Exploratory Keyword Search Over Graphs with Quality Guarantees

Casey Fleck[1], Mohammad Hossein Namaki[2], Xin Zhang[2], Yinghui Wu[2], Lawrence Holder[2]

[1] Coastal Carolina University, Conway, SC 29526

[2] Washington State University, Pullman, WA 99164

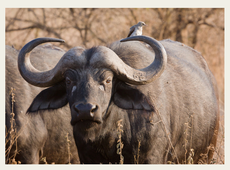
Motivation and Problem

- Querying complex information networks, such as knowledge graphs, is a challenging task for non-expert users.
  - Noisy and incomplete
  - Schemaless, or no schema
  - Highly heterogeneous
- Keyword search provides an intuitive way of querying to get better insight about relationships among entities.
- Keywords approximate user's information need -- calls for an automatic query refinement process.
  - e.g. Q_0: **Buffalo**
- Query Refinement is an incremental process of transforming a query into a new query that more accurately reflects the user's information need (Vélez B 1997).

Q_1: Buffalo NY

Q_1: Bison Buffalo

Q_1: African Buffalo



Q_1: Buffalo wing

Figure 1: some potential examples of "Buffalo" keyword search refinement

- *How can we refine the input query to **reduce the ambiguity** of the original keywords and **guarantee the quality of the answers**?*

Keyword search (KWS) over graphs

- A keyword query is a set *n* of keywords $Q=\{k_1, ..., k_n\}$.
- Given a graph *G*, the query *Q*, a bound *b*, an answer to *Q* is a rooted tree *T(r)* such that
  - ✓ *T* is a subgraph of *G*, and *r* is the root.
  - ✓ *T* contains all the keywords.
  - ✓ For any vertex *v* in *T*, dist(v, r) ≤ *b*.

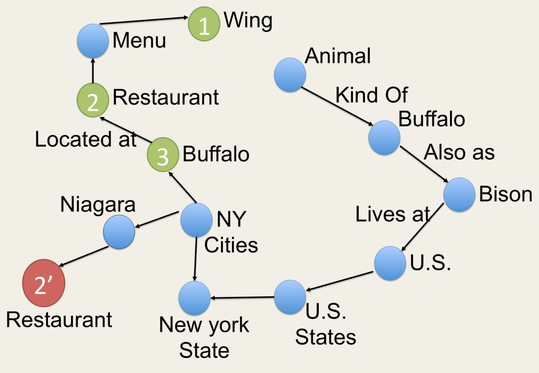


Figure 2: KWS over a graph, searching for "wing restaurant Buffalo"

Quality-preserving KWS Refinement

Quality measure

- Interested in results with closest pairwise keyword matches.

$$weight = \sum_{i=1}^{n} \sum_{j=i+1}^{n} dist(v_i, v_j)$$

where v_i is the match of keyword k_i in the result tree.

- The lower the weight is, the higher the quality is.

Disambiguity measure

- The disambiguity between two keywords k_1 and k_2 is computed using *normalized semantic distance* (De Nies 2016).

$$\frac{\max\{\log|C_\lambda(k_1)|, \log|C_\lambda(k_2)| - \log|C_\lambda(k_1) \cap C_\lambda(k_2)|\}}{\log|V| - \min\{\log|C_\lambda(k_1)|, \log|C_\lambda(k_2)|\}}$$

where $C_\lambda(k_i)$ is the type of k_i neighbors in the data graph.

- The same type of nodes are connected to a pair of keywords with similar meaning.
- ***Div(Q)*** is the sum of pairwise keyword disambiguaties.

Problem Statement

➢ **Input:** Given a labeled data graph G; a keyword search query $Q_i = \{k_1, ..., k_n\}$; a quality loss bound ε; and a distance *bound b*.

➢ **Output:** Retrieving the answers of a refined keyword query Q_{i+1} by maximizing ***Div(Q)*** subject to the quality bound ε

Refinement Framework

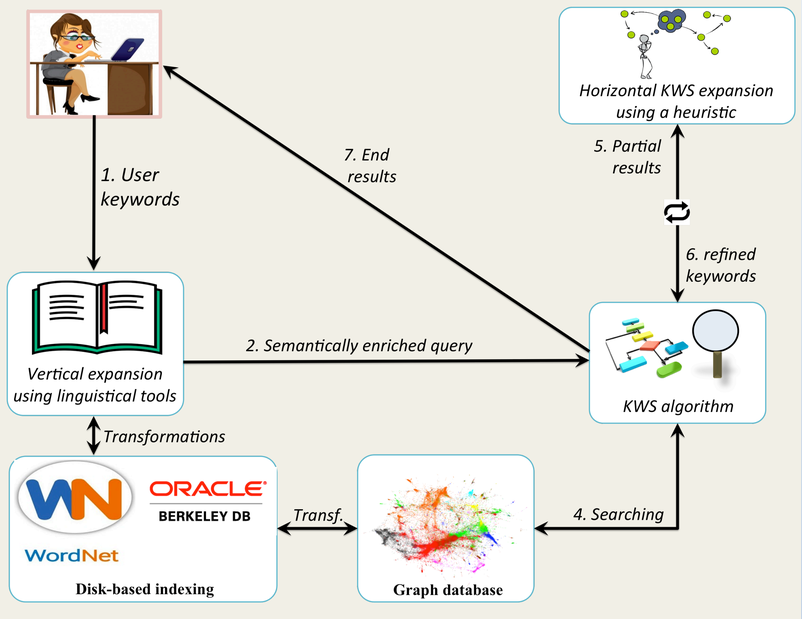


Figure 2: General keyword search refinement procedure

Algorithm

Input: Query Q_i, data graph G, partial results R_i, error bound ε;
Output: a refined keyword search query Q_{i+1};

1. Initialize set **K** = {}, priority queue **D** = empty //ordered by disambiguity;
2. For each result **r** in $\mathbf{R_i}$
3. **K** = **K** ⋃ keywords of the neighbors of vertices in **r**;
4. For each keyword **k** in **K**
5. **D** = **D** ⋃ <**k**, Div(Q_i + **k**)>;
6. While (**D** is not empty)
7. **k'** ← poll from **D**;
8. **UB**(**k'**) ← estimate the upper bound of weight(Q_i + **k'**);
9. if (**UB**(**k'**) ≤ [(1+ε) × weight(R_i)])
10. return (Q_i + **k'**)
11. return Q_i

KWS Performance

- We have implemented **BANK** (Bhalotia et al. 2002), an exact KWS algorithm.
- We have run experiments over DBPedia.
- **DBPedia** is a multi-labeled knowledge graph extracted from Wikipedia including 4M of entities (people, companies, cities, etc.) and 15M of their relationships.

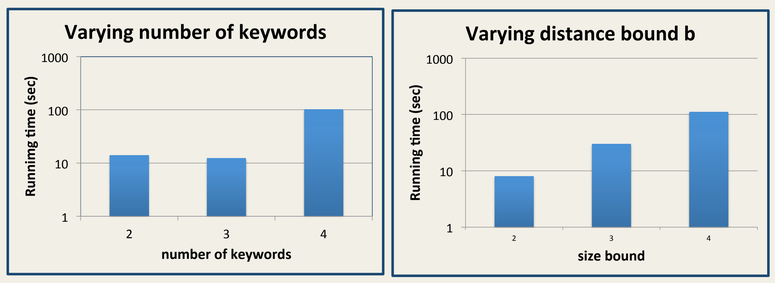


Figure 3: Running time of KWS over DBPedia by varying parameters

Case Study

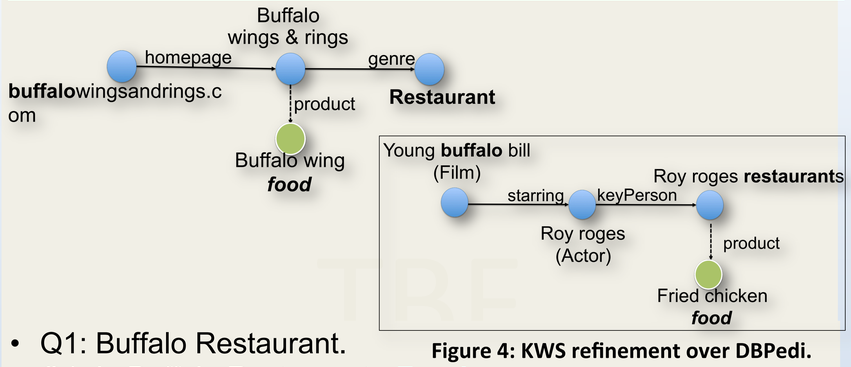


Figure 4: KWS refinement over DBPedi.

- Q1: Buffalo Restaurant.
- Q1+k: Buffalo Restaurant +Food
- A(Q2) = A(Q1)[blue vertices] + green vertex.

Future Work

- Evaluating the refinement algorithm by leveraging both user study and an automatic approach.
- Extending the framework for other query classes.

Acknowledgements

This work was supported by the Washington State University Smart Environments program, and the National Science Foundation's Research Experience for Undergraduates program under Grant No. 1460917.