received much attention in the IR community. Many variations of the basic language model approach have been proposed and studied. Language models have been applied to various retrieval tasks such as cross-lingual retrieval [35], expert finding [2], XML retrieval [17]. In particular, recently Elbassuoni et al. [11] proposed a language-model-based ranking for structured queries over RDF graphs. Also, Nie et al. [22] used language models to retrieve and rank entities given a keyword query. However, our retrieval model is the first to adopt language models for statement search over documents.

An overview of semantic search for textual documents is given in [20], which surveyed 22 systems, none of which providing any document ranking; the authors actually point out the necessity of research in that direction. The Semantic Desktop [29] is a semantic search engine that combines fact retrieval with semantic document retrieval using a triple-based algorithm and graph traversal. Given a natural language query, the approach tries to infer a structured query and retrieve matching triples. If there is no perfect answer, the ontology that the queries are evaluated against is used to expand the user query, and the expanded query is evaluated on a document collection to retrieve matching documents. RankIE [7] is a system for entity-based document search that can exploit structural background knowledge such as ontologies. It includes the ExplainIE tool [4], which was developed to debug IE results in business intelligence applications, mainly by visualizing how entities and documents are related.

Semantic Web search engines try to find RDF content on the Web based on keywords or URIs. To rank the sources of relevant content, they often use variations of the the PageRank algorithm [9, 14] or the tf-idf measure [23]. Semantic Web search engines focus on locating RDF sources and do not consider provenance of RDF data. An important aspect in this context is how to present results to a user, the problem of snippet generation for facts has been studied a lot [1, 24].

3. BACKGROUND ON INFORMATION EXTRACTION

Information Extraction (IE) is the process of automatically extracting structured information such as entities, relationships between entities, and properties of entities from unstructured and semi-structured sources. The process of information extraction typically is a two-fold process. First, the system analyzes documents and extracts *statement candidates* from the text. Second, it integrates these candidates by removing contradictory statements or deriving new ones (through inference for instance).

One common approach to extract statements from text is to use a set of phrase-patterns to match the possible linguistic realizations of the abstract statements. This is usually done by combining natural language processing and statistical inference in order to identify qualifying patterns that are used to extract facts of interest.

Once statements, also referred to as facts, have been extracted, they are stored in a fact repository or a knowledge base. The most common format to store extracted facts is the W3C-endorsed RDF, which is the model adopted in the Semantic Web community. RDF knowledge bases consist of statements in the form of subject-property-object triples. For example, the information that Barack Obama was born in Hawaii can be represented formally as the RDF triple (`Barack_Obama`,`bornIn`,`Hawaii`) with subject `Barack_Obama`, property/relation `bornIn`, and object `Hawaii`. An RDF knowledge base can conceptually be regarded as a large graph with nodes corresponding to entities (subjects and objects) and edges denoting relationships or properties, which we refer to as an RDF graph. Search on this kind of ER/RDF data is naturally expressed by means of structured graph-pattern queries using query languages such as the W3C-endorsed query language SPARQL [31]. A result to a query expressed in such a way is a subgraph of the underlying knowledge graph that consists of a set of triples matching the query graph.

In this paper, we assume that our framework is working with named (canonic) relations originating from information extraction tools working with predefined canonic relations [21, 33], i.e., the system is configured to look for patterns for a set of given relations, e.g., `bornIn`. We can easily extend the framework to work with anonymous relations defined by clusters of similar patterns. For a cluster the confidence of each pattern can be defined by the distance to the cluster center. Some systems [3, 12] extract "interesting" patterns from a document corpus without naming them and try to cluster semantically similar patterns. Each obtained cluster represents an anonymous relation.

4. STATEMENT SEARCH

As explained in the introduction, *statement search* aims at retrieving a set of relevant documents to a set of given statements. This type of search can be motivated by two slightly different information needs. First, a user might want to verify whether a certain statement is true. For instance, a user might wonder whether Barack Obama was born in Kenya and seek documents confirming this statement. Second, a user might want to investigate a certain statement, i.e., learn more details about it. For example, the user might know that Osama Bin Laden died on May 2011 but wants to find more details about his death. In both cases, the user is interested in documents referencing the statement in some way. However, with respect to these two user goals, a document is relevant to a statement query if it either verifies the statement *(persuasiveness)* or provides further information about the statement topic *(on-topicness)*. These two main properties might compete with each other in determining the relevance of a given witness document. There might be documents that very clearly and convincingly support the statement (*persuasive*) but only casually mention the statement and are mainly concerned with another topic (not *on-topic*). For instance, the Wikipedia article about the Kapiolani Medical Center in Honolulu might clearly state that Obama was born there but then would provide no further information about Obama's childhood. A blog post discussing Obama's childhood on the other hand could provide more information about the president's youth but might be a less convincing source for the verification of his birthplace. The quality of a witness for a given query therefore depends on the user's preference on these two aspects. Our ranking model tries to incorporate different features estimating a document's *on-topicness* and its *persuasiveness* in a way that allows to adapt the ranking according to a user's preferences. Still, a perfect witness would satisfy both properties to the full, being informative on the topic and persuasive for the statement(s) in the query. Thus, at the core, our ranking model aims at finding the best documents based on the probability that they are a perfect witness for a given set of statements. In our model, the quality of a witness is therefore based on the following aspects:

1. Given a set of statements g, a witness can only be perfect if it covers all statements in g. This affects both properties, as a statement cannot be verified with a document that does not contain the statement, and it is also unlikely that additional information related to the statements can be found there.
2. A document is only informative to a user if she can learn more about the statement(s) of interest. Therefore, the

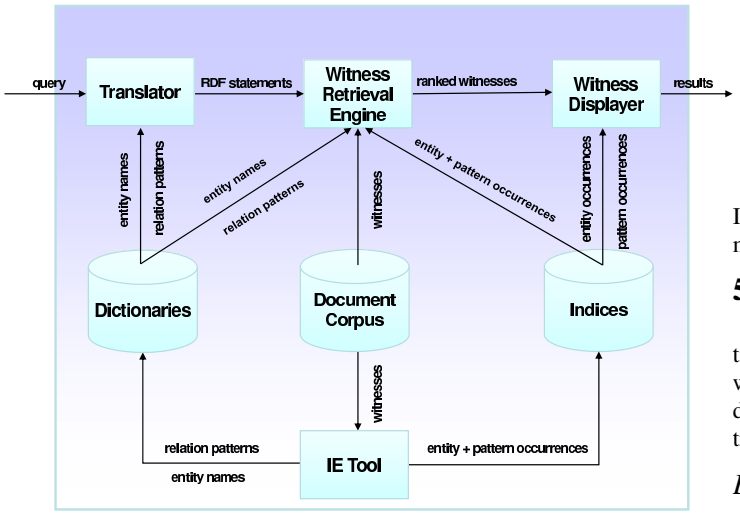


Figure 1: Architecture

document needs to focus on the topic given by the statement(s). This is the main aspect of on-topicness: that more information on the topic needs to be present.

3. Statements can be expressed in various textual forms. Some of these statement representations are more directly associated with the abstract statement than others. A witness can only be used to verify a statement if the statement is expressed clearly within the document's content. A perfect witness will formulate the statement(s) in a way that leaves no room for other interpretations. This is the main aspect of persuasiveness, if the expression of the statement is vague it will not convince a user.
4. In general, we trust information obtained from some documents more than information contained in others. For instance, if the White House website issues a birth certificate for President Obama stating that he was born in Hawaii while a conspiracy blog page states that he was instead born in Kenya, ignoring other factors, most people would rather believe the White House. Both, the statement(s) of interest expressed in the documents (persuasiveness) as well as further information *(on-topicness)* are subjects to trustworthiness.

Our ranking model, introduced in detail in Section 6, tries to take all these considerations into account. Before we dwell into the details of our ranking model and how it achieves the aforementioned goals, we describe our system framework in the next section.

5. SYSTEM OVERVIEW

As explained in the introduction, our system employs an information extraction tool to implement *statement search*. That is, it transforms a user query into RDF statements and retrieves the most relevant witness documents supporting them. Figure 1 gives an overview of our system. In a nutshell, our system works as follows:

1. The document corpus is processed by an information extraction (IE) tool. This process results in two types of information:
   - dictionaires that are used to look up entities based on entity names and textual patterns based on relations, and
   - a set of indices that can be used to find occurrences of a given entity or a statement inside the documents of the corpus.
2. Given a user query, our system utilizes a translation component that transforms the user query into a set of RDF statements referring to entities and relations contained in our dictionaries.
3. The system retrieves the documents that match these statements and ranks them based on how likely they are to be relevant witnesses to the given statements.

In the remainder of this section, we provide more details on the most important components of our framework.

5.1 Dictionaries

The framework uses two types of dictionaries: an entity dictionary and a relation dictionary. With the help of an IE system, we can use these dictionaries to identify statements expressed in documents and to understand user queries and translate them into triple-based statement queries.

Entity Dictionary

Unique entities usually have several non-canonical *names* (Definition 5.1). For instance, the unique entity `Barack_Obama` can be represented by the names *"Obama"*, *"Barack Hussein Obama"*, *"the US President"*, etc. The relationship between entities and their names is an m-to-n relation, i.e., every entity can have many names and every name can refer to different entities. The same entity name could even refer to different entities within the same document. For instance, in a biography about President Obama, the entity name *"Obama"* might refer to President `Barack_Obama` or to his father `Barack_Obama_Sr`. The problem of identifying the right entity for a given name is known as *entity disambiguation* [19] . However, at this point we assume:

1. Each entity name is uniquely addressing exactly one entity within a given document.
2. The disambiguation is solved externally and a mapping from entity names to the entity they refer to is given, e.g., as a side product obtained from information extraction.

DEFINITION 5.1 (ENTITY DISAMBIGUATION). *Given an entity name $\tilde{s}$ that occurs in a document d, the entity it refers to within d is given by: entity ($\tilde{s}$,d).*

The framework makes use of the entity dictionary in three respects: first, when translating user queries into triple form (Section 5.5), second, when matching statements back to their textual occurrences in the document (Section 5.1), and third, when creating the entity occurrence index (Section 5.3).

Relation Dictionary

Similar to the case of entities, a relation can be expressed using various *patterns*. A pattern p is defined as a recurring textual representation of an abstract relation r. For instance, the textual phrase *"was born in"* could correspond to the pattern "X was born in Y" indicating the `bornIn` relation on the subject entity X and the object entity Y. While we assume that the patterns we deal with represent *binary relations* between two arguments, we make no assumptions on the way a pattern is defined. A pattern could, for example, be represented by a textual phrase, an n-gram, or a regular expression. We refer to the occurrence $p(\tilde{s}, \tilde{o})$ of a pattern p with two entity names $\tilde{s}$ and $\tilde{o}$ as a *pattern instance*. If it occurs in document d, we write $p(\tilde{s}, \tilde{o}) \in d$. For example, given the pattern above, the phrase *"Obama was born in Kenya"* would be a pattern instance with X representing the entity name *"Obama"* and Y being instantiated with *"Kenya"*.

Any pattern can indicate multiple relations with varying certainty. For instance, the phrase *"Obama's home country Kenya"*

could refer to the fact that Barack Obama has family roots in Kenya (Barack_Obama,hasAncestorsFrom,Kenya) or it could indicate that he was born (bornIn) in Kenya. Thus, for any relation r that can be expressed by pattern p a confidence value $conf(p, r)$ (Definition 5.2) representing the likelihood that p expresses r is given.

DEFINITION 5.2 (PATTERN CONFIDENCE). *Assume two entity names $\tilde{s}$ and $\tilde{o}$, a pattern p, and a relation r are given. The confidence $conf(p, r)$ represents the probability that given that a pattern instance $p(\tilde{s}, \tilde{o})$ appears in a document d, the author wanted to express the relation $r(entity(\tilde{s}, d), entity(\tilde{o}, d))$. We say p indicates r ($indicates(p, r)$) if and only if $conf(p, r) > 0$.*

Given a query statement (RDF triple), indicative pattern instances can easily be identified by matching the entities and the triple relation to patterns that indicate it. Formally, if D is the document corpus, all indications of a triple statement (s,r,o) are given by $\{p(\tilde{s}, \tilde{o}) | indicates(p, r) \wedge \exists d \in D.entity(\tilde{s}, d) = s \wedge entity(\tilde{o}, d) = o \wedge p(\tilde{s}, \tilde{o}) \in d\}$.

For instance, given the statement triple t = (Barack_Obama, bornIn, Kenya), a document where *"Kenya"* is mapped to Kenya, *"Obama"* is mapped to Barack_Obama, and given that $indicates$("X was born in Y",bornIn) holds, then the pattern instance *"Obama was born in Kenya"* is an indicative pattern instance for t.

5.2 Witnesses

Based on the above introduced formalisms, we can now formally define the notion of a witness (document) for a (set of) statement(s).

Whenever a pattern instance $p(\tilde{s}, \tilde{o})$ of a pattern indicating a relation r occurs in a document d, d is called a witness for the statement $r(s, o)$ with $s = entity(\tilde{s}, d)$ and $o = entity(\tilde{o}, d)$. Thus, given a set of statements $g = \{t_1, ...t_n\}$, where $t_i = (s_i, r_i, o_i)$, then d is a witness for g if there is at least one t_i for that a pair $\tilde{s}$,$\tilde{o}$ exists such that:

1. s_i=entity ($\tilde{s}$,d),
2. o_i=entity ($\tilde{o}$,d),
3. $indicates(p, r_i)$, and
4. $p(\tilde{s}, \tilde{o})$ occurs in d.

Any witness for g is a candidate considered in our ranking algorithm described in Section 6. In order to efficiently identify documents in which pattern instances matching a given statement occur, we make use of a set of indices (Section 5.3).

5.3 Indices

We have two types of indices in our system. The first one deals with relation patterns and utilizes both the entity and relation dictionaries. For each document, it stores the number of times a certain pattern is instantiated with certain arguments in a document. For example, given the statement (Barack_Obama,bornIn, Kenya), and the text snippet *"Obama was born in Kenya"*, the index would store that pattern "X was born in Y" appears with *"Obama"* and *"Kenya"*. In fact, as we assume entity names are disambiguated, the index directly stores the disambiguated entities Barack_Obama and Kenya. Hence, to identify witness candidates given a statement triple, it suffices to match the pattern instances of all patterns indicating the relation against the arguments of the statement.

The second index deals with entities and stores for each document d the number of times an entity is mentioned in d. Since entities do not directly appear inside the documents but are referred to using different natural language names, this index utilizes the entity dictionary in order to look up the entity corresponding to an entity name (see Definition 5.1).

5.4 IE Tool

The information extraction tool is responsible for processing the documents in our corpus and providing the dictionaries. It is also responsible for generating the necessary indices that our retrieval engine operates on to retrieve and rank the witnesses. Generally speaking, we can use any information extraction tool, e.g., [21], if it provides us with some sort of binary patterns, preferably associated with confidence values, to extract entity occurrences. Since information extraction is not the focus of this paper, we omit the details of the information extraction tool and refer the user to related work [21]. The output of the IE tool that our system uses consists of the dictionaries and indices.

5.5 Query Translator

The query translator is responsible for processing a given user query and transforming it into a set of RDF statements that our system can process. In order to do so, it utilizes the dictionaries and the IE tool to perform the translation of the query into a set of RDF statements using the same vocabulary as our dictionaries and indices. Assume, for instance, a user wants to verify that Obama was born in Kenya, she might provide the phrase query *"Obama was really born in Kenya?"*. It would then be translated into the statement triple (Barack_Obama,bornIn,Kenya) and the system would output witnesses containing the statement.

Alternatively, queries might also be given directly as a set of RDF triples, such that our system could, for instance, be used as a verification tool for automatically extracted ontologies. Assume, for instance, a movie knowledge base is generated by an information extraction tool. Then, a user might consider it to be a mistake if she sees four different statements claiming that the role of the character Tony in the movie "The Imaginarium of Dr. Parnassus" was played by four different actors. Using our framework, she could issue the set of all four statements to see which of them is more likely to be true. This might lead her to a document containing expressions of all four statements, which explains that three actors replaced Heath Ledger as he died in the midst of the production.

5.6 Witness Retrieval Engine

The witness retrieval engine is responsible for retrieving *any document that matches the given query* (i.e. at least one statement of the query is indicated in the document) and rank this set of *witnesses* according to the ranking model described in Section 6. It utilizes the entity and relation dictionaries to identify the corresponding possible entity names and relation patterns for the given statements. It uses this information, along with the indices in order to rank the matching witnesses according to the ranking criteria described in Section 4.

Once witnesses are ranked, they are passed to the witness displayer which is responsible for displaying the ranked witnesses to the user. The witness displayer can utilize a variety of techniques to display the witnesses to the user in response to its original query. For instance, the identified statements corresponding to the user query can be highlighted in the returned witnesses, as well as any other automatically extracted information that our IE tool managed to extract from the witnesses. However, this is not the focus of this work, and we thus omit further details about the witness displayer.

6. WITNESS RETRIEVAL AND RANKING

As explained in the previous section, our ranking model is concerned with ranking a set of witnesses that are relevant to a given set of RDF statements. Note that we decouple the problem of transforming the user query into RDF statements from the

ranking of witnesses problem since they are basically independent problems. We thus assume that the transformation of queries into RDF statements is deterministic. Our ranking model can easily be extended to handle the case where this transformation process involves inaccuracies. However, in this paper we focus on the problem of ranking the witnesses given a set of RDF statements.

Our ranking model is based on statistical language models (LMs) [16, 25]. The witness documents are ranked based on the probability of being relevant to the query statements $\{t_1, t_2, ..., t_n\}$, which we denote as $P(d|t_1, t_2, ..., t_n)$ (here, d is a witness). Applying Bayes' rule, we have:

$$P(d|t_1, t_2, ..., t_n) = \frac{P(t_1, t_2, ..., t_n|d)P(d)}{P(t_1, t_2, ..., t_n)} \quad (1)$$

Since $P(t_1, t_2, ..., t_n)$ does not depend on the witnesses, we can ignore it during the ranking. This implies that

$$P(d|t_1, t_2, ..., t_n) \propto P(t_1, t_2, ..., t_n|d)P(d) \quad (2)$$

$P(d)$ is the prior probability that witness d is relevant to any statement. This probability can be estimated in various ways, and in our case we estimate it using the static authority of the page or pagerank [8]. We do this in order to take into consideration the trustworthiness of the witnesses (i.e., give higher weight to witnesses that are authoritative).

As for the probability $P(t_1, t_2, ..., t_n|d)$, we assume independence between the query statements for computational tractability (in-line with most traditional keyword ranking models). Thus,

$$P(t_1, t_2, ..., t_n|d) = \prod_{i=1}^{n} P(t_i|d) \quad (3)$$

To avoid overfitting and to ensure that the witnesses that do not contain all the query statements do not have a zero probability, we use smoothing:

$$P(t_1, t_2, ..., t_n|d) = \prod_{i=1}^{n} [\alpha P(t_i|d) + (1 - \alpha)P(t_i|Col)] \quad (4)$$

The first component is the the probability of the statement t_i being generated by document d and the second component is the probability of generating the statement t_i by the whole collection Col.

We rely on two different methods to estimate the probability $P(t_i|X)$ of generating statement t_i using X, where $X \in \{d, Col\}$:

Unbiased Estimator. Since statements can appear in different forms in witnesses, we need to first fold a statement into all corresponding indicative patterns instances that can be used to represent the statement in order to estimate the probability $P(t_i|X)$. This is similar to translation models [36], when the query is expressed in one language, and the documents retrieved are in a different language. Let the set $Y = \{y_1, y_2, ..., y_m\}$ be the set of all possible pattern instances. The probability $P(t_i|X)$ is then computed as follows:

$$P(t_i|X) = \Sigma_{j=1}^{m} P(t_i|y_j)P(y_j|X) \quad (5)$$

The first component $P(t_i|y_j)$ in Equation 5 is the probability of representing the statement t_i using pattern instance y_j, which is computed as a function of the confidence of representing the relation of t_i using the pattern of y_j, as defined in Definition 5.2. Assuming that the relation of t_i is r_i and the pattern of y_j is p_j, we have: $P(t_i|y_j) = f(conf(p_j, r_i))$. Note that $P(t_i|y_j)$ would be *zero* if r_i cannot be expressed using pattern p_j.

The second component $P(y_j|X)$ in Equation 5 is the probability of generating pattern instance y_j given X where $X \in \{d, Col\}$. This is estimated using a maximum-likelihood estimator as follows:

$$P(y_j|X) = \frac{c(y_j; X)}{\Sigma_{y \in Y} c(y; X)} \quad (6)$$

Here, $c(y; X)$ denotes how often the pattern instance y occurs in X.

Entity-Biased Estimator. The unbiased estimator only takes into consideration the occurrence of the statements in the witnesses. It does not take into consideration what other statements occur in the witness and whether or not they are related to the query statements. For instance, assume the user query consisted of the statement t =*(Barack_Obama, bornIn, Kenya)*. Also assume that witnesses d_1 and d_2 both contain x statements, and that the query statement appears in each witness only once. Thus, $P(t|d_1) = P(t|d_2)$. However, assuming that d_2 contains more statements about Obama than d_1 which is a news directory for instance, we would like to rank d_2 higher than d_1. To this end, let the statement t_i be of the form: *(s, r, o)* with subject s, predicate r and object o. The probability $P(t_i|d)$ is now defined as follows:

$$P(t_i|d) = \beta_s P_e(s|d) + \beta_o P_e(o|d) + (1 - \beta_s - \beta_o)P_t(t_i|d) \quad (7)$$

where $P_e(s|d)$ and $P_e(o|d)$ are the probabilities of generating the subject s and object o using witness d respectively, and $P_t(t_i|d)$ is the probability of generating the whole statement t using d. The parameters β_s and β_o control the influence of each component.

The probability $P_t(t_i|d)$ is estimated using the unbiased estimator from Equation 5. The probabilities $P_e(s|d)$ and $P_e(o|d)$ are in turn estimated using the frequency of occurrence of s and o in d, respectively. For instance, $P_e(s|d)$ is computed as follows:

$$P_e(s|d) = \frac{c(s; d)}{\Sigma_{e \in d} c(e; d)} \quad (8)$$

where $c(e; d)$ is the number of times an entity e occurs in document d.

7. EVALUATION

In this section, we discuss a three-fold evaluation of our ranking model. First, we outline the setup of our evaluation environment (Section 7.1). Then, we discuss what effect the parameters in general have (Section 7.2). Afterwards, we analyze the performance of our ranking model (given the set of witnesses is already determined) based on some concrete settings tailored towards different use cases. We use a naive arbitrary ranking and a keyword-based entity-oriented ranking for comparison (Section 7.3). Finally, our retrieval approach is compared to a purely keyword-based retrieval approach, that is not aware of any statement indication found by the extraction engine (Section 7.4).

7.1 Setup

Dataset & Document Pooling. We use the English part of the ClueWeb 09 document corpus[1], a standard IR benchmark collection containing about 500 million English language documents harvested from the Web. As running our extraction system, SOFIE/PROSPERA [21, 33], on the whole ClueWeb corpus would have been too expensive with our current setup, we constructed a document topic subset by selecting 43 well-known entities, i.e., politicians (e.g., Barack Obama), actors/directors (e.g., Clint Eastwood), soccer players and clubs (e.g., David Beckham) and scientists (e.g., Albert Einstein), to retrieve the top-1000 documents based on a BM25-based score [28] for different names and spelling variants of these entities taken from the general purpose knowledge base YAGO [32]. This resulted in a pool of 182,139 documents, on

[1] http://boston.lti.cs.cmu.edu/Data/clueweb09/

which a modified version of SOFIE/PROSPERA was run that keeps track of fact provenance information. As SOFIE/PROSPERA iteratively learns new patterns, the extraction process was repeated several times.

Queries. We defined 56 queries in the form of statement sets. For instance, one query with a single statement would be the set $\{(\texttt{Barack\_Obama}, \texttt{isLeaderOf}, \texttt{USA})\}$, while one with multiple statements focussing on JFK's assassination would be given by $\{(\texttt{JFK}, \texttt{diedOn}, \texttt{1963-11-22}), (\texttt{JFK}, \texttt{diedIn}, \texttt{Dallas})\}$. We only considered statements that were supported by at least 10 documents (usually more). For each query, a set of potentially relevant documents was generated by pooling the top-50 rankings using various settings of our ranking mechanism. Remember that this includes a filtering step such that only documents containing an indication for at least one statement of the query were considered for the ranking. This resulted in witness pools of 80 witnesses per query on average with a minimum of 11 documents and a maximum of 435 documents for a single query. For the system comparison (see Section 7.4), additional documents retrieved by the (statement indication unaware) runs based on keyword search were added to each pool.

Human Assessments. As outlined earlier, our framework aims at two different use cases. First, a user might aim to learn more about specific statements; second, she might be interested in verifying the validity of some statements. Reflecting these two use-cases our system aims to support, all documents in the pool were assessed 1) as a whole document on their on-topicness (does it contain more information relevant to the query than the actual statement indication) and 2) separately for each statement on their persuasiveness (how strong is its support for this statement). For both assessment dimensions a graded relevance scale with three grades (*non-relevant, somewhat relevant, highly relevant*) was used. For binary measures such as MAP we project the graded assessments to a binary relevance scale, such that a statement or document is relevant in the binary scale if it is highly relevant in the graded scale. For the parameter and ranking (of prefiltered witnesses), evaluation we will mostly assume that documents can only be judged as being relevant for a statement on the *persuasiveness* scale if the extraction system recognized an appropriate indication of the statement in that document. From a usability point of view this makes sense since a user will usually skip a result for which the system cannot highlight a statement indication. However, if the user considers the document, she may be able to identify support for statements not recognized by the extraction engine. In a more aggressive assessment, we have also considered a document as relevant with regard to persuasiveness if the assessor could find the statement, but the system could not. Note that this only affects multi-statement queries since we require at least one indication of a statement to add a document to the pool. We denote the evaluation ignoring indications not recognized by the extraction system as *machine-ignorant* and the one that is more thorough as *human-aware*. For the system evaluation (Section 7.4), where we compare against purely keyword search approaches without any prior knowledge about statement indications, we solely use the *human-aware* method.

Measures. We apply different measures for result quality that loosely correspond to our two evaluation aspects. First, we compute the mean average precision (MAP) and the normalized discounted cumulative gain (nDCG, [18]) for the top 5,10, and 20 results, averaged over all queries. Second, we compute the mean rank (mr), i.e., the first rank at which at least one convincing (highly relevant) indication has been seen for each statement in the query on average, and related to this the mean reciprocal rank (mrr).

7.2 Parameter Analysis

Our model provides several possibilities to adjust it to a user's needs. The entity bias can be controlled by the β_s, β_o variables (see Equation 7). Similarly the influence of the smoothing versus the main ranking function can be influenced by setting the α parameter (Equation 4). Additionally our implementation allows to switch influence of document trustworthiness (i.e., pagerank) on and off. Similarly, the influence of the pattern confidence for computing $P(t_i|X)$ in Equation 5 can be set to no influence, linear influence or quadratic influence.

We have investigated the effects of different settings of these parameters. The pagerank values, which we use as an approximation of a document's probability to contain accurate and extensive information, do not seem to have a significant impact. This might have several reasons: 1) Our query set did not contain many disputed statements where we saw "untrustworthy" witnesses. 2) SPAM pages were already sorted out using the Waterloo Spam rankings[2]. 3) We use pagerank only as a page trust approximation. Pagerank might not always be a good indicator that the information provided is of high quality, in the end we aim at having user feedback provide the trust values. The other parameters, however, provide the expected effects. For instance, increasing the β values tends to improve results on the on-topicness aspect, but might decrease the precision for the persuasiveness (see Table 1), while a higher confidence influence tends to have the inverse effect. However, those parameters are quite interdependent, so that there is no definite behavior that applies to each setting, e.g., increasing one β value means decreasing either the pattern confidence influence or the other β parameter. If one of the β weights or both together reach 1 they completely dominate the ranking formula which might provoke a stronger negative (or sometimes positive) effect depending on the other settings. Similarly, the initial step from both β values being set to 0 towards having at least one of them at 0.1 has a much larger impact as similar increases on a setting where $\beta_s + \beta_o > 0$ already holds. Table 1 shows one of the more extreme cases clearly outlining the influence of the β_s parameter on the ranking performance. For changes of the α value we could not find a noteworthy overall impact on average. However, we did not look into extreme values (e.g., $\alpha = 1$ or 0). Still, a high value of α can lead to overfitting as the smoothing looses influence.

7.3 Ranking Evaluation

In this section, we analyze the performance of our ranking method, given a set of documents identified as witnesses for (part of) the query, in terms of *on-topicness* and *persuasiveness.* We compare the proposed ranking method to a naive ranking (*naive*), which ranks the witness set randomly, as well as to a keyword-based entity-oriented ranking method (*lucene*) implemented with *Apache Lucene*[3] using its default parameter setting. *Both alternative ranking methods were applied on the prefiltered witness pool generated for each query*, as we only evaluate the ranking of witnesses, not the identification of potential witnesses.

For the Lucene-based rankings, we generated keyword queries based on the entities contained in the statement(s) of the query. The relations are therefore not represented in the Lucene keyword query translations. As keyword engines are specialized in finding documents for a certain topic given by keywords, we expected that this method will have good results in the on-topic evaluation.

[2] http://durum0.uwaterloo.ca/clueweb09spam/

[3] `http://lucene.apache.org/`

relevance type	measure	grading scheme	$\beta_s = 0$	$\beta_s = 0.1$	$\beta_s = 0.3$	$\beta_s = 0.5$	$\beta_s = 0.6$	$\beta_s = 0.8$	$\beta_s = 1$
persuasiveness	Avg MAP	binary	0.898	0.813	0.817	0.826	0.816	0.815	0.79
persuasiveness	Avg nDCG	graded	0.898	0.844	0.82	0.808	0.802	0.8	0.78
persuasiveness	Avg mrr	binary	0.89	0.80	0.78	0.76	0.75	0.747	0.746
on-topicness	Avg MAP	binary	0.673	0.703	0.752	0.759	0.755	0.757	0.777
on-topicness	Avg nDCG	graded	0.61	0.71	0.764	0.781	0.780	0.786	0.79

Table 1: The effects of β_s with $\beta_o = 0$, quadratic pattern confidence influence, pattern weighing on and $\alpha = 0.5$

We were mainly interested in a on-topicness-oriented baseline and whether using the knowledge about relations in the ranking would have a noticeable impact in terms of persuasiveness.

Name	α	β_s	β_o	confidence	pagerank
persuade	0.5	0	0	quadratic	on
topic	0.9	0.5	0.5	no	off
mix	0.9	0.1	0.3	quadratic	off
old	0.7	0	0	linear	on
pr	0.5	0	0	no	on
cf	0.9	0	0	quadratic	off

Table 2: The ranking model configurations

For our own system, we make use of three main example configurations of our ranking model representing different use cases (for details see Table 2). One setting (*persuade*) aims at high persuasiveness by ignoring any entity occurrences, but rather relying on the pattern confidence values. Another one (*topic*) mimics an entity search, aiming at on-topicness by focusing totally on the entity occurrences ($\beta_s = \beta_o$= 0.5, thus in a maximal combination) ignoring any other influence like page rank or pattern confidence. A third configuration tries to achieve a good balanced mix of both goals by applying some entity bias while still considering pattern confidences and applying pattern frequency smoothing (*mix*).

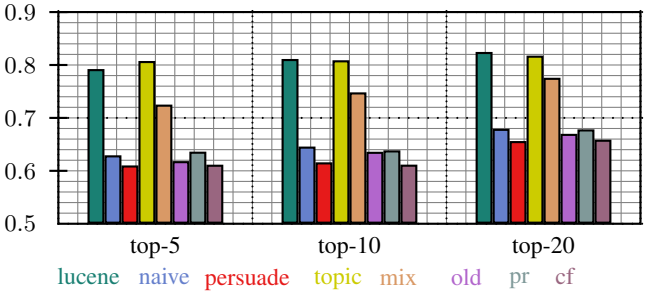


Figure 2: Comparison of *on topic* performance based on *machine-ignorant nDCG* for all queries

Additionally, settings that mainly rely on pattern confidence (*cf*) and respectively pagerank (*pr*) are investigated as well as a setting without any entity bias and with linear confidence influence (*old*) introduced in an earlier demo paper [10].

Fig. 2 shows that *lucene* clearly meets the expectation of performing well in the *on-topic* evaluation in terms of nDCG, and only *topic*, our parameter setting optimized for achieving high on-topicness, can compete with it. Note, however, that the mixed configuration *mix* is not too far behind (∼7%), while all other configurations are at the same level as the naive approach.

The picture changes when we look at persuasiveness of statement indications. Fig. 5(a) shows the mean rank for single-statement queries. Here, the statement-oriented rankings (*persuade*, *old*) are notably better than the entity-oriented approaches: using *persuade* a user would on average have to look only at the first result to be convinced that the statement she is interested in holds, while with

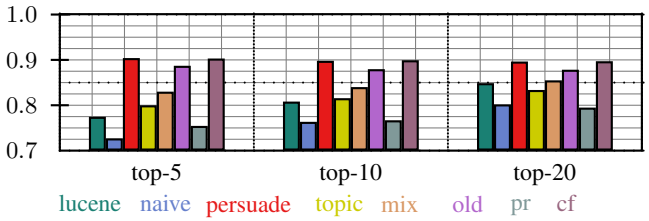


Figure 3: Comparison of persuasiveness performance based on *machine-ignorant nDCG* for all queries

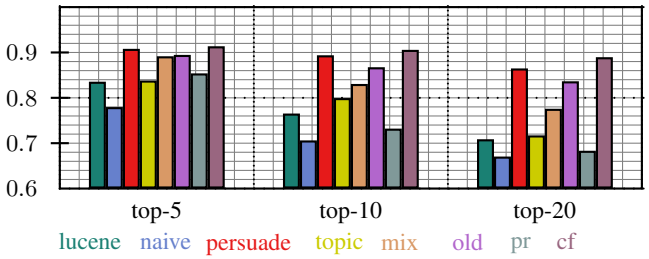


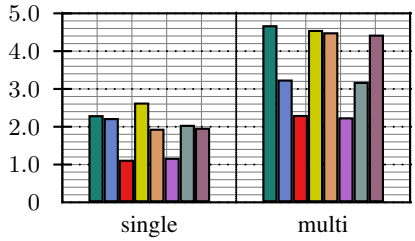
Figure 4: Comparison of persuasiveness performance based on *machine-ignorant MAP* for all queries

lucene she would on average need to look at a second document. Note that *topic* performs similar to *lucene*. The better performance of statement-oriented approaches is reinforced by the mrr values in Fig. 5(b), where *persuade* dominates while pagerank-only ranking is surprisingly good. The reason for the latter might be that those documents with a large pagerank in our corpus are often from Wikipedia. So for most single-statement queries the Wikipedia page of the main entity will be in the top results and would usually contain what we are looking for. The good performance of *pr* breaks down on multiple-statement queries or when we look at the nDCG values in Fig. 3, 6 and 7, as a single good document is not enough for a high nDCG value. For nDCG (Fig. 6) and MAP (Fig. 4), *persuade* is notably ahead of the other approaches.

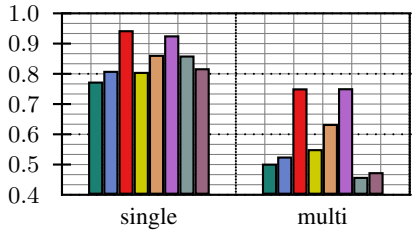
The advantage of using relation information on persuasiveness becomes even clearer when we look at multi-statement queries (Fig. 5(a), 5(b) and 7). Note that the difference between *persuade*, *cf* and *old* and the topic-oriented approaches *lucene,topic* increases further. You may also note that *cf*, which sometimes outperforms *persuade* falls behind on multi-statement queries (Fig. 7). This is probably due to the higher α value, which may lead to overfitting. Also, pagerank cannot help as a counterweight. It might also be surprising that the naive approach has obviously quite a convincingly small mr value, however, the nDCG is about half that of the better pattern aware settings.

So far we have only considered results from the *machine-ignorant* evaluation. However, results from a *human-aware* evaluation recognizing also statement indications not found by the extraction system as support for the persuasiveness judgements are, in general, not too different as can be seen in Fig. 5(c). While the advantage of *persuade* and *mix* over lucene stays similar for single fact queries, lucene can gain on multi-statement queries.

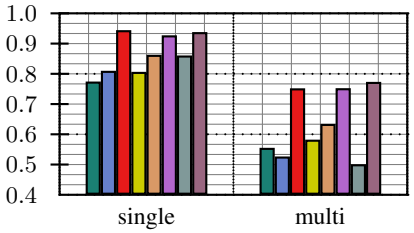
Finally, in Fig. 8 we investigate a combined relevance measure that only counts documents as being relevant if they are highly



(a) Comparison based on *machine-ignorant mr* for single- and multi-statement queries



(b) Comparison based on *machine-ignorant mrr* for single- and multi-statement queries



(c) Comparison based on *human-aware mrr* for single- and multi-statement queries

Figure 5: Comparison of lucene, naive, persuade, topic, mix, old, pr, cf on persuasiveness performance based on *mrr* and *mr* values

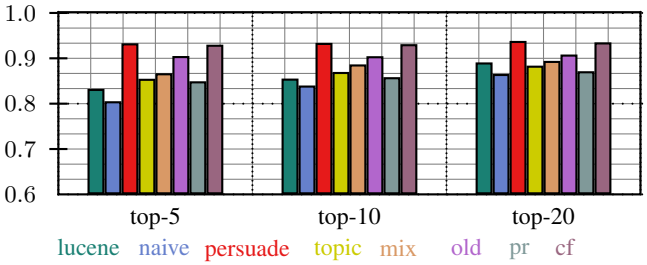


Figure 6: Comparison of persuasiveness performance based on *machine-ignorant nDCG* for single-statement queries

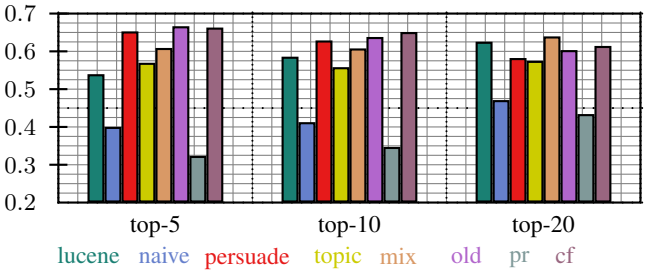


Figure 8: Comparison based on *nDCG* for multi-statement queries in a combined *machine-ignorant* evaluation setup

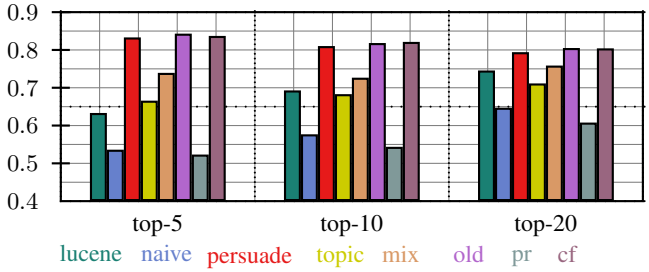


Figure 7: Comparison of persuasiveness performance based on *machine-ignorant nDCG* for multi-statement queries

relevant on exactly one aspect and as being highly relevant if they are highly relevant on both aspects. For instance, a document being relevant in the on-topicness aspect alone is non-relevant, a document being highly relevant in the on-topicness aspect alone is relevant and a document being highly relevant in the on-topicness and persuasiveness aspect is highly relevant. Here we can see that the statement-oriented and mixed settings perform quite well with *mix* being more adept for the long tail while the topic-oriented approach falls behind.

After all, by relying mainly on entity occurrences (*topic*), we can mimic the performance of a state-of-the-art keyword-based ranking system, and this gives good results in terms of on-topicness, while bringing in the relations (*persuade*) will increase the persuasiveness decreasing the on-topicness, but also a balancing is possible with our framework (*mix*). In the next step, we investigate how useful our framework for the overall retrieval task is compared to a keyword query system.

7.4 System Evaluation

The general use case discussed in this paper is end-user statement search, so we also compare against a keyword search retrieval system without any prior knowledge about contained statements. Therefore, we ran Lucene (again with default settings) on the initial set of 182,139 documents retrieved from ClueWeb with three variations of keyword queries. First, each of our queries was translated into a keyword query based only on the involved entities *(fluc:ee)*. Second, each query was fully translated with the relations being translated by a manual mapping, e.g., translating the relation `wasBornIn` to "was born in" *(fluc:ere)*. Finally, each query was translated fully again but the relations were translated based on the best (highest confidence) pattern known to the extraction system for that particular relation *(fluc:epe)*. For each of these variations, we gathered the top-20 documents and assessed them. The results of the evaluation are shown in Fig. 9. As expected, the keyword-based search achieves good results for on-topicness that are competitive to our approach, albeit by comparing *lucene* with *fluc:ee* one can see that the indication-based filtering improves results notably. This becomes even more evident with respect to *persuasiveness*. This can be seen by comparing the keyword-based approaches to the *naive* ranking (with prefiltering) or to the prefilter-supported Lucene run *(lucene)*. Our persuasiveness tailored approach stands out prominently: similarly clear as the three measures shown in Fig. 9 are the mr values: using a keyword-based approach, a user would need to look on average at ∼4.5 results *(fluc:ee)*, ∼3.8 results *(fluc:ere)*, and ∼3.7 results *(fluc:epe)* compared to ∼2.9 results with the prefilter supported *lucene* or ∼1.5 results with the *persuade* setting. Note that in general the effectiveness of our pattern index based ranking strongly depends on the quality of the extraction system. A keyword search approach has the general advantage that it is independent of the extraction system but vice versa does not gain from the extraction system's knowledge about statement indications.

8. CONCLUSION AND FUTURE WORK

In this paper, we introduced the notion of statement search and presented a framework implementing it. Given a phrase query, the proposed framework maps it to a factual statement (or a set of such statements) and retrieves documents containing textual expressions supporting the queried statement(s). We proposed a configurable ranking model based on language models for ranking witnesses, which can be tuned for favoring either documents with strong statement support or further information related to the given statement(s). Our evaluation results show that our ranking model outperforms term-based document ranking in finding strongly supportive documents.

There are a number of possible directions for future work. An important extension of the ranking model would be to automatically derive optimal tuning parameters based on user preferences. Another aspect in this context is diversification, i.e., given multiple statements and a set of documents that only contain subsets of the statements, adapt the ranking of the results in a way that documents

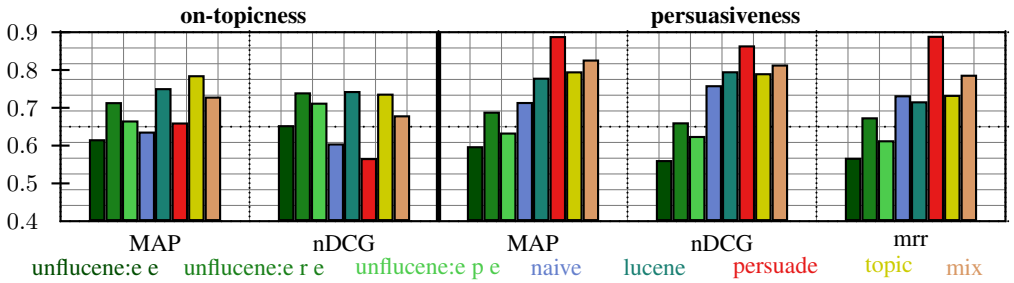


Figure 9: Comparison against pure lucene based approaches with respect to on-topicness and persuasiveness based on the top-10

in the top ranks contain different subsets of the statements. In addition, considering user feedback might help to improve result ranking as well as the precision of the underlying information extraction process, e.g., by exploiting direct feedback on fact quality and counting user clicks to measure the importance of a witness document.

9. REFERENCES

[1] X. Bai, R. Delbru, and G. Tummarello. RDF snippets for Semantic Web search engines. In *OTM Conferences (2)*, pages 1304–1318, 2008.

[2] K. Balog, L. Azzopardi, and M. de Rijke. A language modeling framework for expert finding. *Inf. Process. Manage.*, 45(1):1–19, 2009.

[3] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *ACL*, pages 28–36. The Association for Computer Linguistics, 2008.

[4] W. M. Barczynski, F. Brauer, and A. Mocan. ExplainIE - Explaining Information Extraction Systems. In *ICIQ*, pages 268–269, 2009.

[5] J. Bear, D. J. Israel, J. Petit, and D. L. Martin. Using information extraction to improve document retrieval. In *TREC*, pages 367–377, 1997.

[6] R. Blanco and H. Zaragoza. Finding support sentences for entities. In *SIGIR*, pages 339–346, 2010.

[7] F. Brauer, W. M. Barczynski, G. Hackenbroich, M. Schramm, A. Mocan, and F. Förster. RankIE: Document Retrieval on Ranked Entity Graphs. *PVLDB*, 2(2):1578–1581, 2009.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[9] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM*, pages 652–659, 2004.

[10] S. Elbassuoni, K. Hose, S. Metzger, and R. Schenkel. ROXXI: Reviving witness dOcuments to eXplore eXtracted Information. *PVLDB*, 3(2):1589–1592, 2010.

[11] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum. Language-model-based ranking for queries on RDF-graphs. In *CIKM*, pages 977–986, 2009.

[12] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, 2008.

[13] R. Fagin, B. Kimelfeld, Y. Li, S. Raghavan, and S. Vaithyanathan. Understanding queries in a search database system. In *PODS*, pages 273–284, 2010.

[14] A. Harth, A. Hogan, R. Delbru, J. Umbrich, S. O'Riain, and S. Decker. SWSE: Answers Before Links! In *Semantic Web Challenge*, 2007.

[15] M. Hearst. Direction-based text interpretation as an information access refinement. In *Text-Based Intelligent Systems*, pages 257–274. 1992.

[16] D. Hiemstra. A probabilistic justification for using tf x idf term weighting in information retrieval. *Int. J. on Digital Libraries*, 3(2):131–139, 2000.

[17] D. Hiemstra. Statistical language models for intelligent xml retrieval. In *Intelligent Search on XML Data*, pages 107–118, 2003.

[18] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[19] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in web text. In *KDD*, pages 457–466, 2009.

[20] C. Mangold. A survey and classification of semantic search approaches. *IJMSO*, 2(1):23–34, 2007.

[21] N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, 2011.

[22] Z. Nie, Y. Ma, S. Shi, J.-R. Wen, and W.-Y. Ma. Web object retrieval. In *WWW*, pages 81–90, 2007.

[23] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: a document-oriented lookup index for open linked data. *Int. J. Metadata Semant. Ontologies*, 3:37–52, November 2008.

[24] T. Penin, H. Wang, T. Tran, and Y. Yu. Snippet Generation for Semantic Web Search Engines. In *ASWC*, pages 493–507, 2008.

[25] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281. ACM, 1998.

[26] J. Pound, I. F. Ilyas, and G. Weddell. Expressive and flexible access to web-extracted data: a keyword-based structured query language. In *SIGMOD*, pages 423–434, 2010.

[27] Y. Qiu and H.-P. Frei. Concept based query expansion. In *SIGIR*, pages 160–169, 1993.

[28] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[29] K. Schumacher, M. Sintek, and L. Sauermann. Combining Fact and Document Retrieval with Spreading Activation for Semantic Desktop Search. In *ESWC*, pages 569–583, 2008.

[30] B. Sereno, S. B. Shum, and E. Motta. Claimspotter: an environment to support sensemaking with knowledge triples. In *IUI*, pages 199–206, 2005.

[31] W3c: Sparql query language for rdf. www.w3.org/TR/rdf-sparql-query/.

[32] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.

[33] F. M. Suchanek, M. Sozio, and G. Weikum. SOFIE: A Self-Organizing Framework for Information Extraction. In *WWW*, 2009.

[34] V. S. Uren, S. B. Shum, M. Bachler, and G. Li. Sensemaking tools for understanding research literatures: Design, implementation and user evaluation. *International Journal of Man-Machine Studies*, 64(5):420–445, 2006.

[35] J. Xu, R. M. Weischedel, and C. Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *SIGIR*, pages 105–110, 2001.

[36] C. Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2, March 2008.