Analysing recall loss in named entity slot filling

Glen Pink Joel Nothman James R. Curran
ə-lab, School of Information Technologies
University of Sydney
NSW 2006, Australia
{glen.pink,joel.nothman,james.r.curran}@sydney.edu.au

Abstract

State-of-the-art fact extraction is heavily constrained by recall, as demonstrated by recent performance in TAC Slot Filling. We isolate this recall loss for NE slots by systematically analysing each stage of the slot filling pipeline as a filter over correct answers. Recall is critical as candidates never generated can never be recovered, whereas precision can always be increased in downstream processing.

We provide precise, empirical confirmation of previously hypothesised sources of recall loss in slot filling. While NE type constraints substantially reduce the search space with only a minor recall penalty, we find that 10% to 39% of slot fills will be entirely ignored by most systems. One in six correct answers are lost if coreference is not used, but this can be mostly retained by simple name matching rules.

1 Introduction

The TAC Knowledge Base Population (KBP) Slot Filling (SF) consists of extracting named attributes from text. Given a query, e.g. John Kerry, a system searches a corpus for documents which contain the entity. It then fills a list of *slots*, named attributes such as (`per:spouse`, Teresa Heinz).

The top TAC SF 2013 (TAC13) system scored 37.3% F-score (Roth et al., 2013), and the median F-score was 16.9% (Surdeanu, 2013). Recall for SF systems is especially low, with many systems using precise extractors with low recall. Precision ranges from 9% to 40% *greater than recall* for the top 5 systems in TAC13, and unsurprisingly, Roth et al. (2013) has the highest recall at 33%. Closing the recall gap without substantially increasing the search space is critical to improving SF results.

Ji and Grishman (2011) and Min and Grishman (2012) identify many of the challenges of SF, and suggest that inference, coreference and named entity recognition (NER) are key sources of error. Min and Grishman categorise the slot fills found by human annotators but not found in the aggregated output of all systems. However, this approach only allows them to hypothesise the likely source of recall loss. For instance, it is impossible to distinguish candidate generation errors from answer merging errors. Roth et al. (2014) categorise these errors at a high level, without specific analysis of candidate generation pipeline components such as coreference.

In this paper, we take this analysis further by performing a systematic recall analysis that allows us to pinpoint the cause of every recall error (candidates lost that can never be recovered) and estimate upper bounds on recall in existing approaches. We implement a collection of naïve SF systems utilizing a set of increasingly restrictive filters over documents and named entities (NEs). TAC has three slot types: NE, string and value slots. We consider only those slots filled by NEs as there are widely-used, high accuracy tools available for NER, and focusing on NEs only allows us to precisely gauge performance of filters. String slots do not have reliable classifiers, and value slots require more normalisation than directly returning a token span. Otherwise, this evaluation is not specifically dependent on the nature of NEs, and we expect similar results for other slot types.

We focus on systems which first generate candidates and then process them, the approach of the majority of TAC systems. Our filters apply hard constraints over NEs commonly used in the literature, accounting for a typical SF candidate generation pipeline—matching the query term, the form of candidate fills and the distance between the query and the candidate—but not performing any further scoring or thresholding. We compare sev-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 820–830,
October 25-29, 2014, Doha, Qatar.

eral forms of coreference as filters, motivated by the need for efficient coreference resolution when processing large corpora. Complementing these unsupervised experiments, we implement a maximum recall bootstrap to identify which fills are reachable from training data.

We find ∼10% of recall is ignored by most systems due to NER bounds errors, and despite state-of-the-art coreference, 8% is lost when queries and fills occur in different sentences. Using NE type constraints is very effective, reducing recall by only 2% for a search space reduction of 81%. Without any coreference, 16% of typed fills are lost, but 12% of this recall can be recovered using fast naïve name matching rules, reducing the search space to 59% that of full coreference. 15% of recall is lost if a SF approach, such as a bootstrapping, requires that dependency paths be non-unique in a corpus. We show that most remaining candidates are reachable via bootstrapping from a small number of seeds. Our results provide systematic confirmation that effective coreference and NER are critical to high recall slot filling.

2 Why focus on recall?

In this work, we determine the recall loss caused by candidate generation constraints in SF systems. SF pipelines are typically implemented using a coarse-to-fine approach, where all possible candidates are generated and then filtered by hard constraints and more sophisticated downstream processes. Following this, we maximally generate candidates and assume a high-precision but relatively costly downstream process selects the final extractions. While ultimately any system makes precision-recall trade-offs, the recall of a system's coarse candidate generation process sets a hard upper bound on performance, as candidates that are not generated at all can never be recovered by downstream processes. SF systems could generate every noun phrase in a corpus as potential candidates, but they apply hard candidate generation constraints for efficiency and precision.

We implement these hard constraints as a series of filters, and return every candidate which passes a filter without further ranking or thresholding. These filters are comprised of generic components, such as NER, which are representative of SF pipelines. We are only interested in precision in so much as it corresponds to the size of the search space (the candidates generated), assuming a small, fixed number of answers. The search space determines the workload of later stages responsible for extraction, merging and ranking. Precision can be improved by this post-processing of the candidate set, but recall cannot.

3 Background

Slot filling (SF) is a query-oriented relation extraction (RE) task in the Knowledge Base Population (KBP) track of the Text Analysis Conferences (TAC) (McNamee and Dang, 2009). A SF system is queried with a name and a predefined relation schema, or *slots*, and must seek instances of any relations involving the query entity, and the corresponding slot fills, from a corpus.

Systems typically consist of several pipelined stages (Ji et al., 2011), providing many potential locations for error. The basic pipeline, in Figure 1, consists of four stages (Ji and Grishman, 2011): document retrieval, candidate generation, answer extraction, and answer merging and ranking. The output of the second stage is a set of candidates which are then usually ranked using RE techniques,[1] to precisely pinpoint answers. TAC penalises redundant responses, requiring a final answer merging and ranking stage. The first two stages are the focus of this work, as they inadvertently filter correct answers that cannot be recovered, and they determine the size of the search space for later stages.

Min and Grishman (2012) conducted an analysis of the 140 TAC 2010 SF fills that were found by human annotators but not any system, and manually look for evidence in the reference document and categorise the hypothetical sources of error. They find inference, coreference and NER to be the top sources of error, and that the most studied component (sentence-level RE) is not the dominant problem, contributing only 10% of recall loss. We precisely characterise the contribution of these sources of error.

We follow the SF literature in adopting RE techniques for filtering candidates. RE focuses on identifying relations between entities (or attributes of entities) as mentioned in text. Both relation schema and training data are often provided, and extraction is done using learnt classifiers (Mintz et al., 2009; Surdeanu et al., 2012; Riedel et al.,

[1] We note that question answering techniques have been used directly by SF systems (Byrne and Dunnion, 2011) but RE techniques are the primary method for answer extraction.

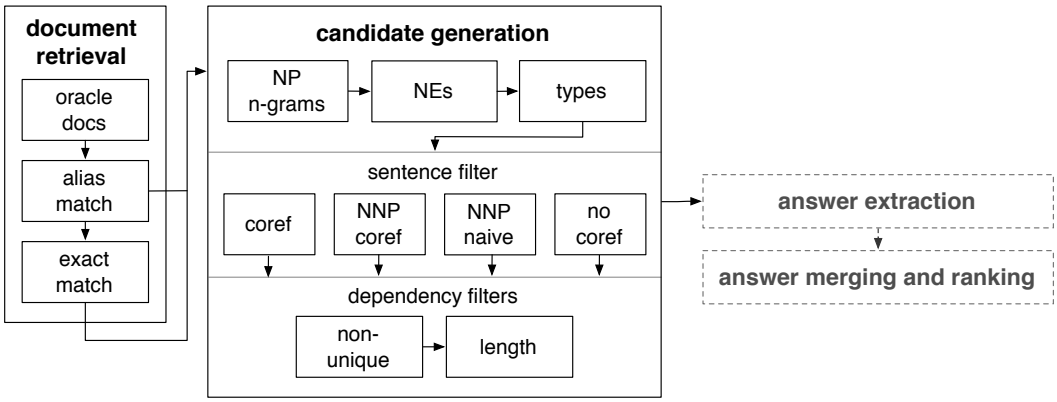

Figure 1: Candidate filters within the standard SF pipeline. Arrows indicate a sequence of filters.

2013; Zhang et al., 2013) or semi-supervised techniques (Agichtein and Gravano, 2000; Wang et al., 2011; Carlson et al., 2010).

Relation phrases or patterns may be identified without labels (Fader et al., 2011; Mausam et al., 2012) or clustered (Yao et al., 2012) into types. Generating candidate entity pairs and using the syntactic or surface path between them to decide whether a relation exists are common threads in RE that also form part of the SF pipeline. In some RE tasks, entities mentioned may already be identified in a document and provided to a RE system; in general, automatic NER is required. Some tasks are defined more generally to include common noun phrases (Fader et al., 2011; Carlson et al., 2010). SF specifically includes slots that can be filled by arbitrary strings such as `per:cause of death`, which make up a large number of slot fills but may require the use of different techniques for extraction, separate from names. NER may be further enhanced by resolving names to a KB (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012; Wang et al., 2011), reducing noise in learning and extraction processes, but we do not take this step in this work.

Typically, a RE system will only consider entities mentioned together in a sentence. When seeking all instances of a given relation between known entities, coreference resolution is necessary to substantially expand the set of candidate pairs (Gabbard et al., 2011). Coreference resolution may not be necessary where each relation is redundantly mentioned in a large corpus, as in SF; in this vein, "Open" approaches prefer precision and avoid automatic coreference resolution (Banko et al., 2007). Moreover, previous analysis attributed substantial SF error to these tools (Ji and Grishman, 2011). Our work evaluates NER, locality heuristics and coreference within a SF context.

Classification features for RE typically encode: attributes of the entities; the surface form, dependency path, or phrase structure subtree between them; and surrounding context (Zhou et al., 2005; Mintz et al., 2009; Zhang et al., 2013). We evaluate the length of dependency path between entities as a variable affecting SF candidate recall, and apply naïve entity pair bootstrapping (Brin, 1998; Agichtein and Gravano, 2000) to assess the generalisation over dependency paths from examples.

4 Experimental setup

We begin with a set of queries (a query being a NE entity grounded in a mention in a document) and, for each query q, the documents D_q known to contain any slot fill for q, as determined by oracle information retrieval (IR) from human annotation and judged system output. Filling every slot in q with every n-gram in D_q constitutes a system with nearly perfect recall. We apply a series of increasingly restrictive filters over this set. As in Figure 1, SF systems in practice must retrieve relevant documents and generate candidates. We propose filters that allow for analysis of recall lost during these stages. We ignore the remaining stages and evaluate the set of candidates directly.

Filters define what documents or NEs are allowed to pass through, based on constraints imposed by query matching, entity form, and sentence and syntactic context. We combine these filters in series in a number of configurations. The use or absence of coreference varies across our configurations, as the need to identify the query mention and terms that refer to the query mention is critical. Finally, we experiment with a boot-

strapping training process, to reflect constraints implicitly applied by a training approach.

The SF typical system pipeline presented in Section 3 applies to most, but not all SF approaches. The following filters directly apply only to systems that use NER as the method of candidate generation, and where candidate generation is distinct from answer extraction. Fourteen of the eighteen teams participating in TAC13 submitted system reports (Surdeanu, 2013). Eleven of these systems identify NEs with NER and pass these to an answer extraction process. The remaining three systems either do not document whether they rely on or do not rely on NER for candidate generation for `name` slots. We include a high recall baseline based on noun phrases (NPs) to cover these systems.

4.1 Filters

The first step in the SF pipeline is to find a relevant document and the query entity mentioned within that document. We use oracle IR to find documents D_q (ORACLE DOCS in Figure 1) but need to find a reference to q in these documents for other filters and downstream stages (ALIAS MATCH in Figure 1). An exact match to the query name is trivial, but some documents may not contain the query verbatim. This primarily occurs in cases where an alias is used, e.g. where the query Fyffes PLC is only mentioned as Fyffes in a document.

SF systems typically implement a query expansion step prior to searching for relevant documents, generating and extracting aliases based on the corpus and external sources (Ji et al., 2011). For documents that do not mention the query verbatim, we manually annotate the longest token span which refers to the query. All of our filters are applied to this base setup. To measure the effect of our manual aliases on recall, we implement a naïve EXACT MATCH filter, which allows a document only if a NE matches the query verbatim.

Entity form filters are based on the form of the entities extracted from documents. We initially consider all substrings of all NPs for a high-recall, yet tractable, baseline. The NP N-GRAMS filter allows every n-gram of every NP. NES allows NEs only; and for TYPES, fill NEs must be of a NER type defined by the slot, e.g. for `per:city of birth` only LOC NEs are allowed.

Sentence filters require the query mention and fill to be in the same sentence, or to have mentions in the same sentence. Sentence filters are COREF: the query and the fill must be mentioned in the same sentence; COREF NNP: as for COREF, but the query and the fill must have coreferent proper noun mentions in the same sentence; NAÏVE NNP: as for COREF NNP, but instead of using a full coreference system and identifying proper noun mentions, we use a naïve proper noun coreference process; and NOCOREF: the verbatim query and the fill must be named in the same sentence.

As dependency paths are often a key feature for extracting relations, we apply further syntactic filters based on dependency paths between NEs and mentions in sentences. Where we use dependencies, we use the Stanford collapsed and propagated representation (de Marneffe and Manning, 2008), e.g. in Alice is an employee of Bob and Charlie the collapsed and propagated dependency path between Alice and Charlie is $\rightarrow$`nsubj`$\rightarrow$employee$\leftarrow$`prep_of`$\leftarrow$.

Syntactic filters roughly capture the complexity of the syntactic configuration between query and filler: LENGTH $\leq$ N requires that the query and fill are separated by a dependency path of at most N arcs, e.g. the above dependency path is two arcs; VERB requires a verb to be present in the dependency path between the query and fill mentions or names; and NON-UNIQUE requires the dependency path between the query and fill to occur more than once in a corpus, modelling a hard constraint on bootstrapping and other learning processes that require a shared dependency context between training and test examples.

4.2 Bootstrapping reachability

In addition to the upper bound set by these explicit hard constraints, we want to reflect constraints that are implicitly applied by an extraction process—are there fills that are never learnable given a set of features and a set of training data? We extend our evaluation to include a training process in a semi-supervised setting. We treat this as a bootstrapping task (Agichtein and Gravano, 2000): given training pairs of NEs in text (each pair effectively a query entity and a candidate slot fill, or vice-versa), extract the context of each pair, and find other pairs in the corpus that share that context. A pair is reachable, and hence learnable, if it can be found by iterating this process. We continue to evaluate maximum recall and do not apply thresholding or ranking that would typically be utilised in a bootstrapping process. We simply output all

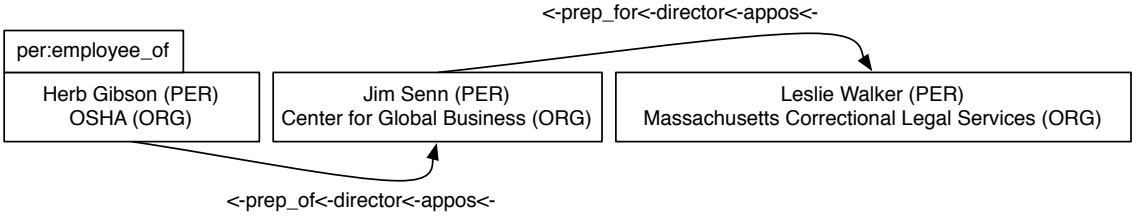

Figure 2: Bootstrapping. The rightmost vertex is labelled with `per:employee of` after two iterations.

possible candidates in order to measure recall loss: as with hard constraints applied by filters, if recall is lost it can never be recovered.

Given a set of training data, we identify if we can reach a test instance by bootstrapping, no matter how remotely it is connected to training instances. We use lemmatised dependency paths as the context for this process as they are relatively precise and discriminative, compared to other features used for SF. In order to simplify processing, we construct a graph of all pairs and paths in the corpus first, and then bootstrap from training instances over this graph. Bootstrapping more general features (e.g. bag-of-words) results in the graph becoming too large to process on our computing resources.

The graph is constructed as follows. Each vertex represents a typed pair of NEs that occur in the same sentence in the TAC KBP Source Data (LDC, 2010), collapsing vertices that have equal names and types into a single vertex. An edge exists between pairs that are connected at least once by the same dependency path. The constructed graph is equivalent to the EXACT MATCH + NOCOREF + NON-UNIQUE filter. Constructing a graph for COREF (which requires many more edges than NOCOREF) was impractical.

Initially, pairs in training data are labelled with their corresponding slots (see Figure 2). In each bootstrap iteration, the labels of each vertex are added to its neighbouring vertices. There is no filtering or competition between labels on a vertex, they are all added. We analyse performance after each iteration, evaluating by mapping the labelled graph back to the equivalent SF queries. This enables us to determine what fills are recoverable from the bootstrapping process.

5 Evaluation

We evaluate our filters on the TAC KBP English Slot Filling 2011 corpus, queries and task specification. As we aim to determine recall upper bounds and recall loss, we use only the documents D from the TAC KBP Source Data (LDC, 2010) that are known to contain at least one correct slot fill in the TAC KBP 2011 English Slot Filling Assessment Results (LDC, 2011).

We restrict the assessment results and the evaluation process to all slot types that are filled by `name` content types as opposed to `value` or `string`. We also do not evaluate the `per:alternate names` or `org:alternate names` slots, as extraction of fills for these slots typically falls outside the RE task: while X also known as Y or similar may appear in text, X and Y are typically mentioned independently across documents.

There are 100 TAC11 queries, 50 PER and 50 ORG. There are 535 fills in our reduced evaluation, 1,171 correct responses over these fills: 56% of the original evaluation slots. The distribution of fills per slot is listed in Table 1. The number of fills per query ranges from 0 (one query has no `name` fills) to 71, with a median of 17. D is comprised of 1,351 documents. The number of documents per query ranges from 0 to 63, with a median of 15.5. We use TAC 2009 and 2010 results and annotations as training data for bootstrapping, with 4,647 relevant training examples.

We evaluate ignoring case and without requiring a specific source document: `nocase` and `anydoc` in SF evaluation. Note that each slot fill is an equivalence class of responses: e.g. for `org:founded by` the correct fills Clifford S. Asness and Clifford Asness are equivalent. Consistent with SF evaluation, we identify at what constraint an entire equivalence class no longer has any member proposed as a fill.

We process documents with Stanford CoreNLP: tokenisation, POS tagging (Toutanova et al., 2003), NER (Finkel et al., 2005), parsing (Klein and Manning, 2003), and coreference resolution (Lee et al., 2011), and these annotations form the relevant components of our filters. Where we use dependency paths, we lemmatise tokens on the

slot	#	slot	#	slot	#
org:top members,employees	118	per:cities of residence	17	per:other family	6
per:employee of	71	per:children	17	per:city of birth	6
per:member of	47	org:stateorprovince of headquarters	17	per:parents	3
org:subsidiaries	32	per:schools attended	16	per:country of birth	3
org:parents	24	per:stateorprovinces of residence	11	org:political,religious affiliation	2
per:origin	23	org:member of	11	per:stateorprovince of birth	1
org:country of headquarters	22	per:spouse	8	per:country of death	1
per:countries of residence	20	org:members	8	per:city of death	1
org:city of headquarters	19	org:founded by	7		
org:shareholders	18	per:siblings	6		

Table 1: Number of fills for slots in the evaluation.

path to increase generality and recall in further analysis. For example, for Alice employs Bob we extract the path ←nsubj←employ→dobj→.

The COREF NNP filter uses CoreNLP coreference, limited to mentions which are headed by NNPs. For NAÏVE NNP we use a naïve rule-based coreference process (Pink et al., 2013), motivated by efficiency reasons, as the full CoreNLP requires parsing and a more complex model. The rules do not require deep processing and can run quickly over large volumes of text. All NEs from a document are matched by processing in decreasing length order. Two names are marked coreferent where, ignoring titles and case: they match exactly; they have a matching final word; they have a matching initial word; or one is an acronym of the other. If multiple conditions are matched, the earliest (the most strict match) is used.

The NON-UNIQUE filter requires that a dependency path occurs more than once between NEs in the full TAC KBP Source Data (LDC, 2010), comprised of 1.8M documents and 318M NE pairs. There are 38.6M distinct lemmatised dependency paths, 5M of which occur more than once.

6 Results

We now analyse where the filters lose recall. Results for non-syntactic filters are listed in Table 2. Figure 3 illustrates our main pipeline which contains filters that would typically be implemented.

NP n-grams We choose all n-grams of NPs (from the CoreNLP constituency parser) to be our highest recall filter, and so our highest baseline has 3% recall loss. We identify the reasons for loss at this filter. There are four errors due to the fill not existing verbatim in text, e.g. Pinellas and Pasco counties does not contain Pinellas County verbatim. Four errors occur where an NP is not correctly identified, which occurs in two different cases: where there is genuine error or where the sentence being parsed is actually a list or other semi-structured data as opposed to an actual sentence. four errors are where a correct answer has not been annotated as correct, we refer to this as ANNOTATION error below, and one case where an incorrect response has been annotated as correct.

While 97% recall is an excellent starting point, 53M candidates is a huge, likely intractable search space for any downstream process. Hence NER is commonly used as the starting point for SF.

NEs Most errors here are due to NER errors, and these errors result in nearly a 10% recall loss. 25 errors are caused where no token in the fill has been tagged as part of a NE (NO NER); and 13 where some tokens were missed (NER BOUNDS). There are two additional cases of ANNOTATION due to determiners not being included in an NE, where they perhaps should have also been annotated. Hence, in agreement with previous analyses, NER error has a large impact on SF.

On this data set we have 10% recall loss that most SF or RE approaches would never be able to extract. However, it is still fairly unconstrained and a high recall bound in comparison to the following filters. Recall errors could be substantially reduced if SF approaches were to take into consideration all NEs in documents as a set of candidates, and take a more document-based approach to RE as opposed to sentence-based. While there has been some work in extracting relations across sentences without coreference (Swampillai and Stevenson, 2011), RE across sentence boundaries is effectively limited to coreference chains between sentences. Currently whole document extraction is not a research focus for SF, and the implementation of whole document techniques throughout SF pipelines would likely be beneficial.

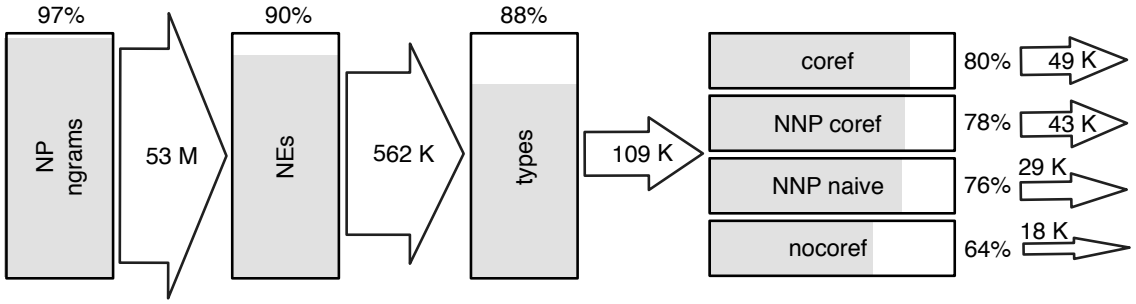

Figure 3: Results for NP N-GRAMS + NES + TYPES, followed by sentence filters with a range of coreference configurations. Grey fill and % indicates recall after each filter, and the number in the arrow is the size of the result set passed to the next filter or to the downstream process.

experiment	R (%)	|search space|
NP N-GRAMS	97	53966773
...+ NES	90	562318
...+ TYPES (1)	88	109241
...+ EXACT MATCH (2)	85	105764
(1) + COREF	80	49170
(1) + NNP COREF	78	43476
(1) + NNP NAÏVE	76	29171
(1) + NOCOREF	64	18331
(2) + COREF	77	47439
(2) + NNP COREF	73	30089
(2) + NNP NAÏVE	73	27770
(2) + NOCOREF	61	16978
(1) + COREF + NON-UNIQUE	65	19958
(1) + NNP COREF + NON-UNIQUE	62	17692
(1) + NNP NAÏVE + NON-UNIQUE	61	13960
(1) + NOCOREF + NON-UNIQUE	48	8084
(2) + COREF + NON-UNIQUE	63	18953
(2) + NNP COREF + NON-UNIQUE	60	16712
(2) + NNP NAÏVE + NON-UNIQUE	56	13064
(2) + NOCOREF + NON-UNIQUE	43	7236

Table 2: Results on D given sets of filters configurations. The ellipses indicate the previous line.

Exact match Requiring that the query name is exactly matched (EXACT MATCH) loses a further 2% recall. Effectively this is the recall error created by the IR component of SF. Five error cases occur when an alias is required, e.g. Quds Force for IRGC-QF; Chris Bentley for Christopher Bentley. Eight errors occur where the query term is a reference to an entity but not its name, all pertaining to the query GMAC's Residential Capital LLC.

Types All errors created by the TYPES filter are due to incorrect NER types on mentions proposed by CoreNLP. We do not aggregate the NE type over the coreference chain. Applying this filter cuts down the search space substantially, with minimal loss to recall. Adding TYPES results in a recall loss of 2%, but cuts down the search space by 80%.

Coref This filter is the starting point for many recent SF approaches: we consider entities that are either named or mentioned in the same sentence. Table 3 shows that coreference is the largest category of recall error created by the COREF filter. NN COREF, NNP COREF and PRP COREF indicate failure to resolve common noun, proper noun and pronoun coreference.

The remainder of the errors are cases where mentions of the fills do not occur in the same sentence. ROLE INF indicates that an individual's role is mentioned, e.g. Gene Roberts, the executive editor, where The Inquirer is mentioned in a previous sentence. LOC INF where additional location knowledge is required: a French company is headquartered in France. The search space has been substantially reduced, by a further 55% to 0.1% of the original space. However, the recall upper bound has dropped to 80% of all fills.

Coref NNP and naive NNP While coreference is important for high recall, more difficult coreference cases (common noun and pronoun coreference) may generate a large number of spurious cases. Using COREF NNP as the sentence filter loses 2% recall, to an upper bound of 78%, for a 12% reduction in the search space. However, using a full coreference system generates may more candidates than using simple NNP coreference. NAÏVE NNP has an upper bound of 76%. This is only 4% lower recall than COREF, but for a 41% reduction in search space. In addition, CoreNLP coreference is much more expensive than our naïve approach as it requires parsing.

No coref Errors for NOCOREF are listed in Table 3. INF indicates that inference or more sophisticated analysis is required to find the fill, such as correctly identifying the relation between entities

Experiment	NN COREF	NNP COREF	PRP COREF	ROLE INF	LOC INF	INF	NO NER	ANNOTATION
COREF	9	6	13	4	3	0	8	1
NOCOREF	16	52	20	4	3	2	14	3

Table 3: Error types for COREF and NOCOREF.

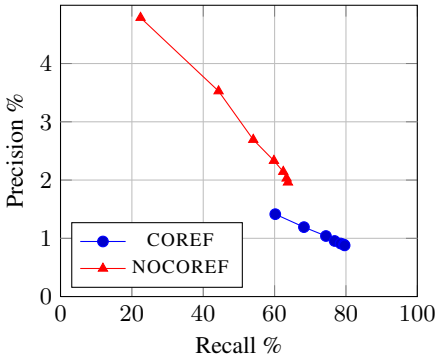

Figure 4: Effect of COREF.

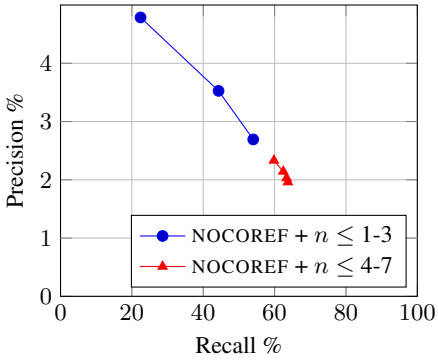

Figure 5: Effect of short dependency paths, taking the NOCOREF points from Figure 4.

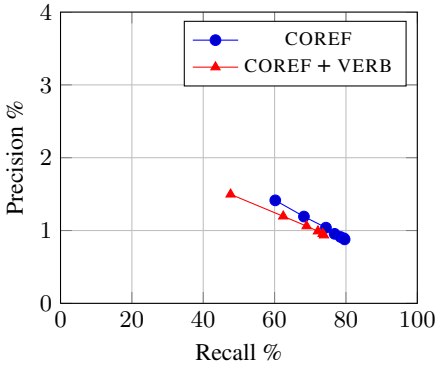

Figure 6: Effect of the VERB filter.

referred to in an interview. NOCOREF results in a recall upper bound of 64%. While this gives us a small search space, we are now losing a substantial proportion of the correct fills.

Precision-recall curves for the dependency path filters are given in Figures 4, 5 and 6. We choose to report precision for simplicity, and note that the downstream search space is the inverse of precision multiplied by the number of correct fills. Dots from low recall to high recall indicate maximum dependency path length from $n = 1$ to $n = 7$. Dependency paths of length 7 give maximum recall in our experiments. Results for the addition of the NON-UNIQUE constraint are given in Table 2.

Use of coreference While critical for recall, use of coreference generates a large number of candidates and presents a key trade-off for SF, as indicated by Figure 4. At maximum dependency path length, coreference gives 16% greater recall at a cost of 1.1% precision, roughly half the precision of no coreference.

Higher precision indicates that fewer candidates are generated. Fewer candidates allows for SF approaches to be scaled to larger amounts of data, and enables techniques that take advantage of redundancy or clustering to be used. Hence the higher precision no coreference approach may allow for more precise learning methods to be used, which may provide better results overall than an approach using coreference.

Short dependency paths In all of our filter configurations, a short dependency path length is sufficient for extracting the majority of slot fills for that particular configuration. Improving precision of fills found on short dependency paths may be a more effective and scalable approach to improving F-score rather than focusing on long paths.

In Figure 5 we consider NOCOREF. Limiting the dependency path length to three loses 11% recall, but gains 0.7% precision. While this loss of recall is high, the reduction in unique dependency paths is substantial. For maximum path length three there are 10,732 paths (1,551 unique); for all paths there are 17,394 paths (2,863 unique).

Verb Figure 6 shows the VERB filters has less impact or recall or precision than some other dependency filters. For COREF with all paths, adding the VERB filter loses 6% recall for a 0.1% gain in precision. Some slots not included in this analysis, such as `per:title`, tend to be described

by shorter paths that often do not include verbs. These slots are also frequent in the TAC11 dataset.

Non-unique The frequency of a dependency path may be a critical feature for learning, as paths that occur only once will not been seen by a bootstrapping process or may not be considered by other machine learning approaches. Applying the NON-UNIQUE filter (Table 2) has a large effect on recall: COREF loses 15% recall for a 41% reduction in the size of the search space; NOCOREF loses 15% recall for a 44% reduction in search space. To recover this recall, the strictness of this filter could be relaxed by further generalising dependency paths or using a different similarity metric to direct match of paths. However, this is the upper bound for approaches which consider only exact dependency paths as a feature.

Bootstrapping A small amount of training data quickly finds slot fills via bootstrapping. One iteration has a recall of 24%, with 7,665 candidates generated. Two to four iterations have recall of 37%–39% (maximum recall), with 31,702–37,797 candidates. The recall upper bound for these configurations is 43%—more training data will allow for better precision, but will only minimally improve recall in this setup. We note that limiting bootstrap to one or two iterations is ideal for the best trade-off between recall and search space. However, closer analysis of discriminative paths is required for a full SF system.

Note that even when bootstrapping through every dependency path in the corpus, there is an upper bound on recall of 39%. Even if we used the test data as additional training data the recall would still be limited to 43%. This demonstrates that systems need distributional features, dependency tree kernels or other similarity comparison as opposed to exact feature matching if dependency paths are to be a useful feature for SF.

7 Discussion

We present an analysis of SF recall bounds given hard constraints applied by standard system components. Pipeline error is common across all NLP tasks. Our analysis suggests that high-precision naïve tools, e.g. naïve coreference, can lead to state-of-the-art performance.

However, the SF task is not strictly an exhaustive evaluation for each query, as the evaluation data is comprised of the time-limited human annotation plus aggregated system output only. There may be fills that are missed in the evaluation results but are correct and returned by our high recall filters—affecting our reported precisions.

We manually evaluate a small sample of the queries, the first five person and the first five organization queries, to identify missed fills in the COREF output (2,903 of 49,170 total fills, or 5.9%). For these fills, there were 29 fills in the assessment data. Of these fills, 21 are returned by COREF, however there are two correct fills found by COREF that are not in the assessment data. One of these two errors would be identified with correct coreference, and the other requires complex long range inference. These additional correct fills that are identified will not have a large impact on the absolute precision, as there are two of 2,903 more fills. However, the relative difference in true positives, 21 to 23, results in some uncertainty in results when comparing them relatively.

8 Conclusion

Recent TAC KBP Slot Filling results have shown that state-of-the-art systems are substantially limited by low recall. In this work, we perform a maximum recall analysis of slot filling, providing a comprehensive analysis of recall error created in the document retrieval and candidate generation stages. We focus on recall error in candidate generation as a performance limitation, as candidates that are lost in the pipeline cannot be recovered by downstream processes.

We find ∼10% of recall is ignored by most slot filling systems due to NER error, and while state-of-the-art coreference provides a substantial recall gain over no coreference, 8% of recall is still lost when queries and fills occur in different sentences. Using NE type constraints is very effective, reducing recall by only 2% for a search space reduction of 81%. Without coreference, a further 16% of fills are lost, but 12% of this recall can be regained using efficient naïve name matching rules, while still reducing the search space by 41%, making such an approach possibly preferable over full coreference. We confirm that coreference and accurate NER are critical to high recall slot filling.

We find that using maximum recall bootstrapping, 39% of test slots fills are reachable from the TAC09 and TAC10 training data, limited by an upper bound on non-unique paths of 43%.

In the future, we intend to assess how specific

slots are affected by recall and search space trade-off, and perform evaluation over all slot types: names, values and strings. In addition, we intend to expand the bootstrapping experiments with variations over the training data.

This work highlights NER, coreference and typing as the areas that have the most impact on slot filling recall, enabling researchers to focus on problems that will most improve performance.

Acknowledgements

We would like to thank the anonymous reviewers for their useful feedback. This work was supported by an Australian Postgraduate Award, the Capital Markets CRC Computable News project and Australian Research Council Discovery grant DP1097291.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, San Antonio, Texas, USA.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of IJCAI*, pages 2670–2676, Hyderabad, India.

Sergey Brin. 1998. Extracting patterns and relations from the World Wide Web. In *Proceedings of the 1998 International Workshop on the Web and Databases*, Valencia, Spain.

Lorna Byrne and John Dunnion. 2011. UCD IIRG at TAC 2011. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of AAAI*, pages 1306–1313, Atlanta, Georgia, USA.

Linguistic Data Consortium. 2010. TAC KBP Source Data. LDC2010E12.

Linguistic Data Consortium. 2011. TAC KBP 2011 English Slot Filling Assessment Results. LDC2011E88.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, United Kingdom.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying Relations for Open Information Extraction. In *Proceedings of EMNLP*, pages 1535–1545, Edinburgh, United Kingdom.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of ACL*, pages 363–370, Ann Arbor, Michigan, USA.

Ryan Gabbard, Marjorie Freedman, and Ralph Weischedel. 2011. Coreference for Learning to Extract Relations: Yes Virginia, Coreference Matters. In *Proceedings of ACL*, pages 288–293, Portland, Oregon, USA.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*, pages 541–550, Portland, Oregon, USA.

Heng Ji and Ralph Grishman. 2011. Knowledge Base Population: Successful Approaches and Challenges. In *Proceedings of ACL-HLT*, pages 1148–1158, Portland, Oregon.

Heng Ji, Ralph Grishman, and Hoa Dang. 2011. Overview of the TAC2011 Knowledge Base Population Track. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL*, pages 423–430, Sapporo, Japan.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of CONLL: Shared Task*, pages 28–34, Portland, Oregan, USA.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of EMNLP-CONLL*, pages 523–534, Jeju Island, Korea.

Paul McNamee and Hoa Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Bonan Min and Ralph Grishman. 2012. Challenges in the Knowledge Base Population Slot Filling Task. In *Proceedings of LREC*, pages 1148–1158, Istanbul, Turkey.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, pages 1003–1011, Singapore.

Glen Pink, Will Radford, Will Cannings, Andrew Naoum, Joel Nothman, Daniel Tse, and James R. Curran. 2013. SYDNEY_CMCRC at TAC 2013. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of HLT-NAACL*, Atlanta, Georgia, USA.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2013. Effective Slot Filling Based on Shallow Distant Supervision Methods. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Benjamin Roth, Tassilo Barth, Grzegorz Chrupała, Martin Gropp, and Dietrich Klakow. 2014. RelationFactory: A Fast, Modular and Effective System for Knowledge Base Population. *Proceedings of EACL*, pages 89–92.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP-CONLL*, pages 455–465, Jeju Island, Korea.

Mihai Surdeanu. 2013. Overview of the TAC2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling. In *Proceedings of TAC*, Gaithersburg, Maryland, USA.

Kumutha Swampillai and Mark Stevenson. 2011. Extracting Relations Within and Across Sentences. In *Proceedings of RANLP*, pages 25–32, Hissar, Bulgaria.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL*, pages 173–180, Edmonton, Canada.

Yafang Wang, Bin Yang, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. 2011. Harvesting Facts from Textual Web Sources by Constrained Label Propagation. In *Proceedings of CIKM*, pages 837–846, Glasgow, Scotland, UK.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised Relation Discovery with Sense Disambiguation. In *Proceedings of ACL*, pages 712–720, Jeju Island, Korea.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. 2013. Towards Accurate Distant Supervision for Relational Facts Extraction. In *Proceedings of ACL-HLT*, pages 810–815, Jeju Island, Korea.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring Various Knowledge in Relation Extraction. In *Proceedings of ACL*, pages 427–434, Ann Arbor, Michigan, USA.