which refinements in each group are fluent and relevant

You will see a check box next to each refinement. You should check these boxes if the refinement is:

- **Fluent**: A refinement is *fluent* if it looks like grammatical English.
  - **Fluent**: "*Musicians*" -> "*Singers*".
  - **Not fluent**: "*Musicians*" -> "*Of singers popularlyist*".
- **Relevant**: A refinement is *relevant* if it describes a concept that is a more specific version of the original topic. Only fluent refinements should be counted as relevant.
  - **Relevant**: "*Musicians*" -> "*Singers*". Singers are a specific type of musician.
  - **Not relevant**: "*Musicians*" -> "*Dancers*". Singers and dancers are both artists, but dancers are not a specific type of singer
  - **Relevant**: "*American Musicians*" -> "*Musicians from New Jersey*". Musicians from New Jersey are a particular group of musicians from America.
  - **Not relevant**: "*American Musicians*" -> "*Canadian musicians*".

You will continue to Step 2 only if there are least 3 relevant and fluent refinements for each group; otherwise you're done.

Step 2: Compare the refinements in Group A and Group B

In Step 1, you looked at properties of individual refinements. In this step, you will compare the refinements in Group A vs. those in Group B overall, based on the following criteria:

- **Comprehensiveness**: Do the queries in the group provide a good *overview* of the topic?
- **Informativeness**: Do the queries provide useful *information* about the topic?
- **Non-redundancy:** Are all the refinements unique, or are there some refinements that express the same idea?
- **Overal usefulness**: Overall, which group is more useful in learning more about the topic?

Example refinements

Here are some example refinements, together with judgements on the three of the four criteria shown above.

- *"Musicians" -> ["Classical musicians", "Rock and roll musicians", "Jazz musicians", "R&B musicians", "Hip-hop musicians"].*
  - ✅ Comprehensive, since it provides a good overview of different musical genres that musicians might fall into.
  - ✅ Informative, since it provides interesting information about different musical genres.
  - ✅ Non-redundant, since it covers different musical genres.
- *"Musicians" -> ["Vocalists", "String instrumentalists", "Percussionists", "Horn instrumentalists", "Composers"].*
  - ✅ Comprehensive, since it provides a good overview of different types of instruments musicians might play.
  - ✅ Informative, since it provides information about different types of instruments.
  - ✅ Non-redundant, since each refinement covers a different instrument group.
- *"Musicians" -> ["Oboists", "Violinists", "Pianists", "Cellists", "Timpanists"].*
  - ❌ Not comprehensive, since many musicians play instruments other than the ones in this list.
  - ✅ Informative, since it does provide information on different types of musical instruments.
  - ✅ Non-redundant, since no instruments are repeated.
- *"Musicians" -> ["Musicians from Europe", "Musicians from Africa", "Musicians from Asia", "Musicians from North America", "Musicians from South America"].*
  - ✅ Comprehensive, since most musicians are from one of these continents.
  - ❌ Not informative, since you can always categorize people by the continent they're from; these refinements provide no information unique to musicians.
  - ✅ Non-redundant, since no continents are repeated.
- *"Musicians" -> ["Male musicians", "Musicians who are men", "Female musicians", "Musicians who are women", "Musicians who do not identify as male or female"].*
  - ✅ Comprehensive, since all musicians fall into one of these categories.
  - ❌ Not informative, since any group of people could be categorized this way.
  - ❌ Redundant, since male and female musicians are each repeated twice, with slightly different wording.

Figure 7: Annotation UI and summary of annotation instructions.