UNED Slot Filling and Temporal Slot Filling systems at TAC KBP 2013. System description

Guillermo Garrido, Anselmo Peñas and **Bernardo Cabaleiro**
NLP & IR Group at UNED
Madrid, Spain
{ggarrido,anselmo,bcabaleiro}@lsi.uned.es

Abstract

This paper describes the system implemented by the NLP GROUP AT UNED for the Knowledge Base Population 2013 English Slot Filling (SF) and Temporal Slot Filling (TSF) tasks. For the Slot Filling task, we implemented a distant supervision approach, using Freebase as a source of training relations and news sources to retrieve training examples. For the Temporal Slot Filling task, our approach is based on learning the temporal link between relation mentions and previously identified contextual temporal expressions. This is realized using distant supervision to match temporal information from a knowledge base and textual sources. Evidence is then aggregated into an imprecise temporal anchoring interval. For both systems, we extract features from a rich document representation that employs a graph structure obtained by augmenting the syntactic dependency analysis of the document with semantic information.

1 Introduction

This paper describes the NLP GROUP AT UNED 2013 system for the English Slot Filling (SF) and Temporal Slot Filling (TSF) tasks. The goal of SF is to extract, from an input document collection, the correct values of a set of target attributes of a given entity. This problem can be more abstractly described as *relation extraction*:

Relation extraction: acquiring relation instances from text. Relation instances are binary relations between a pair of entities, or an entity and an appropriate attribute value:

$$\langle entity, relation, value \rangle$$

The TSF task asks to provide additional temporal anchoring of extracted relations. Many interesting relations are dynamic: their truth value is dependent on time. We call these relations *fluents* (Russell and Norvig, 2010). For a known relation value, how to establish the period of time for which the value is correct? The temporal anchoring problem consists in obtaining from the document collection the temporal validity of the slot values.

Temporal anchoring: obtain the temporal validity of a relation instance:

$$\langle entity, relation, value, temporal\ anchor \rangle$$

It is possible to attempt relation extraction and temporal anchoring sequentially (a *pipelined* approach), or together (a *coupled* approach). It is also possible to consider the relation extraction problem solved and focus only on temporal anchoring. This last option is the one proposed by the KBP 2013 Temporal Slot Filling task: participants are provided with the relation instances and their goal is to anchor them temporally, extracting temporal information from a source document collection.

The rest of this paper is organized as follows. Section 2 provides a concise overview of the design of our approach. The document-level representation we use is described in some detail in Section 3. The specifics of the Regular Slot Filling subtask are described in Section 4, and those of the Temporal Slot Filling task in Section 5. Evaluation results are reported in Section 6, and our conclusions in Section 7.

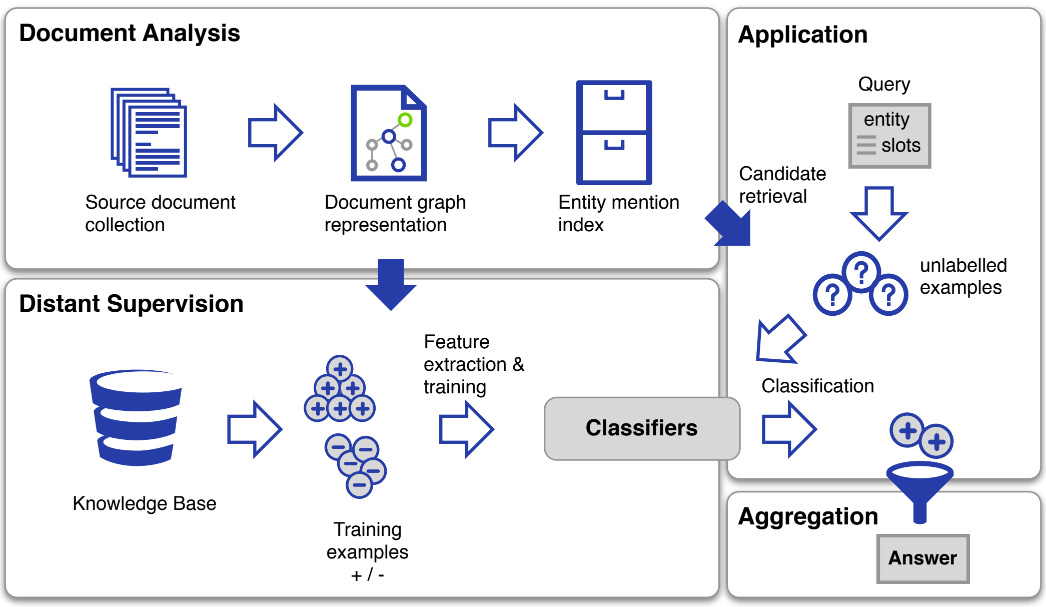

Figure 1: System overview diagram.

2 System overview

In this section, we describe the NLP Group at UNED 2013 Slot Filling and Temporal Slot Filling systems. Both hinge on a distant supervision approach, following the paradigm described by Mintz et al. (2009), which is popular among participants in this task (Agirre et al., 2009; Surdeanu et al., 2010, among others). Distant supervision automates the generation of training data by heuristically matching known facts to text. These obtained examples can then be used to train otherwise supervised extractors.

Slot Filling: distant supervision is employed to learn a battery of binary classifiers for each target slot (*extractors*).

Temporal Slot Filling: distant supervision is applied to automatically label training data to learn a *n-way* classifier to decide the *temporal link* between a relation instance and a contextual temporal expression.

Figure 1 depicts an overview of the system. It follows a straightforward machine learning pipeline design. First, the system is *trained*, using information available to learn a model for the task at hand. In the system *application*, the models we have learnt are used to extract new information. Last, an *aggregation* phase is necessary to reconcile possibly conflicting pieces of evidence, extracted from multiple documents.

The system training has two sub-phases: (1) document analysis, to process unstructured text and generate a useful representation of the information they contain; and (2) distant supervision, to automatically gather training examples. The document analysis phase has two sub-components:

- Document representation. We aim at capturing long distance relations by introducing a document-level representation and deriving novel features from deep syntactic and semantic analysis (see Section 3).
- Entity mention indexing. In order to obtain training examples, we will match KB entries and entity mentions in the document collection. This matching is based on an entity mention index, compiled after processing the full document collection.

Distant supervision is applied to train classifiers for both SF and TSF, although with different purposes. In the case of the SF task, the inputs (KB and document collection) are used to learn a set of *slot extractors*. Each slot extractor can decide if a given unlabelled example is an instance of that slot. For the TSF task, we learn a temporal link classifier (in the TSF task). This classifier assigns a temporal link to a pair $(relation\ mention, temporal\ information)$.

In the system *application*, the models we have learnt are used to extract new information: in SF, the extractors must decide if a given candidate value is a valid for a slot; in TSF, what is the

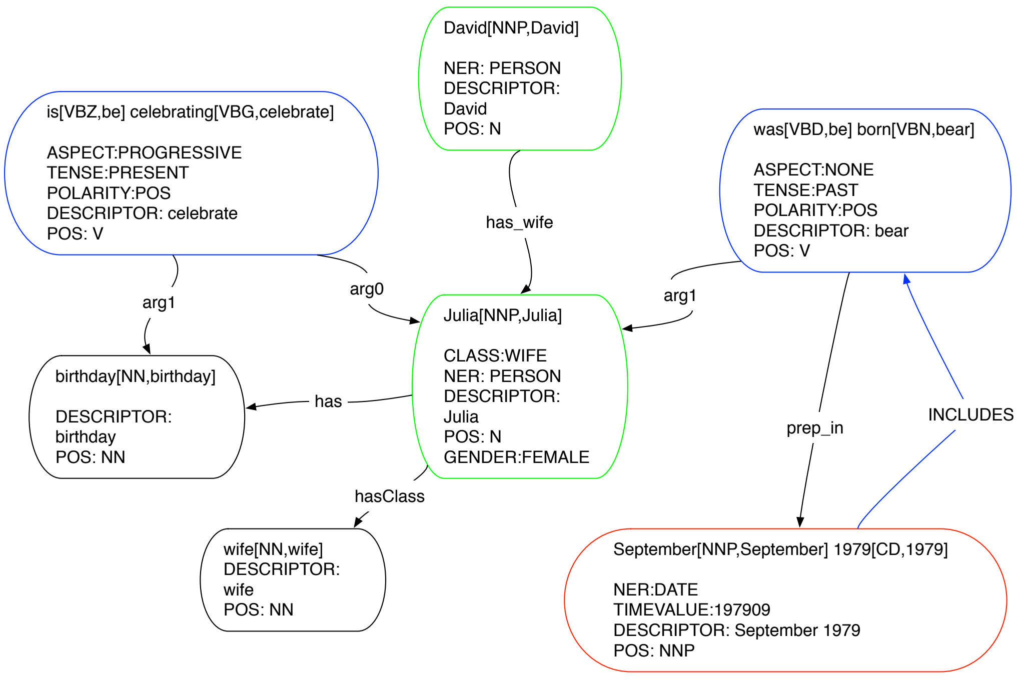

Figure 2: Document graph representation, G_D, for the sample text document "David's wife, Julia, is celebrating her birthday. She was born in September 1979".

temporal link between a candidate mention and a piece of temporal information.

The NLP GROUP AT UNED Slot Filling and Temporal Slot Filling systems build on our participation in the KBP 2011 edition, as reported in (Garrido et al., 2011). We have rebuilt the core components from the previous system, and made changes and improvements across all of them. Some of the main changes are: (1) substitute the previous IR-based passage retrieval for an entity-mention index approach; (2) improve on document representation and feature generation; (3) limited the scope of training examples to sentences, although co-reference allows to gather information from different parts of the documents; (4) in the Temporal Slot Filling task, we have integrated distant supervision into the temporal linking module.

3 Document analysis and representation

Our system relies on a rich document representation that integrates several layers of lexical, syntactic and semantic information in a compact graph structure (Garrido et al., 2012; Cabaleiro and Peñas, 2012). This document-level representation is built upon the set of syntactic dependency trees of each of the sentences, upon which the following operations are performed:

- Lexical and syntactic analysis, named entity recognition and coreference resolution, using Stanford CoreNLP (Klein and Manning, 2003).
- Labelling events and temporal expressions, augmenting dependency trees with edges representing temporal relations, using the TARSQI Toolkit (Verhagen et al., 2005).
- Collapse nodes into discourse referents.
- Rule-based simplification and normalization of the resulting graph structure.

This representation is document-level in the sense that a single graph representation is built from the set of dependency trees for each sentence. A document D is represented as a *document graph* G_D; with node set V_D and edge set, E_D. Each node $v \in V_D$ represents a word or a sequence of words (we group words in two cases: multi-word named entities and a verb and its auxiliaries). Co-referent nodes are collapsed into a single node, representing a *discourse referent*. The graph structure allows navigating between different sentences that contain mentions to the same *discourse referent*.

Each node is labeled with a dictionary of attributes: the words it contains, their part-of-speech annotations (POS) and lemmas, and their positions in the phrase and in the sentence. Also, a representative *descriptor*, which is a normalized string value, is generated from the chunks in the node. Certain nodes are also annotated with one or more *types*. There are three families of types: Events (verbs that describe an action, annotated with tense, polarity and aspect); standardized Time Expressions; and Named Entities, with additional annotations such as gender or age.

Edges in the document graph, $e \in E_D$, represent four kinds of relations between the nodes: (1) syntactic; (2) semantic relations between two nodes, such as `hasClass`, `hasProperty` and `hasAge`; and (3) temporal relations between events and time expressions.

Additional semantic information is also blended into this representation: normalization of genitives, semantic class indicators inferred from apposition and genitives, and gender annotation inferred from pronouns. A graph example is pictured in Figure 2.

4 Slot Filling system description

This section describes with more detail the implementation of the NLP GROUP AT UNED 2013 Slot Filling system. The system core component is a battery of slot-specific classifiers (*extractors*), trained using examples gathered automatically using a distant supervision approach.

Gather training examples. From a Knowledge Base (KB), we extract a set of relation triples or *instances*: $< entity, relation, value >$. For a relation instance, any textual mention of *entity* and *value* is assumed to express the *relation*. By matching the instances to the documents in the source collection, we obtain positive examples for the relation. As negative examples for a relation, we use both: (a) positive examples for any other relation; and (b) examples generated from entity-value pairs that are not connected by any relation in the KB.

Feature extraction. From positive and negative training examples, lexical and syntactic features are generated.

Learning specialized classifiers (extractors) for each target relation. The application of the extractors learned allows us to perform slot filling. We first retrieve candidate sentences, using the entity mention index. And then apply the extractors to obtain new examples for the relation.

Aggregation. Finally, aggregation is needed to produce a single system response from possibly conflicting pieces of evidence, extracted from multiple documents.

4.1 Implementation details

To implement the distant-supervision approach sketched above, an existing Knowledge Base and a source document collection are needed. For the NLP GROUP AT UNED 2013 system, we used the knowledge base Freebase.[1] The original data dump file is in RDF format, and contains over 1 billion triples. From all triples we extract those that are instances of the Freebase relations relevant to any of the KBP slots.

We decided to use a document source corpus for training that was independent from the documents used for evaluation. In particular, we used the source data from the TAC 2010 Knowledge Base Population Evaluation.[2] Note that these source documents are used only for training and are not part of the evaluation corpus for the current 2013 edition of the task.

Table 1 reports on the number of training instances extracted from Freebase, and of training examples obtained by matching Freebase and the source document collection.

Notice that, to retrieve training examples, we exploit a pre-built entity mention index. For this participation, we have used only sentences as valid passages for training and extraction. That means that both subject and value of the relation must be mentioned within the same sentence. As our document-level representation (see Section 3) includes co-reference information, we can use not only explicit mentions, but also pronouns and other referring expressions.

Each example was represented by binary features, which are inspired by previous work (Surdeanu and Ciaramita, 2007; Mintz et al., 2009; Riedel et al., 2010; Surdeanu et al., 2010; Garrido et al., 2012). For a summary of them, see Table 2.

For this task, we used a battery of binary classifiers; each of them was a SVM classifier with

[1]We downloaded a *data dump* of Freebase, dated 2013-05-26, from the site: `https://developers.google.com/freebase/data`.

[2]This data release consisted in 1 777 888 files. It includes the former TAC 2009 KBP Evaluation Source Data (LDC2009E57), which are 1 289 649 documents, most of them newswire; and up to 490 596 web text documents, most of them from earlier GALE Web Text Collection releases. For more details, see (TAC-KBP, 2011).

KBP Slot	**KB instances**	**found in some doc (%)**	**training examples**	***docs/instance***
org:alternate_names	-	-	-	-
org:city_of_headquarters	183762	6458 (3.51)	89200	13.81
org:country_of_headquarters	13748	138 (1.00)	3302	23.93
org:date_disolved	1969	144 (7.31)	4276	29.69
org:date_founded	63740	1387 (2.18)	9025	6.51
org:founded_by	21641	2314 (10.69)	73004	31.55
org:member_of	6390	1013 (15.85)	192802	190.33
org:members	6390	1013 (15.85)	192802	190.33
org:number_of_employees_members	2745	276 (10.05)	2897	10.50
org:parents	223086	1359 (0.61)	26226	19.30
org:political_religious_affiliation	2225	46 (2.07)	468	10.17
org:shareholders	-	-	-	-
org:subsidiaries	223086	1359 (0.61)	26226	19.30
org:top_members_employees	169432	5488 (3.24)	119621	21.80
org:website	470601	597 (0.13)	1692	2.83
per:alternate_names	-	-	-	-
per:age	-	-	-	-
per:cause_of_death	38275	1572 (4.11)	20371	12.96
per:charges	1271	7 (0.55)	15	1.67
per:children	205467	1752 (0.85)	10996	6.28
per:date_of_birth	1181841	5716 (0.48)	25665	4.49
per:date_of_death	427503	6333 (1.48)	45590	7.20
per:employee_or_member_of	157219	7112 (4.52)	362729	51.00
per:origin	695258	33627 (4.84	842791)	25.06
per:parents	205467	1752 (0.85	10996)	6.28
per:place_of_birth	744550	15643 (2.10)	127247	8.13
per:place_of_death	169775	3727 (2.20)	34470	9.25
per:place_of_residence	186475	9650 (5.17)	126655	13.12
per:religion	47127	1153 (2.45)	23360	20.26
per:schools_attended	319561	5210 (1.63)	18735	3.60
per:siblings	160512	1590 (0.99)	10912	6.86
per:spouse	108854	2196 (2.02)	36075	16.43
per:title	2330869	11788 (0.51)	446356	37.87

Table 1: Training data per slot breakdown. Each row lists, for each slot: the number of relation instances obtained from Freebase (KB instances); the number and percentage of instances found in some document; the number of training examples produced; and the ratio of documents to instances.

linear kernel (Joachims, 2002). We used the SVMLight implementation available at `http://svmlight.joachims.org/`. We did not tune the classifiers' default values.

4.2 Limitations of distant supervision

Unfortunately, an automatic labelling procedure such as the one described does not produce training examples for every interesting target relation. Only relations *popular* enough to be included in the initial Knowledge Base schema and populated with enough instances can be used for distant supervision. For any relation not verifying these requirements, an alternative procedure must be used. Methods based on *bootstrapping* (Brin, 1998; Agichtein and Gravano, 2000) are an alternative to distant supervision. Iteratively, examples of a relation (*seed tuples*) are used to retrieve textual instances of the relation; those mentions can be abstracted into extraction patterns (*seed patterns*), that can then be used to search for addi-

Feature name	**description**
path	dependency path between ENTITY and VALUE in the sentence
X-annotation	NE annotations for X
X-pos	Part-of-speech annotations for X
X-gov	Governor of X in the dependency path
X-mod	Modifiers of X in the dependency path
X-has_age	X is a NE, with an age attribute
X-has_class-C	X is a NE, with a class C
X-property-P	X is a NE, and it has a property P
X-has-Y	X is a NE, with a possessive relation with another NE, Y
X-is-Y	X is a NE, in a copula with another NE, Y
X-gender-G	X is a NE, and it has gender G
V-tense	Tense of the verb V in the path
V-aspect	Aspect of the verb V in the path
V-polarity	Polarity (positive or negative) of the verb V

Table 2: Summary of features included in the SF model. X stands for ENTITY and VALUE. Verb features are generated from the verbs, V, identified in the path between ENTITY and VALUE.

```
N:Prep(X,Y) <- X:N:hasClass, N:Y:Prep, prep(Prep), X\==Y.
N:Prep(X,Y) <- X:N:is, N:Y:Prep, prep(Prep), X\==Y.
has:N(X,Y) <- Y:N:hasClass, X:N:Has, has(Has), X\==Y.
has:N(X,Y) <- Y:N:is, X:N:Has, has(Has), X\==Y.
has:N:Prep(X,Y) <- X:N:Has, N:Y:Prep, has(Has), prep(Prep), X\==Y.
V:N:Prep(X,Y) <- V:X:arg0, V:N:arg1, N:Y:Prep, prep(Prep), X\==Y.
V:N:of(X,Y) <- V:X:arg0, V:N:arg1, Y:N:Has, has(Has), X\==Y.
```

Listing 1: Graph mining patterns. On the left side of the rules are the relations, and on the right side are the constraints for the patterns. Syntax is Prolog.

tional tuples. Instead of a large number of annotated examples, a small set of seeds (either seed patterns or seed tuples), is needed to initiate the extraction process.

We experimented with a similar approach for three of the task slots: (1) `org:shareholders`, which is not represented in Freebase's schema; (2) `per:age`, whose values can be derived from *date of birth*, but for which finding training examples would require some reasoning about document dates; and (3) `per:charges`, which is in Freebase, but produced very little training examples.

The basis of this approach is to use 7 patterns over the graph-based document representation explained in section 3. These patterns are aimed at over-generating possible expressions of open relations, without reference of any particular predefined set. The patterns are shown in Listing 1.

After processing the complete collection, frequencies are aggregated. Now, if we are interested in gathering some seeds for a particular relation we can prepare a query over the resource and retrieve them. For example, the relation `org:shareholders` is defined between two organizations (ORG) and we are looking for expressions involving "share". The result of this query is shown in a aggregated view in Table 3. Table 4 shows the number of additional training examples generated for each slot.

Pattern	instances
ORG . buy:share:of . ORG	56
ORG . sell:share:of . ORG	33
ORG . hold:share:of . ORG	33
ORG . shareholder:in . ORG	30
ORG . own:share:of . ORG	26
ORG . purchase:share:of . ORG	13
ORG . acquire:share:of . ORG	12
ORG . downgrade:share:of . ORG	9
ORG . say:shareholder:of . ORG	8
ORG . say:shareholder:in . ORG	7
PER . age:of . NUMBER	66
PER . turn:in . NUMBER	6
PER . face:charge:of . CHARGE	204
PER . face:count:of . CHARGE	95
PER . face:allegation:of . CHARGE	6
PER . face:one:of CHARGE	3

Table 3: Some of the most productive patterns used to obtain relation instances for the `org:shareholders`, `per:age` and `per:charges` slots. Notice instances are not necessarily unique.

KBP Slot	instances	examples	*docs/instance*
org:shareholders	366	18822	51.43
per:charges	292	16391	56.14
per:age	16	32	2.0

Table 4: Additional examples. For these slots, we tried obtaining additional training examples by pattern matching and bootstrapping. Here, instances are de-duplicated.

4.3 Description of the runs submitted

The main difference among the two runs submitted is in the aggregation of pieces of evidence from different documents. The classification process yields a predicted class label, plus a real number indicating the margin (linear distance from the example to the SVM hyperplane boundaries).

SF RUN 1: Aggregated. The answer filler is chosen after aggregating the scores obtained for 'equivalent' extractions. Each candidate filler is normalized, and the classification margin obtained from the SVM classifier is used as a measure of the confidence of the extraction (after normalization). The filler is given as score the sum of the normalized scores from all extractions of the same filler.

SF RUN 2: Not Aggregated. For this run, the scores are not aggregated. Rather, the response with the largest margin value is returned as output.

5 Temporal Slot Filling system description

The TSF task consists in obtaining temporal constraints for the validity of a relational fact. In all but the simplest cases, multiple pieces of temporal evidence have to be considered to make a decision. Both explicit and implicit temporal pieces of information are potentially useful signals to anchor a relation mention. To address the full problem, it is important to assess what are the sub-problems involved. Our approach for Temporal Slot Filling is based on the following methodology (Garrido et al., 2012):

Temporal information extraction and representation. The first step is to represent the temporal information available that is useful for the task. There are many temporal cues around a relation mention:

- **Document-level metadata.** The *document creation time* (DCT), if available, is useful to temporally anchor the relations expressed in the document.
- **Contextual temporal expressions.** Temporal evidence also comes from the temporal expressions present in the context of a relation.

Selecting relevant temporal information. For each document and relational instance, we have to select those pieces of temporal evidence that are *relevant* to the relation, and discard those that are not.

Learning the temporal links. The third step is deciding how a relational fact and its relevant temporal information are themselves related. We use the term *link* for a relation between a temporal expression (a date) and an event; we want to avoid confusion with the term *relation* (a relational fact extracted from text).

Temporal interval aggregation. The last step is aggregating the temporal constraints found for the same relation and value across multiple documents into a common temporal representation.

5.1 Implementation details

To extract, normalize and represent contextual temporal expressions we do the following:

- We use TARSQI to extract temporal expressions and link them to events. In particular, TARSQI uses the following temporal links: *included*, *simultaneous*, *after*, *before*, *begun_by* or *ended*.
- We focus also on the syntactic pattern [*Event-preposition-Time*] within the lexical context of the candidate entity and value.
- Both are normalized into one from a set of predefined temporal links: *within*, *throughout*, *beginning*, *ending*, *after* and *before*.

Document meta-data, such as the *document creation time* is also useful to anchor relations temporally, and simple to extract, when available.

For each document and relational instance, we have to select those temporal that are *relevant* to the relation, and discard those that are not related to it. We compare two approaches:

DCT baseline: This baseline uses only document level meta-data (the document creation time) to anchor relations mentioned in a document. We assume that the DCT signals a point in time when the mentioned relation is true. In other words, it assumes a *within* temporal link from the document creation time to any relation expressed inside the document. In a previous participation in the TSF task (Garrido et al., 2011), we found out that using the document creation time provides a strong baseline.

Contextual temporal information: We define the *contextual* temporal expressions of a relation mention as those that are connected in the document graph to the *subject* and *object* of the relation. Note that this definition crosses sentence boundaries, using coreferent mentions. In our particular implementation, we limited the context to the *three* time expressions closest to the shortest path between relation subject and object.

The NLP GROUP AT UNED 2013 TSF system implements a distantly supervised classifier to learn the temporal links between relation instances and contextual temporal expressions. Similar classification systems have been proposed for this task (Artiles et al., 2011; McClosky and Manning, 2012). The temporal link classifier works as follows:

- Gather training relation instances that have some temporal information from an existing knowledge base (Freebase).
- Distant supervision: heuristically matching

relation instances and textual sources, to obtain training examples.

- Generate a feature representation for evaluation candidates. Our features were extracted from our document-level representation.
- Learn a temporal link classifier.

The classifier is a classical maximum entropy model, that learns the probability that a relation mention r has temporal link t:

$$P(t \mid r) = \frac{1}{Z} \sum_{i=0}^{N} w_i f_i$$

The MaxEnt classifier learns the weights, w_i, from the binary features f_i provided in training. We used the MALLET (McCallum, 2002) implementation of MaxEnt.[3]

To aggregate the individual pieces of evidence found across different documents, we use an heuristic procedure, similar to (Artiles et al., 2011). The desired output is an ordered 4-tuple of time points: (t_1, t_2, t_3, t_4). It should hold that: $t_1 \leq t_2$, $t_3 \leq t_4$, and $t_1 \leq t_4$. If we found that a relation started after two dates d and d', where $d' > d$, the closest constraint to the real start of the relation is d'. Mapped to temporal constraints, it means that we would choose the latest t_1 possible. Following the same reasoning, we would want to maximize t_3. On the other side, when a relation started before two dates d_2 and d_2', where $d_2' > d_2$, the closest constraint is d_2 and we would choose the smallest t_2. In summary, we will maximize t_1 and t_3 and minimize t_2 and t_4, so we narrow the margins.

5.2 Description of the runs submitted

Two runs use contextual temporal information. They differ on the labelling used for temporal links. The temporal linking component of the system solves two problems, as distinguished in the methodological discussion above: (1) selecting relevant temporal information; and (2) learning the temporal links between relevant temporal evidence and the target slot. Considering UNRELATED as one of the labels, both problems are posed as a single n-way classification one. Given a piece of evidence, the classifier decides the temporal link to the relation, which might be UNRELATED.

[3] `http://mallet.cs.umass.edu/`

A question that we want to address is which temporal expressions are useful to anchor a relation and which are unrelated, and should be ignored. In particular, if a contextual temporal expression marks a time before or after the relation, *should that be considered unrelated?* Earlier research (Artiles et al., 2011; McClosky and Manning, 2012) has considered before/after temporal expressions as unrelated. To answer this question, we have run two system configurations:

TSF RUN 1: Contextual temporal expressions. 7 labels. In this run, the n-way classifier uses seven classes: BEGINNING, ENDING, WITHIN, THROUGHOUT, UNRELATED, BEFORE and AFTER.

TSF RUN 2: Contextual temporal expressions. 5 labels. In this run, the n-way classifier uses five classes: BEGINNING, ENDING, WITHIN, THROUGHOUT and UNRELATED. Temporal expressions that were annotated as BEFORE/AFTER in 1 are annotated as UNRELATED.

The other run is the following baseline:

TSF RUN 3: DCT-WITHIN Baseline. The baseline uses only the document creation times of documents containing a mention of the relation as temporal signal.

Label	Training examples RUN 1. 7 LABELS	RUN 2. 5 LABELS
WITHIN	776	
THROUGHOUT	616 344	
BEGINNING	29 337	
ENDING	18 267	
BEFORE	106 758	–
AFTER	121 503	–
UNRELATED	1 771	230 032
Total	894 756	

Table 5: Number of training examples. Labelling breakdown. In Run 3, UNRELATED temporal expressions are labelled as either BEFORE, AFTER, or UNRELATED

6 Evaluation and results

In this section, we present evaluation and results for the two tasks: English Slot Filling and Temporal Slot Filling. For some of the slots, we were not able to retrieve training data, and therefore our system produced no responses: `org:alternate_names`, `org:date_dissolved`, `per:alternate_names`, and `per:other_family`. We exclude these slots from our analyses below.

6.1 Slot filling results

An unfortunate but trivial bug caused our system to produce responses only from a fraction (about a third) of the Gigaword documents in the source collection. As a consequence, recall of our Slot Filling submissions was needlessly low. Table 6 shows the official results for both runs, and the evaluation of the Fixed Run 1, that includes all Gigaword documents from the document source collection. Figure6.1 shows the evaluation of the two submitted runs and a new one including the previously missing Gigaword sources.

The fixed runs were evaluated with the official task scorer, ignoring case of the fillers and filler provenance.[4] While average precision is similar, recall increases, producing a score of $F_1 = 0.173$. Around 30% of the responses had no assessment in the pooled responses.

Does aggregation help? We hypothesized that the aggregation of pieces of evidence extracted from multiple documents would provide better results than selecting the most promising pieces. In order to compare this two approaches, we run two experimental settings. In one, Run 2, the values from the best scored extractions are kept. In the other, Run 1, the scores from multiple extractions are aggregated, and those aggregations with better scores are produced.

The aggregated run (Run 1) is better than the not aggregated one (Run 2): Run 1 represents a modest 5.4% gain over Run 2.

Let us break down the results by slot, and test whether the difference among the two settings is statistically significant. Figure 6.1 showed the F_1 scores per slot for each of the submitted runs. The difference is statistically significant at a 0.05 significance level (the p-value is 0.024).

Nevertheless, the aggregated and not aggregated configurations of the fixed runs are not statistically significantly different. This has to be weighted by the fact that the evaluation is coarser than the manual judgements, and a number of responses from each run did not receive judgements.

[4]This is obtained by lenient matching with the official task scorer (program arguments: `nocase`, and `anydoc`). That is, we only match the fillers (ignoring case) and do not check for provenance. Notice than leaving out provenance, two judgements might be inconsistent if two responses with the same filler but different offsets received different judgements. In those cases, we consider the filler correct if any judgement considered it correct. In not considering offsets, the evaluation is more lenient than the official one. On the other hand, there were up to 394 system responses that had no assessment, and some of them could be correct.

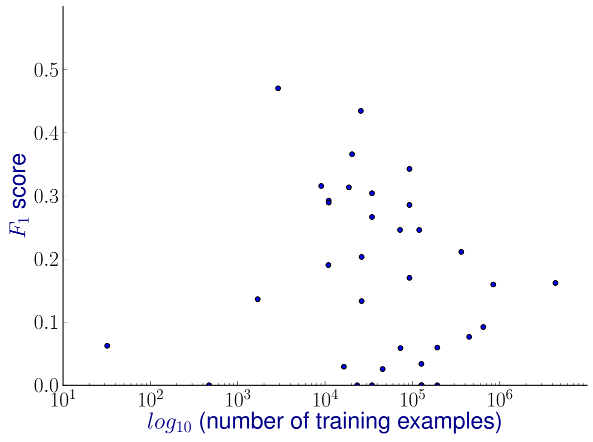

Figure 4: Slot Filling (Run 1): number of training examples per slot does not correlate with classifier performance.

Quality or quantity of training data? While manually annotated data is expensive to obtain, distant supervision offers a way to cheaply produce large quantities of training data. As the automatic labelling is noisy, quality of the training data is a concern. Does having large quantities of training data help to overcome the issues of quality?

Our distant supervision procedure is uniform for all slots, but the quality of the training data per slot varies, as does the number of training examples obtained. Do the classifiers for the slots with more training data behave better than those with less training? In our experimental setting, it does not: the number of training examples per slot do not correlate with the F_1 scores. Figure 6.1 shows the lack of trends between those variables.

Run	Precision	Recall	F_1
SF Run 1	0.176	0.093	0.122
SF Run 2	0.167	0.089	0.116
Fixed SF Run 1	0.172	0.174	0.173

Table 6: Slot Filling. Official Scores and Fixed scores, computed after including all Gigaword documents.

Obtaining training data for unpopular relations. In the absence of tuples form the Knowledge Base, we proposed to retrieve examples by mining the documents with a set of manually defined rules. We did so for three of the slots: `org:shareholders`, `per:age` and `per:charges`.

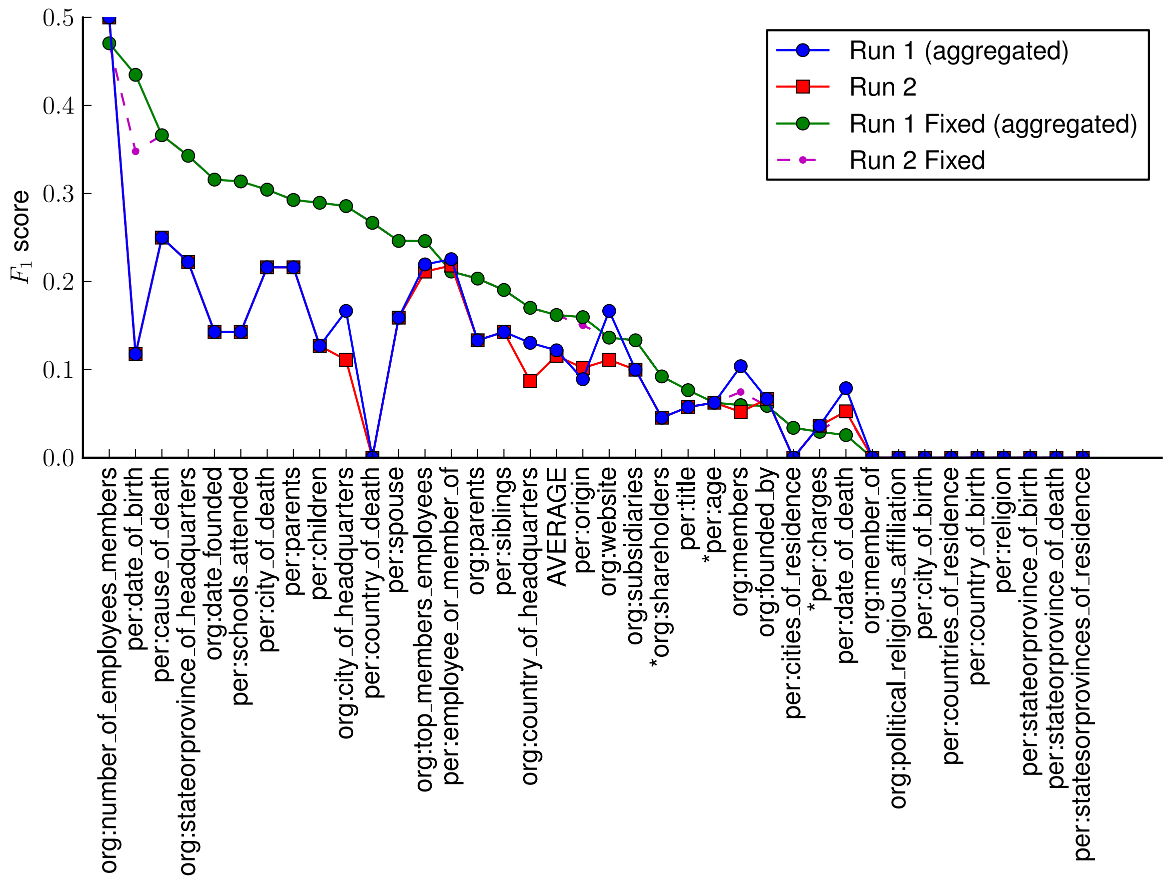

Figure 3: Slot Filling, run performance comparison. F_1 for each of the task slots. Shows the results per slot for each of the submitted runs, and the fixed run including all Gigaword documents. The slots for which we obtained training examples using manually defined rules (see Section 4.2) are marked with a star ($*$).

Although the proposed solution is an alternative to not having training data (for each of the three relations, the system produces valid responses), the results show that these three relations obtain results below average. Obtaining relations tuples through the proposed procedure is noisy, and not suitable for distant supervision. Some effort in producing cleaner training examples, through some form of supervision, is needed to obtain relation tuples for these relations prior to training.

6.2 Temporal Slot Filling

Table 7 summarizes the results for the Temporal Filling task. The DCT Baseline (TSF Run 3) obtains the best results, which are hard to beat by using only contextual temporal expressions. Figure 6.1 show a per-slot breakdown. The performance for each slot varies widely. This can be caused by the way in which examples for each of the slots are retrieved: the *residence* slots are more ambiguous, and therefore the examples we use for temporal extraction are less reliable.

As it is also shown in Table 7, the contextual information approaches are able to retrieve more correct temporal constraints than the DCT baseline. However, the quality of these temporal constraints is worse.

Run	# correct constraints	avg constraint score	official score
TSF Run 1	385	0.061152	0.116
TSF Run 2	290	0.07413	0.106
TSF Run 3	281	0.105778	0.148

Table 7: Temporal Slot Filling Scores. We report the official scores, and also the number of individual constraints judged as correct and the average score per correct constraint. While TSF Run 1 finds more correct constraints, they are scored worse on average. In aggregated, TSF Run 3 outperforms the two classifier-based runs.

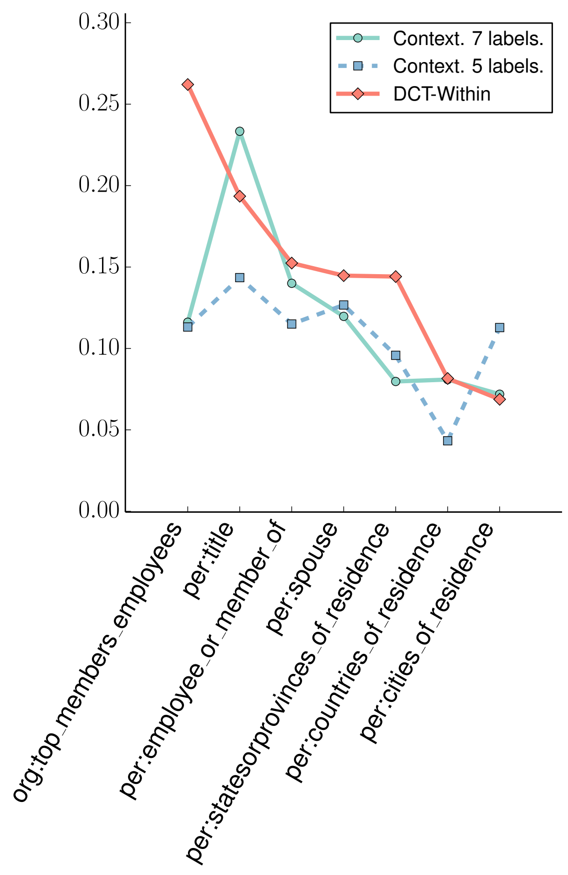

Figure 5: Temporal Slot Filling, performance per slot.

Why the DCT is a strong baseline? In the previous edition of this task, it was reported by the organizers that in the diagnostic task "evaluation documents included a much higher concentration of explicit time information than regular newswire documents" (Ji et al., 2011). This effect might still be present in the present evaluation.

Which of the proposed labelling schemes is better? In Section 5.2, we described two temporal linking configurations. In TSF Run 2, there are 5 possible labels: UNRELATED and four other. In TSF Run 1, the UNRELATED labels are separated in either BEFORE, AFTER, or still UNRELATED.

Despite having different number of labels, the configurations TSF Run 2 and TSF Run 1 are equally difficult for the MaxEnt classifier. We tested this by performing 10-fold cross validation of the two classifiers. The average accuracy of TSF Run 2 (5 labels) temporal link classifier is 0.83, while the average accuracy of TSF Run 1 (7 labels) is 0.84. The difference is not statistically significant.

The DCT baseline proves hard to beat by the classifier-based approaches.

The three approaches are able to capture a different number of temporal constraints.

7 Conclusions and future work

In the Slot Filling task, we have shown that aggregation of the answers from multiple documents produces an improvement over selecting the most promising extractions. Still, there is room for improvement in the aggregation method.

We have seen how, for distant supervision, quantity of training data does not correlate with performance across different relations. Quality of training data is an important concern for distant supervision methods.

Unpopular relations are at a disadvantage under distant supervision schemes, as it is difficult to gather training data for them. Finding relation instances by mining the documents for certain syntactic structures is an alternative to having Knowledge Base tuples. Nevertheless, the produced instances are noisy, causing poorer results when used to gather training examples for distant supervision. More careful training gathering methods need to be studied.

In the Temporal Slot Filling task, we have found that the Document Creation Time baseline is hard to beat. Nevertheless, contextual information approaches are able to retrieve more correct temporal evidence than the DCT baseline. In the immediate future, we will:

- Better aggregation schemes of individual temporal constraints.
- Design a system configuration that leverages both the DCT baseline and contextual temporal information.

Acknowledgments

This work was partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements 288024 (LiMoSINe project), and by the Spanish Government (MINECO) in the framework of the European Community's CHIST-ERA program (READERS project). The authors would also like to thank Hoa Trang Dang and the rest of the TAC KBP Workshop organizers for their excellent effort.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: extracting relations from large plain-text collections. In *DL'00*.

Eneko Agirre, Angel X. Chang, Daniel S. Jurafsky, Christopher D. Manning, Valentin I. Spitkovsky, and Eric Yeh. 2009. Stanford-UBC at TAC-KBP. In *TAC 2009*, November.

Javier Artiles, Qi Li, Taylor Cassidy, Suzanne Tamang, and Heng Ji. 2011. Cuny blender tackbp2011 temporal slot filling system description. In *Proceedings of Text Analysis Conference (TAC)*.

Sergey Brin. 1998. Extracting patterns and relations from the World Wide Web. In *WebDB'98*.

Bernardo Cabaleiro and Anselmo Peñas. 2012. Representación gráfica de documentos para extracción automática de relaciones. *Procesamiento del Lenguaje Natural*, 49(0). In Spanish with English Abstract.

Guillermo Garrido, Bernardo Cabaleiro, Anselmo Peñas, Álvaro Rodrigo, and Damiano Spina. 2011. A distant supervised learning system for the TAC-KBP Slot Filling and Temporal Slot Filling Tasks. In *Text Analysis Conference, TAC 2011 Proceedings Papers*.

Guillermo Garrido, Anselmo Peñas, Bernardo Cabaleiro, and Álvaro Rodrigo. 2012. Temporally anchored relation extraction. In *ACL '12: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers*. Association for Computational Linguistics, July.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the tac2011 knowledge base population track. In *Text Analysis Conference, TAC 2011 Workshop, Notebook Papers*.

T. Joachims. 2002. *Learning to Classify Text Using Support Vector Machines – Methods, Theory, and Algorithms*. Kluwer/Springer.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL 2003*, pages 423–430.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

David McClosky and Christopher D Manning. 2012. Learning constraints for consistent timeline extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 873–882. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL 2009*, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In José Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 6323 of *LNCS*, pages 148–163. Springer Berlin / Heidelberg.

Stuart J. Russell and Peter Norvig. 2010. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education.

Mihai Surdeanu and Massimiliano Ciaramita. 2007. Robust information extraction with perceptrons. In *ACE07*, March.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X. Chang, Valentin I. Spitkovsky, and Christopher D. Manning. 2010. A simple distant supervision approach for the tac-kbp slot filling task. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*, Gaithersburg, Maryland, USA, November. NIST.

TAC-KBP. 2011. Proposed Task Description for Knowledge-Base Population at TAC 2011. Technical report, May.

Marc Verhagen, Inderjeet Mani, Roser Sauri, Robert Knippen, Seok Bae Jang, Jessica Littman, Anna Rumshisky, John Phillips, and James Pustejovsky. 2005. Automating temporal annotation with TARSQI. In *ACLdemo'05*.