Resolving Common Analytical Tasks in Text Databases

Sebastian Arnold
Beuth Hochschule für Technik
Luxemburger Straße 10
13353 Berlin, Germany
sarnold@beuth-hochschule.de

Alexander Löser
Beuth Hochschule für Technik
Luxemburger Straße 10
13353 Berlin, Germany
aloeser@beuth-hochschule.de

Torsten Kilias
Beuth Hochschule für Technik
Luxemburger Straße 10
13353 Berlin, Germany
tkilias@beuth-hochschule.de

ABSTRACT

With the convergence of data warehousing, online analytical processing and the Semantic Web, analytical tasks are no longer only designed and executed by experts. Instead, various users expect to query keyword search engines with analytical intentions. One efficient approach to answer these tasks is to leverage the factual information stored in large-scale text databases. These systems enable analysts to access unstructured text sources from the Web with structured query languages. The challenge of mapping keyword queries to structured queries has been approached in various forms. However, these systems are not able to detect the underlying intent of a task. Thus, they cannot infer the user's expectations towards specificity and form of the results. Moreover, a large fraction of queries for retrieving analytical results is rare. As a result, services for intent-aware task recognition perform poorly or are not even triggered on these long-tail queries. We report from a study over 102,360 query and click patterns from a factual search engine. Our analysis reveals six common analytical tasks: *explore*, *relate*, *resolve*, *list*, *compare* and *answer*. To distinguish among these, we study the effects of syntactical structures in the query, methods for interactive entity detection and query segmentation techniques. We evaluate these features on language models and Naive Bayes classifiers. From our evaluation we report a combined F1 score of 90% for the prediction of task intent from keyword queries.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation

General Terms

Human Factors, Languages, Experimentation, Design

Keywords

informational search; query intent; user goals; keywords

1. INTRODUCTION

Data analysis tasks are an important part of the search process, which involves multiple iterations of lookup, learn and investigatory subtasks [29]. One efficient approach of processing analytical tasks is to leverage the information stored in large-scale text databases. These systems are able to query structured data that has been extracted from text documents in the Web [21] and return factual results as well as text snippets and lineage information.

Please consider the following scenario (see Figure 2): The overall objective of an enterprise analyst is to select a supplier for VOIP technology. The analyst translates her possibly complex information need into a sequence of search tasks [12]: Who are suppliers for VOIP? What are the key facts about one of these companies, e.g. Cisco Systems? Who is the CEO? Is Cisco related to my strongest competitor? During that search process, she gathers the results in form of a spreadsheet table.

Each of these tasks requires the formulation of queries to the search system. Most business users can not translate these information demands into multiple structured queries. Rather, they desire to express complex tasks and intentions in a sequence of keyword queries. Current systems for keyword queries over relational databases [2, 17, 34] are able to parse unstructured queries and execute them against a structured database schema. However, these systems can neither discriminate between the required complex task intentions, nor do they understand the overall query objective of the user. We present an approach that is able to classify six of the most common complex analytical tasks at query time with very high F1 score. Our system maps these analytical tasks to visualization operators and generates an appropriate SQL query after the detection.

Our contribution. Our research bases on the analytical search engine GoOLAP, which provides an interactive search interface for exploratory queries on a text database.

1. In a long term study, we analyze a log of 102,360 keyword queries and click behavior posed to the GoOLAP search engine between 2008 and 2014. In this log, we spotted that 86.4% of the queries represent analytical tasks, such as undirected and directed informational searches [32] on structured data. By combining related work from Web search engines [7, 11, 12, 32], keyword queries in databases [2, 5, 17, 34] and our observations, we introduce a scheme of the most important analytical tasks, in particular tasks that seek to *explore*, *relate*, *resolve*, *list*, *compare* or *answer* factual

results. To our knowledge, this is the first work that combines these common tasks in a single taxonomy.

2. Our approach is based on high-level objectives of analytical search and focuses on the sub-tasks of the process. This vision is highly related to text databases and goes beyond keyword queries on structured databases.

3. During our analysis, we systematically spotted common patterns in query formulation that may help analytical search engines (e.g. OLAP systems) to predict these common tasks. The patterns are based on features that we derive from query syntax and schema information. We tested these features on a scheme of binary classifiers for the prediction of six common classes. From our evaluation of 520 expert-labeled queries we report an overall prediction accuracy of 96.8% and F1 score of 90.1%.

4. We implemented our results in a demonstrator[1]. Moreover, labeled query data sets for resolving analytical tasks are rare in the academic community. Therefore we publish our data set[2].

The rest of the paper is organized as follows: In Section 2, we discuss the background of related work. In Section 3, we introduce our model which combines previous work with our own observations. In Section 4, we describe the interactive process of feature extraction and classification. In Section 5, we present the evaluation setup and discuss our results. We draw a conclusion in Section 6.

2. BACKGROUND

We begin with a comprehension on related work. Then, we describe the process of analytical search tasks and user interaction. Finally, we introduce the implementation of this process using our prototype system GoOLAP.

2.1 Related Work

Spotting search task intentions is an active research area for the Web search and information retrieval community. The top-level classification of *informational*, *navigational* and *transactional* search has been published long before search engines included factual results and is based on the classic model of document retrieval [7, 32]. Informational queries make up the biggest proportion of Web search tasks with roughly 45–60% of queries [7, 30, 32, 38]. The specificity of these tasks can vary between very narrow (e.g. fact finding) and very wide (e.g. exploratory search) [29], while the complexity of tasks ranges from simple lookup to complex transformations spanning an entire query session [3]. Several attempts have been made to further structure informational intents, such as ambiguous and multi-faceted classification [4, 11], and query representation [16, 24]. While most of the spotted tasks can be expressed as structured queries, detecting intent from unstructured query formulation has only been approached before on high level [22, 27, 30, 38]. However, resolving search intentions to evaluate arbitrary analytical queries is a relatively new phenomena.

Obviously, commercial search engines investigate such techniques to retrieve parts of the knowledge graph [9, 36], but we are not aware of results available to the public. Specialized systems are able to rewrite keyword queries into structured SQL or SPARQL statements. DBXplorer [2] is one of the early systems that interpret unstructured lookup queries to relational databases. SQAK [34] is highly specialized on interpreting tasks that contain aggregation. STICS [17] is a search engine for Web and news data that incorporates entity resolution to enhance document lookup on the YAGO-NAGA ontology [23]. However, all these systems are specialized in fixed types of result presentation, such as tables or documents. To our knowledge, our work is the first academic result that incorporates six very common analytical tasks with different structured result types in a single search interface.

Text databases [1, 21] enable the user to query structured data out of text sources. The database utilizes an information extraction system to extract facts from text, each holding a set of pointers to the sources where it is mentioned. The user can then access these facts and their lineage using a structured query language, such as SQL or SPARQL. Our work on INDREX [26] extends the idea of text databases with the ability to express information extractors directly in SQL. Therefore, it provides a span-based data model which is able to store the results of information extraction and natural language processing. The user can formulate extractors with SQL using the proposed span predicate extensions.

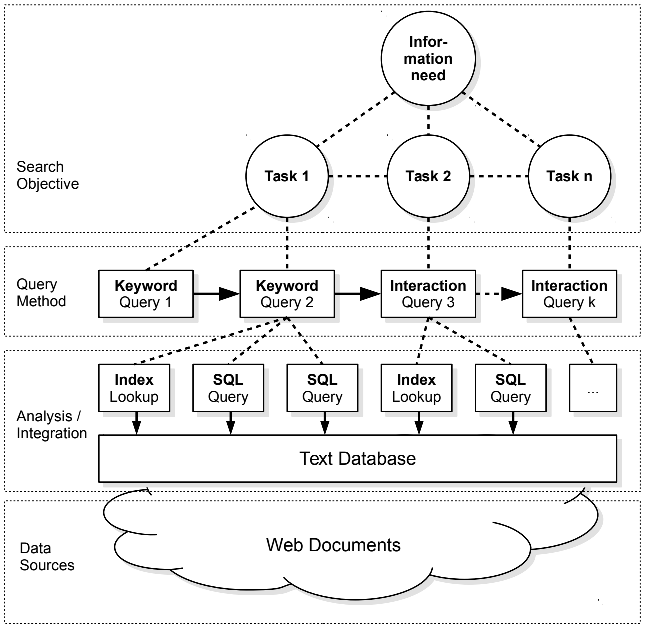


Figure 1: Hierarchy of search tasks (based on [12]). The user's search objective is divided into multiple hidden tasks $1 \ldots n$. The user expresses the tasks as unstructured queries. Each query is executed using one or more structured operators on a text database.

2.2 Process of Analytical Search

We define an analytical task as the specific request for a structured informational result. We assume a simplified model of search that is based on recent information behavior literature [12] and models an overall search *objective* as consecutive sequence of immediate search *tasks*. Each of

[1]http://www.goolap.info
[2]https://dbl43.beuth-hochschule.de/goolap/site/publications/

these tasks solves a specific goal that is required by the analyst to satisfy her information need. We can observe these tasks only by the *methods* of interaction, such as query formulation (see Figure 1). Next, we summarize the principal properties of search objective and query method that have individually been introduced in related work [3, 11, 24, 29].

Dimensions of search objective. The intentions of the analyst define a space with four dimensions. *Specificity* [11] depicts a degree of specialization of the task, such as specific, broad or undirected. While specific tasks have a clear scope and require an exact answer (e.g. `birthplace of barack obama`), undirected or exhaustive [24] tasks have a wider range of results which can be of various types (e.g. `travels of barack obama`). *Explorativeness* [3] describes the clarity of expectations that the user has on the result. For instance, the result may be known to exist or the user may be looking for suggestions. A task with no explorativeness has one distinct answer (e.g. `map of honolulu`); for a task with exploratory intent (e.g. `summer travel locations`), the user is first unsure about the amount of information that is required to satisfy her information need. *Complexity* [3] describes the level of complexity that is required to answer the task. For instance, lookup tasks (e.g. `location of honolulu`) are of low complexity and can be answered by classic retrieval algorithms. Learn tasks (e.g. `cities in hawaii`) require multiple iterations of processing, interpretation and aggregation. Investigatory tasks (e.g. `how to stop global warming`) require the analysis and synthesis of knowledge [29] and therefore are of high complexity. *Result size* describes the number of items that the user expects from the result. For example, some tasks expect a single answer (e.g. `declaration of independence date`), while other tasks may return a list of fixed size (e.g. `list of U.S. presidents`) or even a near endless result (e.g. `pictures of america`).

Dimensions of query method. Methods are the interactions of the user with the search engine. *Query formulation* often is the only explicit expression that we can observe from the interaction. Therefore, our approach focuses on the size and content of the query as one direct indicator for task classification. However, the complexity of scale is very low, while the variance of query formulation is extremely high [31]. Established measurements for syntactical query features are number of terms, number of tokens, distribution of query terms, detection of operator keywords (e.g. `pictures`) and part-of-speech information [22]. Many of these features can be summarized in an *n-gram language model*, which is able to predict words based on probability distribution of the previous $n-1$ words in a corpus of queries [8]. *User interaction* such as click-through information can be analyzed to reconstruct previous user sessions from the log file of the system. By observing the chronological order of refinement actions, we can extract context-based features such as the number of queries, number of reformulations and the identification of initial, expanded, contracted or repeated queries [12]. The observation of user client-side interactions with the result pages can lead to detailed implicit feedback of relevance. An analysis method of click distribution on search engine result pages has been proposed by [27]. The authors of [38] achieve good results using the median amount af clicks made during sessions associated with a single query. The authors of [12] have included more fine-grained human-device

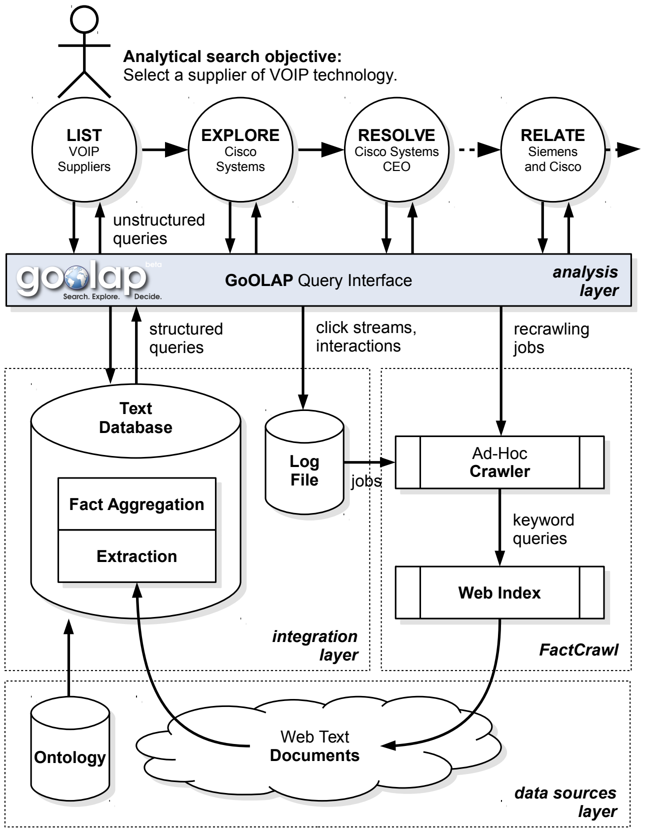


Figure 2: Architecture of the GoOLAP search prototype. The user interacts with the Web front-end on analysis layer. The integration layer is part of the iterative ETL process and holds the data store. The crawler accesses the data sources layer using a Web index.

interface events such as number of mouse moves, scrolling, hovering, key-press events and time until first click. Additional measures based on cursor movement, such as cursor trail length and maximum hover time are proposed by [19].

2.3 GoOLAP Fact Retrieval

We have developed GoOLAP [28], which is a prototype that implements the process of analytical search on unstructured Web text data. The system considers the Web as large text database [1] and executes a focused crawl to retrieve potentially fact-rich English language pages. GoOLAP is based on a scheme of typed facts (n-ary relations) and typed entities, following the span model that was introduced in [26]. Currently the database holds over 5 million distinct named entities of 39 entity types and 29 million fact instances of 84 relation types[3]. From 2014 on, each day between 500 and 800 users query the knowledge base with analytical tasks in form of keyword queries and interactions.

The search process of GoOLAP is shown in Figure 2. An analyst queries the exploratory search interface using keywords and interactions. She expresses her objective as sequential search tasks. Each request is classified, accordingly

[3]we utilize the ontology defined by the OpenCalais Web Service http://www.opencalais.com/documentation/calais-web-service-api/api-metadata/entity-index-and-definitions

Analytical Task	Verbal Interpretation	Query Examples	Features SEQpos	Features SEQtype	speci-ficity	com-plexity	explora-tiveness	result size	Terminology

Explore	tell me all about X	`cisco systems`
		`VOIP`
			`NNP`
			`NNP`
				`Company`
				`Technology`
					undi-rected	learn	high	large	undirected [32]
									infor. gathering [25]

Resolve	tell me all about the specific Y called X	`paris hilton model`
		`cisco systems CEO`
			`NNP NNP`
			`NNP NNP`
				`Person Position`
				`Company Position`
					undi-rected	learn	medium	small	interpretation [29]

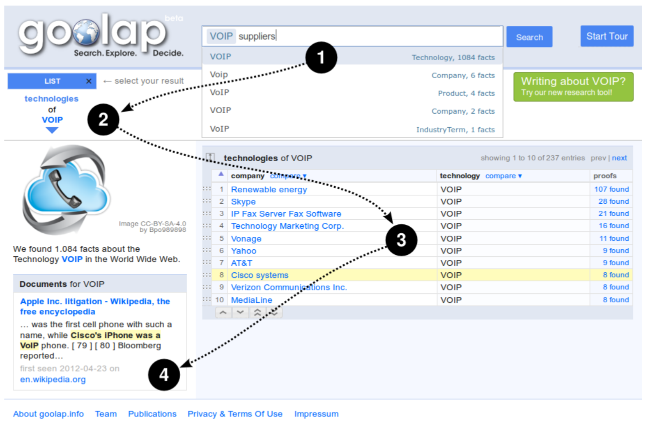
Relate	tell me all about the relation of X1 and X2	`cisco and siemens`
		`angela merkel USA`
			`NNP and NNP`
			`NNP NNP`
				`Company and Company`
				`Person Country`
					undi-rected	learn	medium	large	integration [29]

List	show me a list of X	`VOIP suppliers`
		`products of apple`
			`NNP NN+`
			`NN+ of NNP`
				`Technology CompanySupp+`
				`CompanyProduct+ of Company`
					directed	learn	low	large	directed open [32]

Compare	compare X1 and X2	`iOS vs android`
		`tcp udp comparison`
			`NNP vs NNP`
			`NNP NNP NN`
				`Product vs Product`
				`Technology Technology NN`
					directed	learn	medium	medium	comparison [29]

Answer	answer question X	`barack obama birthday`
		`foundation of siemens`
			`NNP NN`
			`NN of NNP`
				`Person PersonAttributes`
				`Formation of Company`
					directed	lookup	low	small	directed closed [32]
									fact finding [25]

Table 1: Overview of informational analytical tasks with query examples and categorization in the space of specificity, complexity, explorativeness and result size. The table shows simplified sequences of part-of-speech (SEQ_{pos}) and schema (SEQ_{type}) tags for each example and references to prior research terminology.



technologies of VOIP — showing 1 to 10 of 237 entries prev | next

	company compare ▾	technology compare ▾	proofs
1	Renewable energy	VOIP	107 found
2	Skype	VOIP	28 found
3	IP Fax Server Fax Software	VOIP	21 found
4	Technology Marketing Corp.	VOIP	16 found
5	Vonage	VOIP	11 found
6	Yahoo	VOIP	9 found
7	AT&T	VOIP	9 found
8	Cisco systems	VOIP	8 found
9	Verizon Communications Inc.	VOIP	8 found
10	MediaLine	VOIP	8 found

We found 1.084 facts about the Technology VOIP in the World Wide Web.

Documents for VOIP

Apple Inc. litigation - Wikipedia, the free encyclopedia
… was the first cell phone with such a name, while Cisco's iPhone was a VoIP phone. [79] [80] Bloomberg reported…
first seen 2012-04-23 on en.wikipedia.org

Figure 3: Interaction flow on the GoOLAP Web front-end: (1) implicit query annotation using auto-complete, (2) explicit correction of query classification, (3) exploratory result visualization, (4) the lineage of every fact is accessible in our text database.

translated into a structured query to the text database and visualized on the Web front-end. Inside the iterative extract-transform-load (ETL) process, we analyze the log file and trigger crawling jobs. The crawler queries a Web index to retrieve documents that contain unseen facts [6]. The system utilizes a natural language processing pipeline to extract relational facts from these documents and pre-aggregates the results for fast answering times. The data is represented in an interactive user interface (see Figure 3).

3. TASK AND QUERY ANALYSIS

Our study is based on a large query log that we gathered in the time span between 08/2008 and 10/2014 on the GoOLAP Web front-end. We logged query strings and result page interactions for 102,360 distinct queries with non-empty result. From this data set, we inspected a sample of 520 queries in a systematic process (see Section 5.1). In the expert-labeled data set we notice that 86.4% of the queries represent analytical tasks (see Table 2). The remainder of

Data Set	Q-EXP	Q-ML
total queries in data set	520	102,360
queries after filtering	477	85,430
explore	58.07%	10.69%
resolve	12.37%	39.64%
relate	6.08%	14.03%
list	4.82%	6.99%
compare	0.63%	0.21%
answer	4.40%	4.05%
non-analytical	13.63%	24.39%

Table 2: Overview of two data sets derived from 102,360 queries posed to GoOLAP. The supervised Q-EXP set is hand-labeled by experts, the Q-ML set was machine labeled using an initial classifier.

this set consists of navigational and transactional tasks, requests for unstructured and investigatory results as well as apparently senseless requests. The result of our inspection is the base for the following classification.

We introduce six common classes of analytical search tasks. The fundamental categorization of *directed* and *undirected* tasks is based on the observations of [32]. We extend this classification by structured analysis of the query log according to the previously introduced space of specificity, task complexity, explorativeness and result size. Our subsequent goal is to utilize observable properties of query formulation and user interaction methods to predict the task intention at query time. Table 1 gives an overview on the categorization and the corresponding concepts in related work.

We notice that the user's query objective is not always clearly classifiable into one of our proposed classes. It was also shown that search intent of a single user can vary at different times [12]. However, for the generation of practical search results, we are interested in providing a single classification according to the most expected representation of user intent. We therefore follow the approach of [16] and apply single labels.

3.1 Classes of Analytical Tasks

During our query log analysis, we developed the following six classes of common analytical tasks. In this paper we focus on structured informational tasks only (e.g. tasks specific to text databases and OLAP) and classify non-analytical search tasks (e.g. with navigational, transactional or unstructured intentions) as OTHER. This choice also ensures that non-analytical and senseless tasks can be handled correctly by the implementation.

Explore. With 58.1% of all queries, this task is the most common in the query log. The task targets exhaustive information gathering (e.g. "Tell me all about X") and is widely known by related work as *undirected* search [7] or *information gathering* [25]. The parameter X is a broad undirected search term, mostly a single known item or entity (e.g. `cisco systems`). This term is used as a starting point for further exploratory investigation by the user. The user expects the most relevant tuples of a universal relation (e.g. an overview of semi-structured facts), which is built from a large set of relationship types.

Resolve. We discover that a large fraction of 12.4% queries in the log target undirected search, but include disambiguation terms in the query: "Tell me all about the specific Y called X". X is an optional parameter to specify the name of the entity, as in EXPLORE. Y is either a broad term (e.g. a profession in `paris hilton model`) or a relation that needs to be resolved (e.g. `cisco systems CEO` $\rightarrow$ ⟨?,CEO of,cisco⟩). RESOLVE queries are often used to combine the ANSWER to a "which" question with an EXPLORE task in one single task. This task is mostly unrecognized in research, but it is very frequent in our data set and has been mentioned before as *interpretation* [29]. It is supported by Google Knowledge Graph as *association queries* [9].

Relate. 6.1% of the queries focus on the connection between two or more items: "Tell me all about the relation of X_1 and X_2". The parameters X_n are named entities (e.g. `cisco systems` and `siemens`). The user wants to investigate the (unknown) link between these entities. The result may be a semi-structured, explorable set of data (e.g. facts that mention all entities) or a ranked list of information (e.g. the most characteristic facts of the relation in question). This task is supported for structured queries by NAGA as *relatedness queries* [23]. Our work is the first to interpret and resolve RELATE tasks from unstructured keyword queries.

List. We notice 4.8% of queries that target the widely known *directed open* search [32] task: "Show me a list of X". Here, X is an entity type (e.g. `products`) or relation with one unbound component (e.g. `VOIP suppliers`). The user aims to obtain an open-ended list of possible answers, so the result is always a structured list of items. This task is supported by NAGA as *discovery queries* [23] and by Google Knowledge Graph as *set queries* [9].

Compare. With only 0.6% of all queries, the directed task "Compare X_1 and X_2" is rare, but still one of the most common analytical tasks. In contrast to RELATE, the parameters X_n are single entities of the same type (e.g. products `iOS` and `android`). Therefore, the user does not expect an intersection of the entities, but wants to obtain a feature-wise discrimination between them. Therefore, the result is best visualized as a table with the entities as columns and their attributes as rows. While this specific intention has been mentioned as *comparison* [29], COMPARE queries are often supported by special vertical search engines only.

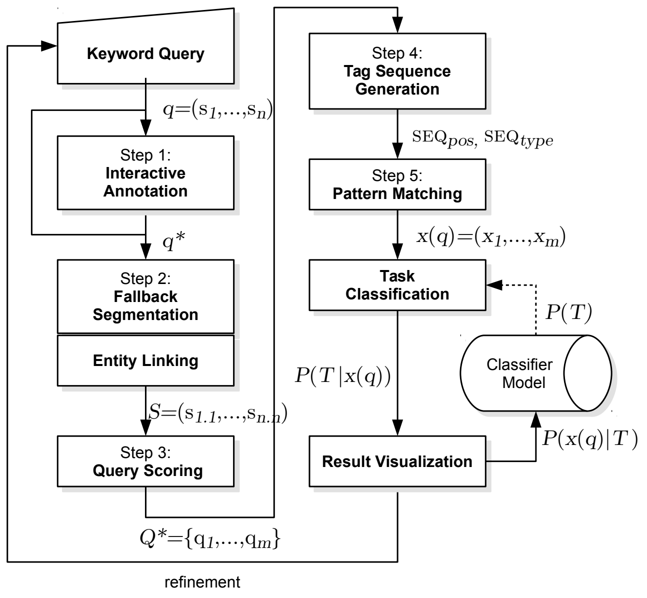


Figure 4: Technical process of intent-aware search result generation. A keyword query is interactively annotated by the user (step 1) or automatically segmented and linked (step 2). A scoring function is applied to identify the best interpretations (step 3). Then, features are extracted by tag sequence generation (step 4) and pattern matching (step 5). Finally, the query is classified and visualized. The user sends implicit and explicit feedback to the system.

Answer. We observe 4.4% of question answering tasks that target a single answer X of type *who*, *what*, *where* and *when*. X can be of various complexity from simple lookup [29] (e.g. `age of barack obama`) to aggregate functions (e.g. `average age in new york`). The result is mostly a single answer (e.g. fact, quotation, coordinates, numeric or boolean value) and may contain meta-information (e.g. source of the answer, rating of trustworthiness or possible alternative answers). This task is also referred to as *fact finding* [25] or *directed closed* search [32]. It is supported by all structured search engines such as NAGA [23] or SQAK [34] and Google Answers from Tables [9].

4. RESOLVING TASKS FROM KEYWORDS

In this section, we introduce a process to resolve the six operationalized task intentions at query time. In order to classify the task from query formulation, we have to extract features from the keywords. This process is shown in Figure 4: while typing, the analyst is offered an auto-complete field to select and disambiguate named entities. In case of no explicit user annotation, we guess the most likely segmentation of the query and detect the named entities that were not annotated using a scoring algorithm. We use these segments to build sequences of syntactical and semantic tags. These sequences are then transformed into a feature vector of low dimensionality. We then utilize this vector to build a probabilistic classifier model.

4.1 Query Segmentation and Annotation

Table 1 shows some representative query examples. We observe that 85% of all queries are 6 words or shorter, and longer queries are long tail distributed (rare). Most queries include one or several multi-word representations of entities or relation names while very few queries contain proper English language sentences. For resolving tasks we must segment the query into representations of entity names, attribute names, relation names and part-of-speech (POS) tags. According to authors of [5], segmenting queries into semantical meaningful multi-word units is a NP-hard problem. We approximate a solution by utilizing user interaction and a simple scoring model based on fact distribution. An example of this process is given in Table 3.

Step 1: Interactive entity segmentation. For segmenting query tokens into multi-word segments we provide the user with an interactive search field. While typing, the system matches the user input against our index of named entities (see Section 2.3) and proposes disambiguations. For each entity, the index contains a canonical name, alternative writings and uses fuzzy matching to tackle typing errors. The user can select the right entities from a ranked auto-complete list. We obtain a sequence of n user-annotated segments $q^* = (s_1, \ldots, s_n)$ that include schema information. Moreover, the annotation provided by the user has possibly fewer segments than the literal query q.

Step 2: Fall-back entity segmentation. The user might not resolve some segments. Therefore we present an algorithm to find potential segmentations. We cleanse and tokenize the query into atomic single-word segments and process them with a POS tagger and lemmatizer[4]. To limit runtime complexity we restrict the maximum query length to 100 characters and 12 words. Next, we merge all segments in q with user annotations q^*. We build n-grams of adjacent unmatched segments with a maximum length of 4. We insert these candidate segments $\hat{s}$ together with all literal and annotated segments s into a prefix tree, indexed by $begin.end$: $q \mapsto S = \{s_{1.1}, \hat{s}_{1.2}, \hat{s}_{2.3}, \ldots, s_{n.n}\}$. This allows us to efficiently access all possible sequences of segments. All segment candidates $\hat{s}$ are then matched to our schema index. We use a top-1 strategy to keep only the highest ranked matches and leave the disambiguation of rare entities to the user. We adapt the POS tag of compound segments to represent a single proper noun (`NNP`). In case of plurals, we use a heuristic to decide if the resulting segment will be annotated as plural.

Step 3: Scoring and pruning segmented queries. We prune all unmatched segment candidates from the tree. Similar to [34], we apply scoring to all remaining segments in S based on fact distribution in the text database. $f(s)$ denotes a score calculated by the number of facts for the given match. We boost candidates using $0 < w(s) \leq 2$ as weight depending on the strength of the match (e.g. fuzzy match or explicit annotation). $l(s)$ denotes the length of a segment in words. We prefer longer matches and calculate the score of a single segment as follows:

$$scoreS(s) = w(s) \cdot f(s) \cdot l(s)^2 \tag{1}$$

Next, our system builds a set of query interpretations Q that consists of all possible ordered non-overlapping segmentations: $S \mapsto Q = \{\hat{q}_1, \ldots, \hat{q}_m\}$. Then we calculate the score of a single query as mean of all token scores. For scoring, we omit function words F (determiners, conjunctions, comparators, symbols) by transforming $\hat{q}$ into the subsequence $r(q) = ((s_1, \ldots, s_n) | s_i \in q \wedge s_i \notin F)$. We return a list of highest-ranked query interpretations $Q^* \subseteq Q$.

$$scoreQ(q) = \frac{1}{|r(q)|} \sum_{s \in r(q)} scoreS(s) \tag{2}$$

Step 4: Generalizing tag sequences. For feature extraction, we must reduce the dimensionality of the highest scored queries. Our basic idea is to transform all segmented queries $\hat{q} \in Q^*$ into two types of tag sequences SEQ_{pos} and SEQ_{type}. We use $pos(s)$ to access a generalized POS tag of a segment by transforming the tag using simple replacement rules: replace conjunctions, prepositions and interrogatives by their literals (e.g. `and`, `of`, `who`), replace all verb forms by the base form `VB`, use `+` as plural modifier instead of `S`. We use $type(s)$ to get the schema type of the matched segment or the POS tag in case of an unmatched token. We skip few stop words W (determiners and symbols) in both sequences.

$$\text{SEQ}_{pos}(q) = \big((pos(s_1), \ldots, pos(s_n)) | s_i \in q \wedge s_i \notin W\big) \tag{3}$$

$$\text{SEQ}_{type}(q) = \big((type(s_1), \ldots, type(s_n)) | s_i \in q \wedge s_i \notin W\big) \tag{4}$$

Step 5: Extract common patterns for each task. We then use the sequences SEQ_{pos} and SEQ_{type} to extract tag-based features. A list of the most descriptive features is given in Table 4. We use the number of segments as the most simple feature. Additionally, we count the frequency of 28 POS tags and 22 literals in SEQ_{pos}, e.g. `NNP` (proper nouns), `NN+` (plural nouns), `of` (preposition), `and` (conjunction). Then, we count the frequency of 30 matched entity

step		s1	s2	s3
(1)	q* =	`cisco systems`:company		`CEO`
(2)	S =	`cisco`/NNP	`systems`/NNS	`CEO`/NNP:position
		`cisco systems`/NNP:company		`CEO`/NNP:position
		`cisco`/NNP	`systems CEO`/NNP	
		`cisco systems CEO`/NNP		
(3)	Q* =	`cisco systems`/NNP:company		`CEO`/NNP:position
(4)	SEQpos	`NNP`		`NNP`
	SEQtype	`Company`		`Position`

Table 3: Example of the query transformation process for q =`cisco systems CEO`. The table shows user annotated segments q^* (step 1), possible segmentation candidates S (step 2), the highest-scored query interpretation Q^* (step 3) and generalized tag sequences SEQ_{pos} and SEQ_{type} (step 4). Matched segments (entity and relation types) are highlighted.

[4]We utilize the pipeline of www.clearnlp.com

tags in SEQpos (count features)		count	ratio	corr
NNP	POS proper noun (singular)	**79,025**	**0.37**	**0.44**
#	number of segments	**85,421**	0.25	**0.34**
NN	POS common noun (singular)	**31,659**	0.19	0.17
NN+	POS common noun (plural)	5,346	**0.44**	0.14
NNP+	POS proper noun (plural)	1,269	**0.35**	0.08
of	literal preposition	2,681	0.14	0.08
tags in SEQtype (count features)		count	ratio	corr
Person	entity type (singular)	**65,821**	0.14	**0.27**
City	entity type (singular)	8,901	0.20	**0.23**
Position	entity type (singular)	8,199	0.20	**0.22**
Company	entity type (singular)	**18,857**	0.06	0.11
PersonAttributes	relation type (all forms)	736	**0.35**	0.05
Position+	entity type (plural)	487	0.34	0.05
patterns in SEQtype (match features)		count	ratio	corr
"Product Product"~2	proximity pattern	58	**0.47**	0.02
when	match pattern	47	0.13	0.01
who	match pattern	119	0.04	0.01

Table 4: Tag-based features derived from tag sequences SEQ_{pos} and SEQ_{type} from 85,430 queries. *Count* shows the number of observations with value > 0, *ratio* denotes the gain ratio [14] of each feature, *corr* shows Pearson correlation [14] with the class label assigned to each query. Features are sorted by correlation, top 5 scores are highlighted.

types (e.g. Person, Company) and 84 relation types from the ontology (e.g. PersonCareer, CompanyCompetitor) in SEQ$_{type}$. Finally, we utilize pattern matching of 15 expert-defined patterns in SEQ$_{type}$. These values are concatenated as feature vector of length 185, which is then used for training and classification.

4.2 Predicting Complex Search Tasks

We implement different classifier configurations to predict the search task T from query formulation q. We restrict our study to tasks of type EXPLORE, RESOLVE, RELATE, LIST, COMPARE, ANSWER, and OTHER. Accordingly, our classifier produces an estimate task intention $\hat{T}$ from $K = 7$ classes $\{T_1, \ldots, T_K\}$. We execute two stages of classifier training to build a supervised and a semi-supervised classifier model. The training data sets Q-EXP (expert-labeled) and Q-ML (machine-labeled) are taken from the query log that is described in Section 5.1.

Baseline: Language models. Class prediction must happen at query time. Search engines utilize for this task fast language models (LM) which are our base line. We estimate a LM for each query in our labeled data set and keep all models in an index[5]. Because our labeled data set is very sparse, we use the SEQ$_{pos}$ sequence introduced in Section 4.1 to generate the language models. For the classification of an unseen query, we query the index and pick the result with the maximum likelihood according to the LM. We return the labeled task of the resulting query as classifier output. Since we expect multi-word queries we compare effects of *Jelinek-Mercer* and *Dirichlet* prior smoothing methods [39].

[5]We use the Lucene 4.7.2 implementation.

Supervised classification. We solve this multi-class classification problem by decomposing it into K probabilistic models and decide for the maximum a posteriori class. We derive the feature vector $x(q) = (x_1, \ldots, x_m)$ from query processing and use the Naive Bayes classifier model. This simple approach of classification performs well and can be implemented very efficiently. We discuss further techniques in future work.

$$P(T|x(q)) = \frac{P(T)P(x_1, \ldots, x_m|T)}{P(x_1, \ldots, x_m)} \qquad (5)$$

Assuming class conditional independence between features $(x_1, \ldots, x_m)$, we can decide for the class that maximizes the estimate using the 1-against-all approach:

$$\hat{T} = \arg\max_{k=1,\ldots,K} P(T_k) \prod_{i=1}^{m} P(x_i|T_k) \qquad (6)$$

We train the initial classifier C-SUP using expert-defined class labels in data set Q-EXP, which contains analytical, navigational and transactional queries. We apply class label OTHER to all tasks that are not in our target space. Using this approach, we obtain a classifier that is more robust towards unseen queries of non-analytical tasks.

Semi-supervised classification for rare queries. We may observe a deficit in classifying rare queries taken from the long tail of our data set (e.g. queries with intended task COMPARE). The main reason for this is the small sample size and therefore low variance of features. We therefore execute a second stage of semi-supervised classifier training. We use a significantly larger data set Q-ML and use the initial classifier to apply class labels to the whole data set. We use these examples of analytical and OTHER tasks to train a second classifier C-SEMI.

In the next section, we evaluate the different approaches of task classification and discuss the effects.

5. EVALUATION

We evaluate the prediction of search task from free-text keyword queries by comparing the classification approaches introduced in Section 4.2: A trivial baseline (always decide EXPLORE), language models (LM), supervised Naive Bayes (C-SUP) and semi-supervised Naive Bayes (C-SEMI).

We start with a description of our data set and define the measurements we use for our experiments. Then, we explain the overall evaluation of different classification approaches and compare the prediction performance of single classes. We investigate the most important features and discuss the runtime performance of our implementation. Finally, we show limitations and common errors of the approach.

5.1 Evaluation Setup

Data set. For the classifier evaluation, we process 102,360 queries from the GoOLAP query log (see Section 2.3). We select a random sample of 520 queries with non-zero result from non-bouncing sessions. The sample was manually segmented and labeled by two experts (strong agreement using Kappa threshold > 0.8) inside the interactive Web interface shown in Figure 3. The queries were further processed by the pipeline described in Section 4.1. After filtering the set Q-EXP consists of 477 distinct expert-labeled samples. The

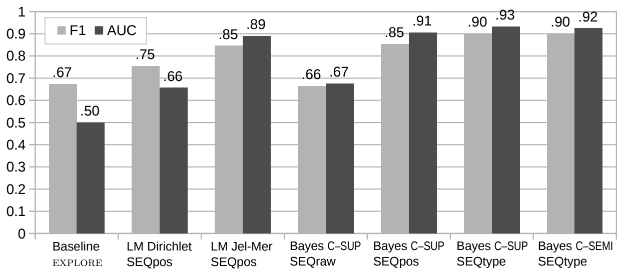


Figure 5: Evaluation of seven classifier configurations with combined F1 and AUC score.

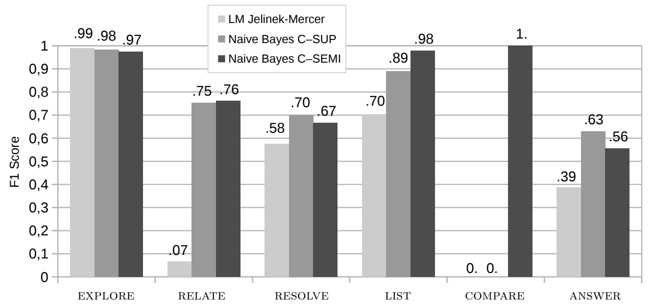


Figure 6: Evaluation of F1 score per class for Language Model, C-SUP and C-SEMI Bayes classifiers.

larger set Q-ML contains 85,430 queries which were constructed by applying machine labeling on the full sample (see Table 2). Q-ML and Q-EXP are disjoint.

Measurements. We use 10-fold cross validation on the analytical examples in Q-EXP to evaluate the classifiers. We measure $Precision_i$, $Recall_i$, $Accuracy_i$, AUC_i and $F1_i$ score for each binary classifier $C_{i=1...K}$. For the composite multi-class classifier, we measure micro-averaged scores for $Precision_\mu$, $Recall_\mu$, $Accuracy_\mu$, AUC_μ and $F1_\mu$ score. We define these measures according to [33]. To calculate Area Under ROC Curve (AUC), we use simple trapezoid calculation because our 1-against-all class decisions are discrete and therefore produce only a single point in ROC space [10]. The overall AUC_μ is accordingly calculated as weighted average over the per-class AUC values.

5.2 Evaluation Results

Overall classifier evaluation. Figure 5 displays combined $F1_\mu$ and AUC_μ values for seven different classifier configurations (n=477). First, we notice that classification of raw POS tag sequences cannot improve the trivial baseline (baseline 67.2% F1, Supervised Naive Bayes 66.3% F1). Therefore, our query transformation process is crucial to achieve better results from the long tail of analytical queries. Both language models incorporate $\textsc{seq}_{pos}$ transformation and reveal a better overall score of 75.5% F1 (Dirichlet) and 84.5% F1 (Jelinek-Mercer). The Naive Bayes classifier C-SUP with the same feature set performs slightly better than the Jelinek-Mercer language model with 85.4% F1 and 90.5% AUC. By including features that base on $\textsc{seq}_{type}$ transformation, we can improve C-SUP to 90.1% F1. Finally, we observe that the semi-supervised Naive Bayes classifier C-SEMI performs equally with the supervised approach, reaching 96.8% micro-averaged accuracy. We now discuss the differences between the best three configurations.

Comparison of classes. We compare classification performance for the single task classes. Figure 6 shows the quantitative differences between the three models. While the results of binary classifiers for tasks EXPLORE, RESOLVE and ANSWER are similar, the supervised Naive Bayes approach performs significantly better for the tasks RELATE (75.4% F1) and LIST (88.9% F1). We observe that the rare task COMPARE (only 3 occurences in Q-EXP) cannot be identified at all by the language models and supervised methods (0.0% F1). However, using the much larger semi-supervised sample as training data, all three examples can be correctly classified without false positives (100.0% F1).

Discriminative features. The inspection of features with highest class label correlation reveals a better view into the long tail of analytical queries (see Table 4). As expected, the task EXPLORE shows high correlation with the binary features `#(segments)`≡ 1 and `#(NNP)`≡ 1 as well as the occurrence of single entity types such as `Person`, `City` or `Position`. Queries labeled as RESOLVE show similar measures, but can be distinguished by the occurrence of `City`, `Position` and `Organization` tags. RELATE shows a high evidence of `NN` tags and can be well detected using the pattern `"Person and Person"`. We observe a very different set of features for the task LIST: the most important patterns are plural forms `NN+`, `NNP+`, `Position+` and `Product+` as well as the occurrence of numbers (`CD`). Queries labeled as COMPARE proved to be hard to detect and are best matched by conjunction tags and proximity patterns such as `"Product Product"~2`. ANSWER tasks are highly correlated with the evidence of relation types such as `PersonAttributes`, `PersonRelation` or `Indictment`, as well as the preposition `of` and adjectives `JJ`. Table 5 shows discriminative patterns for three classes of common analytical tasks.

Runtime measurements. All classification models we use for our implementation (e.g. Naive Bayes) are very efficient. That means, the execution time of the classification process depends mainly on the number of index lookups during fallback segmentation. Our commodity test system (Intel Core i7) classifies the median single word query in 10ms, unannotated queries of length 12 stay under 350ms (worst-case). As this is still maintainable and the majority of queries is much shorter, our system is ready for productive use.

5.3 Discussion

To conclude the evaluation, we discuss the limitations of our approach and point out typical classification errors.

Limitations. The evaluation results show the limitations of our approach. (1) Our dataset originates from many queries posed to search engines that refer to GoOLAP result pages. First, this implies that the variance of direct queries to GoOLAP might be not as high due to the auto-complete guidance. Second, we are able to match a large fraction of named entities because the sample is a self-selection of successful queries to our own text database. Our approach is not yet tested on external data sets. (2) The raw output of

intent	SEQ$_{type}$	SEQ$_{pos}$	freq
RESOLVE	`Person Position`	`NNP NPP`	3.3%
	`Person City`	`NNP NPP`	2.6%
	`Position Person`	`NNP NPP`	2.4%
	`Person Person Person`	`NNP NPP NNP`	2.4%
	`Person Person NN`	`NNP NPP NN`	2.4%
RELATE	`Person Person`	`NNP NPP`	36.9%
	`Person Company`	`NNP NPP`	19.2%
	`Person Organization`	`NNP NPP`	5.4%
	`Company Person`	`NNP NPP`	4.6%
	`Person and Person`	`NNP and NPP`	4.0%
COMPARE	`VB % to %`	`VB NNP to NPP`	3.4%
	`VB % and %`	`VB NNP and NPP`	2.8%
	`Product Product`	`NNP NPP`	2.2%
	`Company and Company`	`NNP and NPP`	1.7%
	`Company Product Product`	`NNP NPP NNP`	1.7%

Table 5: Most discriminative sequences for our novel intent classes RESOLVE, RELATE and COMPARE. Frequency denotes the occurrence of the sequence in all queries classified to the given intent.

the POS tagger is error-prone due to the lack of syntactical information in keyword queries. We manage to correct a large number of tokens by our matching strategies, but there are drawbacks, which are now discussed.

Inspection of wrong classifications. We systematically inspected misclassified examples, Table 6 shows some typical errors. Example 1 shows an error where a named entity could not be resolved, yielding a wrong POS tag and therefore wrong classification. Example 2 is a navigational task that is classified as RELATE because our ontology is not aware of the semantic type *Website* and therefore detects `IMDB` as *Company*. Example 3 shows a token mismatch for the verb `compare`, which is resolved as entity from our long-tail ontology of entities. Example 4 shows a misclassified transactional task, because our system is not aware that `picture` leads to a resource. Example 5 shows the misclassification of a query for advice as RESOLVE which is caused by incorrect training of the semi-supervised classifier: `NN VB` is also a frequent POS tag error for `NNP` (see Example 1).

Possible improvements to ontology matching. We observe a low precision in the detection of natural terms that identify real-world concepts such as `map`, `picture`, `biography` or `court`. Additionally, a high amount of queries include keywords that implicitly serve as operators, such as `compare`, `top`, `download` or `list`. Our current model considers only prepositions (e.g. `of`, `by`) and conjunctions (e.g. `and`, `or`) as possible operator keywords. It was also shown that users frequently use keywords such as `wikipedia` for addressing certain query types [24]. We therefore need to extend our ontology to a broader scheme and include a larger set of possible operators into feature extraction, scoring and classification models. Possible solutions are the integration of noun synset dictionaries such as WordNet[6] or collaborative knowledge bases such as Wikidata [35]. The retrieval model proposed in Biperpedia [13] addresses these issues by extending entities from Freebase with attribute classes from the long tail of queries and integrates synonyms and common misspellings into the data set.

[6]http://wordnet.princeton.edu

#	query	SEQ$_{pos}$	labeled	classified
1	`brigido borges`	`NN VB`	EXPLORE	OTHER
2	`fiona loudon imdb`	`NNP NNP`	OTHER	RELATE
3	`compare ben bella to mao`	`NNP NNP to NNP`	COMPARE	RESOLVE
4	`pictures of paris hilton`	`NN+ of NNP`	OTHER	LIST
5	`password reset facebook`	`NN VB NNP`	OTHER	RELATE

Table 6: Representative examples of wrong classifications. *SEQ_{pos}* is the sequence of POS tags generated by our processing pipeline.

6. CONCLUSION AND FUTURE WORK

We present an approach to resolve analytical tasks in text databases that leverages knowledge about the user's search expectations to generate the results. Our work goes beyond the traditional translation of keyword queries into structured query languages by inferring a normally hidden model of search objective from the query methods of the user. We refer to a large data set of 102,360 keyword queries and user interaction to identify and operationalize six of the most common analytical tasks: EXPLORE, RELATE, RESOLVE, LIST, COMPARE and ANSWER. We publish this data set to the research community. We are able to classify six common tasks from user queries at query time with very high accuracy of 96.8% and an F1 measure of 90.1%. We present the result in form of a demonstrator that utilizes interactive entity recognition, disambiguation and query segmentation methods.

In future work we will address two main issues in the query segmentation process. First, the recognition of candidate annotations (e.g. named entities and relation predicates) needs to be more adaptive towards short queries and specialized domains. Our approach to fix incorrect POS segmentation might also be improved by replacing POS tag features with pure character-based features. A syntax-based extractor would generate features from the token stream while the user is typing the query, e.g. utilizing tri-grams as proposed by [15]. We can use these features in deep neural networks to predict annotations on the fly. Second, annotation disambiguation can be improved using semantic word hashing and interaction mining techniques [20]. We aim to extend the feature set with word embeddings to incorporate latent relational dependence between annotations [18, 37]. In the long term, these improvements might allow prediction of high level search objectives from the hidden iterative states of our search process model. Moreover, we aim to incorporate the client-side interaction stream [12, 19] into a subsequent study and evaluate the system using a representative dataset with higher variance.

Acknowledgments

The work is funded by the Federal Ministry of Economic Affairs and Energy (BMWi) under grant agreement 01MD15010B (Project: Smart Data Web).

7. REFERENCES

[1] E. Agichtein and L. Gravano. Querying Text Databases for Efficient Information Extraction. In *ICDE'03*, pages 113–124. IEEE, 2003.

[2] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *ICDE'02*, pages 5–16, San Jose, CA, USA, 2002. IEEE.

[3] A. Aula and D. M. Russell. Complex and Exploratory Web Search. In *ISSS'08*, Chapel Hill, NC, USA, 2008.

[4] R. Baeza-Yates, L. Calderón Benavides, and C. González Caro. The Intention Behind Web Queries. In *String Processing and Information Retrieval*, pages 98–109. Springer, 2006.

[5] S. Bergamaschi, F. Guerra, M. Interlandi, R. Trillo-Lado, and Y. Velegrakis. Quest: A Keyword Search System for Relational Data Based on Semantic and Machine Learning Techniques. *PVLDB*, 6(12):1222–1225, 2013.

[6] C. Boden, A. Löser, C. Nagel, and S. Pieper. FactCrawl: a fact retrieval framework for Full-Text indices. In *WebDB at ACM SIGMOD*, 2011.

[7] A. Broder. A Taxonomy of Web Search. In *ACM SIGIR Forum*, volume 36, pages 3–10. ACM, 2002.

[8] W. B. Croft. Processing Text: Document Parsing. In *Search Engines: Information Retrieval in Practice*, pages 86–101. Addison-Wesley, Boston, 2010.

[9] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *KDD'14*, pages 601–610. ACM Press, 2014.

[10] T. Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[11] C. González-Caro and R. Baeza-Yates. A Multi-Faceted Approach to Query Intent Classification. In *String Processing and Information Retrieval*, pages 368–379. Springer, 2011.

[12] Q. Guo and E. Agichtein. Ready to Buy or Just Browsing?: Detecting Web Searcher Goals from Interaction Data. In *SIGIR'10*, pages 130–137. ACM, 2010.

[13] R. Gupta, A. Halevy, X. Wang, S. Whang, and F. Wu. Biperpedia: An Ontology for Search Applications. *VLDB'14*, 7(7):505–516, 2014.

[14] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 2006.

[15] L. Heck and H. Huang. Deep Learning of Knowledge Graph Embeddings for Semantic Parsing of Twitter Dialogs. In *GlobalSIP'14*, pages 597–601. IEEE, 2014.

[16] M. R. Herrera, E. S. de Moura, M. Cristo, T. P. Silva, and A. S. da Silva. Exploring Features for the Automatic Identification of User Goals in Web Search. *Information Processing & Management*, 46(2):131–142, 2010.

[17] J. Hoffart, D. Milchevski, and G. Weikum. STICS: Searching with Strings, Things, and Cats. In *SIGIR'14*, pages 1247–1248. ACM Press, 2014.

[18] H. Huang, L. Heck, and H. Ji. Leveraging Deep Neural Networks and Knowledge Graphs for Entity Disambiguation. *arXiv*, abs/1504.07678, 2015.

[19] J. Huang, R. W. White, and S. Dumais. No Clicks, No Problem: Using Cursor Movements to Understand and Improve Search. In *ACM SIGCHI'11*, pages 1225–1234. ACM, 2011.

[20] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *CIKM'13*, pages 2333–2338. ACM, 2013.

[21] A. Jain, A. Doan, and L. Gravano. Optimizing SQL Queries over Text Databases. In *ICDE'08*, pages 636–645. IEEE Computer Society, 2008.

[22] I.-H. Kang and G. Kim. Query Type Classification for Web Document Retrieval. In *SIGIR'03*, pages 64–71. ACM, 2003.

[23] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The YAGO-NAGA approach to knowledge discovery. *SIGMOD'09*, 37(4):41–47, 2009.

[24] M. P. Kato, T. Yamamoto, H. Ohshima, and K. Tanaka. Investigating Users' Query Formulations for Cognitive Search Intents. In *SIGIR'14*, pages 577–586. ACM Press, 2014.

[25] M. Kellar, C. Watters, and M. Shepherd. A Goal-based Classification of Web Information Tasks. *ASIS&T'06*, 43(1):1–22, 2006.

[26] T. Kilias, A. Löser, and P. Andritsos. INDREX: In-Database Relation Extraction. *Information Systems*, 53:124–144, 2015.

[27] U. Lee, Z. Liu, and J. Cho. Automatic Identification of User Goals in Web Search. In *WWW'05*, pages 391–400. ACM, 2005.

[28] A. Löser, S. Arnold, and T. Fiehn. The GoOLAP Fact Retrieval Framework. In *Business Intelligence*, pages 84–97. Springer, 2012.

[29] G. Marchionini. Exploratory Search: From Finding to Understanding. *Communications of the ACM*, 49(4):41–46, 2006.

[30] D. Nettleton, L. Calderón Benavides, and R. Baeza-Yates. Analysis of Web Search Engine Query Session and Clicked Documents. In *Advances in Web Mining and Web Usage Analysis*, pages 207–226. Springer, 2007.

[31] G. Pass, A. Chowdhury, and C. Torgeson. A Picture of Search. In *InfoScale*, volume 152, page 1. ACM, 2006.

[32] D. E. Rose and D. Levinson. Understanding User Goals in Web Search. In *WWW'04*, pages 13–19. ACM, 2004.

[33] M. Sokolova and G. Lapalme. A Systematic Analysis of Performance Measures for Classification Tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[34] S. Tata and G. M. Lohman. SQAK: Doing More with Keywords. In *SIGMOD'08*, pages 889–902. ACM, 2008.

[35] D. Vrandečić and M. Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[36] Wolfram Alpha LLC. Technology of Wolfram Alpha. http://www.wolframalpha.com/faqs9.html, 2015. [accessed 2015-03-08].

[37] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv*, abs/1412.6575, 2015.

[38] J. Zamora, M. Mendoza, and H. Allende. Query Intent Detection Based on Query Log Mining. *Journal of Web Engineering*, 13(1-2):24–52, 2014.

[39] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001.