UCD IIRG at TAC KBP 2013

Lorna Byrne, Caroline Fenlon, John Dunnion
School of Computer Science and Informatics
University College Dublin
Ireland
{lorna.byrne@ucd.ie,caroline.fenlon@ucdconnect.ie,john.dunnion@ucd.ie}

Abstract

This paper describes the IIRG system entered in the TAC 2013 Knowledge Base Population Track. The overall goal of KBP is to automatically identify salient and novel entities from multiple languages, link them to corresponding Knowledge Base (KB) entries (if the linkage exists) in a target language, then discover attributes about the entities (extract temporal spans about the attributes if there exist dynamic changes), and finally expand the KB with any new attributes (Ji et al., 2011).

1 Introduction

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. The Entity Linking tasks focus on three 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 we participated in the Regular English Slot Filling Task and submitted three runs. In an effort to improve our Regular Slot Filling system to enhance recall we decided to also implement a Machine Learning approach to solving this task and have submitted two runs which apply Naïve Bayes methods, IIRG1 and IIRG2. IIRG3 implements a rule-based pattern-matching approach to Slot Filling.

2 Regular English Slot Filling Task

The objective of Slot Filling systems is to return a slot-value in response to a slot query. Each slot query or target-entity is limited to an entity of the form PERSON or ORGANISATION. Given the similarities between the SF task and Question Answering (QA), we have adapted a classical QA architecture (Pasca, 2003) for use in the Slot Filling task. Our system follows a simple pipeline architecture and consists of three main modules: Pre-Processing, including query and document collection processing, Passage Retrieval (PR) and Slot-Value Selection. Firstly, we retrieve a set of documents in response to the target-entity from the initial query using an off-the-shelf document retrieval component, and then select and extract a slot-value based on an identified slot-value type. For Pattern-Matching methods (IIRG3), we have implemented a two-stage retrieval model, where the second stage reduces the search space to passages or segments of that adhere to generated candidate patterns.

2.1 Pre-Processing

In this phase, the source document collection was indexed using Terrier 3.0 (Ounis et al., 2006). During this phase, we also used the Slot Filling Gold Standard files from the previous SF tasks as a source of training data. A subset of the source document collection was identified as relevant training documents using the Gold Standard files. These documents were then processed using the Stanford Suite of Core NLP Tools[1]. We created a training set of examples by extracting sentences from this document collection which contained a correct slot-value. Occurrences of slot-values and target entities were annotated in each of these sentences.

For example, consider the following training sentences identified for the slot name per:date_of_birth:

```
Sean Preston was born in September 2005.
<target-entity> was born in <slot-value>.
```

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

Machine Learning Attributes						
slot-value	WL	WR	before_2	before_1	after_1	after_2
BNC Mortgage LLC	2	2	subsidiary	,	.	END

Table 1: Features identified for Machine Learning Approach

```
Hugo Chavez was born on July 28,1954.
<target-entity> was born on <slot-value>.
```

In order to generate patterns for use in any pattern-matching approaches, candidate patterns were generated around the slot-value using a combination of the Stanford Part-of-Speech (POS) tagger and Named Entity Recognition (NER) tools. To allow for greater coverage, all verbs were reduced to their canonical form. This produces a candidate pattern of the form:

```
<target-entity> VB:bear <slot-value>
```

for the slot per:date_of_birth from the previous examples.

The slot-value and target-entity often occurred within the same sentence in the training examples. Having identified all of the slot-values, it was also possible to generate an Expected Value Type (EVT) for each slot name. For example, per:date_of_birth requires a date, per:spouse requires a person and org:website requires a URL as a slot value. Sentences containing a slot-value were used as input training snippets for Machine Learning approaches. The tokens surrounding the slot-value were used to extract attributes for use in Machine Learning.

USFD at KBP 2011 (Burman et al., 2011) found promising results using Machine Learning with the words found in windows around the target entity and candidate slot value. This involved binary attributes for each word in the training data, marking its occurrence in the sentence containing the candidate answer. Rather than directly using every word from the training data as Machine Learning attributes, the tokens contained in a window around the prospective answer were extracted. The number of tokens to select from each side of the answer was controlled by two window sizes, WL and WR, denoting window size left and right respectively, which could be changed separately to assess the effect of including additional tokens to either the right or left of the answer. Consequently, the total number of these features was WL + WR.

To account for variation in tenses, plurals and other grammatical categories, the canonical form of tokens were taken as features. Punctuation was not ignored in this process; punctuation marks inside the boundaries of the windows were included without alteration. Regardless of window size, sentence boundaries were maintained. For windows extending beyond the start or end of sentences, the features were set to START or END markers.

Number of Training Examples	
Year	#Examples
2010	0.018392371
2011	0.010217983
2012	0.028610354

Table 2: Number of Input Training Examples by Year

Table 1 illustrates the extracted attributes for the following example:

Lehman Brothers said it would absorb charges and costs totaling 52 million dollars as a result of it closing down the mortgage subsidiary, BNC Mortgage LLC.

For each slot-value, the slot-value length was calculated in terms of tokens, so that we could determine if slot-value length is significant for categorising answers. The position of the candidate slot-value in the sentence is determined relative to the length of the sentence. This is calculated based on number of tokens, and does not take token length into account. Relative answer position is calculated by

$$\frac{\frac{1}{2}(s+e)}{l}$$

```
where
s = the index of the first token of the
    slot-value within the sentence
e = the index of the last token of the
    slot-value within the sentence
l = the length of the sentence in tokens.
```

The e1071 implementation of the Naïve Bayes algorithm in R[2] was trained on these features. Several different combinations of window sizes were tested by training on data from a single year and testing on the entire training set. The performance of the classifiers increased with the size of the window, except when trained on the significantly larger 2012 dataset (see Table 2). As a result, two of the largest window sizes tested (WL1WR2 and WL2WR2) were selected for

[2]http://cran.r-project.org/web/packages/e1071/e1071.pdf

use in the testing phase. The results for tests with an empty window at one side of the slot value indicate that words before the value are more useful for determining which slot it belongs to.

2.2 Passage Retrieval

The Passage Retrieval (PR) module identifies passages of text that are likely to contain a slot-value. Passages can be segments of texts or entire documents. Terrier retrieves a set of documents in response to each target-entity as an input query. This set of relevant passages are processed using the Stanford Core NLP tools and occurrences of the target-entity (query) are annotated, after which, occurrences of target entity tokens are also identified and annotated as candidate-target-entity. The sentences are then ranked based on the features they contain, that is, sentences containing a target-entity or candidate-target-entity will be ranked higher than sentences which contain no entity mentions. These sentences are then used 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 the most likely slot-value. Each returned slot-value must also contain the docid of the supporting document from which the slot-value was extracted, along with the relevant document offsets identifying the slot-value within the document. The Slot-Value selection module processes the candidate slot-bearing passages returned by the PR module. Passages that are of the Expected Value Type are selected. ML runs IIRG1 and IIRG2 ignore the sentence rank feature produced during the PR phase, considering all sentences as candidate slot-bearing passages. IIRG3 considers only the highest ranked sentences as input for the slot-value selection phase, that is sentences that contain the target-entity.

For IIRG3, if these segments of texts contain an EVT and/or comply with the candidate patterns for that slot name, the candidate slot-value is selected and added to the set of all candidate slot-values for that slot-name. Similar slot-values which occur in the KB and in the candidate answer set are identified using Levenshtein Distance (Levenshtein, 1966), thereby removing redundant values. The candidate slot-value set is then ranked according to frequency of occurrence, where the highest ranked slot-value occurs in the most documents. For list-value slots, the entire set of candidate slot-values is returned, for single-value slots, the highest ranked slot-value in the set is returned. If no candidate slot-values have been found, a value of "NIL" is returned as the slot-value. That is, if the PR module fails to return any candidate slot-bearing passages or if an EVT is not identified within these passages, or a given passage fails to satisfy any of the generated candidate patterns that slot is given a "NIL" value.

Official Scores for Submitted SF Runs			
RunID	Precision	Recall	F1 Score
IIRG1	0.018392371	0.0199262	0.019128587
IIRG2	0.010217983	0.013748854	0.011723328
IIRG3	0.028610354	0.07720588	0.041749503

Table 3: Results of IIRG SF Runs Submitted to TAC 2013

For runs IIRG1 and IIRG2, the Naïve Bayes classifier was used to select slots for potential answers in the test data. The generation of candidate slot-values was performed using every sentence in the documents linked to the target entity; sentences without some reference to the target were not excluded. Candidates were selected by extracting series of tokens tagged as the entities used in training (LOCATION, PERSON, ORGANIZATION, DATE and MISC). For each candidate answer, the classifier calculates the probability that the answer belongs to each of the possible slots. The proposed slot is the one with the highest probability. Where multiple candidate values have the same proposed slot, the one with the highest probability is selected as the slot value. If the classifier does not predict a certain slot for any candidate answers, that slot is given a "NIL" value.

3 Results

We submitted three official runs for the Slot Filling task, two runs which implemented a Machine Learning approach using Naïve Bayes methods (IIRG1 and IIRG2) and one run which implemented a rule-based pattern matching approach (IIRG3). All three submitted runs performed very poorly on this year's tasks, as illustrated in Table 3. While the pattern matching run, IIRG3, slightly out-performed both of the ML runs, neither approaches achieved any reasonable coverage of the document collection for this task, with the best run only scoring 2.9% in terms of recall.

4 Additional Experiments

An additional run was performed using a similar method to IIRG1 and IIRG2. This consisted of a move towards a more intelligent hybrid of a rule-

based system and Machine Learning. This again involved extracting tokens from a window bounding a candidate answer, creating some other attributes, and training Machine Learning classifiers.

For each slot, a Naïve Bayes classifier was used to select the correct answer(s) for the target-slot pair. In contrast to the first ML runs, this was performed on a per-slot basis; a distinct classifier was trained on data pertaining to each slot. For most slots, candidate answers were selected by extracting strings of NER tagged tokens from the XML files generated for each document. Candidate values for the org:website, per:cause_of_death and per:charges slots were extracted using regular expressions.

Each classifier predicts the probability that a slot value is correct, independently of the other possible answers. For slots needing only a single value, the candidate with the highest probability of being correct is selected. If no candidates are predicted likely to be correct (i.e. probability below 0.5), a NIL value is returned. For slots taking lists of values, all distinct candidates predicted to be correct are chosen.

Some of the other attributes used in earlier runs were retained; answer length and answer offset were again calculated. To reduce and improve the lists of candidates, only values taken from a sentence also containing the target entity were used. As another measure of the relevance of a value to the target, a binary attribute indicating the target's presence in the preceding sentence was included.

Boundary windows up to five tokens to the left and right of the candidate were considered. All combinations of window sizes were tested to find the optimal solution for each slot. This was carried out for three different token extractions: canonical forms (lemmas), NER tags, and part of speech tags. The testing for this run was carried out using gold standard data from this task for the years 2010 and 2011 to train the classifiers, which were then tested using the same dataset from 2012.

4.1 Results

The Naïve Bayes classifiers for each slot were evaluated using the F-score measure. The performance of the classifiers varies greatly between slots. As evident in Table 4, the most accurate token type and window sizes also differ from slot to slot. Predictions could not be carried out for some slots, due to the small amount of positive instances in the gold standard data.

5 Conclusions

This year we participated in one KBP task, Regular English Slot Filling. We submitted three runs for this task, one run based on pattern matching techniques and two runs which implemented Naïve Bayes methods. All of the submitted runs performed poorly at this task, with the highest F-score limited to 4%. The additional implementation of Machine Learning methods did not improve the coverage of our system, this approach did not obtain anything reasonable in terms of recall. While we have yet to complete a comprehensive error analysis, upon initial inspection it would seem that the search space identified for the ML approaches is too large, while the search space identified for pattern-matching methods is too narrow. The ML runs (IIRG1 and IIRG2) fail to achieve any reasonable accuracy in identifying slot-values due to the large volume of noise that is not filtered before the Slot-Value Selection phase begins, while the narrow search space identified in IIRG3 means that this run fails to identify many candidate slot-values and in most cases this run reverts to returning a "NIL" value. It remains an extremely challenging task for our system to achieve any reasonable recall score in the Slot Filling Task.

The additional hybrid run carried out after the task greatly outperformed the initial ML runs. Future work on this task will involve implementing an increasingly hybrid approach based on both of these approaches, perhaps using the generated candidate patterns as a means of reducing the search space for the final Machine Learning system. In addition, an approach involving a window of tokens around the target value itself, or a linked entity would potentially offer more information about the probability of the candidate value being correct and related to that specific target. Producing more positive examples, perhaps manually, may help improve classifiers that performed poorly, or could not predict outcomes at all, for slots with few instances in the gold standard data.

References

Amev Burman, Arun Jayapal, Sathish Kannan, Madhu Kavilikatta, Ayman Alhelbawy, Leon Derczynski, and Robert J. Gaizauskas. 2011. Usfd at kbp 2011: Entity linking, slot filling and temporal bounding. In *Proceedings of the TAC 2011 Workshop*.

H. Ji, R. Grishman, and H.T. Dang. 2011. Overview of the TAC2011 Knowledge Base Population Track. In *Proceedings of the TAC 2011 Workshop*. NIST publication.

I. Ounis, 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)*.

Scores for Additional ML Run						
Slot	Tag	Window_L	Window_R	Precision	Recall	F1 Score
org:alternate_names	NER	1	0	0.51	0.43	0.46
org:city_of_headquarters	lemma	2	1	0.13	0.46	0.20
org:country_of_headquarters	NER	0	1	0.27	0.36	0.31
org:date_dissolved	no predictions					
org:date_founded	lemma	3	0	0.58	0.54	0.56
org:founded_by	lemma	4	0	0.23	0.37	0.28
org:member_of	NER	1	4	1.00	0.11	0.20
org:members	NER	4	5	0.54	0.38	0.45
org:number_of_employees_members	lemma	1	1	0.64	0.75	0.69
org:parents	lemma	4	1	0.14	0.44	0.22
org:political_religious_affiliation	POS	3	5	0.11	0.71	0.19
org:stateorprovince_of_headquarters	no predictions					
org_subsidiaries	lemma	1	5	0.27	0.30	0.28
org_top_members_employees	lemma	2	4	0.48	0.77	0.59
per:age	lemma	1	4	0.65	0.81	0.72
per:alternate_names	POS	3	0	0.65	0.60	0.62
per:cause_of_death	POS	1	1	1.00	0.60	0.75
per:children	lemma	3	0	0.65	0.31	0.42
per:cities_of_residence	lemma	2	0	0.81	0.73	0.77
per:city_of_birth	lemma	3	5	0.57	0.76	0.65
per:city_of_death	lemma	1	0	0.53	0.48	0.50
per:countries_of_residence	lemma	3	1	0.60	0.48	0.53
per:country_of_birth	lemma	4	0	0.18	1.00	0.31
per:country_of_death	no predictions					
per:date_of_birth	POS	0	1	0.64	0.67	0.65
per:date_of_death	no predictions					
per:employee_or_member_of	lemma	4	1	0.70	0.74	0.72
per:origin	lemma	1	1	0.38	0.55	0.45
per:other_family	lemma	3	4	0.33	0.60	0.43
per:parents	lemma	2	0	0.55	0.33	0.41
per:religion	POS	0	1	0.45	0.56	0.50
per:schools_attended	lemma	2	1	0.86	0.84	0.85
per:siblings	lemma	2	0	0.72	0.62	0.67
per:spouse	lemma	4	3	0.63	0.53	0.58
per:stateorprovince_of_birth	POS	4	5	0.29	0.27	0.28
per:stateorprovince_of_death	no predictions					
per:statesorprovinces_of_residence	lemma	3	4	0.47	0.41	0.44
per:title	POS	1	1	0.80	0.86	0.83

Table 4: Results of IIRG Additional ML Run