# TCAR at TAC-KBP 2009

P. Schone, A. Goldschen, C. Langley, S. Lewis, B. Onyshkevych, R. Cutts[§],
B. Dawson[‡], J. MacBride[†], G. Matrangola[*], C. McDonough[※], C. Pfeifer[‡], M. Ursiak[‡]

U.S. Department of Defense
Ft. George G. Meade, MD 20755-6000

## ABSTRACT

The TCAR team developed multiple systems in just a matter of weeks for both participating in the TAC-KBP evaluation under the entity linking and the slot filling paradigms. TCAR's entity linking processes consisted of one that leveraged information retrieval and a separate that sought to appeal to extraction-based techniques. With regard to slot filling, since the TAC evaluation was touted as a task that would appeal to both the question answering and content extraction communities, we opted to build two systems where the first made use of our question answering system and the latter exploited relation-finding from a number of our content extraction systems. We here provide detailed descriptions of our systems and their performance at TAC-KBP.

## 1. INTRODUCTION

Automatic knowledge base population (KBP) is a challenging problem which advances the state of the art in language processing while fusing the efforts from multiple communities. As represented by the TAC-KBP organizers, KBP draws heavily upon techniques that have been studied and analyzed through previous NIST evaluations, such as automatic content extraction (ACE), automatic question answering (QA), and information retrieval (IR). Given our interest in each of these areas, and particularly having a serious desire to help foster KBP efforts, we elected to participate in this year's evaluation. Our goal in participation was primarily to spend several weeks of research experimenting with KBP in a number of different directions and gain serious experience in the difficulties and pitfalls of the field.

[§] Henggeler Computer Consultants, Columbia, MD
[‡] MITRE, Hanover, MD
[†] BBN, Columbia, MD
[*] SRA International, RABA Center, Columbia, MD
[※] Northrop-Grumman, Columbia, MD

We are keenly interested in content extraction capabilities, so it took a primary role for us in each task where we participated. Yet since the claim had been that KBP draws upon ACE, QA, and IR technologies, we likewise desired to study each of these capabilities and discover the kinds of KBP performance that could be realized by using such tools. Thus, for entity linking, we chose to develop two distinct systems with one built largely on IR principles and a separate one which sought to leverage content extraction. In terms of slot filling, we likewise created two systems--one based on our moderately-performing QA system and a separate one that would try to leverage content extraction.

We recognized, based on past experience, that many participant sites would have interest in leveraging the Internet and versions of Wikipedia itself as a means of improving their performance. Yet we have a stronger interest in determining the information that can actually be drawn from a set of documents provided. Hence, we intentionally disregarded use of the Web for the purposes of our research and instead focused our efforts on the core systems.

In this paper, we will describe the systems that we prototyped in this effort and we will discuss the merits and weaknesses of each of these processes. We will also comment on the overall performance of each system. We had, and will identify herein, a number of performance problems that are consistent with pilot-year experiences for many evaluations. Yet more importantly, we mention limitations to our systems which are actual weaknesses in the algorithms themselves. Lastly, we will indicate some of our future directions in these areas.

## 2. CREATING TASK RESOURCES

Before providing descriptions of the systems, it is beneficial to first provide the reader with a explanation of the resources which were developed and exploited to support various tasks. Specifically, we made use of tools for content extraction, topic tagging, information retrieval, and question answering, and we also created auxiliary resources for specific tasks. Each of these resources are described here, and each was made available to potentially any of our core entity linking or slot filling systems.

## 2.1. Extraction-related Capabilities

As was mentioned, one of our core research areas is content extraction. Correspondingly, we made use of, and in some cases expanded, a number of extraction resources identified here.

### *2.1.1. PhoenixIDF*

For entity tagging, we made use of our hybrid entity tagger, PhoenixIDF. This tagger uses the statistical substrate of an older version of BBN's Identifinder [1] but its language analytics has been replaced with a new TCAR-provided substrate of language-universal processing. It is a hybrid system because it also augments the statistical processes with rule-based processing to enhance overall performance.

PhoenixIDF's inventory of entity tags are heavily based on the general tag sets defined by the ACE program [2]. Perhaps one exception that should be mentioned here is that PhoenixIDF has separated adjectival tags describing nationality (such as American, Iraqi, etc.) from nominal person references of ACE. For the purposes of this evaluation, PhoenixIDF also was extended to incorporate two new types of tags: the "Occupation" and the "Age" classes. Since origin, occupation, and age would all be of interest for PERSON entities, these new entity classes are needed additions.

### *2.1.2. CARDS*

TCAR also has created a rule-based relation finding system called "CARDS" (Coreference and Relation Detection System). CARDS uses hand-crafted templates to identify lexical patterns that express relations. The templates use information from all downstream processes: tokenization, sentence segmentation, part of speech tagging, named entity recognition (NER), nominal mention recognition, coreference resolution and word lexicons. Several new relations were built with CARDS for this evaluation like Person-died_of-Cause_Of_Death and Person-charged_with-Charge.

### *2.1.3. LCC's CiceroCustom*

We made use of LCC's CiceroCustom [3] trainable extraction suite for some entity tagging and especially for relation-finding. Two TAC Person slots, “cause of death” and “charges”, required the development of entity-tagging classes other than those available in PhoenixIDF. These classes were therefore developed using Cicero Custom.

### *2.1.4. SERIF*

One of our major tools in this effort was the BBN SERIF system [4]. At past ACE evaluations, SERIF was consistently a top-performing system. We were able to leverage that system to produce relations and coreference analysis as well as to obtain high-quality constituency parses. We worked with BBN to get SERIF to be able to incorporate and pass through PhoenixIDF's entity-tagged input, and to make SERIF better able to process Wiki documents.

## 2.2. Information Retrieval Tools

Major components for both entity linking and slot filling were based on information retrieval techniques. We used commonly-available resources for these processes. In particular, three of our four systems leveraged information retrieval from the CMU/UMass Lemur Toolkit [5]. The fourth of our systems took advantage of the Apache Lucene [6]. In each of these cases, we made no changes to the base systems themselves and treated each as a black box.

## 2.3. Topic/Concept Taggers

Our hope was that if we could topically annotate the raw TAC-KBP documents, and perhaps the knowledge base nodes themselves, we would likely get better entity disambiguation. To explore this possibility, we employed two different kinds of topical or concept taggers that are mentioned here.

### *2.3.1. Semantic Forests*

In the 1990s, we developed a high-speed topic tagger called Semantic Forests that leveraged statistics and machine readable dictionaries (MRD) to produce topical lists. This resource was demonstrated as a mechanism for information retrieval at past TRECs and is described in greater detail in those contexts [7].

For this evaluation, we extended Semantic Forests' breadth of "knowledge." Semantic Forests' original MRD in English previously incorporated about 65,000 terms, yet none of these included more modern terminology. Assuming TAC-KBP would require a system with more up-to-date information, we expanded the MRD by incorporating freely-available knowledge in Wiktionary and Wikipedia. We extracted all of the new terms available in a mid-2008 Wiktionary and morphed these into dictionary entries of the kind desired by Semantic Forests. We then extracted titles with first sentences from the March 2008 Wikipedia dump as a means of creating yet other dictionary entries. When completed, the new Semantic Forest dictionary consisted of approximately 2.2 million dictionary entries. Semantic Forests was used to tag the entire raw document collection.

*2.3.2. Zymurgy Concepts*

We also made use of our Zymurgy concept analysis technology. This system produces document characterizations in terms of ontology concepts that are explicitly or implicitly referred to in the document. Zymurgy's ontology is an extension of the OMEGA ontology developed by USC/ISI [8]. Zymurgy also has the potential of concept-based clustering. For the purposes of this evaluation, Zymurgy was used to annotate a portion of the raw document collection as well as some raw text portions of the knowledge base, and provide some content clustering based on these automatic annotations.

### 2.4. Question Answering Tools

Since question answering bears a strong resemblance to slot filling, another one of the resources we used was our question answering system, QACTIS. This system was described in detail at past TREC evaluations [9]. Since modifications of QACTIS were a significant effort for slot filling, more detailed of these modifications will be provided later.

### 2.5. Auxiliary Resources

In addition to resources that came in the form of analytics and their corresponding output, we also rapidly developed two resources that would be of benefit to the entity linking task itself.

*2.5.1 Evaluation resources*

Before the TAC-KBP coordinator released an entity-with-link development set, we were concerned that we would not be able to adequately test our systems without some truth data. Therefore, we ran PhoenixIDF on the entire collection and identified all entities that occurred between 5 and 100 times. Starting at those that occurred 100 times and working down, we used various models of our information retrieval engines to look for, vet, and create an entity-with-link development set of over 300 elements. This collection attempted to balance entities that had no link with those that had links, and there was very little repetition of real-world entities in this collection. This development set proved very helpful in creating our systems, though, as we discovered at test time, it had significantly fewer organizations (and especially, acronyms) than were used in the final test collection. When the TAC coordinator later provided about 300 new entities and links, this provided us with a total of over 600 development links.

*2.5.2. Name Aliases*

We also developed a rudimentary name-equivalence list for persons for helping to identify English name variants. This resource was developed first by identifying those names from speech pronunciation dictionaries that were spelled differently but have the same pronunciation (such as "Jon" and "John.") It was then augmented with common nicknames.

*2.5.3. Relation Patterns*

For the benefit of slot filling, we also devised a process to help us recognize textual contexts which might give rise to legitimate slot fills. We used the KB and the 1.2 million documents to help in this endeavor. Assume $X$ is the title of a KB node, $S$ is the slot to fill, and $Y$ fills $X$'s slot $S$ (i.e., $Y=S(X)$). We searched the document collection for the co-occurrence of $X$ and $Y$ within a narrow window of each other and created a repository of these that we collated by $S$. The application of these will be described later.

## 3. ENTITY LINKING SYSTEM

Figure 1 depicts the general TCAR entity linking system architecture for the TAC competition. The task was to develop a correlation algorithm that links a given entity from a given document to a specific Wikipedia page from a set of 860,000 Wikipedia pages. We developed two basic types of correlation algorithms: an information retrieval (IR) based system and an information extraction based system.

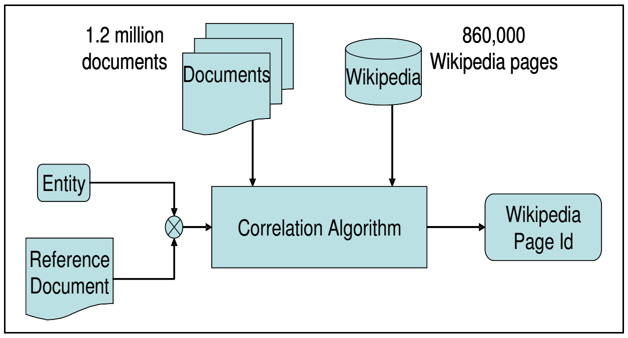

***Figure 1*** *TAC entity linking system consists of correlation algorithms that link a given entity from a reference document to a specific Wikipedia page.*

## 3.1 Information Retrieval Based Linking

### *3.1.1. Resource Usage*

Figure 2 illustrates various resources each of the entity linking systems used. The document repository contains the following for each document: entities, relations, within-document co-references, and parses from the various extractors. Also, the repository contains semantics of each document using semantic topics/concepts from Semantic Forests and Zymurgy. The Wikipedia repository contains similarly-extracted information from the text associated with each Wikipedia page. We wrote algorithms to determine the Wikipedia node's title and whether the node represented a person, organization, or geopolitical entity. The relation names from the document and Wikipedia repositories were mapped to one of the TAC relations.

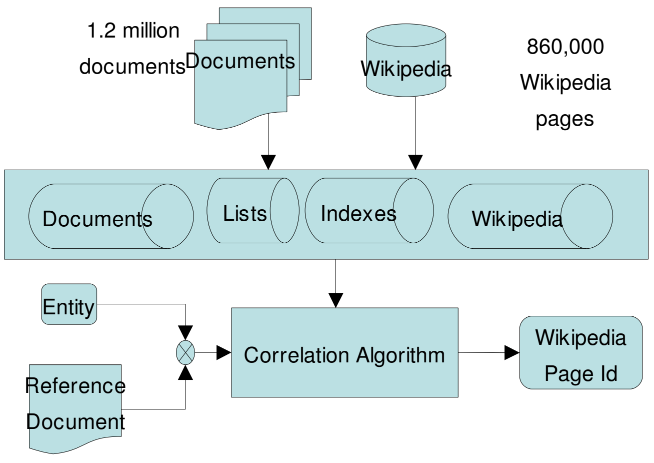

***Figure 2*** *TAC entity linking system consists of context information about documents, Wiki pages, lists, and indexes.*

We also leveraged the previously-mentioned list of name variants and augmented it with nicknames and acronyms which originated from within-document co-reference chains and their various name parts. We also wrote a script to strip diacritics and accents from foreign names. Although we likewise converted the documents into overlapping *n*-grams to be used with *n*-gram based queries, we were not able to fully integrate these resources for our competition so we will not further describe them here.

Our IR-based system used multiple indexes of the document and Wikipedia repositories, each built using the Lemur information retrieval system. Our "*Main*" index is an index of all 1.2 million documents in the document repository, with the stop words removed. We built the remaining Lemur indexes from the 860,000 Wikipedia pages. Our "*Wiki Titles*" indexes the Wikipedia page titles. The "*Wiki Titles and Name Slots*" index is similar, but contains values from Wikipedia page *alternate name* slots. The "*Wiki KB*" index contains information from the Wikipedia title, Wikipedia slots, and Wikipedia text, with stop words removed. Our "*JOINT*" index contains information from both the document repository (*Main* index) and from the Wikipedia repository ("*Wiki KB*" index).

### *3.1.2. Overall System Description*

The IR-based linking system uses information form Figure 2 to see if either the Wikipedia node that corresponds to the given entity or that no node corresponds to the given entity. The system consists of filtering and analysis phases, as Figure 3 depicts. The filtering phase consists of generating a ranked list of fifteen out of 860,000 potential Wikipedia nodes that could link to the entity in the reference document; and, the analysis phase consists of selecting the particular Wikipedia from the set of fifteen Wikipedia nodes.

Our filter phase consists of identifying and re-ranking the top fifteen Wikipedia KB nodes most likely to match the query entity from among the entire set of 860,000 Wikipedia nodes. The first step in this process is to retrieve a set of fifteen candidate nodes based on how likely the query entity name matched the Wikipedia node titles through a Lemur search of the "*Wiki Titles*" index. Bear in mind that the query entity name might differ significantly from the Wiki nodes due to the inclusion of name variants, appostive information, acronyms, and abbreviations. We further reduce our retrieved set of candidate nodes by removing all Wikipedia pages that were not about a person, organization, or geopolitical entity.

Our second filtering step expands the initial query beyond the entity name; the query includes semantic information about the reference document. This semantic information uses output from Semantic Forests and Zymurgy to respectively identify semantic concepts from the reference document. The system queries the *JOINT* index using the entity name and semantic concepts from the reference document. The system then returns the top fifteen documents and Wikipedia nodes from both the document and Wikipedia repositories. The system again removes Wikipedia nodes from this set that are not about a person, organization, or geopolitical entity. The *JOINT* index allows for the system to return NIL if no Wikipedia nodes emerge in the set of fifteen; and, the *JOINT* index query returns documents similar to both the reference documents and reference name. Although we submitted a system that uses semantic information from the reference document using both Semantic Forests and Zymurgy algorithms, we also submitted a system that uses only the semantic information from reference document using only the Semantic Forests algorithm. Either way, recall was statistically indistinguishable between using Semantic Forests or Zymurgy; but in each case, recall was 1 to 2% better with semantic terms than without. However, using Zymurgy's clustering capability between both the reference document and candidate wiki nodes, we were able to advance in the ranking those candidate pages that were more similar in topic and in the inventory of explicit and implicit concepts.

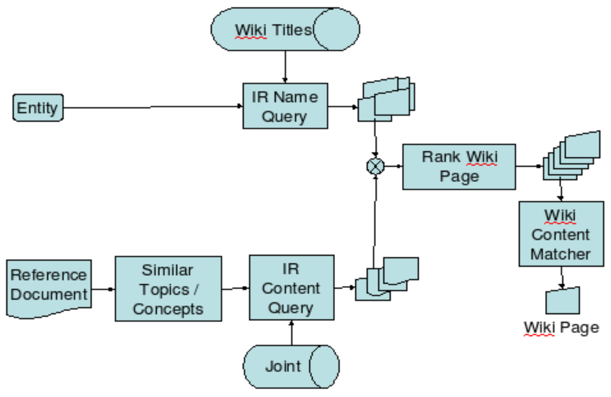

***Figure 3****: TAC system components for Information Retrieval based linking approach.*

The system next re-ranks the initial set of Wikipedia nodes provided by the "*Wiki Titles*" query based on rankings from the *Joint* corpus query. The system only considers Wikipedia pages that occur in both the *Wiki Title* and *JOINT* index query sets. If there are no Wikipedia pages in common, the system just returns the *Wiki Title* query set.

The analysis phase consists of comparing the entity name to the ranked list of at most fifteen Wikipedia nodes from the filtering phase. The Wikipedia nodes are processed in ranked order using the Wikipedia content matcher. The matcher first creates a set of names from the given Wikipedia node title, slot, or extracted from the text. Generally, the Wikipedia node is returned, if one of these names matches the entity name according to the name variant list mentioned earlier. However, if the entity name matches an extracted name from the text body and this extracted name links to another Wikipedia node, that other Wikipedia node is returned instead. The system returns NIL when no names associated with the set of Wikipedia nodes match the entity name.

### 3.2 Linking from Extraction-Matching

Our extraction-based linking system used a four step process to link entities to their knowledge base (KB) entries. These system leveraged entity and relation extraction on the corpus, indexed the knowledge base, searched the KB index, and analyzed the search results.

#### *3.2.1. Corpus Content Extraction*

Our system made use of the entity extraction of PhoenixIDF, and the relation and coreference extractions of SERIF and CARDS.

#### *3.2.2. Indexing the Knowledge Base*

We uses Apache Lucene to index the knowledge base ahead of time. The KB ID was used as the document “name” and the fields included the entity name, and other attributes (or facts).

#### *3.2.3. Searching the KB Index*

The query provided by the TAC contest contains an entity (*e*) and exemplar document (*d*). The query is built using a two step process. First, combining various forms of *e*, and second finding relations (*r*) of *e* within *d*. Finally, the query is run against the KB index. The details of which are described below.

The results of extraction of *d* were searched for occurrences of *e* (a “fuzzy” match with an edit distance of 2 was used to account for variations in spelling). Then the extraction were searched for coreferences of *e* (excluding pronouns) yielding set of strings that are considered to be aliases of *e*. A set of attributes (*a*) was then collected using any extracted relation to *e* (or a coreference of *e*) within *d*.

Finally, a Lucene query was created. The query searched for *e*, and aliases of *e* in the name field and for any attributes that match *a*. “Fuzzy” edit distance matches were used for name searches to account for variations in spelling.

#### *3.2.4. Analyzing the Search Results*

Lucene allows search criteria to be weighted and each result receives a hit score. Heavier weight is given to matches that matched *e* and aliases more exactly. The attributes were used to break ties where there are similar names for the same value of *e* or where no *e* is found. If the Lucene results did not include any values with a hit score greater than the threshold of 1.8 then it was assumed that that entity did not exist in the KB and the response should be null.

Advantages to this approach were that it was relatively easy to create using the available tools we had readily available. This solution was also created in with about 30 hours of design and development effort and is scalable.

## 4. SLOT FILLING SYSTEM

As mentioned previously, we were very interested in exercising various aspects of human language technology as we developed our KBP system. For slot filling, whose goal was to identify new KB information as distilled from the 1.2 million documents, we chose to build two types of systems: one that would be based on question answering and another that would be built using content extraction engines. By building both, we would get a sense of what both the ACE and the QA communities could bring to bear on the general slot-filling task and, likewise, what level of effort would be required.

### 4.1. Question-answering Based Slot filling

Our first methodology for attempting to fill slots was based on the premise that slot-filling as a process is very similar to process of automatically answering questions. We therefore built one slot-filling algorithm using QACTIS, our question answerer. We assumed that the overall TAC evaluation would likely be heavily dominated by NIL answers (and, as discovered after the evaluation, this turned out to be quite true). QACTIS, on the other hand, tries to produce a non-NIL answer for every question that is posed to it. Therefore, we recognized that we would likely significantly overgenerate slot fills, but we were very interested in seeing the results that would come about through such a mechanism. Our hope was that this version of a slot-filler would favor recall even if not high precision.

In previous TREC-QA competitions, participants were given a target entity or event, *E*, and it was expected that every subsequent question that would be asked would be about *E*. For this TAC-KBP competition, we equated *E* with the user-selected entity as depicted in Figure 4. This E value was submitted to a wrapper script that was developed specifically for this task, and which will be described here.

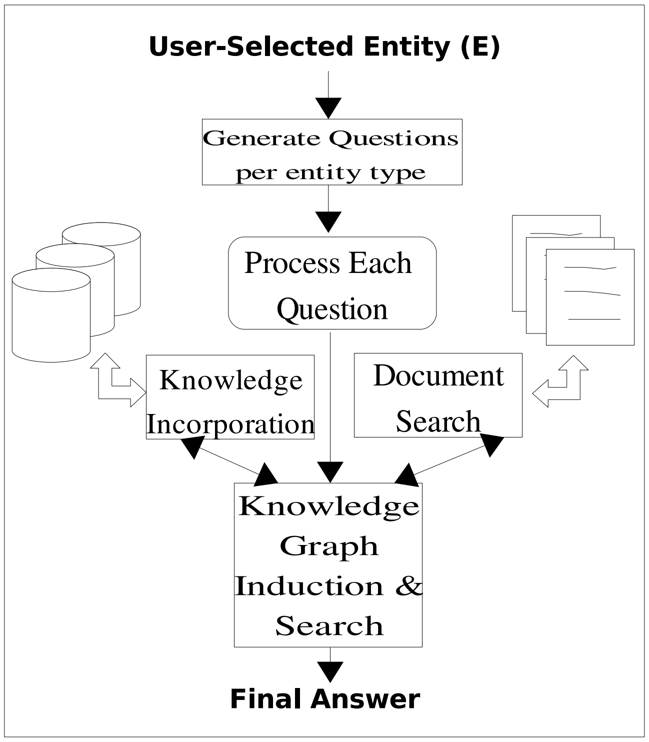

***Figure 4*****:** Slot Filling by Answering Questions

#### *4.1.1. Wrapping QACTIS*

Rather than expecting a series of follow-on questions about *E* as in past TREC's, the wrapper script we built around QACTIS automatically generated a sequence of questions in attempts to fill each potential slot for *E*. For example, if *E* were a person, the system would generate questions such as:

- What is the alternate_name of *E*?
- How old is *E*?
- When was *E* born?
- What nationality is *E*?
- When did *E* die?
- Where was *E* born?

For some multivalued fields, the wrapper script generated multiple questions with the intent of filling only one slot. For example, for the spouse of *E*, the system posed questions:

- Who is the husband of E?
- Who is the wife of E?

After QACTIS attempted to respond to these questions, the wrapper merge-sorted the results from each set based on QACTIS scores, and then returned the *M* best results. The value of *M* was specific to the type of question. For fields where only one answer would be possible, *M* was set to 1. Otherwise, *M* was chosen to be a fixed number: 2 for spouse; 3 for employee_of, title, parents; 5 for residences, member_of, schools_attended; and 10 for children, siblings, other_family.  Even though it is unlikely that the average PERSON entity might have 10 children, it has been our experience with QACTIS that it will get about 1/3 of its answers correct.  Since our goal is recall, we expected that if there are about three children on average and we ask for 10 responses, we might get all three.  This strategy had also been helpful to us before when we used QACTIS on list-style questions.

*4.1.2.  QACTIS-specific Modifications*

The wrapper was not the only element that changes for our slot filling.  QACTIS had to be modified in a number of new ways.

4.1.2.1.  Resource Consistency:  The version of QACTIS that we had used previously at TREC-QA competitions was designed to use BBN's Identifinder and the Charniak constituency parser.  However, given our access to PhoenixIDF results and to SERIF's parsing output, we needed to make changes to QACTIS to support these new types of resources. PhoenixIDF's results appear to be at least as useful as the older entity tagger, if not more, so the overall system does not appear to have been degraded through its use.  On the other hand, much of the existing QACTIS code was optimized to leverage Charniak parses, so the change in parsers did come at some cost which we did not measure.  Yet since SERIF's parser is very fast, we can definitely see it as the parser that we would like to use for our future studies.

4.1.2.2.  Question-type extensions:  There were a number of question types that were completely new to QACTIS.  It was originally optimized on the basis of TREC-QA questions, but question about organization subsidiaries or parent companies almost never appeared in TREC-QA.  Since QACTIS was expected to provide an answer for each of these and other previously-unseen questions, we needed to improve its ability to answer such questions.  To do this, we leveraged the relation patterns described earlier.  For each kind of slot, *S*, with which we expected QACTIS to be unfamiliar, we analyzed the character sequences from the raw documents that illustrate the presence of *S* and we correspondingly developed structure in QACTIS to handle these situations.  This represented over 400 new lines of code.

4.1.2.3.  Person Name Handling: In previous versions of QACTIS, person names were mainly treated as multiword units.  Just as "content extraction" is a hyponym of "extraction," QACTIS previously treated "John Smith" as a hyponym of "Smith."  Since a focus for these new questions was on people and relations between them, it was incumbent on us to create better analysis of people names in order to improve system reaction to names.  More specifically, in the previous QACTIS, the observation of "Robert Smith" would have been counted as an observation "John Smith" because both names would have fallen under the subcategory of "Smith."  The new system changes in QACTIS sought to overcome these.

## 4.2.  Filling Slots Using Extraction

Our second methodology for attempting to fill slots was based on the premise that slot filling can be seen as an information extraction (IE) task similar to the ACE [2] evaluation. Each entry in the knowledge base constitutes a triple consisting of <target entity> <relation type> <filler value> which matches the structure of ACE Relations and ACE Values. ACE Events were also used to fill TAC slots. As the events are not strictly triples, but tuples, the events were mapped into a set of binary relations to facilitate slot filling. Our suite of extractors included all those mentioned in Section2.  Our expectation was that the IE approach would produce a high precision, but perhaps a lower recall system.

*4.2.1. Extraction Mechanisms*

The IE based approach consisted of an ordered processing pipeline: tokenization, sentence segmentation, part of speech tagging, named entity recognition (NER), nominal mention recognition, coreference resolution, and multiple relation/event extractors. The resulting relations and events were processed into slot fillers.

*4.2.2. Building the extraction system*

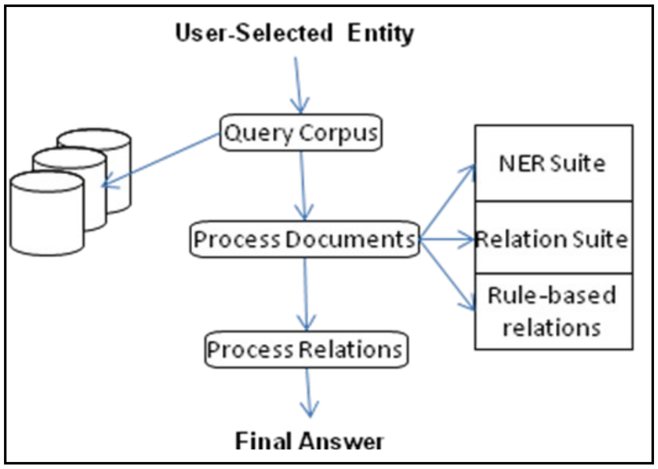

***Figure 5*:** Slot Filling through extraction

The first step in constructing the extraction based system was to understand the guidelines of valid fillers for each TAC slot. In the guideline analysis phase, for each TAC slot a triple was created consisting of <target entity class> <relation type> <filler value class>, and the first and third elements were mapped to the NER classes that could constitute valid fillers. For example, the TAC slot “place of birth” for Person entities, the <target entity class> could be filled by values identified as MUC/ACE type PERSON, and <filler value class> could be filled by values identified as MUC/ACE type GPE. This is a trivial example where a TAC slot filler maps perfectly to an existing named entity class.

Several TAC slots were narrower than their corresponding NER classes, and required world knowledge to fill properly. The TAC person slots “employee of” and “member of” can both be filled by a MUC/ACE type ORGANIZATION. However, depending on the real-world type of organization and the nature of the person’s association it may or may not be a valid filler for either TAC slot. The “top members/employees” slot of TAC organizations is similar in that it could be filled by a MUC/ACE type PERSON depending on the real-world position of that person in the organization. The TAC slot “residences” also details a constraint on valid fillers that would eliminate many MUC/ACE type GPEs; no street addresses, countries are only acceptable if accompanied by a city. In this case, the constraint is not as much semantic as it is structural, as there are resources available to determine the organizational level of a given GPE string (e.g. GeoNames).

Once the possible fillers were determined, an inventory was taken of all the relation classes that our current suite of extractors produce and mapped each relation type to one or more TAC slots. This mapping revealed ambiguities where a single extraction class could be used to fill multiple slots. For instance, the ACE Family relation could fill the TAC slots: spouse, children, parents, siblings, or other family depending on the actual relation expressed.

Processing the competition queries was a multi-step process. First, the query string and example document were submitted to the information retrieval (IR) component to find the 15 most relevant documents from the competition corpus. These 15 documents were processed with CARDS to extract relations and events. CARDS incorporated information from the NER extractor suite, the relation extractor suite and its own templates to produce relations. Any events were transformed into a set of binary relations. All relations that the query string participated in were identified via within-document coreference chains. The mapping of relations to TAC slots was used to produce candidates for competition output. Candidates were eliminated with lexical patterns to account for constraints specified in the guidelines. The remaining set of candidates was used to produce the competition output.

## 5. SYSTEM PERFORMANCE

### 5.1. Linking Performance

Figure 6 illustrates the scores for both the information extraction (IE) and information retrieval (IR) approaches to entity linking using our internal *development* data of 600 queries.

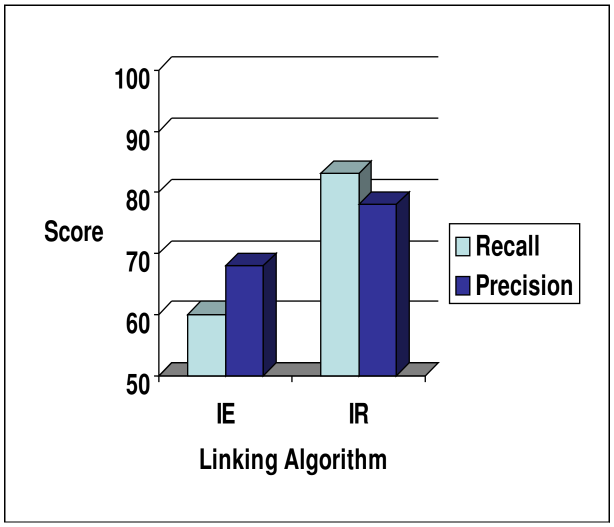

***Figure 6**: Recall and precision scores on internal test data for information extraction and information retrieval approaches.*

The best IR-based approach has a recall of 83% which indicates that the system filtering phase typically has the correct Wikipedia page in the set of fifteen Wikipedia pages. However, the system correctly determines NIL and finds the correct page for a precision of 78% from this set ranked set of fifteen Wikipedia pages.

Figure 7 depicts the results for the three systems submitted to TAC-KBP. "IE" is the information extraction based system. "IR-1" is the information retrieval based system that uses only the semantic forest algorithm to generate semantic words from the reference document for the filtering phase. "IR-2," similar to algorithm IR-1, uses both the semantic words and semantic topics from the reference document, respectively, using the Semantic Forests and Zymurgy algorithms during the filtering phase. The *micro average* is the official score from all of the TAC 2009 3904 test queries, *micro average non nil* is the score from queries that should not return NIL, and *micro average nil* is the score from queries that should return NIL. Since the TAC 2009 competition submitted multiple queries with the same entity name string with different reference documents, the TAC 2009 scorer computes macro average scores. The *macro average* scores are similar to the *micro average* scores, but they averages queries with the same entity name string. Figure 7 suggests that IR-2 has marginal improvement over algorithm IR-1, and both IR algorithms perform better than the IE algorithm.

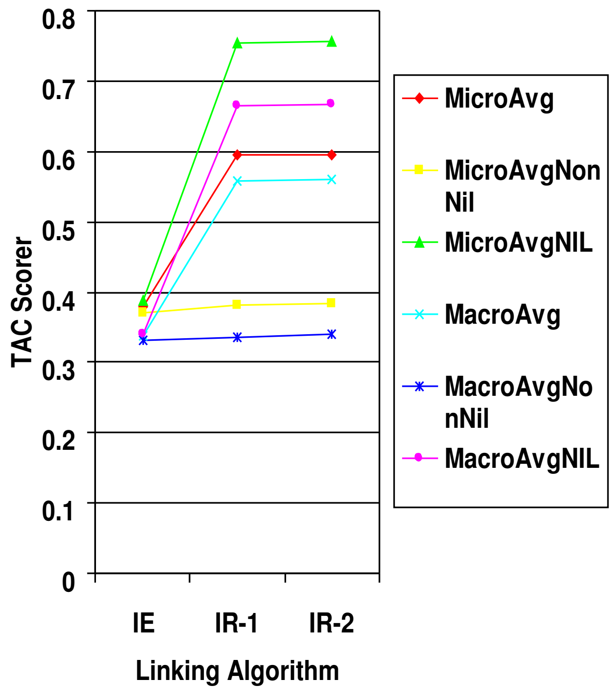

***Figure 7:*** *TAC system scores for information extraction algorithm and two information retrieval algorithms.*

### 5.2. Slot Filling Performance

We were not able to assemble a development set for slot filling, so our primary studies prior to the evaluation were based mostly on visual inspection. For that reason, we will focus here on evaluation performance.

As we expected, our QACTIS-based system provided better recall for non-NILs and our content extraction system favored precision. The differences in these two systems are drastic, so we will articulate them individually.

#### *5.2.1. QA-Based Performance*

We mentioned before that our QA-based system would attempt to fill every single slot. It would only report "NIL" when all searches failed to find relevant information. Table 1 illustrates the output of our system for attempting to fill the slots for one target entity, "China News Agency."

***Table 1****. QA System Result for TAC-KBP SF2*

| Field | Answer |
|---|---|
| Alternate Names | Evening News and Hong Kong China News Agency |
| Political/Religious Affiliation | Communist Party<br>spirituality<br>Democrat<br>worship<br>Buddhism |
| top members/employees | parliamentary<br>intelligence<br>Wenchuan<br>Xu Dingming<br>George Shabad<br>Curley<br>...<br>Alvaro de Molina |
| Founded | 1931 |
| Headquarters | Qingdao |
| Founded by | Li Changchun<br>Paul Julius Reuter<br>Fredy Bush<br>Tian<br>Xinhua |
| Dissolved | 1998 |
| #employees | About 12,000 |
| Member of | Politburo<br>State |
| Members | Politburo<br>Xi<br>Xia |
| Subsidiaries | NIL |
| Parents | NIL |
| Website | www.beijing2008.news.cn |
| Shareholders | CNAC<br>Xu Minjiong<br>Dongbei |

As can be observed from Table 1, many if not most of the answers have the right "shape" of a desired slot fill. Those that do not are in some cases the product of the wrapper that we built around QACTIS as opposed to QACTIS itself.

Nevertheless, according to assessors, the only correct answers were those marked by QACTIS as "NIL." In fact, QACTIS was one of the few systems to even produce any output for this entity. So either systems recognized that no information should exist for this entity, or the evaluation pool from the various systems is impoverished. As expected, such situations will result in low precision, but hopefully fair recall.

In terms of single-valued slots, out of 255 such slots, the TAC-KBP evaluation set only had 39 that were non-NIL. Of the multivalued slots, out of 499, only 129 had non-NIL values. These are surprisingly low numbers. Of the 39, QACTIS found only 5, but there were several others for which it got no credit because of a last-minute Perl scripting error in QACTIS that will be described in Section 6. Of the 129 non-NIL list slots, QACTIS found at least one correct answer for 54 (and it got 9 more that were inexact).

Overall, the QA-based system produced the following scores according to official evaluations:

***Table 2**. Slot Filling Results Using QA*

| | **Single** | **List** | **Total** |
|---|---|---|---|
| Overall Score | 0.196 | 0.237 | 0.216 |
| TAC median | 0.514 | 0.439 | 0.461 |
| NIL Score | 0.208 | 0.262 | |
| TAC NIL median | 0.597 | 0.558 | |
| Non-Nil Score | 0.129 | **0.166** | |
| TAC Non-Nil Median | 0.154 | 0.141 | |

Given the density of NILs in the final test collection, the QA-based system significantly underperformed in comparison to other competitors. Its best result was in the identification of non-NIL LIST values (identified in bold), where it did exceed the median value. Yet the maximum value for this as produced by any one evaluated system was 0.292.

*5.2.2. Extraction-Based Performance*

The content extraction based system only generated an answer when it found a relation supporting the answer. Although no participant sites produced an "All-NIL" (we had contemplated such), such a system would have produced the highest score. This suggest that a system which favors reporting mostly NIL answers and reporting only confident non-NILs is likely to perform well. Our extraction-based system therefore performed well in that it actually only produced 12 non-NIL answers and seven of these were correct (as underlined):

| | | |
|---|---|---|
| SF7 | per:member_of | Hezbollah |
| SF7 | per:employee_of | Hezbollah |
| SF13 | per:employee_of | parliament |
| SF15 | per:employee_of | Sinica |
| SF16 | per:employee_of | Cathay Pacific Airways |
| SF16 | per:place_of_birth | Hong Kong |
| SF17 | per:origin | Massachusetts |
| SF17 | per:employee_of | State Department |
| SF22 | per:origin | Chinese |
| SF26 | per:headquarters | Hong Kong |
| SF48 | per:children | monarch |
| SF53 | gpe:top_employees | Steve Wynn |

This system yielded a reasonably good score as seen in Table 3.

***Table 3**. Slot filling Results through Extraction*

| | **Single** | **List** | **Total** |
|---|---|---|---|
| Overall Score | **0.616** | **0.558** | **0.587** |
| TAC median | 0.514 | 0.439 | 0.461 |
| NIL Score | **0.713** | **0.746** | |
| TAC NIL median | 0.597 | 0.558 | |
| Non-Nil Score | 0.077 | 0.020 | |
| TAC Non-Nil Median | 0.154 | 0.141 | |

There was a huge system problem that occurred during our extraction-based process which will be described in the next section. If that problem had not occurred, our performance in this section would have been significantly higher.

## 6. ERROR ANALYSES

It is meaningful to consider the sources of error in our system. We analyzed our linking and slot-filling results and we here capture information about systematic errors and other issues which made an impact on overall system capability.

### 6.1. Linking Errors and Missing Processes

Being primarily dependent upon IR, our linking systems were fast, but had certain weaknesses.

*6.1.1. Contextual mismatch*

If the system found a perfect or nearly perfect entity match it would often return it even if the context was wrong. Knowledge base nodes that repeat the entity name enough times will be chosen regardless of the lack of other contextual terms in the query. For example, in attempting to link "The Lions" to a KB node, a certain node about a pair of Canadian mountains called "The

Lions" was erroneously chosen every time because of its abundant repetition of the entity name. Disambiguation from the Detroit Lions was further hampered by the node's brief mention of a Canadian football team in British Columbia that was named after these mountains. Error on "The Lions" alone constituted **1%** absolute error, and there were other errors of this type.

*6.1.2. Mishandling of Ambiguous Names*

Many of our linking errors occurred because our system ultimately did not handle particularly common or ambiguous names well. This was largely due to the fact that our 600-query development set did not match the evaluation set well. Our IR systems used weighted queries, with the *names* of query entities being weighted very highly and a lesser weight being given to *concepts* extracted from reference documents. Our development set consisted almost entirely of full-name queries. For such queries, it was likely that the relevant documents would be returned within the top 15, and the concepts would help to disambiguate the rarer case of persons who share the same full names. The evaluation set, however, contained many single-word names and our system performed particularly poorly on those. An example of this is the query "UT". This term is repeated so often in the 1.2 million document repository that no KB nodes appeared within the set of documents returned by the IR, and NIL was erroneously chosen every time. Absent such queries, our performance would increase by **7%** (to 65.9%).

*6.1.3. Insufficient time to incorporate acronyms*

A last major issue that we knew would be one before starting was the handling of initialisms and acronyms. We had made a list of abbreviations and expansions, but we could not integrate them before the deadline, so links that depended upon such expansions suffered as well. Without acronym queries, our performance would have increased by an absolute **3%** (to 62.0%). Some of these are the same as the one-word queries, so we do not expect the errors from Section 6.1.2 and 6.1.3 to be additive.

## 6.2. Slot FIlling Errors and Omissions

There were two major errors in our processing of slots -- one in each system. Each of these will be described as well as an attempt to characterize the impact of the error.

*6.2.1 Flaws in the QA System*

In the QA-based system, the error was extraordinarily simple and was fixed in exactly two seconds after results went out the door. The problem was due to a Perl scripting error.

The QA system ingests PhoenixIDF output. Its tags for nationality and occupation allow for plural answers to be provided (so "presidents" or "Americans" would be legal answers). In the last day before system submission, we wanted to make sure that such answers not be marked as "Inexact" so we inserted a filter in Perl whose task was merely to eliminate the final "s." Unfortunately, the regular expression that was used was "$term=~s/s//;" instead of "$term=~s/s$//;" The latter of these expressions tells the system to delete the final "s" whereas the former deleted the first 's'. We did not recognize this fact until after our evaluation system had started producing faulty values. Instead of such strings as "Chinese", "President", "Secretary," our system was producing "Chinee", "Preident", and "ecretary."

An analysis of our data suggests that this phenomenon occurred at least two dozen times. However, many of these instances were incorrect to begin with, so the errors were not of big issue. Overall, the following were the only answers that could have been impacted (where italicized words could have been "inexact" and boldface could have "correct.") Since three of these were single-valued, these conceivably could have increased the QA system's non-NIL slot answering by as much as **7.6%** absolute.

SF14: title // *miniter*
SF15: origin // **taiwanee**
SF15: title // *miniter*
SF22: origin // **chinee**
SF47: origin // **pakitani**

*6.2.2 Major Issues in the Extraction System*

In terms of our extraction-based slot-filling system, we had a hardware failure and no "safety net" to cover the problem. The hardware where our system was running became overloaded and failed to report any results whatsoever for 12 entities. We forgot to prepare software that would fill slots with "NIL" in the absence of results. So, rather than inserting "NIL" for all of the slots by hand -- which we perceived to be unfair -- we therefore inserted "NO RESPONSE" for each possible slot to impose only error on our system.

We have since rerun and, by ourselves, rescored the system as it should have performed. The system would have reported NIL for all slots from queries 8, 9, 10, 11, 40, 41, 42, 43, 44, 45, 46, and 47 except for the "employee_of" slot for #8 and the "member_of" slot for #9. In these last two cases, the answer that we produced was incorrect. Overall, this would have produced **92** additional correct LIST answers and **55** additional correct SINGLE slots. Table 4 shows how scores would have looked.

***Table 4**. Potential Revised Slot filling Scores*

| | **Single** | **List** | **Total** |
|---|---|---|---|
| Overall Score | **0.832** | **0.743** | **0.773** |
| TAC median | 0.514 | 0.439 | 0.461 |
| NIL Score | **0.968** | **0.995** | |
| TAC NIL median | 0.597 | 0.558 | |

## 7. FINAL COMMENT AND DIRECTIONS

Given that the TAC-KBP data was provided on June 10, 2009 and that entity-linking results were due only one month later and slot-filling results were due two weeks after that, there was very limited time to put all the energy into these two challenging tasks as one might like. We were able to augment our extractors prior to the release of the data; we were able to do some lead studies in random walk and other techniques for cross-document coreference; and at the beginning of May, we had tried to synthesize a development set that would look like the competition set.

Yet receiving the actual data was critical in order to make proper progress. For example, our synthesized set did not resemble the TAC data well, so we needed to re-run all of our annotation tools on the new collection and adjust them to suit the data type and start some processes largely from scratch.

In the long run, by the time of the entity linking competition, there were several major techniques that we had been hoping to incorporate into our evaluation but were not able to given the limited time. We had mostly developed an entity-linking mechanism that made use of machine learning which we would have been pleased to leverage, but were not able to do so this year. Also, our coreference processes did not come to full fruition during the limited development time. We also mentioned that we had some slot-filling bugs that are consistent with first-year participation but which likely would have been eliminated with more development time. Aside from these, however, there were a number of slots which we had hoped to develop relation-finders and question answering capability to support, but ran out of time. We are hoping that if we are able to participate again in TAC, we will be able to include these processes.

Yet overall we were pleased to bring these technologies together to support this new and challenging research direction. We are excited by the existence of the TAC-KBP evaluation, and we are hoping it will be a catalyst for furthering our future research and that of the community.

## 8. REFERENCES

[1] Bikel, D, Miller, S., Schwartz, R., Weischedel, R. (1997) “NYMBLE: A High-Performance Leraning Name-finder.” In Proceedings of the Fifth Conference on Applied Natural Language Processing , ACL, pp. 194-201.
[2] NIST ACE Evaluation, URL: http://www.itl.nist.gov/iad/mig/tests/ace/
[3] Language Computer Corportation, URL: http://www.languagecomputer.com/index.php?page=cicerocustom
[4] Miller, S., Ramshaw, L., Fox, H., and Weischedel, R. 2000. A Novel Use of Statistical Parsing to Extract Information from Text. In *NAACL*, Seattle, WA, pp.226-233.
[5] LEMUR. URL: http://www.lemurproject.org
[6] Apache Lucene. URL: lucene.apache.org
[7] P. Schone, J. Townsend, C. Olano, T.H. Crystal. “Text Retrieval via Semantic Forests.” TREC-6, Gaithersburg, MD. NIST Special Publication 500-240, pp. 761-773, 1997
[8] Philpot, Andrew, Eduard Hovy, and Patrick Pantel. "The Omega Ontology." Proceedings of IJCNLP 2005.
[9] Schone, P., Ciany, G., McNamee, P., Kulman, A., Bassi, T. , "Question Answering with QACTIS at TREC 2004” The 13th Text Retrieval Conference (TREC-2004), Gaithersburg, MD. NIST Special Publication 500-261,2004.