UCD IIRG at TAC 2011

Lorna Byrne
School of Computer Science
& Informatics
University College Dublin
Ireland
lorna.byrne@ucd.ie

John Dunnion
School of Computer Science
& Informatics
University College Dublin
Ireland
john.dunnion@ucd.ie

Abstract

This notebook paper describes the IIRG systems entered in the TAC 2011 Knowledge Base Population Track. In 2010, we participated in the Regular Slot filling Task(Byrne and Dunnion., 2010), and while we scored well with respect to precision our system was found to have very low coverage. This year, we found ourselves dealing with a similar trade-off between precision and recall.

1 Introduction

This is the second year that the IIRG (Intelligent Information Retrieval Group) from University College Dublin has participated in TAC's Knowledge Base Population (KBP) Track. The Knowledge Base Population Track is composed of two related tasks: Entity Linking (EL) which links entity mentions to their corresponding entities in the Knowledge Base and Slot Filling (SF) which augments existing Knowledge Base entities with novel information. Entity Linking tasks process the entity types Person (PER), Organisation (ORG), and Geo-Political Entity (GPE) whereas Slot Filling tasks are limited to PER and ORG entity types only. This year there were five tasks in the KBP track: Mono-Lingual Entity Linking and Regular English Slot-Filling Tasks, both of which also ran last year, and three new tasks introduced in KBP2011, Full Temporal Slot Filling (TSF), Diagnostic Temporal Slot Filling and Cross-Lingual Entity-Linking.

This year we participated in the Regular Slot Filling, Full Temporal Slot Filling and English Entity Linking tasks. TAC provides participants with a large source collection of 1,777,888 documents, consisting mainly of newswire and Web documents, and a reference knowledge base consisting of 818,741 nodes. Participants are required to use the source collection to discover information about target entities; if any attributes are found they should be novel, and thus systems also need to check if any of these attributes already exist in the knowledge base.

In KBP2011, we participated in the Regular Slot Filling task again and the Entity Linking and Temporal Slot Filling tasks for the first time. In last year's submission for the Regular Slot Filling we sacrificed a good recall score for high precision and so with this in mind tried to improve our Regular Slot Filling system to enhance recall. For both the Entity Linking and Temporal Slot Filling task, having never participated in these tasks before we built very simple systems in order to familiarise ourselves with these tasks.

2 Regular Slot Filling Task

The slot filling task is similar to the task of Question Answering (QA). QA systems generate answers in response to queries, normally a natural language question, over large collections. The objective of SF systems is to return a slot-value (exact answer) in response to a slot query. Each slot query could easily be transformed into an equivalent question or set of questions for use in a QA system. The SF task could be viewed as a specialised QA task where the questions remain the same and only the target or focus of the questions change. Given the similarities between the two tasks and our previous research, we decided to approach this task from a QA perspective.

Our system follows a simple pipeline architecture similar to that of a traditional Question Answering System described in Pasca (Pasca, 2003). The SF pipeline consists of three main modules: Query Processing, Passage Retrieval and Slot-Value Selection. We have implemented a two-stage retrieval module. In the first stage, we retrieve a set of documents in response to the target entity in the initial query using an off-the-shelf document retrieval component. In the second stage, we reduce this search space to passages in order to identify passages and segments that match the generated candidate patterns and then extract a slot-value based on an identified slot-value type in as narrow a search space as possible.

2.1 Pre-Processing

In this phase of the system development we used the data from both the 2009 and 2010 Regular Slot Filling Tasks as a source of training data. We created training examples by extracting the slot-value at the sentence-level and annotated occurences of the target entity and the slot-value. There were some slot-values for which it was necessary to create patterns manually; for example, URL patterns to identify websites to extract for the slot org:website. While extracting training data we found in many cases that the slot-value occurs in close proximity to the target. For example, consider the following sentences for the slot per:date_of_birth where the person is the target-entity and the date is the slot-value:

Raul Castro was born on June 3, 1931
Sean Preston, was born in September 2005
Hugo Chavez was born on July 28, 1954

Candidate patterns were then generated around the slot-value in such text segments. These were used later in the selection process, using a combination of the Stanford POS tagger and Stanford NER tools[1]. An example of one simple pattern that can be created from the examples above for the slot per:date_of_birth, is the following:

```
<person> VB:bear <date>
```

Any verb encountered is usually reduced to its canonical form to allow for greater coverage.

Given the similarities between Slot-filling and Question Answering, it is also possible to translate the slots themselves into equivalent questions for use in a QA system. For example, there are a number of questions which correspond to the slot per:age:

```
When was <target-entity> born
How old is <target-entity>
What age is <target-entity>
. . .
. . .
```

Identifying questions such as these allowed us to augment our candidate patterns with previously learned patterns from QA research or from previously run QA tracks such as TREC.

[1]http://nlp.stanford.edu/software

It is also possible to identify an Expected Value Type for each slot from the training examples. For each of the slot queries we generated an Expected Value Type from the associated candidate patterns. For example: per:spouse requires a person as the slot-value, per:date_of birth requires a date, per:age requires a number and org:website requires a URL.

During this phase, we also indexed all the documents of the document collection using Terrier 3.0 (I. et al., 2006). Terrier is a highly flexible, efficient and effective open source search engine, readily deployable on large-scale collections of documents. Terrier implements state-of-the-art indexing and retrieval functionalities, and provides an ideal platform for the rapid development and evaluation of large-scale retrieval applications.[2]

2.2 Passage Retrieval

As indicated above, the Passage Retrieval (PR) module is a two-stage module which identifies passages or documents that are most likely to contain an answer. Passages can be segments of text or entire documents. The slot-value selection module will apply potentially computationally costly NLP techniques to identify candidate slot-values, it is important to ensure that the search space that this module is applied to is as narrow as possible. Although traditional Document Retrieval Methods can reduce the search space significantly, some of the documents can still be quite big and contain a lot of noise. It is therefore necessary to reduce the search space further by identifying useful passages of text within the documents. The documents are processed using the Stanford NER core pipeline. Occurrences of the target entity are annotated in each sentence and target entity tokens are also annotated as **candidate-target-entity**. The sentences are then ranked based on the features they contain, i.e. sentences containing the target entity or candidate target entity are ranked higher than sentences which contain no entity mentions. It is the highest ranked sentences that are presented as input to the Slot-Value Selection Module as candidate slot-bearing passages.

2.3 Slot-Value Selection

The final phase of the pipeline selects and extracts the segment of text that is most likely the relevant slot-value. Each returned slot-value must also contain the docid of the supporting document from which the slot-value was extracted. The Slot-Value selection module processes the candidate slot-bearing passages returned by the PR module and selects passages that are of the Expected Value Type,

[2]http://www.terrier.org

as identified in the Query Processing phase. Smaller passages which conform to the candidate patterns are then selected and added to an answer set. The answer set identified is ranked according to frequency of occurrence, where we prefer to extract a slot-value that occurs in many documents. In order to identify equivalent answers, both in the returned answer set and to identify redundant answers already located in the knowledge base we implement a text similarity metric to identify similar values. We normalise each candidate slot-value, remove any punctuation and spaces etc. Using Levenshtein Distance(Levenshtein, 1966) we then calculate a distance score for each candidate slot-value from other candidate slot-values in the same slot and also from entries in the KB info-boxes that are equivalent to the slots. The most-frequently occurring candidate slot-value is extracted as the slot-value, but it is not returned if it has an equivalent slot-value in the KB info-boxes. This should mean that occurrences of "prime minister", "Prime Minister" and "Prime-Minister" would all be equivalent for the list-valued slot per:title and only one would be returned as the slot-value.

There is no threshold of answers generated in response to a list-slot-value. For list-slot-values, we remove duplicate and redundant candidate slot-values using the same distance metric as before and then return all candidate values of the expected type as slot-values. NIL is returned as the slot-value when no slot-value is found, that is, when the PR module fails to return any candidate slot-bearing passages or when the Expected Value Type is not identified in this phase.

Having identified the set of candidate slot-values it is important to ensure that the slot-value is of the correct type so answers of the form "3-year-old" for per:age where the expected value type is NUMBER should be reduced to "3".

2.4 Results

Approximately 49% of the slot-values that we acquired from the document collection were judged to be the correct slot-values, although, again, our system achieves very low coverage of slot-values across the entire document collection, acquiring only 13% of the novel slot-values (see Table 1). It is clear that we haven't made any improvements on the low recall score from last year's competition. While we have to complete a full analysis of results for these tasks, early inspection seems to indicate that the system is bound by the Slot-Value Selection module, as our system is not aggressive enough at picking out candidate slot-values from the candidate passages and in most cases reverts to NIL. Our results serve to confirm the view that, Slot Filling is a challenging and complex task and it is not easy to achieve good performance here.

Submitted run: IIRG1	
Recall	0.12592593
Precision	0.49173555
F1 Score	0.20050548

Table 1: Results of Run Submitted to TAC 2011

per:spouse
per:title
per:employee_of
per:member_of
per:cities_of_residence
per:stateorprovinces_of_residence
per:countries_of_residence
org:top_employees/members

Table 2: Slots used in Temporal Slot Filling Tasks

3 Full Temporal Slot Filling

KBP2011 saw the introduction of a framework for evaluating and extracting temporal information about a particular slot-value. The Full Temporal Slot Filling task was introduced this year. Temporal information can be distributed across multiple documents. In one document you might find the text snippet "*John Smith was named CEO today....*" and then in another document the following "*John Smith resigned as CEO last year...*". In this task, participants are required to add temporal boundaries to extracted slot-values, so in this example having found that "CEO" is a slot-value for the slot per:title, systems are now required to augment this value with temporal information. The task is limited to eight slots taken from the Regular Slot Filling task, these are listed in Table 2.

The temporal information for each slot can be represented using a four element tuple [T1 T2 T3 T4], indicating that the slot-value became true at some time between teh interval of T1 and T2 and ending at some time between the interval of T3 and T4. This 4-tuple of dates can be associated with a non-NIL slot-value. This representation is chosen so as to keep as close to possible with the Regular Slot Filling Task and the info-box style of the Knowledge Base. We used our basic slot filling pipeline with an additional temporal component which identifies any temporal information in close proximity to the identified slot-value. For this task, we implemented a simple approach with the objective of familiarising

Extract from Four-Tuple Response for SFT257 Slot per:spouse		
T1	19590101	XIN_ENG_20080224.0243.LDC2009T13
T2	19591231	XIN_ENG_20080224.0243.LDC2009T13
T3	20070101	XIN_ENG_20080224.0243.LDC2009T13
T4	20071231	XIN_ENG_20080224.0243.LDC2009T13

Table 3: System Response for SFT257 slot per:spouse

ourselves with this new task and created only a few candidate patterns from the supplied training data.

Consider the following pattern:

```
<target-entity> divorces husband of
<temporal>18 years</temporal> <slot-value>
```

We can now also add the constraint that the keyword "divorces" indicates that the value of per:spouse is no longer true.

Systems are required to return the same output as the regular slot filling task, a slot-value and an associated document identifier that supports the value. The addition of the temporal boundaries is a very challenging task as systems must first correctly identify the slot-value for a given slot and then identify whether there are any temporal constraints associated with this value. If a system identifies temporal boundaries, it must then try to extract the information to complete the 4-tuple if it is available. The sentence: *"Raul was married to fellow revolutionary Vilma Espin from 1959 until her death in June 2007."* where Raul is the target entity. The slot per:spouse can be filled with the value "Vilma Espin". However, as we have also identified some temporal boundaries on this value we now need to augment this slot-value with this information. We need to extract the temporal information in he form of tuples. The phrase "from 1959" indicates that at some point in 1959 this slot-value became true, that is during the interval from 1st January 1959 to 30th December 1959 this slot-value became true. The phrase "until her death in June 2007" indicates that at some point in June of 2007 this slot-value became false, that at some point from 1st June 2007 until 30th June 2007 this relationship and therefore the slot-value became false. With this information now identified we can begin to fill each element of the tuple in turn as illustrated in Table 3

3.1 Results

The Full Temporal Slot Filling Task adds a new dimension to the already difficult Regular Slot Filling Task. We built a very simple system for this task in order to familiarise ourselves with the new temporal constraints placed on slot-values. We have implemented a temporal phase in the Regular Slot Filling pipeline architecture described in Section 2, so that the performance of the Temporal Slot Filling system directly correlates with the performance of the Regular Slot Filling task. The TSF system achieves very low coverage of slot-values, acquiring approximately 13% of the novel temporal slot-values (see Table 4). The TSF system has not filled many slots with temporal information (see Table 5), this is not surprising though given the systems dependency on the Regular SF system, which also achieves low coverage of slot-values across the document collection.

Submitted run: IIRG1	
Recall	0.12995498
Precision	0.25048107
F1 Score	0.17112607

Table 4: Results of TSF Run Submitted to TAC 2011

Slot	Recall	Precision	F1
per:spouse	0.2790	0.4104	0.3322
per:title	0.1663	0.2230	0.1905
per:employee_of	0.2500	0.2635	0.2565
per:member_of	0.01716	0.2187	0.0318
per:cities_of_residence	0.1607	0.2045	0.1800
per:stateorprovinces_of_residence	0.1071	0.1666	0.1304
per:countries_of_residence	0.0292	0.2399	0.0521
org:top_employees/members	0.0	0.0	0.0

Table 5: Temporal Slot Filling Task Slot Results

4 Mono-Lingual Entity Linking Task

The objective of the Entity Linking task is to associate each query entity with the relevant knowledge base file or files. In the Entity Linking task participants are given a list of target entities to process. There are 3 generic entity types included in this task: person (PER), organisation (ORG) and geo-political entity (GPE). Each EL query consists of a target entity and a supporting document id. Systems are required to provide the ID of the KB entry to which the name refers or NIL if the entity does not exists. NILs should be of the format "NILxxxx", where "xxxx" refers to a unique id given to a cluster of similar query entities. The same entity can be referred to by more than one name so systems are required to assign a unique id to all related non-KB entities. NIL clustering allows us to identify novel entities and will support the automatic creation of novel KB entities.

This task is further divided into two subtasks: an entity-linking task using the free text from the KB pages associated with the knowledge base nodes and an entity-linking task without using this free text. Given that the same entity can often be referred to by more than one query string and one entity name can refer to more than one entity this makes Entity Linking quite a challenging task. Again, having never participated in the EL task before, we decided to build a simple system to take part in the Entity Linking task, without using the Knowledge Base text.

4.1 Pre-Processing Phase

In the Pre-Processing phase, we processed the documents in the knowledge base into single documents to conform with standard TREC formats for ease of indexing. We then indexed this collection based on the fields wiki-title and wiki-text. Firstly, it is necessary to normalise the patterns to the same case and remove any punctuation, e.g. EL_02080 Tony“ Rezko contains additional unnecessary punctuation.

Entities will normally occur using different name variants and possibly docids for that reason, in this phase we also calculate the similarity between all of the the query entities in order to cluster similar entities together for each query. We calculate a similarity score using Levenshtein Distance(Levenshtein, 1966). This will allow us to cluster similar entities together based on the threshold set for distance score. For these initial runs we have chosen to cluster exact matches only, and thus have opted for a distance score of zero as the initial threshold.

4.2 Passage Retrieval

In Passage Retrieval, we use the EL queries as input into our search engine and in response we return a list of node ids which correspond to KB entities. The query runs on the wiki-title field for each document. The first run submitted used the EL query as the input query. For the second submitted run we implemented some query expansion techniques. The query expansion techniques involved using simple pattern-matching techniques for resolving acronyms, etc. and we also incorporated our slot filling system and set it to retrieve one slot alternate_name using only the support document as a source of input. The intention here was to capture any information available for the per:alternate_name and org:alternate_name.

4.3 Entity Node Selection

The Entity Node Selection phase selects the best match for the query. The input into this module is the list of wiki-titles retrieved by the PR module. We then apply shallow pattern matching techniques to select the node id that is the closest match for the query entity.

Setting a similarity threshold to 0, we assign all similar NIL nodes to a NIL id of the form NIL00001.

Entity Linking Scores with no wiki-text	
Highest F1 (no Web)	0.714
Median F1 (no Web)	0.521

Table 6: EL Highest and Median B^3 F1 Score

Scores for English entity-linking-no-wiki-text task				
	KBP2010 micro-avg	B^3 Precision	B^3 Recall	B^3 F1
IIRG1	0.560	0.515	0.557	0.535
IIRG3[3]	0.579	0.555	0.545	0.550

Table 7: Results of EL Runs Submitted to TAC 2010

A query which has this id indicates that the queries refer to the same entity and that this entity does not exist in the Knowledge Base.

4.4 Results

For the Entity Linking task we built a very simple system in order to familiarise ourselves with the task. The EL task itself is quite a challenging one and our naive approach to EL should serve to justify the use of more sophisticated techniques in order to disambiguate entities. We didn't expect that these results would be impressive; but rather that they would indicate the need to incorporate the use of the wiki-text and more than likely the supplied document collection in order to disambiguate the KB entities.

Results for the Entity Linking task are calculated based on B^3 Precision, Recall and F1 score and a micro-average accuracy computed across all queries. The run that has implemented some simple query expansion techniques (IIRG3) performs slightly better than the initial run which doesn't implement this feature (IIRG1). Our results for both runs are very close to the median. While we have not yet had sufficient time to examine the Gold Standard for this task, early inspections seems to indicate that our system performs well when identifying entities that do not exist in the knowledge base. For future runs we hope to incorporate the wiki-text and/or the source collection as a feature of our query distance calculations in order to disambiguate query entities from KB entities where they exist.

5 Conclusions

This year we participated in three KBP tasks, Regular Slot Filling, Temporal Slot Filling and Monolingual Entity Linking. Our Regular Slot Filling approach achieved second place amongst the teams that did not access the web and our Entity Linking results were very close to the median results of those runs that did not use wiki-text. Slot Filling remains a difficult task and systems struggle to achieve F-measure higher than 30%. It is clear that we have traded high precision for low recall in the Regular Slot Filling task, the poor performance of our Regular Slot Filling system directly correlates with the low coverage of our Temporal Slot Filling system. In future runs, we hope to adopt a more aggressive strategy for selecting candidate slot values so that we may not only evaluate more of the candidate patterns we have generated, but also in order to participate in more of the additional Temporal tasks, given their dependency on good quality slot-values.

References

L. Byrne and J. Dunnion. 2010. UCD IIRG at TAC 2010. In *Proceedings of the TAC 2010 Workshop*. Nist publication.

Ounis I., Amati G., Plachouras V., B. He, Macdonald C., and Lioma C. 2006. Terrier - A High Performance and Scalable Information Retrieval Platform. In *ACM SIGIR06 Workshop on Open Source Information Retrieval (OSIR 2006)*.

A. Levenshtein. 1966. *Binary codes capable of correcting deletions, insertions and reversals*, volume 10(8). Soviet Physics Doklady.

Marius Pasca. 2003. *Open Domain Question Answering from Large Text Collections*. Center for the Study of Language and Information.