SESSA - Keyword-Based Entity Search through Coloured Spreading Activation

Denis Lukovnikov and Axel-Cyrille Ngonga Ngomo

AKSW, University of Leipzig, Leipzig, Germany
{lukovnikov,ngonga}@informatik.uni-leipzig.de

Abstract. A growing wealth of data is being published as RDF on the Web of Data. The structured character of this data allows for more targeted and complex queries to be answered than on the document Web. However, the query language SPARQL presents a quasi insurmountable obstacle to lay users. We address the problem of entity search based on keyword queries on the Linked Data Web from a Question Answering (QA) perspective. Whereas many systems attempt to generate a SPARQL query that is assumed to encompass the meaning of the user's input, we propose a best-effort approach for entity search. Our method extends the spreading activation principle by also spreading colours through a graph and returns the entity which abides by the colour restrictions derived from the user query. This novel perspective on keyword search has the advantage of the discovery of the correct query segmentation and the term disambiguation being intrinsic to the approach. We evaluate our approach on the QALD-3 benchmark and show that we outperform the state of the art by achieving an F-measure of 56.9% on entity search questions and an F-measure of 41% on the whole of QALD-3.

1 Introduction

Over the last years, a growing amount of data has been published in RDF and according to the Linked Data principles on the Web.[1] The enormous amount of information contained in these datasets has been used in manifold applications.[2] The information contained in the Linked Data Web is most efficiently accessed by drawing upon the query language SPARQL. However, most lay users are not familiar with this language [10]. Thus, alternative approaches for accessing RDF datasets are necessary to ensure that not only experts but also lay users can make use of the wealth of knowledge available on the Web of Data. The currently dominant keyword-based paradigm for searching the Web of documents provides an intuitive means for users to interact with data at Web scale. Consequently, the development of keyword search systems for RDF data has been advocated [10,11].

In this paper, we address the problem of keyword-based entity search over Linked Data. Formally, given a knowledge base K which contains a set of resources R, we consider a query q to be an entity search query if the correct answer to q is a subset of R. Entity search questions generally describe a series of conditions that the entities

[1] http://stats.lod2.eu/

[2] See for example http://challenge.semanticweb.org/2013/.

in the result set must abide by. Thus, in order to find the answer to an entity search question, we can simply explore the facts in the underlying knowledge base and look for resources that fulfil the conditions posed in the question. By these means, we can actually refrain from constructing a SPARQL query. We implement our intuition by first transforming RDF graphs into what we call *reified graphs*, which consists of facts as well as resources and literals that support those facts (see section 3 for a formal definition and construction approach). We then apply a novel, knowledge-driven entity search method that relies on a modified version of the spreading activation algorithm [9] on these graphs. We call our extension the *colour-and-activation-spreading algorithm* (short: *coloured spreading activation*) and use it as a simple yet powerful way for implementing keyword search on RDF knowledge. This paradigm has two main advantages:

1. Many existing systems follow a pipeline-based paradigm [2,5,6,10,11,12,13], where query interpretation consists of several subsequent steps which precede the construction of a SPARQL query that is supposed to retrieve the correct answer. However, such approaches are prone to cascading errors throughout the pipeline. Our method does not suffer from error propagation typical for a pipelined architecture because we do not perform explicit disambiguation or segmentation at any point. Instead, we consider different possible disambiguations and segmentations throughout the search process. This gives the method the ability to avoid mistakes by deferring interpretation decisions to a point where all relevant knowledge is collected and a large number of query interpretation options is implicitly pruned away based on the structure of the relevant knowledge represented by the sub-graph explored during the search.
2. It allows for a best-effort approach that tolerates noise. SPARQL query-constructing methods typically attempt to explain all of the sentence elements that seem relevant. User queries which are noisy may thus not be answered. Instead of trying to filter noisy elements using shallow techniques, our approach tolerates them while exploring the graph and returns the answer that explains most of the query in a manner consistent with the facts in the knowledge base.

The contributions of this paper are as follows:

- A novel QA approach for keyword-based entity search that performs knowledge-driven disambiguation and segmentation while exploring the input knowledge base to find the answer;
- A novel graph search method called colour and activation spreading algorithm (or coloured spreading activation) that is able to spread both colours and activation energies in graphs and uses a colour exhaustion termination criterion;
- An evaluation of our approach on the QALD-3 benchmark dataset.

The rest of this paper is structured as follows. In section 2, we give an overview of related work. Then, we present our algorithm in section 3 and discuss some of its features in section 4. After an evaluation of our approach in section 5, we conclude in section 6.

2 Related Work

Our approach is related to the research area of entity search over Linked Data, which we approach as a subtask of the more general problem of question answering (QA) over Linked Data. Several systems have been developed for the latter task. One of the first systems was AquaLog [6], an ontology-driven QA system for the Semantic Web. Aqualog uses linguistic analysis to transform the input query to a set of query-triples. Then, these query triples are interpreted using lexical resources and the given ontology. The interpreted query-triples are sent to an inference engine to find the answer. One major drawback of AquaLog is that it is limited to one ontology at a time. To address this and other drawbacks of AquaLog, PowerAqua [5] was developed. PowerAqua follows an approach similar to that of AquaLog, reusing AquaLog's linguistic analysis. However, PowerAqua performs more advanced query-triple interpretation using different ontologies simultaneously.

Treo [2] is a method for querying Linked Data that also relies on spreading activation. First, pivot entities in the query are identified. Then, from the dependency structure of the input sentence, Treo constructs a Partially Ordered Dependency Structure (PODS). The PODS is used to resolve the query in the spreading activation search step where semantic relatedness scores are used to rank candidates and subsequently spread activation. Pythia [13] is a more recent QA system. It uses lexica that define mappings between a syntactic and a semantic representation of an expression. With these lexica, Pythia also introduces a distinction between ontology-dependent and ontology-independent lexica. Whereas the former depends on the verbalizations of entities from some ontology, the latter describe ontology-independent expressions (determiners, question words,...). The sentence is parsed using such lexica and the resulting semantic representation is translated to a formal query. Building upon Pythia's dichotomy between ontology-dependent and -independent lexical entries, TBSL [12] presents a template-based QA approach. This approach consists of 3 steps. First, SPARQL query templates are generated by using domain-specific and generic language resources. The template slots are filled by using a combination of resource lookup and natural-language patterns extracted using the BOA framework [3]. The resulting SPARQL queries are finally scored and the best query (i.e., the highest ranking query that returns a non-empty result set) is selected and returned.

More recently, several novel systems have participated in the QALD-3 challenge on QA over Linked Data [1]. CASIA [4] (the currently best-performing system on the QALD-3 benchmark dataset) relies on a three-step approach resembling AquaLog's architecture. During the first step, the question type is determined and text triples are constructed from the dependency parse tree of the question sentence. In the second step, RDF resources which match phrases from the text triples are detected. CASIA uses relation extraction patterns from PATTY[8] to map text fragments to properties and classes. In the final step, a SPARQL query is generated based on the question type and the RDF resources detected in the input question. CASIA achieves an F-score of 0.36 on the QALD-3 benchmark.

Works on keyword-based querying of the Web of Data include SINA [10] and the work of Tran et al. [11]. SINA [10] aims at answering a keyword question using different datasets. First, simultaneous disambiguation and segmentation is performed using

Hidden Markov Models (HMM) and the Hyperlink-Induced Topic Search (HITS) algorithm. The found resources are used to construct an Incomplete Query Graph (IQG) constisting of disjoint sub-graphs. To build the federated SPARQL query that retrieves the results, the IQG's are connected using a Minimum Spanning Tree approach inspired by Prim's algorithm. The work of Tran et al. [11] tackles the problem of keyword search over RDF data. More specifically, the work of Tran et al. is concerned with mapping keywords to a list of ranked conjunctive queries, with a special focus on efficient inference of implied connections. To accomplish this, a top-k algorithm is proposed that computes the best query interpretations of the keyword query using bidirectional graph exploration. The interpretations are then scored and mapped to conjunctive queries. The performance of the proposed top-k approach is evaluated on DBLP.

Our approach focuses on keyword-based entity search over RDF data. Due to the keyword setting, we can not rely on syntactic analysis techniques, such as dependency parsing. In contrast to most existing QA and keyword-based search approaches, we employ a best-effort approach that simultaneously performs segmentation, disambiguation and graph exploration to retrieve the result that explains most of the user query.

3 SESSA

In this section, we describe our QA approach for keyword-based entity search, dubbed SESSA. The intuition behind our approach is that in the context of entity search, a user query usually describes a set of conditions that must hold for a resource to be part of the answer to the query. Thus, by propagating a set of conditions (modelled using colours) within a graph of facts, we can detect those resources that satisfy all conditions and deduce the set of resources that abide by the user query. We implement this intuition by running our approach SESSA on a *reified RDF graph* G, which we introduce formally in the first subsection of this section. Overall, SESSA consists of five steps. First, the input query q is used to build an n-gram hierarchy (subsection 3.2), which is a directed graph H. This graph enables us to consider different possible segmentations throughout the rest of the algorithm. Secondly, for each n-gram in the hierarchy, we use a dictionary D to detect possible meanings of the n-gram (subsection 3.3). The third step is colour assignment, in which the possible meanings are assigned colour within G. The initial colouring of the graph is used as input for the fourth step (subsection 3.4) called the Coloured Spreading Activation step (subsection 3.5) in which the colouring is propagated across G. The fifth and final step is retrieving the answers to the query from the resulting coloured graph (subsection 3.7). In contrast to earlier QA approaches, we explicitly use an extended ontology lexicon, eliminating the need for indirect ontology mapping methods (such as computing semantic relatedness [2]). Given appropriate lexica for entities, SESSA can thus be used for querying any knowledge base in any language. We use the query

"*birthplace bill gates wife*"

on DBpedia as the running example for the rest of this paper. This query aims to retrieve the entities representing the place of birth of Melinda Gates, the wife of Bill Gates.

3.1 Graph Reification

Let R be the set of all RDF resources, B be the set of all blank nodes, L be the set of all literals and P be the set of all properties. An RDF knowledge base K can be regarded as a set of triples $(s, p, o) \in (R \cup B) \times P \times (R \cup B \cup L)$. The corresponding reified graph is a directed graph $G = (V(G), E(G))$ which we construct as follows. First, we set $V(G) = \emptyset$ and $E(G) = \emptyset$. Then, for each triple (s, p, o), we create a fact node f and add the triples nodes and the fact node to G: $V(G) = V(G) \cup \{s, p, o, f\}$. Then, we add the edges (f, s), (f, p) and (f, o) to $E(G)$. We repeat the addition of nodes and edges for all triples from the input knowledge base. Figure 1 shows an example of a reified graph for facts pertaining to the running example.

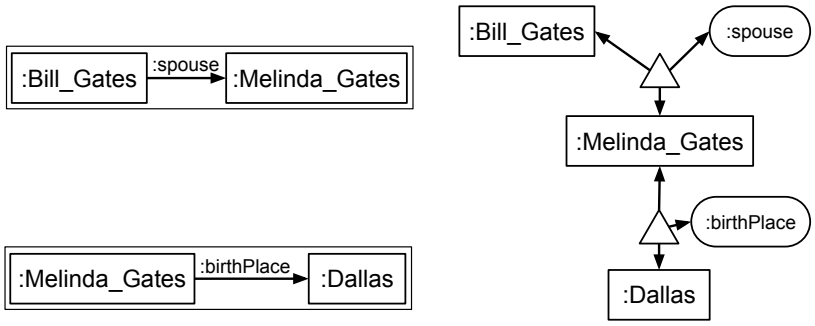

Fig. 1. Graph transformation

3.2 Building the n-gram hierarchy

Formally, SESSA's input is a user query q which consists of a sequence of keywords $k_1 \dots k_m$. For our example, $k_1 = \texttt{birthplace}$, $k_2 = \texttt{bill}$, $k_3 = \texttt{gates}$ and $k_4 = \texttt{wife}$. To construct an n-gram hierarchy, we construct a directed graph $H = (V(H), E(H))$ as follows. The nodes of H are subsequences $k_i \dots k_j$ from q with $1 \leq i \leq j \leq m$. The pair of nodes $(ng, ng') \in E(H)$ iff $ng = k_i \dots k_j \wedge i < j \wedge (ng' = k_{i+1} \dots k_j \vee ng' = k_i \dots k_{j-1})$. Logically, nodes which consist of exactly one keyword do not have any children. In the running example, the bigram *"bill gates"* is the parent of two unigrams, *"bill"* and *"gates"*. The whole n-gram hierarchy for the running example is shown in Figure 2.

3.3 Candidate Generation

The aim of this step is to detect candidate resources from the graph G that map to nodes of the n-gram hierarchy H. To achieve this goal, we rely on dictionaries. Formally, a dictionary (also called lexicon) $D : V(G) \mapsto 2^W$ maps entities (individuals,

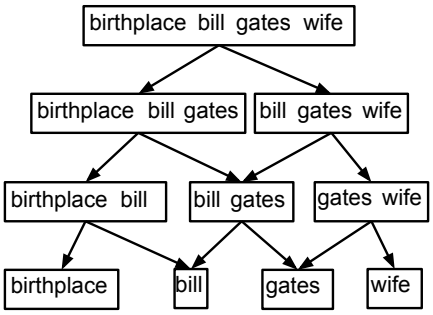

Fig. 2. Running example: N-gram hierarchy

predicates and classes) $e \in V(G)$ from the reified graph G to n-grams $s \in W^*$ whose unigrams stem from the set of words W of natural language. For example, the resource `dbr:Bill_Gates`[3] is mapped to the sequence *"bill gates"* in English. We use dictionaries to assign a set of candidate entities $C(ng) \in R$ to n-grams $ng \in V(H)$ from the n-gram hierarchy H obtained in the previous step:

$$C(ng) = D^{-1}(ng). \tag{1}$$

A simple way to construct the dictionary D is by using label relations (such as `rdfs:label`) available in the knowledge base K. In addition to that, external lexical resources can be used, such as Lemon[7] lexica for ontologies or the results of relation extraction frameworks such as BOA [3]. Such external lexical resources commonly contain lexicalizations of resources that cannot be derived from label properties. For example, the label of the property `dbo:spouse` in DBpedia[4] is *"spouse"*, but *"wife"* or *"husband"* are alternative lexicalization to indicate the `dbo:spouse` relation.

After retrieving a set of candidates $C(ng)$ for each n-gram ng, we prune the sets of candidates. The hierarchy H is traversed to remove candidates $c \in (C(ng) \cap (\bigcup_{ng_i} C(ng_i)))$ of child n-grams ng that are already present in one of ng's ancestors ng_i. In the running example, the bigram *"bill gates"* has the resource `dbr:Bill_Gates` as a possible meaning. However, the entity `dbr:Bill_Gates` may also be a possible meaning of the unigram *"gates"*, which is a child of *"bill gates"* in H. If this is the case, the entity `dbr:Bill_Gates` is removed from the list of candidates for *"gates"*.

After the completion of the steps above, we now have a mapping between the n-gram hierarchy H and the reified graph G. In the following two steps, we aim to use this mapping to detect facts that map the requirements of the user query. To this end, we first map the candidate resources $r \in V(G)$ for each n-gram $ng \in V(H)$ to colours. Then we spread these colours within G to perform an implicit disambiguation and segmentation as well as to obtain facts that match all conditions, i.e., that have all colours.

[3] `dbr` stands for `http://dbpedia.org/resource`.
[4] `dbo` stands for `http://dbpedia.org/ontology/`.

3.4 Colour Assignment

For the sake of simplicity, we assume that $\forall ng, ng' \in V(H) : ng \neq ng' \Rightarrow C(ng) \cap C(ng') = \emptyset$. The colour assignment function Q maps each $r \in V(G)$ to a set of colours. These colours abide by a *colour hierarchy* P. Given the assumption above, the set of colours $Q(r)$ for every resource $r \in \bigcup_{ng \in V(H)} C(ng)$ is a singleton $Q(r) = \{k\}$, which is assigned according to the following rules:

- The **first rule** states that different possible meanings of the same n-gram should be assigned the same colour and candidates of different n-grams should have different colours: $\forall r, r' \in C(ng) : Q(r) = Q(r')$ and $\forall ng, ng' \in V(H) : ng \neq ng' \Rightarrow \forall r \in C(ng), r' \in C(ng') : Q(r) \cap Q(r') = \emptyset$. This enables our spreading activation algorithm to disambiguate internally, as will be discussed later (section 4).
- The **second rule** states that the colour hierarchy should reflect the N-gram hierarchy. Colours can have children and the child-parent relations between colours should be set in the same way as in the N-gram hierarchy H. Formally, the child-parent relations of colours can be defined by a directed graph $P = (V(P), E(P))$ where $c \in V(P)$ are colours and edges $(c, c') \in E(P)$ denote that c' is a child of c. For edges in $E(P)$, the following holds: $(c, c') \in E(P) \wedge c \in Q(r) \wedge c' \in Q(r') \Leftrightarrow (ng, ng') \in E(H) \wedge r \in C(ng) \wedge r' \in C(ng')$. For example, if the possible meanings of *"gates"* are green and the possible meanings of *"bill gates"* are blue, then green should be a child of blue because *"gates"* is a child of *"bill gates"*.

The results of this step are an initial colouring of G as well as a colour hierarchy P. A part of the initial colouring of the reified graph G for the running example is shown on the left side of Figure 3 (p. 9). The candidates for the n-grams *"birthplace"*, *"bill gates"* and *"wife"* in H receive their respective colours triangle, diamond and circle. Some n-grams not shown in Figure 3 are *"bill"* and *"gates"*. Since the n-grams *"bill"* and *"gates"* are both children of *"bill gates"*, their candidates receive colours that have the colour diamond (the colour of candidates of *"bill gates"*) as their parent.

3.5 Initialization of Coloured Spreading Activation

The aim of coloured spreading activation is to propagate the colours set in the previous steps into the graph until facts are found that have all colours and thus fulfil all criteria specified by the query. The original spreading-activation algorithm operates on a graph as follows. Let e_i be the current activation of the node u_i in the graph. The regular spreading activation algorithm computes the new energy e_i of u_i as $e_i = D \sum_{v_j:(v_j,u_i)\in E} w_{ji} e_j$, where w_{ji} is the weight of the edge (v_j, u_i) and $D \in [0, 1]$ is a decay factor. In contrast, SESSA uses a modified version of spreading activation, which we call *coloured spreading activation*. It differs from the standard spreading activation algorithm in two key aspects. First, our approach relies on two energy scores dubbed *energy* and *explanation*. Moreover, it uses coloured activation criteria to prohibit incorrect activations as well as to terminate the algorithm.

The *explanation score* $x : V(G) \mapsto \mathbb{N}$ indicates how many of the input unigrams is explained by some activated resource $v \in V(G)$. For example, the resource

`dbr:Bill_Gates` has an explanation score of 2 because it stands for two unigrams, i.e., *"bill"* and *"gates"*. Nodes that did not belong to the set of candidates of any n-gram are assigned an explanation of 0 during initialization.

The *energy* function $e : V(G) \mapsto \mathbb{R}^+$ approximates the probability that a node v should be part of or lead to the solution generated by SESSA. The initial value of e approximates the probability that the n-gram ng which was used to select the node $v \in V(G)$ as a candidate actually referred to that node (i.e., the probability that v is the right candidate for the given n-gram). While several functions can be used to compute the initial value of e for each $v \in V(G)$, we estimate this value by counting how often ng was used to denote the resource v and dividing this count by the total number of occurrences of ng in a large reference corpus. Note that energy scores are only required to be set such that $\forall v_i, v_j \in C(ng) : e(v_i) > e(v_j) \Leftrightarrow P(v_i|ng) > P(v_j|ng)$ where $P(v_i|ng)$ denotes the probability that resource v_i from $V(G)$ is meant by some n-gram ng. Thus, we could also use other functions such as the string similarity between v's label and $D(v)$. Comparing different approaches for computing the initial value of e is out of the scope of this paper and will be carried out in future work. Nodes that did not belong to the set of candidates of any n-gram are assigned an energy of 0. Overall, while the explanation score is used to retrieve the final result that explains most of the query, the energy scores can be used for ranking within the final result.

3.6 Coloured Spreading Activation

The spreading activation process is iterative in nature and updates the energy, explanation and colours of the nodes $v_i \in V(G)$ which fulfil the following activation criteria:

1. A node can be updated if the *minimum activation criterion* is fulfilled. For fact nodes f, at least two nodes u, v connected to f must be activated (i.e., the following must hold: $\exists((f,u) \in E(G) \wedge (f,v) \in E(G)) : e(u) > 0 \wedge x(u) > 0 \wedge e(v) > 0 \wedge x(v) > 0$). For all other nodes, at least one connected node must be activated.
2. A node can be updated if the colours of its activated predecessors can be combined, where two colours c and c' cannot be combined if $c = c'$ or c and c' are descendants of each other or share descendants in P: $(descendants_P(c) \cup \{c\}) \cap (descendants_P(c') \cup \{c'\}) \neq \emptyset$ where $descendants_P(c)$ is the set of all descendants of colour c in colour hierarchy P derived from ancestral relations $E(P)$.

Nodes which fulfil these criteria are updated as follows:

$$e(v_i) = \sum_{v_j:(v_j,v_i)\in E(G)} e(v_j), \tag{2}$$

$$x(v_i) = \sum_{v_j:(v_j,v_i)\in E(G)} x(v_j), \tag{3}$$

$$Q(v_i) = \bigcup_{v_j:(v_j,v_i)\in E(G)} Q(v_j). \tag{4}$$

Nodes which do not fulfil the criteria are not updated. Note that the activation criteria ensure that the explanation score can never be higher than the number of keywords in

the query, as this can only be the case if the activation inherent to some query keyword is used multiple times while composing the answer. However, a keyword cannot be used several times when computing an answer due to the combinations of a colour with itself or with its parents being prohibited. Similarly, the activation criteria also prohibit combinations of colours that are associated with overlapping n-grams. After each iteration at least one node must get a new colour for the algorithm to continue to run. In most real use cases, the number of spreading activation passes is typically under $2 * (\text{ number of keywords } -1)$.

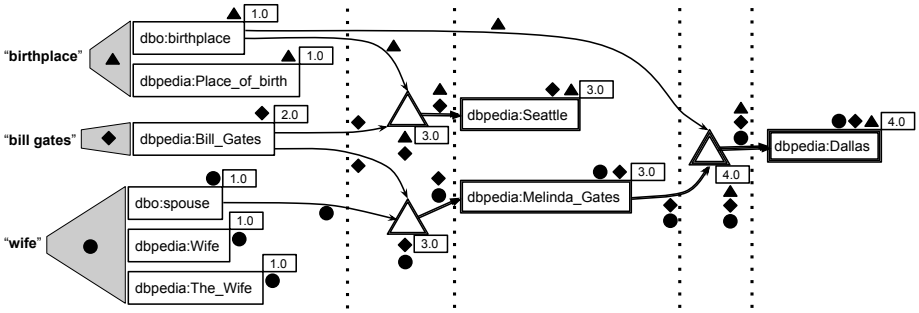

Fig. 3. Running example: spreading activation

The coloured spreading activation algorithm is illustrated in Figure 3, using our running example. The leftmost part of Figure 3 shows the initialization of the coloured spreading activation algorithm. The rest of Figure 3 shows the stabilized state of the subgraph. The numbers in the boxes represent the explanation scores of the activation states of the corresponding node. The triangle, diamond and circle represent colours used by the algorithm. The dotted lines show in which iteration a node was activated.

3.7 Getting results

The algorithm eventually stops, as guaranteed by the colour criterion. The result is a subgraph where nodes have different activation state values. The set of nodes with the highest explanation score is taken as the result. In our running example, the entity node `dbr:Dallas` accumulated an explanation score of 4.0, which is also the highest possible explanation score. Thus the result of the query from the running example is the entity `dbr:Dallas`. Additionally, the energies of nodes can be used to order the results of the approach. Among the nodes with the highest explanation score, a higher energy leads to a higher rank.
It is also possible to retrieve the k best results: All in $V(G)$ can be grouped by their final explanation score into explanation score classes which then are ranked by explanation score. Nodes within the same explanation score class can further be ranked using energies.

4 Discussion

In this section, we discuss some of the interesting aspects of the presented approach. We begin by explaining how our approach performs simultaneous disambiguation and segmentation during the search process. Thereafter, we discuss how the approach can be used to answer analogy and out-of-scope questions. Finally, we discuss how our approach deals with queries which require implied relations.

4.1 Inherent disambiguation and segmentation

The aim of segmentation is to split up the query string in such a way that all the words that refer to a single resource are in the same segment while disambiguation deals with choosing the best meaning from a number of possible meanings of some ambiguous segment. The possible meanings for a segment are determined by the candidate generator. Together, segmentation, candidate generation and disambiguation can be regarded as the process of *interpretation*. The best decision for disambiguation and segmentation is more likely to be found while solving both together. For example, SINA[10] performs disambiguation and segmentation simultaneously.

However, many existing systems (including SINA) follow pipelined approaches that are vulnerable to error cascading, i.e. errors tend to accumulate through the pipeline. With the approach presented in this paper, the propagation of disambiguation and segmentation errors is prevented by performing both simultaneously with finding the answer. This approach uses the structure of the available knowledge to find the best interpretation while searching, which goes beyond the possibilities of disambiguation preceding the actual search. One of the advantages following from this approach is natural support for local disambiguation in keyword queries. With local disambiguation, we mean the correct assignment of possibly different meanings to identical n-grams in the same input, as, for example in the boolean question *"birthplace troy star troy?"* (*Is the star of the movie Troy born in Troy?*).

SESSA considers different possible interpretations of an n-gram and does not perform explicit disambiguation at any point, instead implicitly disambiguating while searching for the answer. The explanation scores define the initial result for the user query and thus perform most of the implicit disambiguation. However, the initial result obtained using explanation scores is refined by taking the interpretations of n-grams with the highest energy score. Thus, the algorithm favours interpretations that lead to a better answer. In the running example, the `:spouse` interpretation of *"wife"* is chosen over other interpretations because of its collaboration in the most explanatory answer.

Like the disambiguation, the segmentation is handled internally by coloured spreading activation. Several possible segmentations of the input query are considered concurrently during the coloured spreading activation process (as reflected by H) but the segmentation giving the result with the highest explanation score is preferred over other possible segmentations.

4.2 Answering analogy questions

The coloured spreading activation algorithm provides support for answering analogy questions like *"Michelle Obama Barack Obama Bill Gates"* (which should return *"Melinda*

Gates"). The algorithm is able to find the relation between two entities and reuse this relation in combination with other entities. However, we do not assess this ability of the algorithm due to a lack of benchmark data for this type of queries.

4.3 Assessing response quality

The final activation state of the graph provides information for assessment of the response quality. The meaning of explanation scores and energies as explained previously can be used for response evaluation. Colours assigned to nodes indicate which parts of the user query the node explains and the number of colours indicates the number of interactions that led to the activation of a node. To illustrate the usefulness of this information, we adapted our system to answer OUT OF SCOPE questions in the QALD-3 dataset, which was not done previously by any QA system.

4.4 Finding and handling implied relations

Sometimes, relations needed to find the answer are not explicitly mentioned in the query but are implicitly assumed based on the meaning of some N-gram. We find possible implied predicates by examining the incoming relations of entities. Predicates are ranked according to the number of incoming relations of the entity that are instances of the predicates. The predicate(s) with the most incoming instances will be activated as the implied predicate(s). For example, in *"tolstoy play"*, the unigram *"tolstoy"* refers to Leo Tolstoy and *"play"* may refer to the theatre play type. However, the desired connections between these concepts is not explicitly mentioned. But we can see that Leo Tolstoy is the `:author` of a large number of things and thus, the implied predicate is probably `:author` whereas *"play"* probably implies that we are interested in things that have `:Play` as its `:type`.

5 Evaluation

5.1 Experimental Setup

We evaluated our approach on questions from the QALD-3 test dataset and used their keyword formulations (as given in the QALD-3 dataset) as input. We chose QALD-3 over QALD-4 because QALD-3 has been used by previous systems and thus makes comparing our approach with the state of the art possible. We used the `answertype` tag to determine the expected type of results for each query because the keyword formulations often did not contain information that could be used to infer the desired type of answer (e.g. boolean or node). While our main goal was to answer entity search questions, we performed the evaluation using *all* 100 questions from the QALD-3 dataset to demonstrate the overall question answering capabilities of our approach and make comparison with previous approaches possible. The results for QALD-3 presented below are given for different subsets of the QALD-3 questions in order to keep the difference between entity search performance over QALD-3 and overall question answering performance clear.

The first set of questions was restricted to entity search queries that did not require the YAGO namespace, did not use comparative or superlative clauses (as in *"Give me the second highest mountain"*) and did not use quantifiers. This set contained 65 of the 100 questions contained in the test dataset of QALD-3. Note that in contrast to the state of the art, we also attempted to detect OUT OF SCOPE questions. This is done by looking at the number of colours in the best nodes in the graph after spreading activation. If the best nodes have only one colour, then these nodes are candidates for the n-grams in the original query and none of the nodes interacted. In this case, the answer to the question is OUT OF SCOPE.

The second set of questions contained boolean queries that do not use comparatives, superlatives or quantifiers and do not use the YAGO namespace. Overall, QALD-3 contained 5 queries of this kind. The answer to a boolean question is `true` if the final graph state contains nodes with maximally possible explanation score and false otherwise. For example, the answer to *"michelle obama wife barack obama"* (question 70) is true because the algorithm is able to activate the fact node <`:Barack_Obama`, `:spouse`, `:Michelle_Obama`>with highest possible explanation score of 5.

All evaluation scores for different sets of questions presented below were derived from the results obtained by running our system on all 100 questions from the QALD-3 test dataset using the same configuration of our system. We used DBpedia and Wikipedia to determine the initial values of the energy function e for resources $v \in C(ng)$, for all $ng \in V(H)$. We only used lexical information for the DBpedia ontology namespace (`dbpedia-owl`) and the DBpedia property namespace (`dbpprop`). The `rdfs:label` properties of entities on DBpedia as well as the English Lemon lexicon for DBpedia [14] were used to construct the dictionary D for candidate generation. Additionally, the Lemon lexicon for the DBpedia ontology namespace was extrapolated to the DBpedia property namespace. Often, the DBpedia property namespace contained predicates of the same name as in the DBpedia ontology namespace. Predicates from the DBpedia property namespace which exactly match a DBpedia ontology predicate were assigned the same lexical information from the Lemon lexicon as the DBpedia ontology predicate. This method produced a collection of lexical mappings[5] for DBpedia. We also added[6] minor optimizations to our system and implemented some features (like support for multiple activation of the same resource node) which are not discussed here.

For each question, the following metrics were computed[7]

$$precision = \frac{\text{number of correct answers}}{\text{total number of answers given}} \tag{5}$$

$$recall = \frac{\text{number of correct answers}}{\text{total number of answers expected}} \tag{6}$$

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \tag{7}$$

[5] `https://raw.githubusercontent.com/lukovnikov/datasets/master/lex.dbpedia.rdf`

[6] see code for details: `https://raw.github.com/lukovnikov/datasets/master/s3x.zip`

[7] `https://github.com/lukovnikov/qald-evaluation`

Table 1. Evaluation results. The second column shows the number of questions of the type described in the first column. The third shows the number of questions for which the system generated an anwer. The column labelled "exact" gives the number of questions that were answered with an F-measure of 1.00. Average precision, recall and F-measure are given in the columns labelled "P", "R" and "F" respectively.

Dataset	Total	Answered	Exact	P	R	F
Entity search only	65	40	31	0.558	0.603	0.569
Boolean only	5	4	4	0.8	0.8	0.8
QALD-3	100	44	35	0.402	0.432	0.410

Results The questions that could be answered are shown in Table 2 in section A (Appendix). The evaluation metrics over the different question types and the whole of QALD-3 are shown in Table 1. Overall, SESSA was able to answer 35 queries with an F-measure of 1.00 of which 31 were entity search queries and 4 were boolean queries. Out of the 65 questions in the entity search set, high-recall (recall $>$0.75) answers for 40 questions were found. Over the portion of QALD that we considered for entity search, the average precision was 0.558, the average recall 0.603 and the macro-averaged F-measure was 0.569. On the whole of QALD-3, this results in an average precision of 0.402, an average recall of 0.432 and a macro-averaged F-measure of 0.410. The system was able to answer 4 OUT OF SCOPE-questions. With these results, we outperform CASIA [4], the winning system of the QALD-3 challenge which achieved an F-measure of 36%. Note that CASIA relies on the full sentence formulations of the questions. Using keywords instead of sentences leads to SESSA having less information (especially for complex questions) and is in most cases not to our advantage.

5.2 Error analysis

The main sources of errors in our system were:

1. *Suboptimal performance of relation implication*: our approach sometimes could not draw connections between two terms because the wrong relation was implied by our system. For example, for *"company Munich"* (question 39), the implied relation for the `dbr:Munich` sense of *"Munich"* was `dbo:birthplace` whereas the relation `dbo:location` is needed to find the answer.
2. *Lack of lexical knowledge for ontologies*: the generated lexica are still incomplete and the Lemon lexicon we relied on covers only the DBpedia ontology namespace. The provision of such lexicons for the whole of the benchmark would enable our system to answer more questions, particularly questions which rely on the YAGO ontology. For example, *"U.S. state governor Sean Parnell"* (question 27) could not be answered because our lexica do not map *"U.S. state"* to `yago:StatesOfTheUnitedStates`.
3. *Complex questions*: the system was not able to answer questions which contained quantifiers, comparatives or superlatives (such as question 5, *"mountain second highest"*). However, such questions are out of scope of the proposed method as it is only concerned with entity search as declared previously.

4. *Ambiguous questions*: some questions are more ambiguous in their keyword formulations than they are in their full sentence formulations as given in the QALD-3 test dataset. For example, the question *"Who founded Intel"* (question 41) has a keyword formulation *"Intel founded"*. However, from this keyword formulation, we cannot imply a resource of type person is expected, which results in poor precision.

6 Conclusion

In this paper, we present a simple yet powerful idea for entity search using keywords. Our approach relies on the colour-and-activation-spreading algorithm, which strives to find the best interpretation of the query keywords by avoiding explicit disambiguation and segmentation. Instead, our algorithm embraces the ambiguity of natural language and tries to find the best segmentations and disambiguations simultaneously with the answers. Our method provides a best-effort approach at entity search, trying to give results explaining the whole query if possible, otherwise giving the most explanatory partial results. Whereas partial results from intermediate iterations of spreading activation do not provide correct answers for the QALD benchmark, they can be useful for users by automatically providing context, alternative interpretations and related entities. Even though the QALD dataset that was used to evaluate the proposed approach leaves it in a disadvantage w.r.t. state of the art QA systems that use full sentence formulations of the questions, the proposed approach achieved an F-score superior to that of CASIA, the winning system of the QALD-3 challenge. However, the QALD dataset did not allow us to evaluate some powerful features of our system, such as finding relations between resources and answering analogy-based questions. We will aim to include such queries in future QA benchmarks. Finally, the proposed algorithm is ontology- and language-independent as it does not rely on language-specific syntactic analysis nor on a specific schema. The only ontology- and language-dependent part of our system are the lexica, which need to be provided for every ontology-language pair in order to be supported by the search system. The system can thus easily be extended in order to support federated search on interlinked datasets, provided qualitative lexical information for their ontologies can be found.

The proposed approach can benefit from different possible improvements, some of which are shortly discussed here. Both searching for implied relations and their handling during spreading activation can be improved, which will allow to answer more questions. Extending coverage of the ontology lexica will allow to verbalize more relations and classes and thus extend the coverage of the system. Finally, energy-based ranking of results within the same explanation class can be further refined, for example by ranking candidates that are relations or classes together with candidates that are entities. Further research includes (1) exploiting local syntactic and morphological clues for more efficient search, (2) adding support for questions containing quantifiers and comparatives. Detecting and processing local syntactic and morphological clues will support more sentence-like formulations of queries and provide additional indications for correctly interpreting the questions.

Finally, our method can be implemented more efficiently over a graph database, allowing for distributed operation, less overhead during graph exploration and efficient

spreading activation. The current implementation relies on Lucene indices for candidate generation and SPARQL queries to a DBpedia endpoint for graph exploration.

References

1. Philipp Cimiano, Vanessa Lopez, Christina Unger, Elena Cabrio, Axel-Cyrille Ngonga Ngomo, and Sebastian Walter. Multilingual Question Answering over Linked Data (QALD-3): Lab Overview. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 321–332. Springer, 2013.
2. André Freitas, João Gabriel Oliveira, Seán ORiain, Edward Curry, and João Carlos Pereira Da Silva. Querying Linked Data using semantic relatedness: a vocabulary independent approach. In *Natural Language Processing and Information Systems*, pages 40–51. Springer, 2011.
3. Daniel Gerber and A-C Ngonga Ngomo. Bootstrapping the Linked Data Web. In *1st Workshop on Web Scale Knowledge Extraction@ ISWC*, volume 2011, 2011.
4. Shizhu He, Shulin Liu, Yubo Chen, Guangyou Zhou, Kang Liu, and Jun Zhao. CASIA@ QALD-3: A Question Answering System over Linked Data.
5. Vanessa Lopez, Enrico Motta, and Victoria Uren. Poweraqua: Fishing the Semantic Web. In *The Semantic Web: research and applications*, pages 393–410. Springer, 2006.
6. Vanessa Lopez, Michele Pasin, and Enrico Motta. Aqualog: An ontology-portable question answering system for the Semantic Web. In *The Semantic Web: Research and Applications*, pages 546–562. Springer, 2005.
7. John McCrae, Dennis Spohr, and Philipp Cimiano. Linking lexical resources and ontologies on the Semantic Web with Lemon. In *The Semantic Web: Research and Applications*, pages 245–259. Springer, 2011.
8. Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. PATTY: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics, 2012.
9. M Quillian. Semantic Memory. *Semantic Information Processing*, pages 227–270, 1968.
10. Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Question Answering on Interlinked Data. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1145–1156. International World Wide Web Conferences Steering Committee, 2013.
11. Thanh Tran, Haofen Wang, Sebastian Rudolph, and Philipp Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 405–416. IEEE, 2009.
12. Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. ACM, 2012.
13. Christina Unger and Philipp Cimiano. Pythia: Compositional meaning construction for ontology-based question answering on the Semantic Web. In *Natural Language Processing and Information Systems*, pages 153–160. Springer, 2011.
14. Christina Unger, John McCrae, Sebastian Walter, Sara Winter, and Philipp Cimiano. A Lemon lexicon for DBpedia. In *NLP-DBPEDIA@ISWC*, 2013.

A QALD-3 results

Table 2. Evaluation results for QALD-3 questions for which an answer was found by SESSA

Question	Input	Precision	Recall	F-measure
2	john f. kennedy successor	0.429	1.0	0.6
3	berlin mayor	1.0	1.0	1.0
4	free university amsterdam students	1.0	1.0	1.0
14	member prodigy	0.833	0.833	0.833
19	people born vienna die berlin	0.8	0.667	0.727
20	tall michael jordan	1.0	1.0	1.0
21	capital canada	1.0	1.0	1.0
22	governor wyoming	1.0	1.0	1.0
24	father queen elizabeth II	0.167	1.0	0.286
28	movie direct francis ford coppola	1.0	1.0	1.0
30	birth name angela merkel	1.0	1.0	1.0
35	minecraft develop	0.5	1.0	0.667
36	copper melting point	1.0	1.0	1.0
37	brno sister city	1.0	1.0	1.0
38	maribor inhabitants	1.0	1.0	1.0
41	intel founded	0.5	1.0	0.667
42	amanda palmer husband	1.0	1.0	1.0
43	breed german shepherd dog	1.0	1.0	1.0
45	rhine country	1.0	1.0	1.0
46	professional surfer born philippines	1.0	1.0	1.0
47	hawaii average temperature	1.0	1.0	1.0
52	invent zipper	1.0	1.0	1.0
54	san francisco nickname	1.0	1.0	1.0
56	hells angels founded	1.0	1.0	1.0
57	astronaut apollo 14	1.0	1.0	1.0
58	salt lake city timezone	1.0	1.0	1.0
62	socrates influence aristotle	1.0	1.0	1.0
64	launch pad operate nasa	0.875	0.875	0.875
65	john lennon instrument	1.0	1.0	1.0
67	juan carlos I wife parents	0.5	1.0	0.667
68	google employees	1.0	1.0	1.0
69	tesla nobel prize physics	1.0	1.0	1.0
70	michelle obama wife barack obama	1.0	1.0	1.0
74	michael jackson die	1.0	1.0	1.0
76	margaret thatcher child	1.0	1.0	1.0
78	margaret thatcher chemist	1.0	1.0	1.0
81	book kerouac publish viking press	1.0	0.8	0.889
83	mount everest high	1.0	1.0	1.0
84	comic captain america create	0.667	1.0	0.8
86	australia largest city	1.0	1.0	1.0
87	harold and maude compose music	1.0	1.0	1.0
90	prime minister spain residence	1.0	1.0	1.0
94	pilsner urquell brewery founding year	1.0	1.0	1.0
100	produce orangina	1.0	1.0	1.0