Keyword Search over Data Services: January Report

Vidmantas Zemleris

14th January 2013

1 From Keywords to Structured Queries

Mainly there are two methods to translating Keywords to Structured Queries: a heuristic rules ranking based exhaustively evaluating possible alternatives, and machine learning approach. The full sentence Question Answering (QA) approach is also possible, however the keyword-based methods (possibly with ideas from QA) shall work better knowing the constraints on the short project duration.

1.1 Exhaustive search & Heuristics (based on Keymantic[BDG+10])

Currently the prototype is based on a simple recursive exhaustive search. It starts with some matching techniques[1] to identify the schema terms or their values to which each keyword may correspond (entry points) estimating each "likelihood". Then these choices are combined with a number of heuristics obtaining a score of possible result, and these are ranked. It's flexible for defining the features deciding the rank, but currently it is hard to add feedback (especially the implicit one) and heuristics' parameters are hand-picked.

Heuristics: Calculating Entry points As an example, for query "configuration of Zmmg-13Jul2012-v1" (the numbers are estimated likelihoods): 'configuration' is matching schema terms: (1.0, u'config.name'), (0.66, u'lumi'), (0.57, u'dataset.name'), (0.54, u'status.name'), (0.51, u'monitor');

While 'Zmmg-13Jul2012-v1' matches as a values of these schema terms:
(0.7, 'dataset.name=*Zmmg-13Jul2012-v1*', (0.5, u'reco_status'), (0.5, u'tier.name')

false positive matche colored Magent

Heuristics: Combining entry points In addition to combining "likelihood" scores of entry points, a number of heuristics is applied to boost the final scores:

- promoting such configurations where nearby keywords refer to related schema terms (e.g. entity name and it's value)
- promoting such configurations that match most of the keywords;
- if requested entity and a filter condition is the same (small increase)
- data service constraints must be satisfied. alternative: identifying interpretations of high rank, that do not satisfy some constraint (a mandatory filter condition is missing), and informing user

[1] including a number of metrics for lexical and semantic word similarity and regular expressions defining allowed inputs for generating entry points (this could result in false positives with the likelihood slightly lower than of the true positives);
Limitation: Semantic Word Similarity is based on NLTK & Wordnet which currently works only on the same parts of speech. E.g. 'located at XX' can not be easily matched to 'site' (that means 'location' in our domain) automatically.

Performance [BDG+10] employed an performance optimization based on "Munkres/Hungarian" algorithm, it makes the search a bit more restrictive[2]. At the moment, we don't combine services[3], so the performance is OK even without this. Another possible optimization could be Dynamic Programming based on the assumption that interdependences exist only over k-nearby keywords reducing the search space (similar to the assumption on HMM, but allowing to any scoring/ranking heuristics).

1.2 Machine learning: HMM method (KEYRY[BGRV11])

Bergamaschi et al. approached the same problem through Hidden Markov Model (HMM) that allow taking into account the query logs or user's feedback. **Keywords that are near to each other are expected to represent interrelated concepts**, and HMM can efficiently model such kind of interdependences [BDG+12].

An HMM Models the keyword translation as probabilistic process (similarly as HMM-based speech recognition or part-of-speech-tagging) under the assumptions that:

1. the label (HMM state) of each keyword (HMM observable) depends only the keyword itself and on a fixed number of labels before/after it – this is mostly true for keyword queries

2. the probability of transitioning from label X to the next label Y depends only on last label(s) and the next label.

These probabilities form the basis of HMM model. They could be estimated using some of the heuristics as in Section 1.1 allowing to use this method even then there is no enough of learning data (More details on implementation are on the report draft).

Advantages of HMM: it can model keyword labeling quite intuitively and allows incorporating users feedback easily (at least positive feedback on what of proposed results were good). *It could work even then query log is not available yet (without training).*

Disadvantages: requires modeling each possible word resulting in large model (at least it seems so); training could work badly with low number of examples (however it is not mandatory). The implementation aspects of the method are not described suffiently clearly in the papers[BDG+12] (even though that's on Springer, IEEE), especially the learning process. The assumptions above too strict, but at least this is better than just modelling individual keywords without their inter-dependences.

Alternatives: CRF and Other Machine Learning Models:

Conditional Random Fields (CRFs) which are less restrictive than HMM, have been quite popular last years. They allow **combining** a number of arbitrary features/functions (based on any term of the input query and the labels before and after the label that is being decided [of even anywhere for more relaxed versions of CRF]) – all this is not allowed in an HMM.

I'm still not sure to which extent the manual initialization/bootstrap of the model is possible, however at least conversion from HMM into a CRF **is** possible. Not yet sure if it's quality of results could outperform the HMM or Exhaustive Search. Also training could be more complex/expensive for CRF. Finally, we may be subject to some limitations of libraries implementing HMM or CRF.

1.3 Question answering and Natural Language Processing

We are already using Semantic similarity based on Wordnet. Some of the other methods could be probably useful, e.g. Word Sense Disambiguation (infer exact meaning of a term based on terms nearby).

For processing full-sentence queries, one may consider giving larger weight to "important" words in the query (e.g. parse the query and take all Noun Phrases), as well as Question Focus extraction (what the question is asking for). Rule-based Relation-Extraction could be effective for decoding semantic meaning, but they need to be composed manually to be effective, which is resource consuming work. TODO:Watson also used automatic relation extraction, however I guess that needs large and good learning corpus.

[2] because they only consider true/false matchings for some terms, whereas in our case many of the matchings (of entry points) are susceptible to false positives (e.g. regular-expression based)

[3] that is not directly supported by data integration system

2 Methods for Feedback and Self-Improvement

These could include simply boosting the most popular queries (e.g. based on structured query logs), like/dislike feedback on particular result (a group of similar structured queries) as used SODA[Kla11][4]), and machine learning methods that could directly incorporate users feedback and reason on that. Explicit Feedback on entry point generation could be also useful (e.g. user choosing that given keyword must correspond to that entity or its attribute).

The machine learning that could be initialized with a number of heuristics even without query logs and improving with the time, looks quite promising, but more effort needed to: a) evaluate if that could at least reach the quality of the exhaustive search, and b) how the machine learning and feedback mechanism shall be designed.

3 Evaluation

In the provided paper on *IR Evaluation* I didn't find anything applicable much for us, except from general ideas that often there more interdependences influencing the evaluation than being modeled, but that's quite OK for directly evaluating the algorithms.

Possible evaluation strategies:

- A predefined set of keyword queries along with their correct interpretation (unit tests)
  - compare different approaches (e.g. pure heuristics, vs. HMM without learning)
- User's opinion (questionnaire):
  - what he prefers: structured queries, keyword queries or natural language queries
  - of the three, what gives the best results? does it make the querying easier?
  - user's background
- User's performance
  - is complex to evaluate. if we compared the time required to compose the same structured query:
    - there may be individual differences
    - if we asked the same user to compose both structured and keyword queries for *the same Information Need*, the keyword query would allow user to learn the schema, and that would influence his approach to structured queries (but that's still better than inverse sequence)
- Runtime results (Like/Dislike feedback; Links clicked) - lack of time for being useful.

Evaluation of incorporating User feedback into the results ranking is much more complex, so I'd be tempted to avoid it (at least for now). *Probably, additional evaluation results (such as Like/Dislike feedback) could be provided during the Defense, that is 3-5 weeks after the Report deadline.*

4 Reflections on Innovativeness & Usefulness of this work

Apart from the fact that it's going to be real world open-source implementation (in comparison to closed-source research prototypes) and targeting explicitly data services (Keymantic & KEYRY[BDG+10, BGRV11] were mainly designed for DB, but also mentioned web services), there's not so much new so far.

On the other hand, usage of Natural Language Processing (NLP) for better query interpretation and different Machine Learning (ML) approaches than applied before (such as CRF instead of HMM) for incorporating both implicit and explicit users feedback, could be something interesting. Still I would not be too much excited too early. Another problem is that performing a good evaluation of ML approaches could be hard as at the moment we do not have learning/evaluation data (gold standard).

[4] that is actually almost same as the "popularity" of an item; this could make more sense if implemented per user role, as information need greatly depend on that

5 Current Status

Done:

- Literature for: detailed look at HMM approach; NLP and Question Answering; some on Feedback and very little on CRFs
- +/- Structural sketch of Thesis Report

Future tasks:

- finish automatic extraction of schema and possible values given the meta-data and service definitions (includes querying services or gathering info during runtime)
- service statistics to be gathered (avg execution time of each service interface + std. deviation) - that can be easily computed online with minimal storage
- **improve keyword query processor**
  - incorporate feedback (on results; on keyword classification) : HMM? CRF?
  - allow both structured queries and keywords at the time
  - support for combination of structured input and keywords at the time
- evaluation

References

[BDG+10] Sonia Bergamaschi, Elton Domnori, Francesco Guerra, Mirko Orsini, Raquel Trillo Lado, and Yannis Velegrakis. Keymantic: semantic keyword-based searching in data integration systems. *Proc. VLDB Endow.*, 3(1-2):1637–1640, September 2010.

[BDG+12] Sonia Bergamaschi, Elton Domnori, Francesco Guerra, Silvia Rota, Raquel Trillo Lado, and Yannis Velegrakis. Understanding the semantics of keyword queries on relational data without accessing the instance. In Roberto Virgilio, Francesco Guerra, Yannis Velegrakis, M. J. Carey, and S. Ceri, editors, *Semantic Search over the Web*, Data-Centric Systems and Applications, pages 131–158. Springer Berlin Heidelberg, 2012.

[BGRV11] S. Bergamaschi, F. Guerra, S. Rota, and Y. Velegrakis. A hidden markov model approach to keyword-based search over relational databases. *Conceptual Modeling–ER 2011*, pages 411–420, 2011.

[Kla11] M. Klausmann. *User feedback integration-incremental improvement.* PhD thesis, Master's Thesis Nr. 31 ETH Zürich, 2011, 2011.