GatorDSR: University of Florida Knowledge Base Acceleration Notebook

Morteza Shahriari Nia[*]
m.shahriarinia@ufl.edu

Christan Grant, Yang Peng, Daisy Zhe Wang[†]
{cgrant, ypeng, daisyw}@cise.ufl.edu

Milenko Petrovic[‡]
mpetrovic@ihmc.us

September 27, 2013

Abstract

In this paper we will present the system design and algorithm adopted by the GatorDSR team, University of Florida to efficiently process TREC KBA 2013 — SSF track. Here we will describe the system evolution as well as the details of our algorithm to extract slot values for the given slot name. Scalability and efficiency in precision and recall are the major goals of this work, given the overly limited time limitation and available computational resources.

1 Introduction

One of the important challenges in maintaining the most popular free, web-based, collaborative, multilingual encyclopedia in the internet (Wikipedia) is to make sure its descriptions are upto date. It has been observed that there is a considerable time lag between the publication date of cited news articles and the date of an edit to Wikipedia (WP) creating the citation. In a case-study, the time lag for a sample of $\tilde{6}0$,000 web pages cited by WP articles in the Living_people category, The median time is over a year and the distribution has a long and heavy tail [1]. It is also noted that the majority of wikipedia entities have updates on their WP article much less frequently than their mention frequency.

Such stale entries are the norm in any large knowledge base (KB), because the number of humans maintaining the knowledge base is far fewer than the number of entities in the KB. Further, the number of mentions is much larger than the number of entities. This mismatch between human maintainers and the large stream of mentions to entities of interest is effectively a definition of a “large” KB [1]. So, knowledge-bases are important and we are working on accelerating populating them.

In this article we will describe the approach the GatorDSR team from the University of Florida in addressing the challenge at NISTs Text REtrieval Conference (TREC) — Knowledge Base Acceleration track 2013 In the first section we describe the system design and articulate its evolution during the whole process, challenges and needs that we have overcome. In the second section we will go through the details of the NLP techniques we took at streaming slot filling (SSF): given a slot name (entity + relation type) we fill in the slot value (relation value) on structured text documents. For similar approaches regarding last year's track you can refer to [2].

The task is that given certain Wikipedia or Twitter entities (wiki/twitter URLs) and certain relations of interest given in Table 1, extract as many triple relations as possible (hence, slot filling). This can be used to automatically populate knowledgebases such as free-base or DBPedia or even filling in the information boxes at Wikipedia. The corpus is in English and is a snapshot of part of the web containing document types described in Table 2. Each document is annotated using lingpipe and is called StreamItem, a bundle of

[*] Department of Electrical & Computer Engineering, University of Florida, Gainesville, Florida, USA

[†] Department of Computer & Information Science & Engineering, University of Florida, Gainesville, Florida, USA

[‡] Institute for Human and Machine Cognition, Ocala, Florida, USA

StreamItems are put together and serialized as Apache Thrift objects, then compressed using xz compression with LempelZivMarkov chain algorithm (LZMA2) and finally encrypted using GNU Privacy Guard (GPG) with RSA asymmetric keys. To have a sense of the scale of objects and compression as an example a 6mb gpg.xz files would become 45 mb thrift objects which can contain a couple of thousand StreamItems depending on their size. Some of the documents have null values for their annotation fields.

Since this is *streaming* slot filling, we are only interested in new slot values that were not substantiated earlier in the stream corpus, which ranges from October 2011 to February 2013. Below you can view some slot values and their types, the comprehensive description of which can be found at [3] and [4]. These examples demonstrate that out of an ordinary sentence in the general web, which parts are of interest for us and which should be avoided to be returned as slot values of a given entity. You can view that sometimes it can get fairly complex, so a good extraction scheme is essential. The evaluation metric is that the extractions will be submitted to KBA assessors and they will determine the overall performance for comparison.

Assessors were instructed to “use the wikipedia article to identify (disambiguate) the entity, and then imagine forgetting all info in the WP article and asking whether the text provides any information about the entity” [1]. Documents would be devided according to two metrics: 1) mentioning the entity or not, 2) relevance level to the entity:

- **Mention**
  - **Mention:** Document explicitly mentions target entity, such as full name, partial name, nickname, pseudonym, title, stage name.
  - **Zero-mention:** Document does not directly mention target. Could still be relevant, e.g. metonymic references like “this administration” – >“Obama”. See also synecdoche. A document could also be relevant to target entity through relation to entities mentioned in document – apply this test question: can I learn something from this document about target entity using whatever other information I have about entity?
- **Relevance**
  - **Garbage:** not relevant, e.g. spam.
  - **Neutral:** Not relevant, i.e. no info could be deduced about entity, e.g., entity name used in product name, or only pertains to community of target such that no information could be learned about entity, although you can see how an automatic algorithm might have thought it was relevant.
  - **Relevant:** Relates indirectly, e.g., tangential with substantive implications, or topics or events of likely impact on entity.
  - **Central:** Relates directly to target such that you would cite it in the WP article for this entity, e.g. entity is a central figure in topics/events.

The details of the metric for SSF will favor systems that most closely match the changes in the ground truth time line of slot values. This is done by searching for other documents that mentioned an entity and exactly matched the slot fill strings selected by the assessors.

Table 1: Ontology of Slot Name Categories

PER	**FAC**	**ORG**
Affiliate AssociateOf Contact_Meet_PlaceTime AwardsWon DateOfDeath CauseOfDeath Titles FounderOf EmployeeOf	Affiliate Contact_Meet_Entity	Affiliate TopMembers FoundedBy

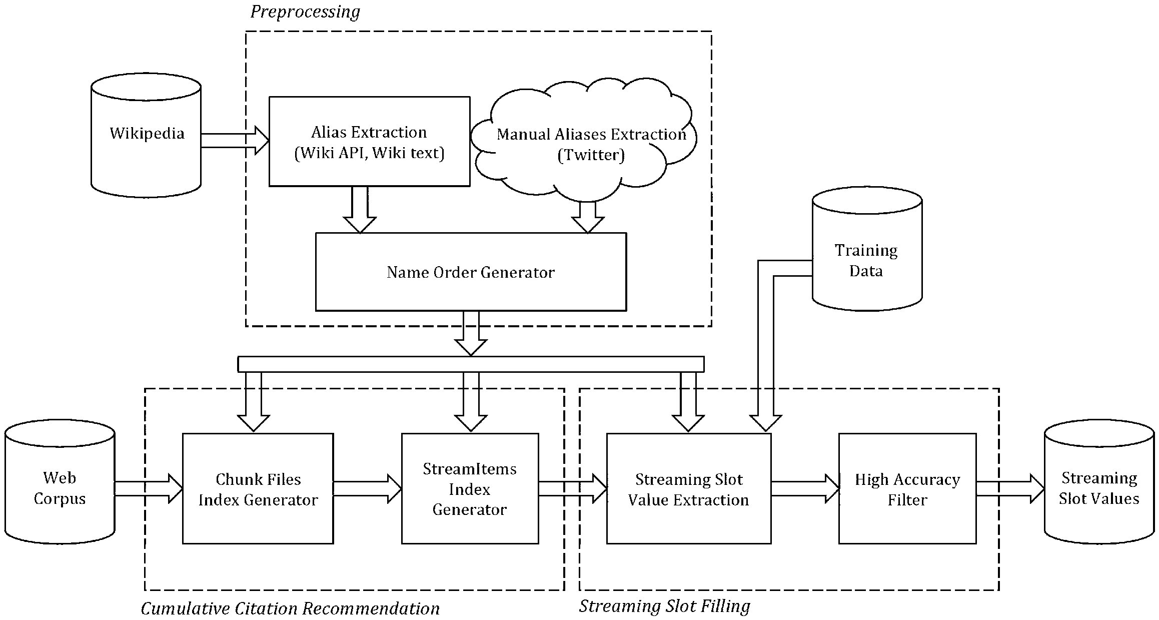

Figure 1: Pipeline Design

Example 1: "Matthew DeLorenzo and Josiah Vega, both 14 years old and students at Elysian Charter School, were honored Friday morning by C-SPAN and received $1,500 as well as an iPod Touch after winning a nationwide video contest."
target_id: http://en.wikipedia.org/wiki/Elysian_Charter_School
Affiliate: "Matthew DeLorenzo" Affiliate: "Josiah Vega"
NOT an affiliate: "C-SPAN" NOT an affiliate: "iPod Touch"
Example 2: "Veteran songwriters and performers Ben Mason, Jeff Severson and Jeff Smith will perform on Saturday, April 14 at 7:30 pm at Creative Cauldron at ArtSpace, 410 S. Maple Avenue."
target_id: http://en.wikipedia.org/wiki/Jeff_Severson
Affiliate: "Ben Mason" Affiliate: "Jeff Severson" Affiliate: "Jeff Smith"
NOT an Affiliate: "Creative Caldron" NOT an Affiliate: "Art Space"
Example 3: "Lt. Gov. Drew Wrigley and Robert Wefald, a retired North Dakota district judge and former state attorney general, unveiled the crest Friday during a ceremony at the North Dakota Capitol."
target_id: http://en.wikipedia.org/wiki/Hjemkomst_Center
Contact_Meet_PlaceTime: "Friday during a ceremony at the North Dakota Capitol"

Table 2: Document Chunks Distribution

# of Documents	**Document Type**
10988	arxiv (full text, abstracts in StreamItem.other_content)
34887	CLASSIFIED (spinn3r)
77674	FORUM (spinn3r)
12947	linking (reprocessed from kba-stream-corpus-2012, same stream_id)
141936	MAINSTREAM_NEWS (spinn3r)
4137	MEMETRACKER (spinn3r)
280629	news (reprocessed from kba-stream-corpus-2012, same stream_id)
6347	REVIEW (spinn3r)
688848	social (reprocessed from kba-stream-corpus-2012 plus extension, same stream_id)
740987	WEBLOG (spinn3r)

2 System Architecture

As shown in Figure 1, our system is built in a pipeline manner which gives it the advantage to run each section separately and allow stream processing without wasting resources being clogged by one operation while others are waiting. Below, we describe each component briefly and in Section 3 we will go through technical details.

2.1 Preprocessing

To begin, we need to have real world aliases of entities. TREC KBA prohibits Wikipedia entities to have any manual aliases being added and only allows automatic ways. We use Wikipedia API backlink references (redirects to a wiki entity) as aliases of the entity. We also use some techniques to extract some aliases from within the text of a wiki entity, which will be described later on. This whole process is referred to as **Alias Extraction (Wiki API, Wiki text)** Regarding Twitter entities this limit is lifted and participants are allowed to manually add aliases for them. Part of the reason is that Twitter environment is composed of so much informal and dialectic language which makes it extremely difficult to extract context out of tweets or entities descriptions in their profiles. This process of extracting aliases for Twitter entities is referred to as **Manual Alias Extraction (Twitter)**.

Once aliases are available we pass them through some rules of generating proper name orders which will produce various instances of writing a name. As a basic example Bill Gates can be written as Gates, Bill. This will allow the system to capture various notation forms of aliases. This part of the system is referred to as **Name Order Generator**.

2.2 Cumulative Citation Recommendation

The main goal of Cumulative Citation Recommendation is to have an aggregate list of documents that are worthy of being cited in a Wikipedia page. Regarding Cumulative Citation Recommendation we perform exact string matching and treat all the documents that mention an entity equally likely to be citable. For future iterations of the paper we have ideas on how to score documents based on their likelihood of citation worthiness but for this iterations we treat all mentioning documents equally likely. One of the reasons for this is that in former TREC KBA reports [1] there were observations of how non-mentioning documents have highly low chance of being citable in Wikipedia. So we take on that and ignore non-citing documents. Regarding mentioning but garbage, neutral or central documents we produce more slot values which can be later on filtered out in the high accuracy filter section.

Our pipeline of processing the corpus consists of a two layer indexing system referred to as **Chunk Files Index Generator** and **StreamItems Index Generator**. Chunk Files Index Generator will generate indexes of the chunk files that contain a mention of any of the desired entities. StreamItems Index Generator on the other hand will index StreamItems that contain a mention of a given entity respectively. This two level indexing will eliminate the need to process each and every ChunkFile/StreamItem for every entity.

2.3 Streaming Slot Filling

As described in the Introduction, the general purpose of slot filling is to extract proper values for relations of interest, some of which can be found in Table 1. This is called Stream Slot Filling due to the fact that it is assumed that data is being generated as time goes on and for each extraction we should only consider current or past data. In our schematic diagram we refer to this as **Streaming Slot Value Extraction**. Stream slot filling is done by pattern matching documents with manually produced patterns for slots of interest. The way we do this is by observing a sentence that has a mention of the entity or one of its coreferences. An anchor word in the sentence related to the slot name is located and we match either left or right of the anchor word for potential slot values.

2.4 Post Processing Algorithm

The SSF output of many extractions is noisy. The data contains duplicates and incorrect extractions. We can define rules to sanitize the output only using the information present in the SSF file. The file is processed in time order, in a tuple-at-a-time fashion to minimize the impact on accuracy. We define two classes of rules deduplication rules and inference rules. In our diagram we refer to this component as **High Accuracy Filter**.

3 Implementation

Our contribution is that we perform pattern matching that conforms to each slot value along with post-processings to eliminate noisy outputs. We extract aliases for entities from Wikipedia automatically both using API and using the actual page content, then apply pattern matching rules for slot value extraction.

3.1 Alias Generation

We use Wikipedia API to get some aliases automatically. This is done by retrieving backlink references (redirects of a wiki entity). Unfortunately this is not good enough and to enhance recall we need more aliases. To have better use of a wiki page we parse HTML DOM of the page, then use regular expressions to extract the bold phrases of the first paragraph as alias of the actual entity. Based on our observation this is a very accurate heuristic and provides us with lots of famous aliases of the entities. To consider other typical cases we consider some generic first name last name order swapping conventions such as Bill Gates → Gates, Bill.

3.2 Streaming Slot Value Extraction

In this system, we use pattern matching methods to find corresponding slot values for entities. This is done by observing the training data which is the first four months of the corpus(Oct 2011 - Feb 2012). The procedure is: 1) Find stream items that contain the entities by using alias names of the entities. 2) Inside these stream items, fetch the sentences which contain entities by using alias names and the coreference information provided by Lingpipe tags. 3) Use these sentences to match existing patterns. 4) When patterns matched, generate SSF results.

The format of the patterns is consist of five fields: 1) the type of the entity. 2) the slot name. 3) the pattern content. 4) the direction of the slot value to the entity. 5) the type of the slot values. When we match one pattern, we match all the fields except the third field. The type of slot values could be the entity type tagged by Lingpipe, noun phrase tagged by OpenNLP and the time phrases we hard-code. For these three kinds of patterns, we implement them in different ways accordingly.

Next we will explain the patterns with more details. Letss first look at one example of the first kind of patterns. For slot FounderOf, we have this pattern: <PER, FounderOf, founder, right, ORG>. PER means that the entity we are finding slot values for is a PER entity; FounderOf means this is a pattern for FounderOf slot. Founder is the anchor word we are trying to match in the sentence text; right means that we are going to the right part of the sentence to match the pattern and find the slot value; ORG means the slot value should be a ORG entity.

Here is an example for the second type: <PER, AwardsWon, awarded, right, NP>, NP meaning noun phrase. This pattern can be interpreted as that we are looking for a noun phrase after the awarded" since that noun phrase could possibly represent an award. Since titles and awards are usually not the Lingpipe entities, we use the OpenNLP noun phrase chunker to fetch the noun phrases.

Another example for the third type: <PER, DateOfDeath, died, right, last night>. In this pattern, last night" means we are looking for exactly the phrase "last night" to the right of "died". This pattern is inspired by the intuition that in news articles, people often mention that somebody died last night instead of mentioning the accurate date information and Lingpipe tends not to tag phrases like "last night" as a DATE entity.

To improve recall, we introduce generic patterns that can generate many results. But these patterns generate many errors, too. Thus there is a trade-off between the accuracy and recall. This is one drawback of using pattern matching method, meaning it is really hard to find good patterns with both high accuracy and recall.

For accuracy, we have found that there are three major kinds of errors: 1) wrong entities found; 2) wrong tags by the Lingpipe; 3) wrong results matched by the patterns. We solve the first problem by using the

Wikipedia or twitter information of the entities to get better alias names. And we use post- processing to reduce the second and third types of errors.

3.3 High Accuracy Filter

The SSF output of many extractions is noisy. The data contains duplicates and incorrect extractions. We can define rules to sanitize the output only using the information present in the SSF file. The file is processed in time order, in a tuple-at-a-time fashion to minimize the impact on accuracy. We define two classes of rules deduplication rules and inference rules.

The output contains many duplicate entries. As we read the list of extracted slots we create rules to define "duplicate". Duplicates can be present in a window of rows; we use a window size of 2 meaning we only be adjacent rows. Two rows are duplicates if they have the same exact extraction, or if the rows have the same slot name and a similar slot value or if the extracted sentence for a particular slot types come from the same sentence.

New slots can be deduced from existing slots by defining inference rules. For example, two slots for the task are "FounderOf" and "FoundedBy". A safe assumption is these slot names are biconditional logical connectives with the entities and slot values. Therefore, we can express a rule "X FounderOf Y" equals "Y FoundedBy X" where X and Y are single unique entities. Additionally, we found that the slot names "Contact_Meet_PlaceTime" could be inferred as "Contact_Meet_Entity" if the Entity was a FAC and the extracted sentence contained an additional ORG/FAC tag. We also remove erronious slots that have extractions that are several pages in length or tool small. Errors of extracting long sentences can typically be attributed to poor sentence parsing of web documents. We have some valid "small" extractions. For example a comma may separate a name and a title (e.g. "John, Professor at MIT"). But such extraction rules can be particularly noisy, so we check to see if the extracted values have good entity values.

The aim in post processing was to sanitize the results increasing the precision of the results by eliminating spurrious results.

4 Results

We had a single node cluster. The configuration of the server running the jobs is according to Table 3. All together, The total size of the data after XZ compression and GPG encryption is 4.5TB and just over 500M StreamItems [5]. Data is stored in date-hour directories: From 2011-20-05-00 (12am)) until 2013-02-13-23 (11pm) there are 11952 date-hour combinations. The source code of our system is stored as an open source project where enthusiasts can also contribute to [6], also the relevant discussion mailing list is accessible here [7].

Our results are as follows: Document extraction using query entity matching with aliases, sentence extraction using alias matching and co-reference. Slot extraction using patterns, NER tags and NP tags. 158,052 documents with query entities, 17885 unique extracted slot values for 8 slots and 139 entities, 4 slots and 31 entities missing.

On the performance of the first submission run we performed some random sampling via two processes, the results of which are as in Table 4. Our final submission has been elaborated in Tables 5 and 6. You can observe that only CauseOfDeath is missing from the set of found slot values.

Table 3: Benchmark Server Specifications

Model	Dell xxx 32 cores
OS	CentOS release 6.4 Final
Software Stack	GCC version 4.4.7, Java 1.7.0_25, Scala 2.9.2, SBT 0.12.3
RAM	64GB
Drives	2x2.7TB disks, 6Gbps, 7200RPM

Table 4: SSF Performance Measure on Initial Submission

	Correct	Incorrect Entity name	Incorrect Value
Sampling #1	55%	17%	27%
Sampling #2	54%	15%	31%

5 Discussion

We started off by trying to use the off-the-shelf tools for this task. Mainly we started by using Scala/Spark [8] to benefit from the parallelization performance there. Spark was not as efficient as expected and the overhead was much more than tolerable. We tried to use Scala parallelization itself to do the job, and unfortunately it did not satisfy our needs, there was excessive memory overhead on map reduce jobs. We migrated the core of the system to Java Parallelism APIs and that was not good enough either, we wanted better. So we built our own parallel system which in actual performance had the least of overhead, the least memory consumption and the most robust memory model which avoided unpredictable CPU stalls on garbage collections in processing the corpus. In what follows we will describe each step of this evolution in details. Figure 1 shows a schematic view of the system.

As mentioned before, when we evaluate the results of slot extraction, we find three kinds of problems for accuracy: 1) wrong entities found; 2) wrong tags by the Lingpipe; 3) wrong results matched by the patterns. We also have recall problems: 1) not enough good alias names to find all the entities. 2) not enough and powerful patterns to capture all the slot values.

We will use entity resolution methods and other advanced methods to improve the accuracy and recall of entity extraction part.

For slot extraction part, in order to improve the performance, what we need to are: 1) Using multi-class classifiers instead of pattern matching method to extract slot values in order to increase both recall and accuracy for slots “Affiliate”, “AssociateOf”, “FounderOf”, “EmployeeOf”, “FoundedBy”, “TopMembers”, “Contact_Meet_Entity” and so on. 2) For special slots, like “Titles”, “DateOfDeath”, “CauseOfDeath”, “AwardsWon”, using different kind of advanced methods, e.g. classifiers, matching methods. 3) Using other NLP tools or using classifiers to overcome the drawbacks of the LingPipes inaccurate tags. The first and second tasks are the most important tasks we need to do.

5.1 Shortcomings

The missing twitter entities are more than the 50% of the twitter entities we need to work on and this is a very very bad performance. One reason could be that those entities are not very famous. For example Brenda Weiler google search result is 860,000 documents our of billions of web documents. For our small portion of the web it might make sense. If you pay attention to the histogram of the entities found you’ll

Table 5: SSF Accuracy Measure on Submission Infer

	Correct	Incorrect Entity name	Incorrect Value
Affiliate	1%	95%	5%
AssociateOf	63.6%	9.1%	27.3%
AwardsWon	10%	10%	80%
CauseOfDeath	0%	0%	0%
Contact_Meet_Entity	21%	42%	37%
Contact_Meet_PlaceTime	5%	20%	85%
DateOfDeath	29.6%	71%	25%
EmployeeOf	5%	30%	65%
FoundedBy	62%	17%	21%
FounderOf	50%	0%	50%
Titles	55%	0%	45%
TopMembers	33%	17%	50%

Table 6: Recall Measure on Submission Infer

	Total instances of slot value found	# of entities covered by slot value
Affiliate	108598	80
AssociateOf	25278	106
AwardsWon	38	14
CauseOfDeath	0	0
Contact_Meet_Entity	191	8
Contact_Meet_PlaceTime	5974	109
DateOfDeath	87	14
EmployeeOf	75	16
FoundedBy	326	30
FounderOf	302	29
Titles	26823	118
TopMembers	314	26

note that more than half of the entities have appeared in less than 10 StreamItems. A good portion have appeared only once. Which the document does not necessarily have good information for us.

A theory is that we are falling behind because we are using the cleansed version of the corpus. We believe there has been a reason that TREC has included the actual HTML document as well as the cleansed version. This definitely will convey some information to us. If it was as easy as reading some clean text they wouldn't bother including so much data for teams to be useless. So we guess is that we are missing some information from not using the actual document. And, we are looking for tokens with entity value set which will depend us dramatically on the accuracy of lingpipe, which is a fast algorithm but is not as good as other NLP tools can be e.g. Stanford NLP.

6 Conclusions

We experimented through different tools and approaches to best process the massive amounts of data on the platform that we had available to us. We generate aliases for wikipedia entities using Wiki API and extract some aliases from wikipedia pages text itself. On twitter entities we extract aliases manually as it is part of the rule of the KBA track. We process documents that mention entities for slot value extraction. Slot values are determined using pattern matching over coreferences of entities in sentences. Finally post processing will filter, cleanup and infers some new slot values to enhance recall and accuracy.

We noticed that some tools that claim to be performant for using the hardware capabilities at hand sometimes don't really work as claimed and you should not always rely on one without a thorough A/B testing of performance which we ended up in generating our in-house system for processing the corpus and generating the index. Furthermore, on extracting slot values, pattern matching might not be the best options but definitely can produce some good results at hand. We have plans on generating classifiers for slot value extraction purposes. Entity resolution on the other hand was a topic we spent sometime on but could not get to stable grounds for it. Entity resolution will distinguish between entities of the same name but different contexts. Further improvements on this component of the system are required.

Acknowledgements

Christan Grant is funded by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-0802270.

References

[1] D.A. Roberts F. Niu C. Zhang C. R I. Soboroff J.R. Frank, M. KleimanWeiner. Building an entity-centric stream filtering test collection for trec 2012. *The Twenty-First Text REtrieval Conference (TREC 2012) Proceedings*, (21), 2012.

[2] Heng Ji and Ralph Grishman. Knowledge base population: Successful approaches and challenges. In *ACL*, pages 1148–1158, 2011.

[3] Tac kbp slots. http://www.nist.gov/tac/2012/KBP/task_guidelines/TAC_KBP_Slots_V2.4.pdf.

[4] Ace (automatic content extraction) english annotation guidelines for events. http://projects.ldc.upenn.edu/ace/docs/English-Events-Guidelines_v5.4.3.pdf.

[5] Trec kba stream corpora. http://aws-publicdatasets.s3.amazonaws.com/trec/kba/index.html.

[6] Gatordsr opensource project. https://github.com/cegme/gatordsr.

[7] Gatordsr mailing list. https://groups.google.com/forum/#!forum/gatordsr.

[8] Apache spark. http://spark.incubator.apache.org/.