UCD IIRG at TAC 2010 KBP Slot Filling Task

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 paper describes the IIRG's first implementation of a system for automatic Knowledge Base Population (KBP). The Text Analysis Conference (TAC), first organised by NIST in 2008, promotes further research in Natural Language Technologies. In 2009, NIST added a Knowledge Base Population Track to TAC, the goal of this track was to promote research in to the automatic population of knowledge bases.

1 Introduction

This is the first year that the IIRG (Intelligent Information Retrieval Group) has participated in TAC's Knowledge Base Population Track. The 2010 Knowledge Base Population Track is composed of 2 related tasks Entity Linking (EL) and Slot Filling (SF). Entity Linking is the task of linking entity mentions in unstructured texts to entities in the knowledge base. Slot Filling involves the acquisition of novel values for the predefined attributes of the entities in the knowledge base. We focused on the Regular Slot Filling and Surprise Tasks in this year's track.
Participants are given a list of target entities to process. There are 2 generic entity types included in the Slot Filling task: PER (person) and ORG (organisation). Each Slot Fill query contains a name-string (target-entity), a docid which refers to the context document associated with the entity, an entity type (PER or ORG), a node id which refers to a representation of the entity in the knowledge base if one exists and an optional ignore field containing a list of slots to ignore. For example, (Babyshambles, eng-WL-11-174595-12968292, ORG, E0805901, org:founded)[1] is a slot query for the band Babyshambles. For each entity type there is a pre-defined set of attributes or "slots" to be populated with slot values. KBP 2010 defined 16 slots for organisations and 26 slots for persons. A system should harvest information from the document collection in order to populate these slots. Systems are expected to return novel slot-values, a slot can either be single-valued (e.g. per:date_of_birth) or list-valued (e.g. per:parents). A system should return NIL as the slot-value for a slot for which no novel information has been extracted for that slot.
The Slot Filling task is quite similar to the task of Question Answering. QA systems aim to return an answer to the user in response to a natural language question, SF systems aim to return a slot-value (exact answer) to the user 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 Surprise Slot Filling Task, akin to the regular task, tests the adaptability of a slot filling system under time constraints. The task of Slot Filling could be viewed as a modified QA task where the questions remain the same and only the target or focus of the question changes. Given the similarities between Slot Filling and Question Answering and that we are also currently investigating QA techniques, we decided to approach the SF tasks from a QA perspective.

[1] SF215 from TAC 2010 Regular Slot Filling Task

2 Question Answering System Architecture

The goal of a Question Answering (QA) system is to improve on the output of traditional Information Retrieval systems by returning an answer rather than a ranked list of potentially relevant documents in response to a natural language question.

While research into QA systems continues to refine the approaches taken and improve the outputs by returning more exact answers, they are generally based on a prototypical architecture (Voorhees, 2003). This architecture, described by Pasca (Pasca, 2003) and used by many QA systems (Isozaki, 2004), (Cui et al., 2004), consists of three main modules: Question Processing, Passage Retrieval and Answer Selection. A question is firstly analysed syntactically and semantically to determine what the question is asking about, eg a person, a date, a location, etc. A query is also constructed from selected question terms and is given to the Passage Retrieval module, which identifies passages of text that are likely to contain an answer. The Answer Selection module processes the candidate answer-bearing passages and selects the phrase that is most likely to be a correct answer. It should be noted that processing of a question in this classic QA architecture is normally serial and thus the overall performance of a system is bound by its weakest link.

2.1 Question Processing

The Question Processing module of a QA system identifies the type of information the question is looking for and determines any information needed for subsequent modules. Most QA systems perform some form of Question Classification. Classification of a question usually identifies the semantic type of the entity being sought, thereby identifying the Question Type. Determining the Question Type allows a system to determine any additional constraints on the expected answer and derive an Expected Answer Type. A query is also constructed from the question terms; this query will later be presented to the Passage Retrieval component.

It is known that a high number of errors in question-answering can be attributed to errors in question analysis (Moldovan et al., 2002). Most of the work performed by the QA system is based upon identifying the correct Question Type, as a failure to do so practically disables the entire system: if a question is incorrectly classified, the Passage Retrieval module will retrieve passages containing the wrong type of phrases and the Answer Selection module will extract answer phrases of the wrong type. High quality Question Processing is thus crucial for the overall performance of a QA system.

2.2 Passage Retrieval

Irrespective of the way in which the classic architecture is implemented by a QA system, the QA pipeline almost always involves searching for and retrieving documents as a means of narrowing down the potential documents to be searched (Voorhees, 2003). Given that a typical QA system deals with as much text as a Document Retrieval system, and given that the user will require an answer in a reasonable amount of time, most QA systems include an IR search component which treats the question as a query and returns a list of relevant documents or segments of documents (Burger et al., 2001), (Na et al., 2002). In the context of a QA system, this IR module is usually referred to as the Passage Retrieval or Document Retrieval module. The function of this module is not to find the actual answers to the questions but to identify those passages or documents that are most likely to contain an answer. Passages can be segments of text or entire documents.

The performance of the Passage Retrieval component, especially in terms of passage recall, is critical for the overall success of the entire QA system. Collins-Thompson et al have identified that there is a consistent relationship between the quality of initial document retrieval and overall system performance (Collins-Thompson et al., 2004). If the Passage Retrieval phase fails to identify any of the answer-bearing segments then the Answer Selection module will inevitably fail, as even the most linguistically rich answer selection methods cannot find an answer phrase that does not exist. In a comparison (Roberts, 2002) of passage retrieval and document retrieval using the University of Sheffield's QA System (Scott and Gaizauskas, 2000), document retrieval was found to be less effective at producing a greater number of answer-bearing passages. In general, passage retrieval methods based on NLP techniques produce results that are more accurate.

2.3 Answer Selection

The final phase of the QA pipeline selects and extracts the phrase that is most likely to contain an answer. The Answer Selection module searches the candidate answer-bearing passages returned by the PR module and selects phrases that are of the Expected Answer Type. The answer set identified is usually ranked according to the likelihood that a given phrase contains an answer. The module will then select the best answer from the candidate answer set. An answer may be an exact answer or a short snippet of text containing the answer.

3 Slot Filling Approach

We have adapted the classical QA architecture for use in the Regular Slot Filling task (Figure 1). The SF pipeline consists of three main modules: Query Processing, Passage Retrieval and Slot-Value Selection.

3.1 Pre-Processing

In the pre-processing phase we index all the documents of the document collection using Terrier (Ounis 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] During this pre-processing phase we inspected the available data from the previous TAC Slot Filling task. We extracted training passages from the document collection, these passages consisted of sentences which contained a mention of the target-entity and/or a given slot-value, as well

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

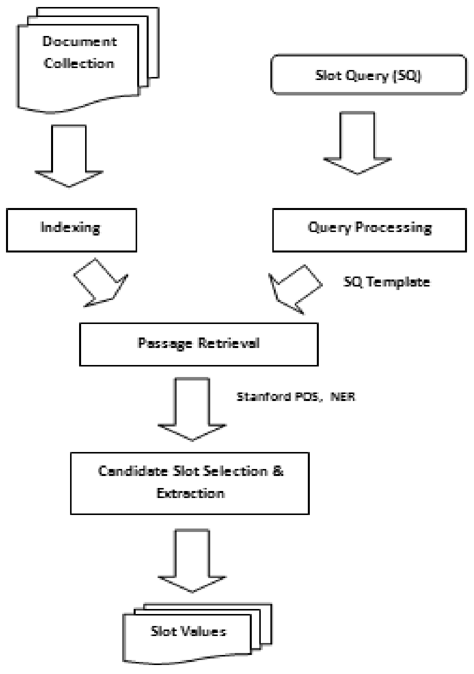

Figure 1: SF System Architecture

as sentences in close proximity to these sentences. A set of initial candidate patterns are then generated around the slot value using a combination of the Stanford POS tagger and Stanford NER tools[3]. A final set of candidate patterns is then created manually from the initial set, to be used later in the selection process. An example of one of the candidate patterns created for the slot per:spouse:

```
<person> VB:marry <target-entity>
```

the verb encountered is reduced to its base form. We also identified any equivalent questions within our own question repository, for example per:age was found to be equivalent to a number of questions including:

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

This process allowed us to augment candidate patterns with previously learned patterns and useful query terms discovered whilst processing questions from previous QA Tracks such as those in TREC(Voorhees, 1999).

3.2 Query Processing

The slot-name lends itself to classifying the type type of information we are looking for so we bypass any equivalent non-trivial question classification phase. For each of the slot queries we built a query template as input for the Passage Retrieval module and mapped each of the slot names to an Expected Value Type (EVT), e.g. the EVT of slots per:spouse and per:siblings is PERSON.

The template contains the PR input queries, additional query terms and relevant phrases extracted from the context document. It is not always sufficient to just use the target-entity as the initial query, in some cases the target-entity might be an abbreviation which may need to be expanded in order to provide relevant search results. Conversely, the target entity may be known more frequently in an abbreviated form. This template will aid us in disambiguating a given target-entity from similar entities in the document collection. Assuming that the target-entity was the focus of the context document, we relied on this text to create the query template.

3.3 Passage Retrieval

The Passage Retrieval module locates passages which are likely to contain a slot-value. 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 process is applied to is as narrow as possible. Terrier retrieves a set of documents likely to contain a slot-value in response to a query template. Once a list of candidate documents has been generated in response to a query, segments of text are extracted at a sentence level. Sentences containing a mention of the entity and also sentences in close proximity to the entity are input to the Slot-Value Selection module as candidate slot-bearing passages.

3.4 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 was extracted. The Slot-Value Selection module processes the candidate slot-bearing passages returned by the PR module and filters out passages that are of the Expected Value Type. If the passage conforms to the set of candidate patterns associated with the slot name then the EVT is extracted and added to the candidate answer set. The answer set identified is ranked using the ranking factor “frequency-of-occurrence”, with the most frequently occurring answer extracted as the slot value. For some slots e.g. per:spouse and per:age it is not sufficient to rank the potential answer set based on frequency-of-occurrence. The most frequently occurring value might not necessarily indicate the most recent or novel for these slots, for example per:spouse may have multiple occurrences of previously temporally viable values. For such slots, the candidate answer set was ranked using the ranking factor “most-recent-occurrence”, and the top answer i.e. the answer which occurred in the most recent news article is returned as the slot-value.

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

Regular Slot Filling	
Recall	0.18665378
Precision	0.6655173
F1 Score	0.29154077

Table 1: Results of Run Submitted to Regular Slot Filling Task

Surprise Slot Filling	
Recall	0.043564357
Precision	0.47826087
F1 Score	0.07985481

Table 2: Results of Run Submitted to Surprise Slot Filling Task

There is no threshold set to limit the number of answers generated in response to a slot-value. For list-slot-values, the system identifies and removes duplicate values from the candidate answer set, there is no ranking factor applied to this answer set and the system returns all of the answers in the candidate answer set. The system also tries to identify slot-values that already exist in the Knowledge Base, such redundant values should not be returned 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 and candidate patterns are not identified.

3.5 Surprise Slot Filling

We also participated in the Surprise Slot Filling Task using the SF pipeline described above. This is a new task for KBP2010 requiring participants to return slot values for new and previously unseen slots within a short time period. This task added 4 new slots, 3 PERSON slots (“diseases’, “awards-won”, “charity-supported”) and 1 ORGANIZATION slot (“product”) and allowed a maximum of 4 days for training the system and running the task. This task was scheduled directly after the Regular Slot Filling task and as such we had less than a day to devote to this task. The strict time constraints meant that we had limited time to generate a reasonable set of candidate patterns for these new slots. The poor results obtained in this task (Table 2) are not surprising given the time constraints and limited training data available.

4 Results

Table 1 describes the final scores obtained in the Regular Slot Filling KBP Task. Our SF system was the second best performing system in the Regular SF task. 66.5% of the slot-values that we acquired from the document collection were judged to be non-redundant values, although our system achieves very low coverage of slot-values across the entire document collection, acquiring only 18% of the novel slot-values. The system also returned a NIL value in response to a high percentage of slots. Although there are a significant number of correct NIL values in this task, analysing the NIL responses is also a meaningful evaluation. There are a number of reasons why NIL is returned as a slot-value. For example, the Passage Retrieval module may not be returning enough slot-bearing passages for the selection process, the expected answer type may not have been discovered in the candidate sentences, there may be no slot-values available for a given attribute in the document collection e.t.c. It is important to ensure that the system is returning a NIL value only when there are no learnable slot-values for a given attribute. In general, this is also an interesting challenge from a QA perspective: rather than aggressively searching candidate passages for any answer, should systems return no answer where no “exact” answer exists? The density of NIL values in the dataset makes this a very labour-intensive analysis, and while we have not yet completed this process we have discovered various examples of the aforementioned reasons why the system returned NIL as a value. Table 2 describes the final scores obtained in the Surprise Slot Filling KBP Task. While we had very limited time to devote to this task, the poor results achieved here especially in terms of recall, highlight again the need to improve the coverage of our system.

4.1 Slot Filling Error Analysis

We have noted that our system was not aggressive enough when removing some redundant slot-values, that is, those values which already exist in the KB. While eliminating such values now has no effect on recall, it has a marginal effect on precision.

Furthermore, we now realise that we should have ensured in a more rigorous fashion that the slot-values returned conformed with the track guidelines. There are some occurrences of slot-values embedded in additional text e.g. “70-year-old” rather than “70” returned for the slot per:age. Conversely, there are examples of incomplete slot-values e.g. “security chief” should have been “national security chief” in response to the slot per:title according to the supporting document returned. Altering these slot-values marginally improves precision, but has no effect on recall.

Finally, the assumption that the focus of the supplied context document was the target-entity was not well-founded. For some slots, this resulted in less effective query expansion, low passage recall with too few candidate slot-bearing passages available for the selection process.

Score	LDC[4]	Top-1	**Top-2**
Recall	0.54061896	0.64796907	**0.18665378**
Precision	0.7013802	0.667996	**0.6655173**
F1	0.6105953	0.6578301	**0.29154077**

Table 3: Results for the KBP Slot-filling task at TAC 2010

5 Conclusions and Future Work

We have presented our first implementation of a Slot Filling system for automatic Knowledge Base Population. Given the similarities between Slot Filling and Question Answering, we chose to approach this task from a QA perspective, adapting a typical QA pipeline to handle slot-values. 66.5% of the slot-values that we did acquire were judged to be non-redundant values, though the system was found to have low coverage across the document collection. There is huge scope for improvement with respect to system recall, as the system only acquired only 18% of the non-redundant slot-values across the document collection. The low coverage of the system is also reflected in the recall scores achieved in the Surprise Task (Table 2).

Eventhough our system achieved a poor recall score for the Regular SF Task, it was awarded the second highest F1 score in this task (Table 3). It is certainly a non-trivial task to achieve reasonable performance in the slot filling task and regular slot-filling remains a very challenging task. We suspect that the poor coverage of the system is surely related to the performance of the Passage Retrieval module. Future work on the system will involve implementing more effective query expansion techniques without relying on the context document and improving the coverage of our system, especially in terms of passage recall.

References

J. Burger, C. Cardie, V. Chaudhri, R. Gaizauskas, S. Harabagiu, D. Israel, C. Jacquemin, C. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weishede. 2001. Issues, Tasks, and Program Structures to Roadmap Research in Question Answering.

Kevyn Collins-Thompson, Jamie Callan, and Egidio Terra. 2004. The Effect of Document Retrieval Quality on Factoid Question Answering Performance. In *ACM SIGIR Conference on Research and development in Information Retrieval*, pages 574–575. Poster.

H. Cui, K. Li, R. Sun, T.-S. Chua, and M.-Y. Kan (National University of Singapore). 2004. National University of Singapore at the TREC 13 Question Answering Main Task. In E.M. Voorhees and L.P. Buckland, editors, *Proceedings of the Thirteenth Text REtrieval Conference (TREC-2004)*, pages –. NIST publication.

H. Isozaki. 2004. NTT's Question Answering System for NTCIR QAC2. In *Proceedings of NTCIR-4*.

D. Moldovan, M. Pasca, S. Harabagiu, and M. Surdeanu. 2002. Performance Issues and Error Analysis in an Open-Domain Question Answering System. In *Proceedings of ACL 2002*. ACM.

S.H. Na, I.S. Kang, S.Y. Lee, and J.H. Lee. 2002. Using Grammatical Relations, Answer Frequencies and the World Wide Web for TREC Question Answering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-2002)*. Nist publication.

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

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

Ian Roberts. 2002. Information Retrieval for Question Answering. Master's thesis, University of Sheffield.

Sam Scott and Robert Gaizauskas. 2000. University of Sheffield TREC-9 Q & A System. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. NIST publication.

Ellen M. Voorhees. 1999. The TREC-8 Question Answering Track Report. In *Proceedings of the Eight Text REtrieval Conference (TREC*, pages 77–82.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC)*, pages 54–68.

[4]LDC produced a manual run for evaluation