Overview of NTCIR-13 Actionable Knowledge Graph (AKG) Task

Roi Blanco
Amazon*
roiblan@amazon.com

Hideo Joho
Faculty of Library, Information and Media Science, University of Tsukuba, Japan
hideo@slis.tsukuba.ac.jp

Adam Jatowt
Graduate School of Informatics, Kyoto University, Japan
adam@dl.kuis.kyoto-u.ac.jp

Haitao Yu
Faculty of Library, Information, and Media Science, University of Tsukuba, Japan.
yuhaitao@slis.tsukuba.ac.jp

Shuhei Yamamoto†
Faculty of Library, Information and Media Science, University of Tsukuba
yamahei@ce.slis.tsukuba.ac.jp

ABSTRACT

This paper overviews NTCIR-13 Actionable Knowledge Graph (AKG) task. The task focuses on finding possible actions related to input entities and the relevant properties of such actions. AKG is composed of two subtasks: Action Mining (AM) and Actionable Knowledge Graph Generation (AKGG). Both subtasks are focused on English language. 9 runs have been submitted by 4 teams for the task. In this paper we describe both the subtasks, datasets, evaluation methods and the results of meta analyses.

Keywords

Actionable knowledge graph, evaluation

1. INTRODUCTION

Actionable Knowledge Graph (*AKG*)[1] task was held at the 13th NTCIR Workshop on Evaluation of Information Access Technologies (NTCIR-13)[2] [5] as one of pilot tasks. The task is an answer to the recent interest in knowledge graphs and an attempt to establish common grounds for designing and analyzing action-focused knowledge graphs as well as related technologies.

Knowledge graphs (KGs) have become an increasingly common and important component in search engine result pages (SERPs). Thanks to knowledge that can be harvested from the Web, search engines can directly return relevant information (alongside typical search results in the form of links to web pages and other media nuggets), saving user effort in reading and comprehending the returned results. It is now relatively common for search engines to respond to entity-centric user queries using data stored in KGs by delivering factoid type information about entities contained in search queries (e.g., birthplace of a celebrity, shop address with a pointer on a map). Since many users use search engines not only for acquiring information but also for completing certain actions and achieving some goals [2], generating readily actionable output should increase searcher' satisfaction.

Equipped with the collected information on the range of possible actions for a given entity, search engines could display actionable information that corresponds to the most probable underlying search intent behind user queries. Users could then directly act on such output data to more effectively and efficiently complete their desired actions. Direct links to services allowing the execution of such actions could be included as a means for improving the search experience. Given that on average 43% of search queries contain an entity [6], solutions for supporting entity-centric actions have high potential to facilitate search on the Web. Although there has been considerable research on entity-centric search [3, 6, 10], few proposals investigated the possibility of automatically deriving actions related to entities in search queries and the properties of these entities related to the actions.

Actionable Knowledge Graph (AKG) is considered as a specialized version of KG that contains data on the range of possible actions and their related information in relation to particular entity types and their instances. Automatically *constructing AKGs based on open information extraction* is then one important research problem. This is the objective of the first subtask of AKG Task: *Action Mining* (*AM*) subtask which requires returning relevant actions for input entities. The other one relates to the problem of *extracting relevant properties for facilitating users' actions* and is the basis of the second subtask: *Actionable Knowledge Graph Generation* (*AKGG*). Participants of the AKGG subtask need to submit relevant properties for the combination of entity and one of its actions.

The remainder of this overview paper is organized as follows. Section 2 introduces the two subtasks in AKG task as well as discusses the way in which results have been evaluated. Section 3 briefly describes the runs submitted by the participating teams. Section 4 provides the results. Finally, the paper is concluded in Section 5.

2. TASKS

Below we describe the content of both the subtasks. Table

*Work done prior to joining Amazon.
†Currently, NTT Service Evolution Laboratories
[1] http://ntcirakg.github.io/index.html
[2] http://research.nii.ac.jp/ntcir/index-en.html

Table 1: AKG important dates.

Date	Event
Aug 24, 2016	NTCIR-13 Kick-off Event
Oct 01, 2016	AM: Dry run topics + training data release
Nov 01, 2016	AM: Dry run submissions due
Dec 01, 2016	AM: Dry run results release
Dec 31, 2016	Task registration due for AM subtask
Jan 01, 2017	AM: Formal run topics release
Feb 01, 2017	AM: Formal run submissions due
Feb 15, 2017	AM: Formal run results release
Mar 15, 2017	AKGG: Dry run topics, sample data release
Apr 15, 2017	AKGG: Dry run submissions due
May 15, 2017	AKGG: Dry run results release
Jun 01, 2017	Task registration due for AKGG subtask
Jun 15, 2017	AKGG: Formal run topics release
Jul 15, 2017	AKGG: Formal run submissions due
Sep 1, 2017	AKGG: Formal run results release
Sep 1, 2017	Partial Overview paper release
Oct 1, 2017	Partcipants' papers due
Nov 1, 2017	Camera-ready due
Dec 5-8, 2017	NTCIR-13 Conference

1 outlines the important dates of AKG task.

2.1 Action Mining (AM)

Teams participating in AM subtask were asked to return actions relevant to an entity of a particular type. For a given entity type (e.g., Product) and instance entity (e.g., "Final Fantasy VIII"), up to 100 potential actions that can be taken in relation to the entity (e.g., "play on android", "buy new weapons", "learn junction system") should be returned. The format of each action contains verb (e.g., "play") and modifier[3] (e.g., "on Android"). Participants are allowed to submit up to three actions that share the same verb because semantics of actions can differ quite much depending on their modifiers.

The formal run dataset of AM task consisted of 200 test entities sampled from a set of query log datasets. To create the dataset, we grouped together the question answer and query data from Yahoo Webscope[4] and run an entity linker [1] over each question/query and selected the top-1 ranked entity. We then have selected entities based on their importance in the datasets estimated by the frequency of occurrence. Table 2 shows several examples of inputs that participants receive.

The actions are to be found by participants based on any data source they wish to use and any methodology. Several example relevant actions for the test instance marked by #1 in Tab. 2 are shown in Tab. 3).

Result evaluation was done in two assessment stages. First, verbs from the submitted actions were judged as for their relevance irrespectively of their modifiers. This was done using CrowdFlower[5] crowdsourcing platform based on results pooled from all the participating teams (depth of the pool was 20). The second level of assessment involved the full actions (verbs+modifiers) such that only the actions judged as the most relevant in the first assessment (L3 score, described later) were considered. Again, the selected results were pooled with the cut-off value equal to 20.

For completing both the assessments, CrowdFlower workers had to choose from the following options:

L3: Some people, organizations or other subjects definitely have taken or will take this action for the entity

L2: This action has been or will be definitely taken by the entity

L1: This action can be relevant for the entity

L0: There is no relevance of the action to the entity

For the performance testing the average values of nDCG@10, nDCG@20, nERR@10 and nERR@20 were used for both levels of assessment.

2.2 Actionable Knowledge Graph Generation (AKGG)

The second subtask is related to detecting descriptive data: entity predicates that are relevant for performing the action. Knowing such predicates is useful for search engines to provide direct interfaces for action completion. Table 5 shows example test instances consisting of a search query, an entity included in that query, the types of the entity, and action. Participants were asked to rank entity properties (as shown in the example in Table 4 which corresponds to the test instance #1 in Table 5) based on their relevance to the query.

The query (input) can be ambiguous as in realistic search queries, and participants need to return the ranked list of relevant entity properties. Properties to be ranked and returned were those defined as attributes of the entity type in the schema.org[6] vocabulary. Participants could submit up to three runs with 20 being the maximum number of ranked attributes.

Actions in the test queries were first taken from the outcomes of the Action Mining (AM) Subtask which were judged as relevant by CrowdFlower workers. Then they were manually selected by the task organizers. For the total of 200 queries, 100 had modifiers and 100 were missing any modifiers.

For evaluation we first pooled the results and then we used Crowdflower to generate assessments per each result. Annotators could select out of 5 assessments:

L4: Perfect

L3: Excellent

L2: Good

L1: Fair

L0: Bad

For the performance testing the average values of nDCG@10, nDCG@20, nERR@10 and nERR@20 were used.

3. PARTICIPATING SYSTEMS

This section provides the system descriptions of submitted runs.

3.1 AM

TLAB team [9] from National Institute of Informatics, Japan employs a probabilistic model based on Bayes rule

[3]The modifier's length is limited to 50 characters. Modifier can be also missing (NULL).

[4]https://webscope.sandbox.yahoo.com/

[5]www.crowdflower.com

[6]http://schema.org

#	Entity	Entity Type(s)	Wikipedia URL
1	Final Fantasy VIII	Product	https://en.wikipedia.org/wiki/Final_Fantasy_VIII
2	Yo-Yo Ma	Person	https://en.wikipedia.org/wiki/Yo-Yo_Ma
3	Zambia	Place	https://en.wikipedia.org/wiki/Zambia
4	York University	Organization	https://en.wikipedia.org/wiki/York_University

Table 2: Example test instances of AM subtask.

Verb	Object
play	on android
buy	new weapons
learn	junction system
watch	videos of other players
compare	with other games

Table 3: Example results for the input given in test instance #1 of Table 2.

Ranked Properties
Agent
ServiceType
Result
Location
StartTime

Table 4: Example results for the input given in test instance #1 of Table 5.

which estimate several distributions which model heterogeneous relationships between query and relevant actions. It utilizes diverse document genres (news articles, movie reviews, web pages, product reviews, medical data, Wikipedia pages) to respond to a high diversity of query types in the test set. The team submitted one run.

TUA1 team [4] from Tokushima University, Japan prepared three runs. They submit each query to different search engines in order to get relevant documents and then to construct the knowledge corpus for each query to extract syntactic information, i.e., the POS and syntactic tree structures from the knowledge corpora. They then collect both the TF-IDF and the embeddings for all words in the corpus. Important verbs are selected for each query by iteratively picking them from the verbs in the query's knowledge corpus based on an importance score (i.e., large TF-IDF), representativeness score in the knowledge corpus (i.e., high mean cosine similarity with semantic vectors of other verbs in the corpus), and diversity when compared to the already selected verbs (i.e., large semantic differences). The selection of the modifiers follows a similar iterative procedure in which, first, the representativeness and the diversity of the verb-related sentences is evaluated, and then modifiers are picked from the syntactic trees of the selected sentences. The runs differ in search strategies and search engines for building knowledge corpora.

ORG team was the the organiser team which provided two runs that differ in their parametrization. We used the following data as an input: Yahoo Webscope[7] (L4-L9) and the first Quora answers dataset[8]. We simply aggregated both datasets and employed the same pre-processing and parametrization.

The proposed model set up by the organizer's team estimates the probability of observing a binary variable A that indicates the presence of an action, given an input word sequence w drawn from a multinomial distribution of W parameters and an input entity E_t drawn from a multinomial of E parameters. It makes use of the part of speech (POS) $s \in$ pos to reflect the fact that some POS's are more likely to represent action forms (most notably verbs) than others ($s \sim multi(pos)$). Finally, we aggregate information from a document corpus D, where each document is represented by its entities E_d and its multinomial language model θ_d. The relationships between the variables are summarized in a graphical model in Figure 1.

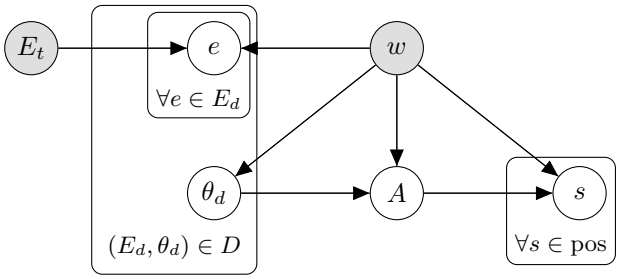

Figure 1: Bayes network for action mining

Given an input query comprised of a target entity E_t we want to rank and retrieve the top k actions $top(E_t; k)$ among all the potential word sequences w based on their conditional probability of $P(A|w, E_t)$.

$$top(E; k) = \underset{w}{\text{argmax-k}}\, P(A|w, E_t) \tag{1}$$

We estimate $P(A|w, E_t)$ using the model described in Figure 1 as follows.

$$P(A|w, E_t) = \prod_{s \in \text{pos}} p(\text{s}|w, A) \left(\sum_{(E_d, \theta_d) \in D} p(w|\theta_d) p(A|\theta_d, w) \prod_{e \in E_d} p(e|E_t, w) \right) \tag{2}$$

The model decomposes across two different components, one document-dependent (the right side of Eq. 2) and one document-independent (left). The right hand side is marginalized over all the documents in the corpus D, and it decomposes the similarity of the input query using two different similarities, using word sequences $p(\theta_d|w)$ and using entities $p(e|w)$. The formulation also incorporates the likelihood of a particular document language model to express an action $p(A|\theta_d, w)$ for

[7] https://webscope.sandbox.yahoo.com/

[8] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

#	Query	Entity	Entity Type(s)	Action
1	request funding	funding	thing, action	request funding
2	kyoto budget travel	kyoto	thing, place	visit a temple
3	consequences of flood	flood	thing, event	live in a flood area
4	how to use google maps	google maps	thing, intangible, service	create a google maps mashup

Table 5: Example test instances of AKGG subtask.

a word sequence. The left hand side of Equation 2 estimates the likelihood of the input sequence w being an action independent of the input entity itself, which we estimate using the part of speech of w and denote as $\prod_{s \in \text{pos}} p(\text{s}|w, A)$.

We now detail the different probabilities that need to be estimated from the data:

$\mathbf{p(e|E_t, w)}$: conditional probability for two entities, given an input query. This probability capture the similarity of e and E_t provided that they are sufficiently close to w. We use a simpler window filter so that if w and e are within the same text window[9] we let $\mathbf{p(e|E_t, w)} = \mathbf{p(e|E_t)}$ and $\mathbf{p(e|E_t, w)} = 0$ otherwise. $\mathbf{p(e|E_t)}$ is a component employed by algorithms that perform entity linking [1, 8] and can be approximated in many different ways, from the entities lexical similarity using their surface forms to their *semantic* similarity, mapping the entities into some vector space and computing their inner product or a sigmoid-normalized cosine distance. We remark that this approach ensures high recall, because we technically do not need to observe E_t in the data but only entities that are similar to E_t. In the case that a very high precision model is desirable, one can simply let this probability be 0 if $E_t \neq e$.

$\mathbf{p(A|\theta_d, w)}$: conditional probability of an action given the language model of the document and the input query. This can account for the fact that some textual items are more likely to contain relevant actions, like those beginning with *How, What* for instance. Interestingly, we could extend this estimation to account for the distance of the word w in a document with respect to the entity e (or an occurrence of a sequence in θ_d highly similar to w). In this case, this conditional would result in $p(A|\theta_d, w, e)$ and reflect the fact that actions and entities that are close to each other are more likely to be related. One simple approach could be to use the positions of w and e, respectively p_w and p_w and to apply a filter that assigns zero probability for large distances, say $|p_w - p_e| > \alpha$. This step would be crucial for an efficient estimation of the model (given that one would only need to compute the probability for windows of size $\leq \alpha$). In our case, the documents are short (e.g., one or two sentences) and we will disregard this estimation, although we note that if the model is to be applied on longer documents it would be required to somehow approximate this word-entity distance relationship with respect to action relevance.

$\mathbf{p(\theta_d|d)}$: estimate of the similarity of the word and the document language model which is commonplace in IR methods, which typically apply Bayes rule and approximate this probability by $p(w|\theta_d)p(\theta_d)$. In our case we will use Dirichlet priors smoothing [11] for unigrams, so that $\mathbf{p(w|\theta_d)} = \Pi_{q_i \in w} \mathbf{p(q_i|\theta_d)}$ and use a uniform prior for $p(\theta_d)$.

$\mathbf{p(\text{s}|w, A)}$: it represents confidence in the POS tag s of the sequence w if we observe an action A. This probability should weigh higher verbs and verb-like particles - in particular we set it to a small value for POS sequences that are not VB or VBP, so that only sequences those POS particles are considered as candidate actions.

In order to calculate entity to entity similarities we first chunk an input document into sentences[10] and then select the *main* entity from the sentence using an entity linker ([1]). We select the *max* scored entity from all the possible entities present in the sentence. Next, we make use of entity vectors generated from a 2016 June Wikipedia dump to calculate $p(e|E_t)$ using the normalized cosine distance between the vectors of e and E_t. To learn more about how these vectors are generated we further refer to [1]. The probability $p(\text{s}|w, A)$ requires to compute a distribution between part of speech taggers. In this case we employ again NLTK's POS tagger and filter tags that are not verbs. Given that many words can act as verbs in rare cases (either functionally or as an artifact of the POS tagger), we estimate $p(verb|w)$ as the ratio of the number of times w was detected to be a verb among all the occurrences of w in the corpus. This simple heuristic works well because we aim to mine actions globally and not in a particular sentence context. Finally, we experimented with different alternatives for estimating $p(A|\theta_d, w)$, however, the straightforward approach of letting all documents/sentences be equally likely to contain an action was satisfactorily simple and worked well in our dataset. We require a minimum entity to entity similarity η, a minimum word and language model similarity γ and a minimum likelihood of an action of being a verb β.

Algorithm Alg. 1 is used to compute all the actions for a set of input queries using one pass over the document corpus. The algorithm runs in two phases, first it aggregates information over a collection of documents (1-13) then sorts and filters out the non-actions from a ranked list (14-29). The aggregation procedure iterates over all the documents d and checks, for all the input query entities, whether the main entity E_d linked from the document and the query are sufficiently similar. This is done using the *embeddings* of the entities and the *sim* function (cosine). The sequence score for the action w is accumulated in a hash h and the potential action modifiers are extracted from d in Alg. 2. Finally, we extract the top scored verb-like word sequences for every query, and we proceed similarly with their associated action modifiers. This process ensures that some spurious actions are discarded, for instance, cases in which a function word acts as a verb or errors coming out of the POS tagger.

In order to detect action objects (modifiers), we filter sequences of words coming after the action form using their part of speech (see Alg. 2). This procedure is performed using a $filter(d, p)$ function which returns true if the word at position p in d is a punctuation sign (“.”), an empty space, a preposition (”PP”) or a coordinating conjunction (”CC”). The functions $is\_POS(w)$ return true if the corresponding part of speech of w matches POS; further $isPreposition$ matches prepositions and subordinating conjunctions (”IN”).

[9] For this work we took every question to comprise a window of text

[10] We use `http://www.nltk.org/`

Algorithm 1: mineActionsAndModifiers()

Data: $Q = \{E\}$ input query set, D document set, k actions to retrieve, t modifiers per verb to retrieve, η, γ, β thresholds

```
for E ∈ Q do
    φ(E) ← embedding(E_t)
    h(E) ← [ ]
    m(E) ← [ ]
for d = (E_d, θ_d) ∈ D do
    e ← max(E_d)
    for E ∈ Q do
        s ← sim(φ(E), embedding(e))
        if s > η then
            for w ∈ d do
                if p(θ_d|w) > γ and isVerb(w) then
                    h(E)[w] ← h(E)[w] + s
                    addModifiers(m(E, w), d)
for E ∈ Q do
    h(E) ← sort(h(E))
    c, i ← 1
    actions(E_t) ← [ ]
    while c < min(k, len(h(E))) do
        w ← h(E).key[i]
        if p(verb|w) > β then
            actions(E)[c] ← w
            c ← c + 1
            modifiers(E, w) ← [ ]
            j ← 1
            m(E, w) ← sort(m(E, w))
            while j < min(t, len(m(E, w))) do
                modifiers(E, w)[j] ← m(E, w)[j]
                j ← j + 1
        i ← i + 1
return actions, modifiers
```

To select the right parameters (η, γ, β) we employed a small development set and chose values for those thresholds accordingly. **Org-1** has a threshold for entity similarity of 0.7 and **Org-2** has a more aggressive threshold of 0.9.

3.2 AKGG

CUIS team [7] from The Chinese University of Hong Kong and Indian Institute of Technology submitted three runs. In general, they score 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. To further enhance the performance, they deploy supervised learning methods by minimizing a simple position-sensitive loss function, where an additional manually annotated training data from the dry run topics is used.

4. RESULTS

This section presents the results of meta analyses conducted for all submitted runs in AM and AKGG.

4.1 AM

Algorithm 2: addModifiers

Data: modifier list to append m, verb of the action w, document d

```
s ← []
p ← positionOf(w, d)
while p < len(d) and filter(d, p) do
    p ← p + 1
if len(d) > p then
    return m
out ← False
in ← isNoun(d, p) or isVerb(d, p)
while p < len(d) and not out do
    if !filter(d, p) then
        s.append(" " + d[p])
    out ← in and not (isNoun(d, p) or isVerb(d, p))
    in ← not in and (isNoun(d, p) or isVerb(d, p))
    p ← p + 1
if isPreposition(d, p − 1) then
    len(s) ← len(s) − len(d[p − 1])
m.append(s)
return m
```

	$nDCG$@10	$nDCG$@20	$nERR$@10	$nERR$@20
TLAB	0.6424	0.7549	0.6831	0.6854
TUA-1	0.5981	0.7699	0.6738	0.6785
TUA-2	0.6345	0.7978	0.7435	0.7467
TUA-3	0.5967	0.782	0.7546	0.7594
ORG-1	**0.7226**	**0.8353**	0.7868	0.7878
ORG-2	0.7175	0.8325	**0.7926**	**0.7935**

Table 6: Results for AM subtask when using verb only for evaluation (first level assessment). $N = 200$**.**

	$nDCG$@10	$nDCG$@20	$nERR$@10	$nERR$@20
TLAB	0.3577	0.4325	0.3272	0.3358
TUA-1	0.3032	0.447	0.3053	0.3248
TUA-2	0.3909	0.524	0.4047	0.4172
TUA-3	0.3016	0.4295	0.3031	0.3238
ORG-1	0.5071	0.6358	0.5446	0.5514
ORG-2	**0.5261**	**0.6507**	**0.571**	**0.5764**

Table 7: Results for AM subtask when using verb+modifier for evaluation (second level assessment). $N = 200$**.**

	$nDCG$@10	$nDCG$@20	$nERR$@10	$nERR$@20
CUIS-1	**0.5753**	**0.7358**	0.6329	0.6352
CUIS-2	0.5736	0.7349	0.631	0.6333
CUIS-3	0.5684	0.7322	**0.6443**	**0.6474**

Table 8: Results for AKGG subtask. $N = 200$**.**

Tab. 6 and Tab. 7 show the results of the 1st and the 2nd assessment (see Sec. 2.1 for description of evaluation) of AM subtask, respectively. The numbers are a mean score of 200 topics. As it can be seen, runs submitted by the organizers team **Org-1** and **Org-2** are characterized by the best results in both the stages of the assessments when using all the metrics. The range of the performance drop of the other systems when compared to the best performing method for the first level assessment is: 11%-17% for nDCG@10, 4%-10% for nDCG@20, 5%-15% for nERR@10 and 4%-15% for nERR@20. In the case of the second level assessment the ranges are as follows: 26%-43%, 20%-34%, 30%-47% and 28%-44%, respectively. While **ORG-2** outperforms **ORG-1** in the second level assessment as shown in Tab. 7, both have actually mixed results in the first level assessment.

4.2 AKGG

Tab. 8 shows the results for AKGG subtask. All the three runs submitted by CUIS team have similar performance. However, it is difficult to select the winning run since different runs are best according to different metrics (nDCG and nERR).

5. CONCLUSIONS

This paper presented the NTCIR-13 Actionable Knowledge Graph (AKG). Our test collections were designed to offer an opportunity to evaluate search technologies for supporting action-focused search in a structured way. Two subtasks were devised to advance action extraction and action's property detection. Both subtasks had a respectable number of queries and topics for system evaluation and user studies.

6. ACKNOWLEDGMENTS

This work was supported in part by MIC/SCOPE #171507010. The authors also thank the NTCIR project at NII.

7. REFERENCES

[1] R. Blanco, G. Ottaviano, and E. Meij. Fast and space-efficient entity linking for queries. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 179–188, New York, NY, USA, 2015. ACM.

[2] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, Sept. 2002.

[3] N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09, pages 1–12, New York, NY, USA, 2009. ACM.

[4] X. Kang, Y. Wu, and F. Ren. Tua1 at the ntcir-13 action mining task: sampling related actions from online searching. In *Proceedings of the 13th NTCIR Conference*, 2017.

[5] M. Kato and Y. Liu. Overview of ntcir-13. In *Proceedings of the 13th NTCIR Conference*, 2017.

[6] T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 589–598, New York, NY, USA, 2012. ACM.

[7] X. Lin, W. Lam, and S. Sharma. Cuis team for ntcir-13 akg task. In *Proceedings of the 13th NTCIR Conference*, 2017.

[8] A. Pappu, R. Blanco, Y. Mehdad, A. Stent, and K. Thadani. Lightweight multilingual entity extraction and linking. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 365–374, New York, NY, USA, 2017. ACM.

[9] M. M. Rahman and A. Takasu. Tua1 at the ntcir-13 action mining task: Sampling related actions from online searching. In *Proceedings of the 13th NTCIR Conference*, 2017.

[10] H. Yu and F. Ren. Role-explicit query identification and intent role annotation. In *Proceedings of the 21st CIKM*, pages 1163–1172, 2012.

[11] 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.