# Underspecified Query Refinement via Natural Language Question Generation

*Hassan Sajjad*[1] *Patrick Pantel*[2] *Michael Gamon*[2]
(1) Qatar Computing Research Institute, Qatar Foundation, Doha, Qatar [*]
(2) Microsoft Research, Redmond, WA, USA
hsajjad@qf.org.qa, ppantel@microsoft.com, mgamon@microsoft.com

ABSTRACT
Underspecified queries are common in vertical search engines, leading to large result sets that are difficult for users to navigate. In this paper, we show that we can automatically guide users to their target results by engaging them in a dialog consisting of well-formed binary questions mined from unstructured data. We propose a system that extracts candidate attribute-value question terms from unstructured descriptions of records in a database. These terms are then filtered using a Maximum Entropy classifier to identify those that are suitable for question formation given a user query. We then select question terms via a novel ranking function that aims to minimize the number of question turns necessary for a user to find her target result. We evaluate the quality of system-generated questions for grammaticality and refinement effectiveness. Our final system shows best results in effectiveness, percentage of well-formed questions, and percentage of answerable questions over three baseline systems.

KEYWORDS: Query refinement, question generation, search as a dialog.

[*]This work was conducted at Microsoft Research.

## 1 Introduction

Vertical search engines, i.e., domain-specific search engines, retrieve ranked entities from an underlying database, often via unstructured keyword queries. Popular engines, such as Yelp, Bing Travel, IMDB, and Amazon share a common aspect with niche engines, such as BlueWine and OpticsPlanet: user queries are often underspecified. Whether because of the seemingly infinite inventory in the larger engines, or because of the esoteric collections in niche engines, or simply because users seek to browse a collection of results, underspecification is pervasive. When the set of specified entities exceeds the common limit of ten blue links, finding the desired entity can be a long and frustrating process.

For example, consider the query “blue polo with purple stripes" issued to the Bing Shopping engine. Pages of results are returned with correctly matching products. Browsing through these, especially in a mobile or handsfree scenario, can be prohibitively difficult. In most result sets, however, entities can form natural clusters based on important attributes. In our example, clusters can be formed based on price, designer, shirt patterns, etc. Some vertical search engines leverage such editorially defined clusters by forming a faceted search experience, where users can refine search results by selecting attribute values. Such experiences are expensive to build, require a heavily curated ontology to define the important attributes, and consequently are only feasible for large sites with significant revenue streams.

In this paper, we explore methods to automatically discover important attributes from unstructured data associated with entities in a database. Further, in a runtime scenario, we propose algorithms for selecting the best candidate attribute to ask about, based on various criteria, and we construct a binary natural language question for the user to answer. The main idea is to let the data guide the user in her search such that she can more quickly and effectively find her targeted entity.

Our first challenge is to select attribute terms from unstructured text that are in general important for a particular class of entities. We employ a Maximum Entropy technique for recognizing strings that appear to be good attribute terms. Our second challenge is to select the most appropriate attribute term given a user query and result set. Here we propose a set of ranking functions that optimize against the number of necessary questions to find the desired entity. On average, our ranking functions ensure finding the target entity using $\log_2(m)$ questions where $m$ is the number of entities in the original result set. Finally, we transform the selected attribute term into a well-formed natural language question by matching against a set of lexico-syntactic question templates. We evaluate our systems' ability to quickly and effectively find target entities in an Xbox Avatar Editor scenario. Question well-formedness (as a function of factors such as grammaticality and spelling) is also evaluated.

The rest of the paper is organized as follows. Previous work on interactive question generation is summarized in Section 2 and in Section 2 we formally define our problem and the Xbox Avatar experimental dataset. We present our algorithms for finding question candidates in Section 3. Finally, we present our experimental results in Section 4 and conclude with a discussion of future work in Section 5.

## 2 Related Work

With the recent evolution in online search systems, question generation systems that improve the retrieval results in response to a user query have become an important area of research.

A major step in this process is *what to ask in the conversation?*. This involves clustering either the entire web (pre-retrieval method) (Voorhees, 1985) or only the retrieved results in response to a user query (post-retrieval method) (Cutting et al., 1993; Hearst et al., 1996; Allen et al., 1993; Leouski and Croft, 1996). In this paper, we follow the post-retrieval method to cluster the retrieved results. The description of the cluster is then used in the form of a question to the user.

Scatter/Gather (Cutting et al., 1993; Hearst et al., 1996) is a cluster-based approach which divides the documents in the collection into K clusters. Based on a query, a subset of clusters are selected which are dynamically reclustered (Carpineto et al., 2009). The scatter/gather method assigns attribute terms (descriptions) to the clusters from the feature vector or centroid. These attribute terms are difficult to use and understand especially in our scenario where we want to use them as questions to the user.

Suffix Tree Clustering ensures that the attribute terms of the clusters are meaningful and usable (Zamir and Etzioni, 1999). The idea is to extract terms from the text which are complete, self contained and meaningful. Zamir and Etzioni (1999) achieved this by taking frequent terms which are not crossing sentence boundaries. The problem with Suffix Tree Clustering is that only terms are used in the similarity metric for documents. This results in a decrease in the quality of clusters – especially for languages with free constituent order where parts of speech may come in various orders in a sentence (Carpineto et al., 2009; Masłowska, 2003).

Another class of algorithms focuses both on the quality of the cluster and the quality of its attribute terms. Vivísimo and Lingo (Osinski, 2006) are algorithms of this type. Lingo follows similar steps as that of Suffix Tree Cluster. It differs at query level where it finds abstract concepts from the query and matches them with frequent terms.

Kotov and Zhai (2010) add a question/answering feature to a search engine in order to improve search results and to guide the user to the output they are looking for. They generate a question for every candidate attribute and rank the questions using various heuristics, such as the number of query words that a question candidate matches. A set of top ranked questions are shown to the user who then selects the most relevant question according to their requirement. This user action leads to a modification of the original query and to an update of the search results. They require user input to select the best question. In this paper, we propose a ranking function to automatically rank the questions in a way that minimizes the number of questions needed to find the target entity.

Ontology-based term selection methods use dictionaries, thesauri and WordNet to learn the association between query terms and candidate terms (Bhogal et al., 2007; Hersh et al., 1992; Basili et al., 2007). Each term is mapped to a concept in an ontology. A term is a good candidate for selection if it belongs to the same concept as the query term. The drawback of using ontologies is that they are not available for all languages and for all domains, and their construction is expensive and time consuming. A detailed analysis of ontology-based query expansion can be found in Bhogal et al. (2007).

In contrast to most of the previous work on methods to browse web results, our domain is a web of entities. We learn good attribute terms from the unstructured data associated with the entities using a Maximum Entropy technique. We propose a set of ranking functions that optimize against the number of necessary questions to find the desired entity. Our

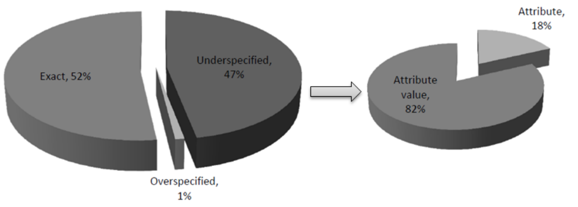

Figure 1: *Left*: Percentage of query types found in a random sample of the Avatar Dataset. *Right*: Percentage of relevant question types for underspecified queries.

method does not require any knowledge-based resource or user feedback, and uses only unstructured text.

In this section, we formally define the problem of helping a user quickly and effectively find her target result in a vertical search scenario.

## 2.1 Query Scope

Queries to vertical search engines can be categorized by the type of result set they generate: 1) an **exact query** is one that leads to a single result; 2) an **underspecified query** is one that leads to multiple results; and 3) an **overspecified query** is one that leads to zero results. For underspecified and overspecified queries, a follow-up user action is necessary in order to satisfy the information need.

The left pie chart in Figure 1 illustrates the percentage of query types in one dataset, introduced in Section 2.3. Although numbers vary by source and domain, the underspecified queries generally greatly outnumber overspecified queries.

Underspecified queries, which form the focus of this paper, result in multiple valid matches called a **confusion set**. When this set is large, it is difficult for a user to navigate through the matches to find her target result. There is an opportunity to help the user by building a system capable of interactively narrowing-down the confusion set.

## 2.2 Task Definition

We refer to the underlying data store being queried as a **Database**. We assume no structure within the database other than: 1) the set of records constituting the full search space; and 2) records belong to one or more semantic categories. For example, in a clothing database, each item of clothing is a record belonging to categories such as *pants, t-shirts, sweaters,* and *socks*. Some databases will contain other structured information such as attributes (e.g., t-shirts have a *color*, *price* and a *designer*) and relations (e.g., a particular t-shirt coordinates well with a set of pants). Although this information can be (and has been) leveraged for guiding users through a faceted search experience, we focus in this paper on the extraction of salient questions from **unstructured data**. Specifically, we aim to leverage user-generated comments and descriptions of database records as the source of information from which we will guide search users.

**Problem Statement:** Consider a database $\mathscr{D}$ where each record $r$ is associated with a set of semantic categories $C_r$ and a set of unstructured textual descriptions $S_r = \{s_{r1}, s_{r2}, ..., s_{rk}\}$.

Given a user query and a matching confusion set $R = \{r_1, r_2, ..., r_m\}$, our task is to ask a natural language question to the user that, based on the user's answer, best reduces the size of the confusion set.

For example, consider our query *"blue polo with purple stripes*" from our Bing Shopping scenario in Section 1. Suppose that there are 30 resulting matches in $R$ consisting of 14 long-sleeved shirts and 16 short-sleeved shirts, as well as 10 shirts each from three fashion designers Ralph Lauren, Calvin Klein and Marc Jacobs. Suppose also that the user is seeking a short-sleeved polo from Ralph Lauren. Candidate questions to ask the user include: $Q_1$ "Are you looking for a long-sleeved shirt?"; $Q_2$ "Do you want a shirt by Marc Jacobs?"; and $Q_3$ "What fashion designer would you like?" The answer to $Q_1$ would result in cutting the confusion set nearly in half, whereas the answers to $Q_2$ and $Q_3$ would remove a third and two thirds of the confusion set, respectively. In this case then, $Q_3$ is the question that best reduces the size of the confusion set. Formulations of how to best reduce a confusion set are discussed in detail in Section 3.

Questions can be categorized as attribute questions or attribute value questions. An **attribute question** is one that seeks the value of an attribute of a semantic category. For our t-shirt category above, $Q_3$ is an example attribute question. It is seeking the value of the *fashion designer* attribute. An **attribute value question** is a binary question that asks if the target result has a particular attribute value. $Q_1$ and $Q_2$ are example attribute value questions, where $Q_1$ is asking if the desired sleeve length is *long* and $Q_2$ is asking if the desired fashion designer is *Marc Jacobs*. Attribute questions generally result in finer reductions in the confusion set, however they are cognitively harder to answer than their binary counterparts. Not considered in this paper are other more complex question types such as set questions (e.g., "*Is the shirt blue, red, or green?*") and compound questions (e.g., "*Is the shirt blue and short-sleeved?*").

We manually inspected the underspecified queries illustrated in Figure 1 along with their resulting confusion sets. We annotated each according to the question type (attribute or attribute value) of the question that would lead to the largest reduction in size of the confusion set. When both question types resulted in the same reduction of the confusion set, we preferred the binary attribute value type since it is easier for users to answer. The rightmost chart in Figure 1 illustrates the result of the study. In 82% of the cases, an attribute value question was deemed more appropriate than a value question. Based on this insight, we limit the scope of this paper to the automatic generation of attribute value questions.

**Textual Grounding:** We learn attribute value questions without access to any ontological structure in the database. In the textual descriptions $S_r$ associated with a record $r$, we leverage the fact that many users will refer to the salient attributes and values of $r$. All unigrams, bigrams, and trigrams will be considered as candidate attributes and values, and we build statistical models to identify them.

## 2.3 Avatar Dataset

Very few people in the research community have access to the underlying databases powering vertical search engines such as Yelp, Bing Shopping, and Amazon. For the experiments in this paper we use a dataset that we believe is sufficiently similar to datasets used in vertical search engines on the Web, building upon publicly available data.

| Category | Count | Category | Count |
|---|---|---|---|
| Trousers | 54 | Shoes | 36 |
| Wrist wear | 16 | Shirt | 131 |
| Ring | 16 | Nose | 18 |
| Mouth | 27 | Hat | 34 |
| Facial other | 26 | Gloves | 16 |
| Glasses | 34 | Hair | 90 |
| Facial hair | 17 | Eyes | 45 |
| Eyebrows | 27 | Ears | 9 |
| Earrings | 34 | Chin | 9 |
| Costume | 27 | | |

Table 1: Semantic categories in the Avatar Dataset.

We consider the domain of Xbox Avatars. Users of the Xbox gaming console associate themselves with an avatar that they can personalize with clothing, body features, and accessories. We refer to each item that can be personalized, such as clothing, as an **asset**. We utilize a dataset that was developed for a separate research project and will be publicly released in early 2013 as part of that project (Volkova et al., forthcoming). Below we briefly describe that project's process for creating the data.

There are a total of 666 assets in the dataset and each asset can belong to one of 19 categories. Each category contains 35 assets on average. Table 1 shows the categories and the number of assets in each category. We define $\mathscr{D}$ as this collection of assets.

The textual descriptions S are collected using Mechanical Turk. To ensure the quality of the annotation, a two-tier process (similar to Chen and Dolan (2011)) was followed. First, the annotations were manually inspected in order to select a group of trusted workers based on the quality of their annotations and their commitment to work with the project for a longer period of time. Only these trusted workers were then allowed to annotate.

For each asset, 50 descriptions were obtained from 50 different workers, where descriptions were produced in a task-independent manner (i.e., the annotators were not aware of the final use of these descriptions). Workers were asked to produce a description of the asset and its distinctive features. Sample descriptions of an asset from the category *Hat* are shown in Table 2.

Category attributes, such as the color of eyes, are grounded in the crowdsourced descriptions. Consider the category *Shirt* which has *sleeves*, *color* and *design* as general attributes. An instance (asset) of the category *shirt* contains the values of these attributes such as *long*, *brown* and *flag on chest* for the attributes *sleeves*, *color* and *design* respectively. These attributes and their values are not explicitly stored for assets. Instead, this information is grounded in the free text description of the assets.

We refer to this data set as the **Avatar Dataset**.

## 2.4 Summary

Our goal is to generate meaningful and well-formed attribute value questions for underspecified queries in the Avatar Dataset. In the next section, we describe our system architecture for question generation.

| **Descriptions of a hat asset** |
|---|
| Green color flower design hat |
| Drab and yellow beanie with flowers |
| Green, flower print elastic clothing hat |
| Green toboggan with flowers and stripes |
| Green and yellow winter hat, daisy design on them and pompom on top |

Table 2: Example descriptions for an Avatar asset from the category *Hat*.

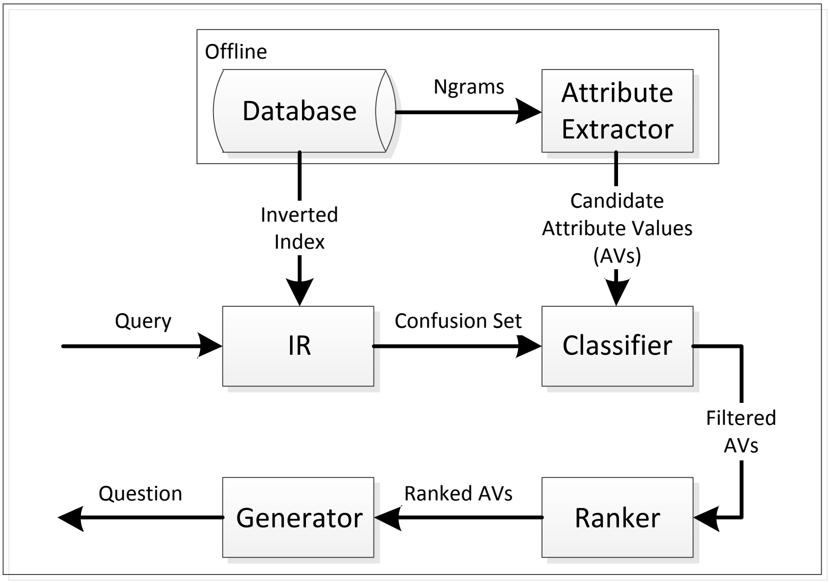

Figure 2: Question generation architecture.

## 3 Question Generation

Figure 2 outlines our question generation architecture. The input is an unstructured and underspecfied user query and the output is a well-formed attribute value question to the user. Offline processes are first applied to build an inverted index mapping each word in $S$ (the textual descriptions defined in Section 2.2) to its corresponding assets, and to extract candidate attribute values from $S$. Given a user query, an IR system retrieves a confusion set consisting of matching assets in the database. A classifier is applied to select the appropriate candidate attribute values, which are then ranked according to how they are expected to reduce the size of the confusion set. The top ranked attribute value is formulated into a question by the generator. Below we describe each component in turn.

### 3.1 Attribute Extractor

As described in Section 2.2, category attributes are not explicitly modeled in the database and must instead be inferred from the textual descriptions associated with each asset. Each unigram, bigram, and trigram is considered as a potential attribute value. The Attribute

Extractor associates with each category the ngrams that appear to be most likely attribute value candidates.

If an attribute value, such as the sleeve length of a shirt, is salient to a category, then we hypothesize that strings referring to the attribute value, such as “short-sleeved” and “short sleeves”, will occur more often in descriptions for assets of the category then for other assets. We therefore seek ngrams that are highly associated with each category, where association can be measured using statistics such as pointwise mutual information (PMI) and log-likelihood. In this paper, we use PMI. Given an ngram $n \in S$, we measure its association with a category $c$ as:

$$\mathrm{PMI}(n;c) = \log \frac{P(n,c)}{P(n)P(c)} \tag{1}$$

where $P(n,c)$ is the probability that an ngram in a description of an asset in $c$ is $n$, $P(n)$ is the probability that an ngram in any description is $n$ and $P(c)$ is the probability of any ngram occurring in a description in class $c$.

Ngrams with a PMI score higher than a predetermined threshold with a category are selected as candidate attribute values for that category. The resulting candidates are noisy and will be further filtered online by the Classifier component.[1] Table 3 lists examples of good and bad candidate attribute values for an asset from the *Shirt* category.

| **Entity** | Blue half sleeved polo with stripes |
|---|---|
| **Attribute values** | |
| **Answerable** | Blue, half sleeved, polo, stripes |
| **Unanswerable** | polo with, with, with stripes |

Table 3: Sample ngrams extracted as candidate attribute values by the Attribute Extractor for an asset from the *Shirt* category.

## 3.2 IR

Following (Salton, 1971), we build an inverted index mapping each ngram $n$ in $S$ to its corresponding asset $r$ along with its tf-idf, defined as:

$$\text{tf-idf}(n,r) = tf(n,r) \times \log idf(n) \tag{2}$$

where $tf(n,r)$ is the frequency of $n$ in $S_r$, and $idf(n)$ is the fraction of textual descriptions $s \in S$ containing $n$.

Let $\mathbf{r}$ be a vector of all ngrams in $S_r$ where the value of each ngram is its tf-idf with $r$. Then, given a query $q$, we form a query vector $\mathbf{q}$ consisting of all ngrams in $q$, where the value of each feature is 1. Our IR component first retrieves from the inverted index all assets matching an ngram with $q$. For each matching asset $r$, we then compute a simple IR rank score as the cosine of the angle between $\mathbf{q}$ and $\mathbf{r}$:

$$\mathrm{cosine}(\mathbf{q},\mathbf{r}) = \frac{\sum_i q_i \cdot r_i}{\sqrt{\sum_i q_i^2 \cdot \sum_i r_i^2}} \tag{3}$$

[1]In this paper, we experimentally set the threshold to 1.

| **Binary features** |
|---|
| unigram, bigram, trigram, POS tag sequence of candidate, separate POS tag of every word in the candidate, candidate is a substring of the query, candidate contains the queried category |
| **Real-valued features** |
| PMI score, log-likelihood score |

Table 4: Features used in our Maximum Entropy classifier.

## 3.3 Classifier

The Attribute Extractor provides a shortlist of salient attribute value candidates, but we still need to further filter this list to arrive at our final list of candidates. For example, we still find spurious and rare candidates as well as non-constituent terms (“long and") in the candidates. In addition, we know that the ultimate usefulness of a salient attribute is dependent on the query: An attribute value candidate may be salient for a category, but given a specific query, it can still be useless as a refinement candidate. For example, a salient attribute value may not lead to any reduction in the confusion set or it could overlap with what is already asked for in the query, making it redundant for a refinement question.

To address these issues, we use a machine learned model that utilizes features derived from both the user's query and the Attribute Extractor provided list, and filters out attribute value candidates that are unanswerable or not relevant given the query.

The model we use is a Maximum Entropy classifier. The selection of the training data for this supervised classification approach is described in detail in Section 4. For every query in the training set, we retrieve a confusion set using the IR component (Section 3.2) and select candidates from the list provided by the Attribute Extractor that also match at least one description of the entities in the confusion set: an attribute value candidate that does not fulfil that criterion is by definition not able to serve as a disambiguator on the set. We use the Ranker module (Section 3.4) to select two attribute-value candidates for every query in the training set and annotate them with nine automatically extracted features as summarized in Table 4. The *unigram* feature indicates that the candidate term is a unigram, similarly for *bigram* and *trigram*. The feature *POS tag sequence of candidate* represents the part-of- speech tags of the words in a candidate. *POS tag of every word in the candidate* indicates all individual POS tags. Information about the relation of the candidate to the query and the category of the queried record is captured using the features *candidate is a substring of the query* and *candidate contains the queried category* respectively. We also utilize real-valued features for the PMI score and log-likelihood score of the candidates wrt the category. Finally, we manually annotate each example as a positive example (good candidate) or a negative example (bad candidate). The Maximum Entropy classifier is trained on the annotated examples. Given a list of candidate attribute values, the Classifier predicts whether the attribute values are good candidates or bad. The values which are good candidates according to the Classifier are then input to the Ranker module.

## 3.4 Ranker

Recall our problem definition from Section 2.2, which states that we aim to ask a question that best reduces the size of the confusion set, thus allowing a user to find her target result with the minimum number of refinement questions. Our goal of finding the target result in an end to end system is difficult to evaluate without deployment. We instead introduce a ranking component that we reasonbaly expect to correlate with that goal of finding the right result. Our Ranker component orders the filtered attribute values from the Classifier according to how well each is expected to reduce the size of the confusion set. The top ranking attribute value will be used in formulating the final question asked to the user.

Since attribute value questions are binary, the most effective questions will be those that result in dividing the confusion set in half. This would result in an optimal interaction strategy where $\log_2 m$ questions are needed to guide a user through a confusion set of size $m$. We define our ranking score for an attribute value $n$ and confusion set $R$, $\text{score}_R(n)$, as a real-valued function ranging from zero to one where *zero* indicates that $n$ will cut the confusion set in half and *one* indicates that $n$ will leave the confusion set unchanged. We seek questions that minimize this score. Formally:

$$\text{score}_R(n) = 2\left|\frac{\sum_{r\in R}\phi_R(n)f(r)}{\sum_{r\in R}f(r)} - 0.5\right|$$

where $\text{score}_R(n)\colon \mathbb{R}\rightarrow[0,1]$, $f(r)$ represents a weight function associated with each asset $r$ in the confusion set, and $\phi_R(n)$ is the number of assets in the confusion set that hold the attribute value $n$ (estimated by whether or not at least one textual description mentions $n$).

If $f(r)=1$, $\text{score}_R(n)$ is minimized when $n$ cuts the confusion set in half. Recall however that each asset in the confusion set has a relevance score assigned by the IR module (see Section 3.2). It is therefore reasonable to assume that items at the head of $R$ will be more likely the target asset than items at the tail of $R$. We derive various definitions of the weight function $f(r)$ to capture this intuition:

$$f(r) = \begin{cases} M_1: & 1 \\ M_{rank}: & \frac{1}{\text{rank}(r)} \\ M_{ir}: & IR(r) \\ M_{dcg}: & \begin{cases} 1 \text{ for } \text{rank}(r)=1 \\ \frac{\text{rank}(r)}{\log_2 \text{rank}(r)} \text{ otherwise} \end{cases} \end{cases}$$

$M_1$ considers all assets in the confusion set equally probable to be the target asset. $M_{rank}$ weighs assets according to their rank in $R$ and $M_{ir}$ weighs them according to their cosine with the user query. Similarly, $M_{dcg}$ weighs assets according to their gain discounted by rank position (similarly to that done in the Discounted Cumulative Gain (DCG) metric used primarily in IR).

The attribute value candidate with the lowest score is selected to form the final question. We use the Attribute Extractor score as tie-breaker (see Section 3.1).

In our experiments, we build our system using $M_{rank}$. $M_1$, $M_{ir}$, and $M_{dcg}$ are used as evaluation metrics (see Section 4).

| Template | POS |
|---|---|
| Should it be | JJ (Cat: NN) |
| Should they be | JJ (Cat: NNS) |
| Do you want | DT JJ NN |
| Do you want (a/an) | JJ NN NN |
| Are these | JJ VB NNS |
| Is it | RB VBN |
| Is it (a/an) | JJ VB NN |
| Does it have | NN NNS |

Table 5: Question templates used to form a question

## 3.5 Generator

The final component in our system takes an attribute value as an input and produces a grammatical binary question from it. We use eight manually created question templates for this purpose. These question templates contain part-of-speech placeholders for the attribute value. Table 5 shows our question templates and a few examples of the part of speech sequences that can be used to complete each question template into a well-formed question. For question templates that only differ by an article (*a*, *an*), we check the first character of the question term to select the appropriate question template.

## 4 Experiments

## 4.1 Data Sets and Systems

From the Avatar Data Set described in Section 2.3, we sub-sampled a set of 160 assets for manual analysis. We refer to this set as the *Sampled Avatar Data Set*. For each asset we randomly chose one of the 50 available descriptions to serve as a set of random queries for the Sampled Avatar Data Set. Of the 160 queries, 75 were underspecified, which forms our *Underspecified Query Set*. Each query in that set is also associated with a confusion set as retrieved by the IR system from all assets in the Sampled Avatar Data Set.[2] We split the Underspecified Query Set into 50 query/asset pairs for training the maximum entropy classifier, called the *Classifier Training Set*, and 25 pairs for testing of the end-to-end system, called the *Test Set*.

We compare our system from Section 3, labeled *SYS*, against three baseline systems. For every query in the Test Set, we generate one question from each of the three baseline systems and our final system.

All three baseline systems extract attribute value candidates from the Attribute Extractor and that are relevant to the assets in the confusion set. As opposed to *SYS*, they do not use Classifier to filter the good candidate attribute values and use the output of the Attribute Extractor directly in Ranker. The systems select a question candidate based on the Ranking Function using $M_{rank}$ as a weight function. PMI is used as a tie-breaker. The baseline systems differ with respect to the ngram size that they consider. System *B1* considers unigram attribute value candidates, *B2* considers bigram candidates and *B3* trigram candidates.

[2]The IR system was built using 49 textual descriptions per asset since one was reserved for the Underspecified Query Set.

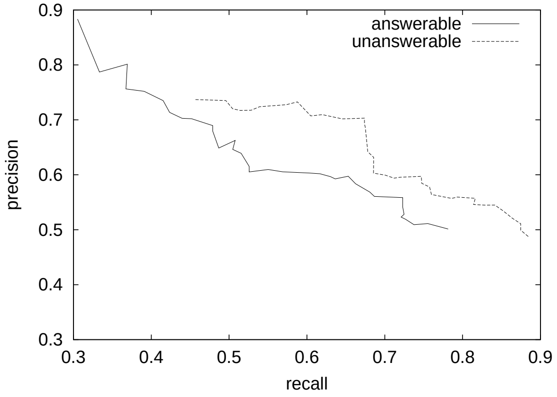

Figure 3: Precision-recall curve of the answerable and unanswerable classes.

## 4.2 Training of the Classifier

From the Classifier Training Set we need to derive a set of positive and negative examples for training. Positive examples are attribute values that are meaningful with respect to the given query and answerable if used in a question, whereas negative examples are either not meaningful or unanswerable. For simplicity, we refer to these examples as *answerable* and *unanswerable*, respectively.

Ideally, we would annotate all attribute value candidates for each query in the Classifier Training Set as answerable or unanswerable. In order to make the annotation task feasible, however, consider that the classifier needs to be optimized for its runtime task of filtering out attribute value pairs from the list provided by the Attribute Extractor. It is reasonable, therefore, to pick examples for annotation that are likely to be relevant in that scenario. For this purpose we first form the intersection of (i) the terms in the descriptions of the assets in the confusion set; and (ii) the list of attribute values produced by the Attribute Extractor. This produces a set of attribute value candidates just like the ones that the classifier will be exposed to at runtime. We also need to focus on finding answerable training cases, since the negative (unanswerable) cases are in the majority and hence much easier to come by. Of particular importance are cases that are selected by the Attribute Extractor and highly ranked by the Ranker, i.e. "borderline" candidates. We collect the top-10 attribute values (selected by the Attribute Extractor) for a given query and confusion set as measured by the Ranker (using $M_{rank}$ as a weight function). From this top-10 set, we pick the top candidate and a random candidate to annotate as answerable or unanswerable. The resulting training set consists of 100 data points, 47 answerable and 53 unanswerable. We represent each training case as a feature vector as described in Section 3.3.

We evaluate the classifier based on 10-fold cross validation on the training set. The precision-recall curve of answerable (solid line) and unanswerable (dotted line) for different probability thresholds is shown in Figure 3. For our final system, we select a probability threshold greater than 0.7 for the answerable question terms.

## 4.3 Output Judgments

There are two properties of a system-generated question that we want to evaluate. First and most importantly, we want to know how good a final question is with respect to best dividing

| | **ANS** | $M_1$ | $M_{rank}$ | $M_{ir}$ | $M_{dcg}$ |
|---|---|---|---|---|---|
| **B1** | 0.6 | 0.6 | 0.63 | 0.62 | 0.74 |
| **B2** | 0.76 | 0.38 | 0.44 | 0.39 | 0.57 |
| **B3** | 0.68 | 0.41 | 0.48 | 0.46 | 0.61 |
| **SYS** | **0.88** | **0.26** | **0.36** | **0.32** | **0.49** |

Table 6: System effectiveness at reducing the size of the confusion set. Better systems will have a high ANS score (i.e., more questions are answerable) and low values for $M_1$, $M_{rank}$, $M_{ir}$, and $M_{dcg}$.

the confusion set. Second, we evaluate the grammaticality of the question to address the quality of our question generation component. During the latter task, we found some cases where the POS sequence of a question term does not match with any of the POS sequences allowed for question templates in our generation component. In these cases no question can be generated and we distinguish these cases from the "formulated" questions.

We designed an evaluation form which shows a query from the test set with its corresponding gold asset and four questions generated by the four systems. For every question, the judge has to select whether (i) the correct answer to the question is "yes" or "no"; or (ii) whether the question is answerable. The confusion set is then reduced based on the answer to the question by matching the attribute value ngram against the descriptions for each asset in the confusion set. We keep separate statistics for the unanswerable questions as determined by the judge's input to (ii). One of the authors served as the judge.

## 4.4 Results

Table 6 shows the results of all systems on our four metrics (recall that our final system uses $M_{rank}$ in its Ranking Function, so the $M_{rank}$ column is not a bona fide evaluation result and is only included for completeness). The "ANS" column refers to the percentage of questions that are answerable and relevant in context of the gold asset. Better systems maximize ANS and minimize $M_1$, $M_{rank}$, $M_{ir}$, and $M_{dcg}$.

Our final system shows a significant increase in the percentage of answerable question terms in comparison with the baseline systems. It also has the lowest scores for every evaluation metric, which shows that the question terms generated by our system divide the confusion set better than the baseline systems.

One author also rated every question for well-formedness. Each question was judged for grammaticality and for non-grammar errors (e.g., spelling errors), which are accounted for in a separate category "Other". Table 7 summarizes the results. Our system resulted in the most well-formed queries, with fewer mismatches with the POS templates described in Table 5 and fewer grammatical errors.

The lower percentage of well-formed questions for B2 and B3 reflects the fact that bigrams and trigrams tended to contain more rare POS sequences (such as non-constituents) that could not be accommodated by any question template.

## 4.5 Error Analysis

For a few test queries, our system produces meaningless questions like "shaped mustache" in response to a query "long and broad mustache with terror face look". This can happen

| | Well-Formed | Errors | | |
|---|---|---|---|---|
| | | Template | Grammatical | Other |
| **B1** | 0.52 | **0.16** | 0.24 | 0.16 |
| **B2** | 0.48 | 0.36 | 0.16 | **0** |
| **B3** | 0.28 | 0.52 | 0.20 | **0** |
| **SYS** | **0.68** | **0.16** | **0.16** | 0.04 |

Table 7: Question well-formedness. Error types include no matching POS template (i.e., the Generator component did not fire), grammatical errors, and other errors such as misspelling.

when the classifier rejects all question candidates as unanswerable and the system selects a question term from the list provided by the Attribute Extractor with the lowest $M_{rank}$ score.

The answer to 30% of the questions resulted in the removal of the target asset from the confusion set. We found that there are two reasons for this, neither of which is a shortcoming of the system: either the user has made a mistake in answering the question or the description of the gold asset is not correct. We examined the latter cases and found that some textual descriptions associated with a few assets were inaccurate. For example, a pair of red pants was described as "red shorts" by a microtask worker. Similarly, some t-shirts are described as jackets. Suppose a user is searching for a pair of red pants and based on the description of the target asset and the confusion set, the system asks, "Are these shorts?" The correct response "no" will lead to the elimination of the target asset from the confusion set. Answering questions pertaining to the value of scalar facial attributes depends on a user's perception. The answer to questions "are these long lips?" or "are these bushy eyebrows?" depends on the user's notion of long and bushy. Here, again, the target asset may be wrongly excluded from the reduced confusion set. Similarly, the annotators sometimes confuse the position (left/right) of the attribute of an asset (e.g., "mole under the left/right eye").

Examining the ungrammatical questions, we found that the most common source of error is the use of a singular article with a mass noun, due to the fact that we neglected to distinguish between mass and count nouns in our question templates.

## 5 Conclusion

We have proposed a question generation system that produces refinement questions for underspecified queries from unstructured text in a vertical search scenario. We applied our system to an Xbox Avatar personalization dataset and found that compared to three baselines, our system offers the best reductions in confusion set size and the highest percentage of well-formed natural language questions.

There are many opportunities for future research in this area and on the Avatar Dataset. Some examples include automatic detection of inconsistent asset descriptions, refinements in attribute value extraction, and improved question generation including the generation of more complex questions.

## Acknowledgments

The authors thank Bill Dolan, Chris Brockett, Yun-Cheng Ju, and Svitlana Volkova for sharing their Avatar Dataset and providing vision and valuable guidance.

## References

Allen, R. B., Obry, P., and Littman, M. (1993). An interface for navigating clustered document sets returned by queries. In *Proceedings of the conference on Organizational computing systems*, COCS '93, New York, NY, USA.

Basili, R., Cao, D. D., Giannone, C., and Marocco, P. (2007). Data-driven dialogue for interactive question answering. In *AI*IA*.

Bhogal, J., Macfarlane, A., and Smith, P. (2007). A review of ontology based query expansion. *Information Processing and Management*, 43.

Carpineto, C., Osinski, S., Romano, G., and Weiss, D. (2009). A survey of web clustering engines. *ACM Computing Surveys*, 41(3).

Cutting, D. R., Karger, D. R., and Pedersen, J. O. (1993). Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, New York, NY, USA.

Hearst, M. A., Pedersen, J. O., and Alto, P. (1996). Reexamining the cluster hypothesis : Scatter / gather on retrieval results. *Computing Systems*.

Hersh, W. R., Hickam, D. H., and Leone, T. (1992). Words, concepts, or both: optimal indexing units for automated information retrieval. In *Proceedings of the 16th Annual Symposium on Computer Applications in Medical Care*, Oregon Health Sciences University, Portland.

Kotov, A. and Zhai, C. (2010). Towards natural question guided search. In *Proceedings of the 19th international conference on World wide web*, WWW '10, New York, NY, USA. ACM.

Leouski, A. V. and Croft, W. B. (1996). An evaluation of techniques for clustering search results. Technical report, Department of Computer Science, University of Massachusetts.

Masłowska, I. (2003). Phrase-based hierarchical clustering of web search results. In *Proceedings of the 25th European conference on IR research*.

Osinski, S. (2006). Improving quality of search results clustering with approximate matrix factorisations. In *Proceedings of the 28th European conference on IR research*.

Salton, G. (1971). *The SMART Retrieval System–Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Voorhees, E. M. (1985). The cluster hypothesis revisited. In *Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '85, New York, NY, USA.

Zamir, O. and Etzioni, O. (1999). Grouper: a dynamic clustering interface to web search results. In *Proceedings of the 18th international conference on World Wide Web*, WWW '99, New York, NY, USA.