CUIS Team for NTCIR-13 AKG Task

Xinshi Lin
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Hong Kong
xslin@se.cuhk.edu.hk

Wai Lam
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Hong Kong
wlam@se.cuhk.edu.hk

Shubham Sharma
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India
shubhamsharma4060@gmail.com

ABSTRACT
This paper describes our approach for Actionable Knowledge Graph (AKG) task at NTCIR-13. Our ranking system scores each candidate property by combining semantic relevance to action and its document relevance in related entity text descriptions via a Dirichlet smoothing based language model. We employ supervised learning technique to improve performance by minimizing a simple position-sensitive loss function on our additional manually annotated training data from the dry run topics. Our best submission achieves NDCG@10 of 0.5753 and NDCG@20 of 0.7358 in the Actionable Knowledge Graph Generation (AKGG) subtask.

Keywords
NTCIR, Actionable Knowledge Graph, Property Ranking, Word Embedding, Dirichlet Smoothing, Supervised Learning

Team Name
CUIS

Subtasks
Actionable Knowledge Graph Generation

External Resources Used
English Wikipedia

1. INTRODUCTION
The objective of NTCIR-13 Actionable Knowledge Graph (AKG) Task [1] is to foster research on generating knowledge graphs that are optimized for facilitating users' actions (buying, booking, downloading, comparing, creating, etc.). The organizers set two subtasks: Action Mining (AM) subtask and Actionable Knowledge Graph Generation (AKGG) subtask.

Our CUIS team participated in the AKGG subtask, which requires participants to rank entity properties. The query (input) can be ambiguous as in realistic search queries, and participants need to return the ranked list of relevant entity properties to create an actionable knowledge graph. Actions in the test queries will be taken from the outcomes of the Action Mining (AM) Subtask. Properties to be returned will be those defined as attributes of the entity type in schema.org vocabulary.

Roughly speaking, a property consists of one or several words which describe a characteristic of an entity or an action. We assume that a word is more likely to be part of a property owned by an entity or an action if it occurs more frequently in related text descriptions of this entity or context of this action. Therefore, we model the relevance between a property and a query quadruplet (**Query**, **Entity**, **Types**, **Action**) by considering their semantic relevance to an action and document relevance to an entity.

One challenge for AKGG is its lack of annotated data. To overcome this problem, we manually annotated ten queries from the dry run topics additionally. Together with five samples provided on the official website of AKGG, these annotated data become a small training set for training a one-parameter learning model.

2. Our Approach
Our ranking system is composed of three main components as follows:

1. Preprocessing
   - Query preprocessing
   - Extracting candidate properties
2. Indexing
3. Ranking
   - Computation of semantic relevance
   - Computation of document relevance
   - Computation of relevance score

2.1 Query Preprocessing and Extracting Candidate Properties
For each query quadruplet (**Query**, **Entity**, **Types**, **Action**) from the input, a query string is generated by simply aggregating **Query** and **Action**. The string is then tokenized into a list of query terms after stop word removal and stemming.

We extract all attributes listed on the webpage http://schema.org/TYPE as candidate properties for this query where TYPE iterates over each type name in **Types**. Since the naming convention of properties in schema.org is lower camel case (e.g. “startTime”, “mainEntityOfPage”), we use Python’s re package to split each property into a list of terms with regular expression [a-zA-Z][^A-Z]*. Stop words are filtered out and remaining terms are stemmed.

Noting that a few properties in schema.org are partially or fully written in the form of an acronym (e.g. “hasPOS” is the acronym for “hasPointOfSale”, “sku” is the acronym for “StockKeeping Unit”). We manually change those properties to its original form.

2.2 Indexing

As main source of text descriptions for entities, an English Wikipedia dump were automatically indexed to build a database for computing document relevance.

A Wikipedia dump stores data in XML format. We can automatically extract articles with the help of genism, a Python package for Natural Language Processing [2]. The index contains two fields, "title" and "content", which stores the title of a Wikipedia article and its textual content respectively.

2.3 Ranking Entity Properties

Semantic Relevance

We score each property by considering its semantic relevance to the query. The relevance score is defined by the following formula:

$$score_1(q, a|C) = \sum_{t \in T_q} \log(\max\{g(t, t_a)\}), \forall\, t_a \in T_a$$

where T_q is the list of query terms, T_a is the list of attribute terms, and g is a similarity measure in the context of corpus C for query term t and attribute term t_a. For AKGG subtask, we set g to cosine similarity of word embedding representation of two terms.

Document Relevance

Besides semantic relevance, we also consider document relevance of a property given a query quadruplet, which is defined as logarithm of probability of property being relevant given the document related to the query. We apply a Dirichlet smoothing based language model [3] to model the relevance, which is defined by the following formulas:

$$score_2(q, a|C) = \sum_{t \in T_p} \log p(t|d_q)$$

$$p(t|d_q) = \frac{tf(t, d_q) + \mu p(t|C)}{|d_q| + \mu}$$

where d_q is the document related to the query q in the corpus C and $tf(t, d_q)$ is the raw frequency of the term t in the document. The prior parameter μ is set to the average document length in the whole collection of documents.

Computation of Relevance Score

We define the final relevance score as the weighted average of the previous mentioned relevance scores.

$$score(q, a) = \sum_{i=1}^{m} \lambda_i score_1(q, a|C_i) + \sum_{j=1}^{n} \gamma_j score_2(q, a|C'_j)$$

where C_i denotes the i-th corpus for computing semantic relevance, C'_j denotes the j-th corpus for computing document relevance. λ_i and γ_j are parameters such that $\sum_{i=1}^{m} \lambda_i + \sum_{i=1}^{n} \gamma_j = 1$.

We employ supervised learning technique to determine the unknown parameters, the value of model parameter $\boldsymbol{\theta} = (\lambda_1, \ldots, \lambda_m, \gamma_1, .., \gamma_n)$ was obtained by minimizing the following position-sensitive loss function:

$$\theta = \underset{\theta}{\boldsymbol{arg}\min} \sum_{q \in Q} Loss(q, L_q|\theta)$$

$$Loss(q, L_q|\boldsymbol{\theta}) = \frac{1}{|L_q|} \sum_{a \in L_q} \left(Rank(a|\boldsymbol{\theta}) - GTRank(a)\right)^2$$

where Q denotes the collection of all queries, L_q denotes the ranked list of properties according to their relevance score. $Rank(a|\boldsymbol{\theta})$ is the rank of property a returned by our ranking system given parameter $\boldsymbol{\theta}$ and $GTRank(a)$ is the actual rank of property a in the ground truth.

Due to the lack of annotated data, we only use English Wikipedia as the only corpus for computing both semantic relevance and document relevance. Then the model is therefore reduced into its one-parameter version since we have $\lambda_1 + \gamma_1 = 1$, which was trained by performing a grid search for the parameter λ_1 to optimize the corresponding loss function.

3. Experiments

3.1 Experiment Setup

We use the English Wikipedia dump on June 2017 as the main corpus. The CBOW based word embedding model is trained via gensim's word2vec module with parameter (size=400, window=5, min_count=5). Articles extracted from the Wikipedia dump were indexed using Pylucene. For a given entity in the query quadruplet, a related Wikipedia article is defined as the document whose title is the same as the entity name. If such article does not exist, we simply use Lucene's default BM25F search engine to retrieve the most relevant article in the title field given the entity name as the query.

As stated by the organizers, the returned properties from submitted runs were pooled and given to annotators (CrowdFlower workers). The annotators (at least three annotators per each result) had to choose among the following options:

L4: Perfect

L3: Excellent

L2: Good

L1: Fair

L0: Bad

Normalized discounted cumulative gain (NDCG) and normalized expected reciprocal rank (NERR) are used as the evaluation metric and only top 20 properties returned by system were pooled and evaluated.

3.2 Experiment Results

We submitted three runs. Run 1 consists of the results returned by our previously mentioned ranking system and remove those unidentified attributes such as "gtin8" or "gtin12". Run 2 is the unfiltered version of run 1. Run 3 was the bag-of-words version of Run 1, which removes the logarithm part in the computation of the semantic relevance and the document relevance and simply sums up the similarities or probabilities.

As shown in Table 1, the column "Runs" shows the name of each run. The column "NDCG@10" and "NDCG@20" report normalized discounted cumulative gain at top-10 and top-20 for each run respectively. The column "NERR@10" and "NERR@20" report normalized expected reciprocal rank at top-10 and top-20 for each run respectively.

3.3 Discussion

Having examined the results in Table 1, we found that Run 1 achieved best NDCG@10. It is improved by 1.3% compared to run 3. We also found that Run 3 obtained the best NERR@10 and NERR@20. It is improved by 1.8% compared to Run 1. Moreover, the results for each run are similar. This may be because our model does not take sequential information of queries into consideration even we model the relevance in the form of language model.

3.4 Further Analysis

We obtained ground truth for the formal run topics after the results had been released. We converted the ground truth into the standard TREC evaluation format and a subsequent parameter analysis was conducted using the previously mentioned one-parameter model for Run 1. As shown in Figure 1, the NDCG@10 and NDCG@20 remains almost unchanged when the parameter λ_1 is less than 0.3. There is a slight increase for the results when λ_1 is between 0.3 and 0.7. Significant increases in both NDCG@10 and NDCG@20 are observed when λ_1 is between 0.7 and 0.9 and the performance reaches a peak when λ_1 is around 0.9. The figure also shows extreme situation concerning λ_1 equals to 0 or 1, which imply that consideration of both semantic relevance and document relevance improves the overall performance of the ranking system.

We also studied some good cases (NDCG@10 is larger than 0.8) and bad cases (NDCG@10 is less than 0.2) from the results. Roughly speaking, our ranking system generates good estimation if the **Query** and **Action** are specific enough. Take the query quadruplet ("Volkswagen Passat", "Volkswagen Passat", [Thing,Product,Vehicle,Car], "remove the door panel") for an example. Our ranking system returns properties such as "productID", "brand", "material" and "vehicleConfiguration" because attribute terms of these properties occur frequently in the related entity description and the context of the action. On the other hand, our ranking system will generate biased estimation if the **Query** and **Action** are ambiguous. For example, our ranking system returns unreasonable properties such as "isPartOf", "hasPart" and "name" for the quadruplet ("On the road", "On the road", [Thing,CreativeWork,Book], "follow") because the system cannot recognize the phrase "On the road" in the query string as an entity about a film or a book.

4. Conclusions

We have presented the participation of our property ranking system in the NTCIR-13 AKG task. We develop a ranking system that scores each candidate property by combining its semantic relevance to query and action and its document relevance in related entity text descriptions. We use supervised learning technique to improve performance of our model. Experiments show that our system obtains reasonable results for the formal run topics.

One of the future work is to exploit more information in the knowledge base such as DBpedia for the relevance measure. We also consider investigating the effectiveness of weight for each score.

Table 1: Evaluation results for our submitted runs

Runs	NDCG @10	NDCG @20	NERR@10	NERR@20
Run 1	0.5753	0.7358	0.6329	0.6352
Run 2	0.5736	0.7349	0.6310	0.6333
Run 3	0.5684	0.7322	0.6443	0.6474

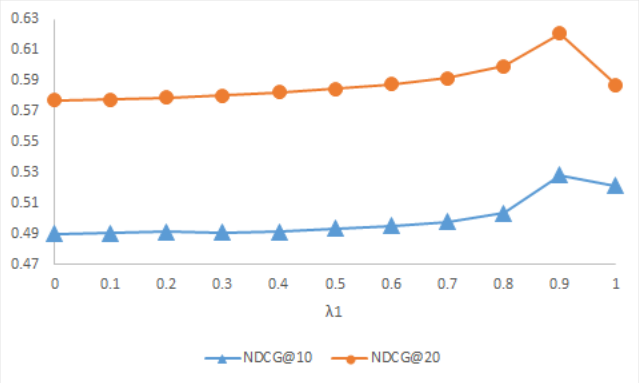

Figure 1: Effect of changing value of parameter λ_1 on the performance of the one-parameter ranking model.

5. REFERENCES

[1] R. Blanco, H. Joho, A. Jatowt, H. Yu, S. Yamamoto. Overview of ntcir-13 actionable knowledge graph (akg) task. In *Proceedings of the NTCIR-13 Conference*, 2017.

[2] R. Rehurek, and P. Sojka. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010.

[3] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 334-342, New York, NY, USA, 2001. ACM.