LCC Approaches to Knowledge Base Population at TAC 2010

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung, and Ying Shi
Language Computer Corporation
1701 N. Collins Blvd.
Richardson, TX, USA
john@languagecomputer.com

Abstract

The Knowledge Base Population (KBP) track at the Text Analysis Conference 2010 marks the second year of this important information extraction evaluation. This paper describes the design and implementation of LCC's systems which participated in the tasks of Entity Linking, Slot Filling, and the new task of Surprise Slot Filling. For the entity linking task, our top score was achieved through a robust context modeling approach which incorporates topical evidence. For slot filling, we used the output of the entity linking system together with a combination of different types of relation extractors. For surprise slot filling, our customizable extraction system was extremely useful due to the time sensitive nature of the task.

1 Introduction

Language Computer Corporation (LCC) participated in the Entity Linking, Slot Filling and Surprise Slot Filling tasks at TAC 2010. This paper describes the systems we built for each of these tasks, our results, and comments on our approaches.

The remainder of the paper is organized as follows. Section 2 details our Entity Linking approach. In Section 3, we describe our Slot Filling system, and in Section 4 we discuss how the Slot Filling system was rapidly customized to four new slots in the Surprise task. Section 5 states the conclusions.

2 Entity Linking

The task of entity linking involves resolving an input query of an entity mention to its corresponding entry, or *sense*, in the Knowledge Base (KB). A query is specified by its entity mention string and source document identifier. The KB is derived from a subset of roughly 800K entries from Wikipedia.[1] When no KB entry represents the entity mention in question, the correct system response is NIL. Entity linking was a task in both TAC 2009 and 2010, and the primary change was the expansion of the source document collection to include blog and USENET genre documents in addition to newswire.

To solve this problem, we developed the ELITE (Entity Linking Informed by Topical Evidence) system which employs a three step process. First, the system generates all possible candidate senses for the mention string. Next, it incorporates a variety of evidence to rank the candidates and identify the most likely sense. Finally, ELITE detects if the top-ranked candidate is the correct one, or if the actual sense is unknown and NIL should be returned. We describe our implementation of these steps in the following sections.

2.1 Sense Generation

In sense generation, we attempt to identify every potentially correct sense of the query mention string. We prefer to err on the side of generating too many candidates, since a recall error at this stage would likely be irreparable.[2] To a lesser degree, we attempt to avoid the excessive inclusion of spurious senses.

The candidate senses to be returned are ultimately referenced by their KB node id. Since the KB entries are a subset of Wikipedia articles, in intermediate

[1] October 2008 snapshot.

[2] A correct answer could still be possible if NIL is correct.

processing we represent candidates in the space of Wikipedia article names. We then treat mapping the names into the KB as a special case which is handled during NIL sense detection, described in 2.3.

To generate senses, we utilize several different resources, or *sources*, which each map entity strings to candidate senses based on different Wikipedia-specific features.

Normalized Articles and Redirects (NAR). This source maps the normalized forms of each article's name and redirect page names to the original page name. A normalized name is lowercased and stripped of whitespace and disambiguation labels.

Surface Text to Entity Map (STEM). This source maps all hyperlink anchor texts to their target pages. It is particularly useful because popular targets are more frequently referenced, and can yield a wide variety of aliases.

Disambiguation Page (DP). This source maps every disambiguation page name to each of the hypertext anchors on that page which are superstrings of that page name. A relaxed form of this source (DPR) requires no such string overlap. The DPR contains more noise, but occasionally contains a target not in the DP.

Each of these mappings are stored in a custom read-only in-memory data structure that facilitate efficient storage and retrieval of candidate senses for entity queries. This representation leverages binary search, parallel arrays of Java primitives, and sorted ordinal offsets into them to provide fast access and reasonable space efficiency to millions of relations.[3]

In addition to these context independent mappings, we use several approaches that leverage source document context.

Longer Mentions (LM). We identify senses from STEM associated with longer mentions of the entity string in the source document (e.g., query:"Black Panthers" $\rightarrow$ text/sense:"New Black Panthers").

Soft Mentions (SM). To account for near but inexact string overlap with the mention, we add *soft mentions* as well, which have a high Dice coefficient with known entity senses (e.g., mention:"Moss" $\rightarrow$ text:"Carrie Ann Moss" $\rightarrow$ sense:"Carrie-Ann Moss").

[3]In total these resources contain 28 million terms and 56 million relations, but can still be used on standard hardware (e.g., 2 GB of RAM).

Expanded Acronym Bootstrap (EAB). The system also utilizes document context if it identifies the mention to be an acronym which is defined in its expanded form (e.g., "in the Democratic Republic of Congo (DCR)"). Instead of generating additional senses from this expansion, however, we bootstrap with this information by updating the query mention string to the expansion and restarting the linking system. In this way, sense generation is performed again on this new and more informative text.

Search Engine (SE). Finally, the Google search engine is very effective at identifying some of the more difficult mappings between strings and senses. In our web-allowed runs, we perform queries using the Google API limited to the English Wikipedia site. The queries are run offline, and the results are stored. Online, we filter results by requiring significant Dice or acronym-based similarity to the query, and utilize the top three results as sense candidates.

Using the 2009 entity linking data, we benchmarked sense recall at 97% when using all of these sources in combination.

2.2 Sense Ranking

In sense ranking, the system orders the candidates to identify the most likely sense. To achieve this, we extract a battery of features representing contextual, semantic, and surface evidence. Context features rely on a Wikipedia link-based similarity model using low ambiguity context terms. Candidates are ranked with both heuristic and machine learning methods.

2.2.1 Context Modeling

The importance of context in disambiguation cannot be understated. In particular we are interested in comparing the source document context and candidate sense context. The sense context is derived from KB elements, especially the backing text, or article. A typical approach to measuring similarity between these documents is the comparison of their weighted term vectors. This approach was used by top performing systems in 2009 (Li et al., 2009; McNamee et al., 2009). Terms however, suffer from several problems including ambiguity and sparsity. Sparsity refers to the low likelihood of a direct match of a literal term or phrase in two documents, even when the same concept exists in both documents.

Name	Size	Type	Description
Surface			
LINK_PROB	1	D	percent of mention string links in STEM which target the candidate sense
DICE_TEST	2	B	true if a Dice coefficient score passes the threshold
ACRO_TEST	2	B	true if passes an acronym test
SUBSTR_TEST	1	B	true if candidate or mention is a substring of the other
WEAK_ALIAS	1	B	true if all three surface tests fail
Contextual			
CTX_SIM	1	D	candidate's average LLS score to context terms
CTX_WT	1	B	sum of all context term scores
CTX_CT	1	I	number of context terms
ALIAS_HIT	1	B	true if high precision alias of this candidate is found
FACT_HIT_PTS	1	D	points awarded if a candidate's fact phrase is found
Semantic			
QUERY_TYPE	1	E	semantic type of the query string according to NER system
CAND_TYPE	1	E	semantic type of the candidate according to KB, DBpedia and WRATS
SEM_CON	1	B	true if query and candidate type are not inconsistent
Sources			
SOURCE	7	B	true for each source which generates the candidate
SOURCE_CT	1	I	total number of sources that generates the candidate
Other			
LINK_COMBO	1	D	weighted average between CTX_SIM and LINK_PROB
WEAK_ALIAS_SRC	2	B	joint feature between WEAK_ALIAS and SE or DPR sources
LOG_LINK_CT	1	I	log of total link count to the candidate sense page
SENSE_CT	1	I	number of candidate senses generated
IS_BLOG	1	B	true if source document is detected to be a blog

Table 1: Entity Linking Feature Groups.

We address these challenges using an approach from (Milne and Witten, 2008) which models context terms as Wikipedia page concepts. This notion of a concept term differs in that 1) the term is disambiguated, and 2) the term is represented by references from other terms. This approach was originally used for cross-linking documents with Wikipedia articles, where disambiguation is performed at the document level. By comparison, our focus is on correctly linking specific spans of text, which are individual entities.

Context term candidates are first identified from low ambiguity spans of text.[4] These concepts are then used to disambiguate less clear terms, namely the entities requiring linking. Context terms and entity senses are modeled using their corresponding Wikipedia page, which affords comparison by the extensive hyperlink graph. Sense and term *link similarity* is computed with the Google Normalized Distance (Cilibrasi and Vitanyi, 2007) using inbound Wikipedia links as features. This extra step of indirection from the original terms provides a rich set of topical features, which achieves much higher hit rates than using direct terms.

To select the final set of context terms, each term is given a score from the average of its *linkability* and *relatedness*. Linkability is the percentage of times this span is linked in Wikipedia. Relatedness is the average link similarity to all other candidate context terms. We also prefer close *proximity* terms by selecting the nearest 2*N context terms to the entity mention, reranking them by score, and retaining the top N.[5] Finally, where N context terms do not meet this criterion, we *iterate* by relaxing the maximum ambiguity conditions and resampling the terms.

2.2.2 Features

In the sense ranking step, we use a total of 20 feature groups which are divided into five categories. A

[4]Measured in terms of link probability and observed frequency. In our experiments we found best performance for the first iteration with respective values of 0.01 and 4.

[5]We experimented with different values of N and used 8 in the evaluation.

summary of the features and their types[6] is listed in Table 1.

Surface Features. The first category of features focuses on the entity mention independent of context. LINK_PROB indicates the *link probability* based on the percent of mention string links in STEM which target the candidate sense.

Three groups of binary features each test the similarity between the mention string and the candidate sense's name. DICE_TEST indicates if the maximum of two cases of the Dice coefficient exceeds a threshold. One case compares both full length strings, and the other case takes the maximum of the left and right aligned scores.[7]

For all-uppercase mention strings, the ACRO_TEST features indicate whether the mention might be an acronym of the candidate sense name. The first variant tests whether these letters in any order form the uppercase sequence of the words of the candidate's name. The other merely requires that they begin with the same letter (allowing "TSEC" to match "Taiwan Stock Exchange").

A third feature, SUBSTR_TEST, indicates if the candidate or mention is a substring of the other. Finally, a joint feature indicates whether all three of these surface tests have failed.

Contextual Features. This feature category utilizes portions of the source document outside of the entity mention. The contextual similarity feature CTX_SIM stores a candidate sense's average link similarity to each of its context terms. Additional features, CTX_CT and CTX_WT, encode the number of context terms and the sum of their scores.

Two other features represent contextual clues in the source document with high correlation to the candidate sense. ALIAS_HIT identifies the presence of an alternative alias of the sense. Aliases are generated from the STEM source, and are carefully filtered to retain more specific forms to avoid incidental hits.[8] The goal is to add confidence to a candidate sense if one of its aliases – which is different and more specific than the query string – is found elsewhere in the document. For example, the query "Secret Service" in a document with the text "US Secret Service" would fire this feature for the sense "United States Secret Service" because the presence of "US" has added additional evidence.

The other contextual clue, FACT_HIT, tests the presence of a related entity, or *fact*, known through a relation contained in DBpedia (Auer et al., 2007). This feature's weight is inversely proportional with the frequency of that fact phrase in an independent corpus.

Semantic Features. Surface and contextual evidence is combined to provide the entity types for the mention and candidate, along with their compatibility. The semantic type for the entity mention is determined using LCC's CiceroLite NER system (Lehmann et al., 2007). The highest scoring type is identified by averaging the type confidences across all matching mention strings in the document. If the greatest confidence is below a threshold, it is set to NIL.[9] This approach yields 65% recall with 96% precision. The mention type feature QUERY_TYPE allows the classifier to adjust for type-specific differences, such as a reduced expectation for contextual evidence.

The candidate sense's entity type feature CAND_TYPE is set using a cascade of resources beginning with the KB. If the type is unknown in the KB, DBpedia is consulted. As a last resort, LCC's WRATS ontology[10] resource is consulted. Using this cascaded approach, we observe 97% precision with 95% recall.

Entity types from CiceroLite, DBpedia, and WRATS are all reduced to the types in the KB: person, organization, geopolitical and unknown. Having identified the types for the mention and candidate, a consistency feature SEM_CON is set to true if the types are consistent.[11] While this feature fired to indicate inconsistency in only 10% of queries, it did so with 96% precision.

Generation Features. One binary SOURCE feature is created for each source which generated the candidate sense. This allows the classifier to hedge

[6]Type indicates the feature's value type, which can be (B)oolean, (I)nteger, (D)ouble, or (E)numeration.

[7]Left and right alignment methods truncate the longer string so that they have equal length.

[8]This filtering shares logic with the surface test features.

[9]A confidence of 50% was used.

[10]WRATS contains Wikipedia page names with one of twelve semantic types and classification confidence, with state-of-the-art 93% accuracy.

[11]Types are consistent if they are equal, or if either is unknown.

depending on the origin of the sense. Another joint feature SOURCE_CT provides the number of generators which recommend the sense.

Other Features. Several additional features are used which do not fall cleanly into the above categories. A joint feature LINK_COMBO provides the weighted average between the link similarity and link probability. The WEAK_ALIAS_SRC features combine surface feature tests with generation sources. Two other features, SENSE_CT and LOG_LINK_CT, provide information about the polysemy and popularity of the candidate sense. Finally, a genre feature, IS_BLOG, indicates whether the source context is believed to be a blog.

2.2.3 Ranking Methods

Having these features, we use one of two approaches to rank the candidate senses. Our heuristic approach combines contextual, surface, and semantic features into a numeric score. It initializes the score to the joint feature LINK_COMBO, which is the average of context similarity and link probability. It adds a bonus if a high precision alias is encountered.[12] Finally, it eliminates candidates which are identified to be semantically inconsistent with the mention.

Our machine learning approach applies the NIL Sense Detection classifier described in Section 2.3. This classifier's outcome label confidence is used to re-rank the top N candidates, which have been identified and initially ranked with the heuristic.[13]

2.3 NIL Sense Detection

After selecting the preferred sense candidate by ranking, the final step is to determine if it is believed to be the correct link for the entity mention. In order for the system to output a non-NIL response, it must be sufficiently confident that the top candidate is the correct one, and that the sense is contained in the KB.

To determine the likelihood of a correct link between the query entity and the top sense candidate, we employ a binary logistic classifier. To train this model, we utilize all senses which our system ranked to position N or higher, which also had non-NIL keys.[14] For development we mixed 2009 and 2010 data evenly into a 60/40 train-test split. For the final system we trained on all available data. In the heuristic ranking mode of our system, we utilize a threshold that was empirically determined over the development data. If the classifier rejects the candidate, the linked entity target is considered to be an *unknown NIL*.

Since the set of Wikipedia articles is a superset of the KB, it is possible for the selected candidate to be outside the KB. In this case, we call the linked entity target a *known NIL*. While this can benefit the system by providing a concrete target for known NILs, it also adds a few challenges.

First, sometimes the system will link to an equivalent concept which is not in the KB, and wrongly emit a NIL. For example, the system selected "Abbot and Costello" as a known NIL for the query "Abbot" in the text "which included several Abbot and Costello comedies", when the runner up answer "Bud Abbot" was the keyed one.

Second, since we utilized existing resources from a different version of Wikipedia than the KB's backing version[15], our process required mapping back into the relevant version. To do so, we built a "reverse map" derived from redirects and the Wikipedia change log. This process is a somewhat tedious and imperfect one.[16] Otherwise, mapping into the KB from the relevant version of Wikipedia is trivially performed, and unmappable candidates result in a NIL response.

2.4 Experimental Results

We submitted three runs for the linking task, including one heuristic-based NIL classification (LCC1) and two machine learning-based runs (LCC2, LCC3). The main run LCC1 used no live web access. LCC3 varied from LCC2 in that the classifier bias was relaxed to produce a greater proportion of

[12]We selected a weight of 0.2.

[13]We experimented with N values of 1 and 3.

[14]The value of N is determined by the ranking settings. NIL keys cannot be used, since the actual target is unknown.

[15]January 2010 snapshot.

[16]Because the page history is a graph, it is sometimes impossible to infer the original article. Also, not all types of page evolutions are recorded (e.g., when content is moved, as with the page *Movement for Democratic Change*). On the 2009 dataset, reverse mapping errors resulted in 2% accuracy loss.

Submission	ALL	KNOWN	NIL	PER	ORG	GPE
LCC1	85.8	79.2	91.2	**96.0**	82.4	78.9
LCC2 (*Max*)	**86.8**	**80.6**	**92.0**	95.6	**85.2**	**79.6**
LCC3	86.4	82.4	89.8	95.3	84.5	79.4
Median	68.4	-	-	84.5	67.7	59.8

Table 2: Entity Linking Submission Scores.

non-NIL responses.[17]

Table 2 shows the results of LCC's three runs, with overall accuracy as well as accuracy on semantic subsets of the data. Scores which were best across all 46 task submissions are bolded, and the median score is shown for reference. LCC2's overall accuracy of 86.8% was the high score for the entire task, though LCC1's performance in the person (PER) category was slightly higher. Both ML runs exceeded the heuristic system; however, the heuristic run lacked live access to the web.

The actual NIL proportion of queries was 54.7%. LCC3, with its relaxed NIL bias, predicted the most accurate proportion of NILs, at 54.4%; however, LCC2 still outperformed LCC3, even with a less accurate ratio of 56.4%. LCC3 generated more errors overall by attempting to produce more non-NIL entities. Also, NIL outcomes have many ways of being correct, as opposed to a particular sense, which has only one way of being correct.

We found that our lowest performing semantic category of queries was location (GPE). Analysis revealed that locations often had insufficient or misleading document context, particularly in web documents. In more than one instance, a document had a strong entertainment context, which then caused the selection of an entertainment-oriented entity over the correct location. For example, "AZ" was resolved to the rapper, rather than the U.S. state. This was compounded by a configuration bug in LCC2 and LCC3 which did not utilize the semantic consistency feature during candidate ranking, which would have eliminated person matches for a location type.

Another class of errors were due to excessive leniency in candidate generation. In an effort to obtain all correct senses for entities, the system produced too many candidates which then resulted in more difficult disambiguation. The DPR source was a prolific offender.

[17]This parameter adjustment was selected due to a perceived overweighting of NIL instances in the training corpus.

In general, our system successfully utilized the single sense per discourse (SSD) principle. However, in several cases, such as with the dateline query "BUFFALO[, N.Y.]", the system mistook context elsewhere in the document to be indicative for the query entity. Consequently, the document context for the sports teams "Buffalo Bills" and "Buffalo Sabres" beat out the location.

In one case, finer grained semantic types were required to distinguish "Bengkulu" as a city rather than the province. Adding to the challenge was the overwhelming prior probability on this entity as a province, at 86% versus 13%. Our systems do have access to finer grained distinctions, but time was not available for system development to properly utilize these resources for the submission.

Evaluation results were consistent our expectations, based on previously observed performance. In post-hoc evaluation on the 2009 data and 2010 training data, scores ranged from 86-90%, indicating better than state-of-the-art accuracy.

3 Slot Filling

The Slot Filling task consists of extracting relationships and attributes for target entities. A slot filling query consists of the text of an entity, the document it occurred in, and a reference to a node in the KB if applicable. In the 2010 KBP task specification, there are 42 different slot types defined: 26 for persons and 16 for organizations. Each of these slots are defined as being single- or list-valued.

The task of slot filling can be thought of as a constrained form of question answering. For example, filling the *Subsidiary* slot for the target entity "Samsung" is like asking the question, "What are the subsidiaries of Samsung?" For this reason, our approach to slot filling follows the same series of processing stages typical to a question answering system (Harabagiu et al., 2001).

Our system's first stage is query processing, which analyzes the query in order to understand which entity is being targeted and how to locate it in text. Next is passage retrieval, which selects relevant passages from the corpus that contain references to the target entity. After selecting these passages, candidate answers are extracted. Finally, these candidate answers are merged and the top slot fill(s) are selected. We describe our implementation of these steps in the following sections.

3.1 Query Processing

Given a slot filling query, the system's first task is to properly understand that query, namely the identity of the target entity. For entities contained in the KB, a *node id* is provided. But for entities outside the KB, the node id is marked NIL. Here we employ the entity linking system in order to distinguish between NIL and known NIL target entities.

Having identified and linked all non-NIL entities, the system next generates aliases. This process uses the resources described in Section 2.1, except the mapping performed is the opposite of that in sense generation. Since these aliases are used for document retrieval, we increase their precision using a layer of statistical filtering.

3.2 Passage Retrieval

During passage retrieval, the system finds passages which could contain a reference to the target entity. A query is created to search the source corpus for all documents containing a reference to the entity. The source corpus is stored in a Lucene[18] index for this purpose. Ideally, we would run the linking system over the entire source document corpus and index the results for searching (as we do with named entity types); however, since we did not have time, we use the entity name and generated aliases to perform searches.

Results are filtered to include only passages containing an alias or an entity coreferential to an alias. For this task we use precision-oriented name and pronoun coreference methods.[19]

3.3 Answer Extraction

For answer extraction, the system identifies relationships between the target entity and potential slot fills by running a set of relation extractors over the retrieved passages which are associated with each slot to be filled. The relation extractors leverage entity extraction. In several cases, custom entities were created to handle new entity types. Additionally, we make use of answer *projection*, which obtains answers outside the source collection, to both locate candidates as well as validate the answers found by relation extractors.

Relation Extraction. A variety of LCC tools were used to perform relation extraction. The first was *CiceroCustom*, a rapidly customizable event and relation extraction system, which uses automatic topic detection, active learning, and cascades of machine learning classifiers. Snapshot models are continually created throughout the annotation process, and new training examples are selected based on the model's uncertainty of those examples. This reduces the amount of human interaction required by avoiding the annotation of examples about which the system is already confident. The second tool our approach utilized was a template-based system built on top of a semantic parser. The final relation extraction tool was a set of rules built with LCC's proprietary Packrat grammar system (Lehmann et al., 2005).

Custom Entity Extraction. KBP slot filling includes several relations which require entity types that were not defined by CiceroLite. The primary tool used to fill these gaps was *Welder*, a system that allows the rapid creation of extractors for new entity types using data from Wikipedia as well as the web. *Welder* derives a rich array of semantic features from both semi-structured data, including Wikipedia categories and hyperlinks, as well as unstructured data, such as subject complement patterns (Hearst, 1992). Higher precision is obtained using features similar to those in the entity linking system, such as LINK_PROB. In addition, topically-related contextual terms relevant to the entity class, obtained from weakly-supervised learning, are used for disambiguation. To handle exceptional cases, additions to CiceroLite's customizable rule layer were made.

KBP Customization. Existing LCC extractors covered approximately 30 of the KBP slots and new extractors were created for the rest. Special handling had to be added for certain slots. For example, the LCC *Hiring* extractor denotes an employment relationship, while our *Join* extractor may de-

[18]http://lucene.apache.org

[19]Later analysis showed that nominal coreference would have been more beneficial than anticipated.

note employment or membership, depending on the type of organization joined. Most of the entity types required existed in CiceroLite, but Welder was used to extend our coverage of crimes, death causes, and the different types of organizations (to distinguish membership/employment).

Answer Projection. We also developed a projection module with the aim of taking known slot-filling attribute values from structured reference sources and finding evidence to verify them in the source document corpus. First, the module gets input attribute values for the target entity from structured reference resources like DBpedia. We selected attributes from the reference source which corresponded to the requested slot fills. We picked the appropriate reference fields using a combination of analysis techniques utilizing the semantics of the field names and verifying known values from available keyed data. The attribute values were alternated using different heuristics depending on the nature of the value; for example, for birth date, only the year was required. An answer was extracted if the entity mention and attribute value mention occurred in the same sentence. This approach was beneficial in raising the recall of our system.

3.4 Answer Processing

After running all of the relation extractors, the candidate answers are merged and ranked to form the final answers for system output.

Merging. Merging of answers is necessary because the same concept can be referred to in multiple ways. This includes synonymous texts, writing styles and conventions, usage of popular aliases, and even misspellings. We use four steps to merge answers. First, coreference chains are used to normalize pronouns and partial names to their canonical form. Next, answers are combined if they are found to have synonym, hypernym, or hyponym relationships according to WordNet (Fellbaum, 1998). Then, answers which are linked to the same Wikipedia concept are merged. Finally, surface-level variations (e.g., typos) are detected using the Dice coefficient.

Ranking. For single-valued slots, candidates must be ranked in order to select the best answer for a query. We use a scoring function based on answer's merged group size, with candidates from higher precision retrieval strategies weighted more. For known or known NIL target entity queries, we again apply our entity linking system. We give preference to answers in context with an entity with high link confidence to the target entity. For list-valued slots, one value was selected from each set of merged values, selected first by frequency and then by length.

3.5 Experimental Results

We submitted three different runs for the slot filling task. Our first submission consisted of our core extraction system with no web access or projection. The second submission included projection for better recall, but used aggressive context filtering to maintain precision. Our third and final submission did not filter projection or use entity linking, but considered fewer candidates. This was intended to enhance the recall.[20] The scores obtained by each of our runs are listed in Table 4 and the breakdown between single- and list-valued slots are in Table 3.

Submission	**P**	**R**	**F**
LCC1	45.3	18.8	26.6
LCC2	44.9	19.4	27.1
LCC3	43.6	19.2	26.7
LCC1 + no aliases	48.1	15.8	23.8
LCC1 + manual aliases	48.9	16.8	25.0
Median	21.4	10.5	14.1
Max	66.8	64.8	65.8

Table 4: Slot Filling Submission Scores.

Both the precision and recall of the system were lower than expected. Our development set scores had a precision of around 80% with recall around 35%. The majority of precision errors tended to be from imprecise relation extractors, incorrect named entity recognition, or coreference errors.

One source of errors was caused by an issue in linking target entities. For example, the entity "Sean Preston" was linked to his mother "Britney Spears", because on Wikipedia his name redirects to her page. This caused the entities to be considered aliases of each other, which hurt our precision. Surprisingly, this issue was never encountered during development of our entity linking system.

We thought that the entity linking system might significantly improve our precision in the case where

[20]Because of the way the candidates were filtered, this actually had lower precision and recall.

	LCC1			LCC2			LCC3		
	Single	**List**	**Total**	**Single**	**List**	**Total**	**Single**	**List**	**Total**
P	74.7	36.5	45.3	73.6	39.4	44.9	71.6	38.2	43.6
R	26.5	16.9	18.8	26.5	17.8	19.4	26.5	17.6	19.2
F	39.1	23.7	26.6	39.0	24.5	27.1	38.6	24.0	26.7

Table 3: Single-valued vs. List-valued Slot Filling Scores.

an entity target name was polysemous. In practice, this did not occur, as 98% of the known NILs were the most popular entity for that name. Since the answers extracted almost always concerned the target entity, this left little opportunity for the linking system to improve results by filtering out spurious candidates.

On the other hand, the linking system did provide benefits to system recall. Using no aliases at all resulted in a F-score of 23.8. During post-evaluation analysis, a human annotator went through the aliases and created a manual set of unambiguous aliases, which resulted in a F-score of 25.0, actually lower than with automatic aliases, as illustrated in Table 4. While automatic aliases caused precision errors, they benefited our system by improving recall.

Another source of precision errors was mapping from LCC extractors to KBP slot types. Both *Membership/Employment* and *Org_Member_Of/Subsidiary* can be tricky to differentiate. For a real world extraction system, these errors are not as problematic since it would be reasonable to collapse these into *Affiliation* and *Org_Part_Of*.

The recall errors are spread out across the different slot types, and are mostly due to the variety of patterns not seen during system training.

4 Surprise Slot Filling

The surprise slot filling task is new for KBP 2010. The purpose of this task is to test the efficient portability of extraction systems to novel tasks. Both CiceroCustom and Welder were designed to rapidly develop extractors, and both were heavily used. The surprise slot types are *Awards Won*, *Charities Supported*, *Diseases*, and *Products*.

4.1 Customization

Our approach to the surprise slot filling task was to create custom extractors for each new slot type.

Awards Won. This extractor made use of an existing *Award* entity type in CiceroLite supplemented by a fresh Welder-generated lexicon of 11,700 common awards. To extract the *Awards Won* relations, we created a CiceroCustom extractor around this entity type, as well as a grammar rule.

Charity Supported. This slot type requires having a strong lexicon of charities. We used Welder to learn the names of 7,400 charities. Unfortunately, many of the charities mentioned in the source data collection are not famous[21] enough to learn with Welder. However, many of these unknown charities contain words such as "Society", "Fund", or "Save". We used a statistical approach over the 7,400 known charities to learn these terms. To extract charities supported, we created a custom relation extractor. This extractor found a contextual relation between charities and persons within a certain distance of each other. Because this is a low precision approach, we used Welder to create a lexicon of giving/supporting terms, which were required within the context.

Diseases. Welder was used to create a lexicon of 27,900 diseases. A CiceroCustom extractor was created to extract *Diseases* relations, along with a supplementary grammar rule.

Product. The *Product* slot was not only the most difficult slot in the surprise task, but also the most frequently occurring one. One reason product extraction is difficult is because traditional NER specifications do not provide a parent type from which to begin customization. Also, products are often named after other entities, and occur in a wide variety of contexts.

We found Welder to be very effective in creating lexicons for specific product types. For example, we created a lexicon of 872 microprocessors. However, it was infeasible to manually "weld" lexicons for every conceivable product category, in the span of this

[21] A famous entity being either frequently mentioned on the Internet or having a page in Wikipedia.

evaluation. Therefore, we primarily relied upon a recall-oriented rule-based approach.

4.2 Experimental Results

We made two submissions for the surprise task. The first was made after 11 hours, with a total of 29 man hours. The second was after 34 hours, with an additional 29 man hours spent. The second day was mainly spent improving our recall, specifically for the product extractor. The results obtained by our system for the Surprise Task are listed in Table 5. The additional day of work between our two submissions provided significant improvements, adding 57% in recall. With only this small amount of work, our scores on the surprise task are close to those of the regular slot filling task.

Slot Type	LCC1			LCC2		
	P	R	F	P	R	F
Award	56.5	19.7	29.2	55.6	22.7	32.3
Charity	48.1	19.1	27.4	48.1	19.1	27.4
Disease	42.8	22.2	29.3	46.7	25.9	33.3
Product	50.5	13.3	21.1	53.0	25.3	34.3
Total	50.3	15.4	23.6	52.3	24.2	33.1

Table 5: Surprise Slot Filling Scores per Slot Type.

Table 6 shows our results which were the best obtained by any completely automatic approach for this task.[22] LCC was the only team to obtain scores with F-measure of greater than 10% (Ji et al., 2010). The table also displays elapsed time, which shows the increase in performance after a second day of customization.

Submission	P	R	F	Time
LCC1	50.3	15.4	23.7	11 hours
LCC2 (*Max*)	52.4	24.2	33.1	34 hours

Table 6: Surprise Slot Filling Submission Scores.

This surprise task also showed the usefulness of Welder in generating large entity lexicons in a short period of time. Without these lexicons, the system performance would have been significantly worse. Finally, the challenges experienced during this evaluation have already helped us create improvements to our customization software.

Our performance was lower than expected, especially the recall. Both precision and recall on the development set were around 50%; however, the size of that set was very small. The *Awards Won* extractor suffered from our system's lack of structured data extraction. Several documents contained bulleted lists of award winners, which our system was unable to parse. The *Diseases* extractor was trained on fairly simple sentences, and struggled with several of the more complicated sentences that occurred in the test data. Despite our attempt to make the *Charity Supported* extractor high precision, our system occasionally struggled to tell when someone was supporting the charity or just mentioned in context with it. Finally, the *Product* extractor was fairly imprecise. An example of an error was "Porsche Corral", which looks a model of Porsche, when in fact it is an event. More time is required in order to build a high quality all-encompassing product extractor.

5 Conclusions

The KBP track at the TAC 2010 marks the second year of this important information extraction evaluation. KBP emphasizes the need for linked data, where information mined from text can be connected to and stored with preexisting knowledge of entities in the world.

This year, additional changes were made to the tasks which further increased their realism. One such change to the Entity Linking task extended its document collection to include blog and USENET genre documents in addition to newswire. Another change was the addition of a new surprise version of the Slot Filling task. It tests the adaptability of extraction systems to new domains under time constraints.

This was LCC's first year in the evaluation, and we participated in three tasks. For entity linking, we used a three step process of sense generation, sense ranking and NIL sense detection. It achieved high recall of entity senses through both context-free and context-dependent sense generation sources. Key to our system was a robust context modeling approach which used Wikipedia link-based similarity with low ambiguity context terms. NIL sense detection was achieved with a logistic classifier and twenty feature groups. Our system achieved accuracy scores in the upper 80's, which proved to be best in the task this year.

Our slot filling system utilized processing stages

[22]One team submitted better results, but these required manual intervention by human users.

similar to that of a question answering system. It applied four independent relation extraction systems to obtain answers, including both machine learning and rule-based methods. The entity linking system was employed to disambiguate known NIL queries, generate aliases, and link final candidate answers. Despite being one of the higher performing systems, results were lower than expected, underlining the difficulty of the task. Factors included overfitting our system on the limited training data, and a special breed of linking errors which harmed system precision.

For the surprise system, we customized our slot filling system to four new slot types in a matter of hours. This involved not only the creation of several new relation but also entity extractors. Results were submitted after one day and again after a second day, where day two results demonstrated a 40% improvement in F-measure. At a fraction of the time invested, performance exceeded that of the system on the regular slot filling task. Though scores were lower than anticipated, LCC's submission achieved an F-measure three times higher than any other fully automated approach. Challenges included overfitting on the very limited training data and the task of rapidly developing a general purpose product extractor from scratch. We appreciated the opportunity to participate in this new exercise, and are continuing to improve our customization process based on this experience.

A demo of LCC's extraction capabilities incorporating some of the described linking and slot filling technology can be found online.[23]

6 Acknowledgements

The authors gratefully acknowledge the support and funding of part of this work by the Air Force Research Laboratory (AFRL). Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the AFRL or the US government.

[23]http://demo.languagecomputer.com/cicero

References

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *6th Intl Semantic Web Conference, Busan, Korea*, pages 11–15. Springer.

R.L. Cilibrasi and P.M.B. Vitanyi. 2007. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*, 19:3:370–383.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunsecu, R. Girju, V. Rus, and P. Morarescu. 2001. The Role of Lexico-Semantic Feedback in Open-Domain Textual Question-Answering. In *ACL01*.

M. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. In *14th International Conference on Computational Linguistics*.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2010. Overview of the TAC 2010 Knowledge Base Population Track. In *TAC (Text Analysis Conference) 2010 Workshop*.

J. Lehmann, P. Aarseth, M. Deligonul, L. Nezda, and A. Hickl. 2005. ACE 2005 TERN System Description: TASER. In *Proceedings of 2005 Automatic Content Extraction Conference*.

J. Lehmann, P. Aarseth, L. Nezda, Sarmad Fayyaz, Arnold Jung, Sean Monahan, and Meeta Oberoi. 2007. Language Computer Corporation's ACE 2007 System Description. In *Proceedings of 2007 Automatic Content Extraction Conference*.

F. Li, Z. Zheng, F. Bu, Y. Tang, X. Zhu, and M. Huang. 2009. QUANTA: KBP (Entity-linking Task, Slot-filling Task), RTE (Two-way Task). In *TAC (Text Analysis Conference) 2009 Workshop*.

P. McNamee, M. Dredze, A. Gerber, N. Garera, T. Finin, J. Mayfield, C. Piatko, D. Rao, D. Yarowsky, and M. Dreyer. 2009. HLTCOE Approaches to Knowledge Base Population at TAC 2009. In *TAC (Text Analysis Conference) 2009 Workshop*.

D. Milne and I.H. Witten. 2008. Learning to link with Wikipedia. In *ACM Conference on Information and Knowledge Management (CIKM'2008)*.