Applying the espresso algorithm to large parsed corpora

Gosse Bouma and John Nerbonne, Center for Language and Cognition, Groningen

Abstract. Information extraction (IE) culls information from text including relations, our focus here, such as head-of(Sergej-Brin, Google). The Espresso algorithm was developed to do this (Pantel & Pennacchiotti 2006), and we extend their work here first by using as input not raw text but rather syntactic analyses derived from the text, and second by applying the algorithm to Dutch. This required parsing hundreds of millions of words of text, which was regarded as infeasible only ten years ago.

1. Introduction

It is interesting to extract relations automatically from text to use in inference, to populate ontologies and to study the history of ideas (Kizito et al. 2009). Information extraction may be seen as an application of work in natural language semantics of the sort pioneered by Frans Zwarts (Zwarts 1983). Information extraction systems learn patterns for extracting pairs of words or phrases instantiating a given relation from text. For instance, for the relation *capital of*, a system might learn extraction patterns such as `Arg1 *is capital of* Arg2', or *'The ambassador of* arg2 *was called back to* Arg1'. Lightly supervised information extraction systems learn extraction patterns by means of a bootstrapping procedure, where a set of seed pairs is used to find patterns associated with the seeds, and where the patterns thus found are used in turn to find potential instantiations of the relation. The process then iterates with the (best) new instantiations to find more patterns, until some termination criterion is met.

Bootstrapping procedures like this require large text collections for learning patterns, so it is not surprising that most work in this area has used unannotated corpora and has been aimed at learning extraction patterns based on surface strings. In learning patterns based on surface strings, one encounters a certain amount of morphological and word order variation (i.e. present and past tense verbs, singular and plural forms of nouns, presence of adjectival and adverbial modifiers) which may hinder identification of the most general extraction patterns. Using parsed data can help here, as it allows the use of syntactic patterns instead of surface patterns. Xu, Uszkoreit & Li (2007) argue that abstract syntactic patterns, represented as attribute-value matrices, may actually be used to learn not just binary relations, but also relations with three or four arguments. In this paper, we restrict ourselves to binary relations, and we use the shortest path between the two arguments in a dependency graph as extraction pattern. Dependency paths abstract over morphological and word order variation, and thus can be used to identify relevant patterns more reliably than surface patterns.

We apply a well-known information extraction algorithm, *Espresso* (Pantel & Pennacchiotti 2006), to large, syntactically parsed, Dutch corpora (110M - 700M words). Although the experiments in *Espresso* are based on learning surface patterns, there is nothing in the algorithm which requires this, and thus using dependency paths instead of surface strings is relatively straightforward, except of course for the amount of additional cpu time required for parsing large text collections.

We show that applying the *Espresso*-algorithm to a parsed version of Dutch Wikipedia (110M words) allows us to obtain state of the art results for learning the *part-whole* relation. Next, we discuss a number of experiments in learning relations between named entities (*politician - political party, soccer player - club, company owner - company, institute - location*) based on a large corpus of newspaper text and Wikipedia (700 M words). Accuracies vary from 97% for the *politicians - political party* relation to only 30% for the *soccer player - club* relation. In the latter case, accuracy can be improved significantly by requiring that the arguments of extracted instance pairs must be distributionally similar to seeds or previously extracted instance pairs.

2. Corpora and Parser

Alpino (Van Noord 2006) is a wide-coverage, robust, parser for Dutch. Its grammar is designed following ideas of Head-driven Phrase Structure Grammar (Pollard & Sag 1994). It uses a maximum-entropy model for statistical disambiguation, and coverage has been increased over the years by means of semi-automatic extension of the lexicon based on error-mining (Van Noord 2004). Efficiency is improved by using a part-of-speech tagger to filter out unlikely pos tags before parsing (Prins & Van Noord 2001) , and by means of a technique which filters unlikely derivations based on statistics collected from automatically parsed corpora (van Noord 2009).

Alpino has been used as a crucial component in *Joost*, a question-answering system for Dutch (Bouma et al. 2005). Joost has been used in the CLEF evaluation campaigns, where it achieved the best results for Dutch, and it has also been used to develop a QA system for Dutch Wikipedia, and as part of an interactive, multimodal, medical QA system (Tjong Kim Sang, Bouma & de Rijke 2005; Fahmi 2009). Whereas most QA systems only use parsing to analyze the question and sometimes to analyze text snippets returned by the ir component, we used Alpino to parse the complete text collections used for all of these systems (80M, 110M, and 2M words, respectively). The benefits are that syntactic information can be used to optimize the ir process, and that off-line answer extraction can be based on dependency patterns. Jijkoun, Mur & de Rijke (2004) show, for instance, that both the recall and the precision of patterns for extracting answers off-line improve if patterns are dependency paths, instead of surface strings. Fahmi (2009) argues that syntactic information is crucial for identifying the complex noun phrases that are the arguments of medical relations (*cause, symptom, treatment*).

Although wide-coverage, robust, statistical parsers exist for a number of languages, it is often simply taken for granted that these are not fast or robust enough for processing the large volumes of text that are required in ie applications. Pantel, Ravichandran & Hovy (2004) observe that full parsing of a 15 GB corpus[10] would require 54 days of processing by a dependency parser, and 5.8 years of processing for an (unidentified) syntactic parser. Given the availability of a large cluster of CPU's (for instance by means of a grid or a cloud computing service), this objection is beginning to lose its force. The corpora used in the experiments below have all been parsed by the Alpino parser. The high-performance cluster of the University of Groningen[11] was used to run large numbers of jobs in parallel, thus making the task practically feasible.

3. The Espresso Algorithm

We adopted *Espresso* (Pantel & Pennacchiotti 2006), a lightly supervised algorithm that is initialized using a small set of seed pairs as an ie algorithm. Pantel & Pennacchiotti (2006) show that their method achieves state-of-the-art performance when initialized with relatively small seed sets over the Acquaint corpus (~ 6M words). Recall is improved with web search queries as additional source of information. We adopt the *Espresso* method for computing pattern and instance reliability. Instead of working with unannotated data, we apply this method to parsed corpora.

In *Espresso*, the reliability of a pattern *p*, $r_\pi(p)$, given a set of instance pairs *I*, is computed as the average strength of association with each instance pair *i*, weighted by instance reliability, $r_\iota(i)$:

[10] Corpus size is usually reported in number of words or sentences. We were not able to determine the number of words in this corpus.

[11] Accessible at http://www.rug.nl/cit/hpcv/faciliteiten/HPCCluster. It takes approximately 1 week to parse 100M words of text on this 400-node cluster. Processing times fluctuate strongly, depending on the number of scheduled jobs.

$$(1) \qquad r_\pi(p) = \frac{\sum_{i \in I} \left(\frac{pmi(i,p)}{\max_{pmi}} \times r_\iota(i) \right)}{|I|}$$

In this equation, *pmi(i,p)* is the pointwise mutual information score (Church & Hanks 1990) between a pattern, *p* (e.g., *part-of*), and an instance pair *i* (e.g., *engine-car*), and max_{pmi} is the maximum pmi score between all patterns and instances. The reliability of the seed pairs used to initialize the process is set to 1.

The top-k most reliable patterns are selected to find new instances. The reliability of an instance pair *i*, $r_\iota(i)$ is:

$$(2) \qquad r_\iota(i) = \frac{\sum_{p \in P} \left(\frac{pmi(i,p)}{\max_{pmi}} \times r_\pi(p) \right)}{|P|}$$

The recursive discovery of patterns from instance pairs and instance pairs from patterns is repeated until a threshold number of patterns and/or instance pairs been extracted.

3.1 Pattern Creation

Whereas Pantel & Pennacchiotti (2006) use surface strings as patterns, we used the (shortest) dependency path between (the root form of) two nominal words (i.e. nouns or proper names) as patterns. Given a dependency tree, we extract the information that two entities are connected by means of a dependency pattern. A dependency pattern in our implementation is the shortest path through the tree connecting two nodes, where the nodes themselves are replaced by placeholders Arg1 and Arg2.

For example, for the sentences in (3), Alpino produces the dependency trees given in Figures (1) and (2).

(3) a. Begin volgend jaar treedt ook het Spaanse Telefónica tot Unisource toe
Early next year, the Spanish Telefónica will also join Unisource

b. Alle delen van de planten bevatten alkaloïden en zijn daarmee giftig
All parts of the plants contain alkaloids and therefore are poisonous

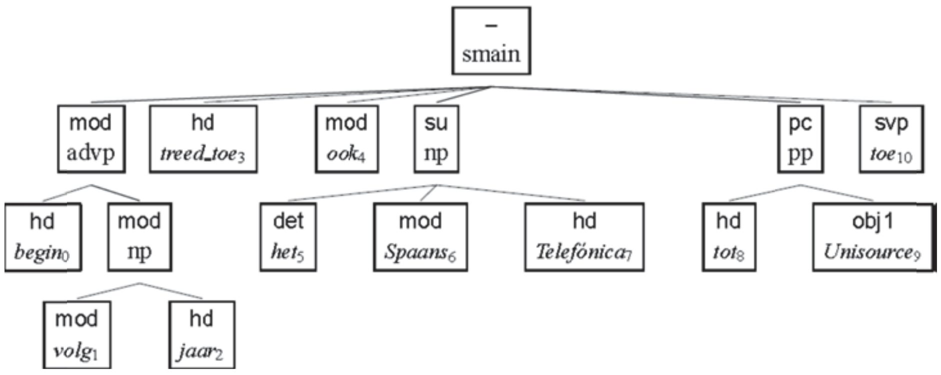

Figure 1 Dependency tree for (3a).

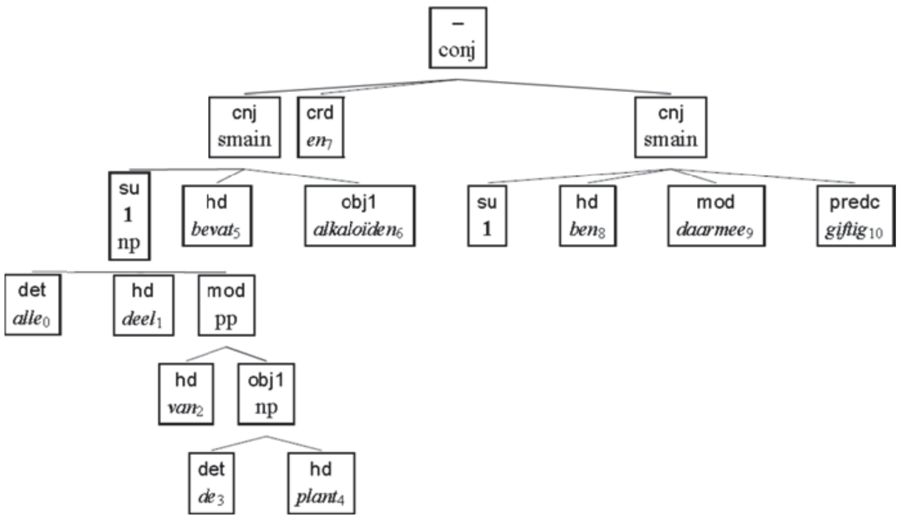

Figure 2 Dependency tree for (3b)

The dependency patterns connecting *Telefónica* and *Unisource* and *alkaloïde* and *plant*, respectively, are:

(4) a. Arg1+*su* ← *treed_toe* → *pc+tot+obj1*+Arg2
b. Arg1+*obj1* ← *bevat* → *su+deel+mod+van+obj1*+Arg2

One advantage of using dependency paths over patterns based on surface strings, is that dependency paths are able to deal with word order variation. Note that this is especially relevant for languages like Dutch or German, where there is considerable word order freedom, as illustrated by the (somewhat abbreviated) grammatical variants of (3a) in (5).

(5) a. Ook Telefónica treedt begin volgend jaar tot Unisource toe
b. Ook Telefónica treedt begin volgend jaar toe tot Unisource
c. Telefónica treedt begin volgend jaar ook toe tot Unisource
d. Begin volgend jaar treedt Telefónica toe tot Unisource

For surface-based approaches, each word order variant may lead to a separate pattern, whereas our method extracts the same dependency path in each case. Another advantage is that dependency paths often capture more of the relevant context than surface patterns. Note, for instance, that the verb stem in (3a) (*treedt*) precedes the subject, while a verbal particle (*toe*) follows the object. Surface based pattern extraction methods tend to concentrate on the string between the two arguments in a relation, and not always capture enough of the preceding or following context to obtain an accurate pattern. Finally, note that the preceding context contains an adverb, *ook*, and the name *Telefónica* is prefixed with a determiner and a modifier (*het Spaanse*), which most likely are not relevant for formulating an accurate pattern, and thus would have to be ignored.

Though dependency paths are more abstract than surface patterns, some spurious variation remains. One source of variation in dependency patterns is coordination:

(6) a. Unisource sloot eerder allianties met Telefónica en SITA
b. Arg1+su ← sluit → obj1+alliantie+pc+met+obj1+en+cnj+Arg2
c. Unisource sloot eerder een alliantie met Telefónica
d. Arg1+su ← sluit → obj1+alliantie+pc+met+obj1+Arg2

Note that (6a) and (6c) give rise to two different dependency patterns linking *Unisource* and *Telefónica*. In many scenarios, it is safe to ignore the fact that in (6a) *Telefónica* is part of a coordination inside a prepositional phrase which is a dependent of the verb. Therefore, we normalize such dependency paths by removing coordinations embedded in a longer dependency path. After normalization, the dependency pattern for sentence (6a) is identical to that of (6c). Note that this normalization does not apply to entities that are directly connected by means of a coordination, such as *Telefónica en SITA* in (6c). In those cases, we preserve the pattern Arg1+*cnj* ← *en* → *cnj*+Arg2. We observed that applying this normalization step reduces the number of unique dependency patterns by over 20%.

Stevenson & Greenwood (2009) compare various methods for using dependency tree information in pattern creation for ie. Methods extracting only *subject-verb-object* tuples have limited coverage, whereas methods extracting the minimal subtree containing both arguments suffer from lack of generality. Their *linked chain* method corresponds to our shortest path pattern extraction method, and performs well in an evaluation using the Wall Street Journal and biomedical data.

It should be noted that the *Espresso* algorithm requires that mutual information scores be known for instance pairs and for dependency patterns connecting these pairs. Thus, for any two entities (i.e. a noun or proper name) occurring in a given sentence in the corpus, we need to determine the shortest path connecting the two. For a sentence containing *N* entity denoting words, *N*(N-1)* patterns are extracted. Statistics on the number of instance pairs, dependency patterns, and the combination of these two are given in Table 1. Wikipedia refers to a parsed version of a dump of Dutch Wikipedia (from June 2008). Wiki+News is a combination of the Wikipedia corpus with a large, 600M-word, newspaper corpus (Ordelman et al. 2007).

Table 1 Pairs and patterns extracted, in millions. The last three lines give statistics for the data remaining after applying a minimum frequency cut-off of 2.

	Wikipedia		Wikipedia + News	
	all	unique	all	unique
words (approx.)	110		700	
pairs	67.7	35.6	299.5	116.5
patterns	67.7	10.8	299.5	47.6
pair-patterns	67.7	51.3	299.5	225.2
pairs ($f \geq 2$)	38.7	6.6	211.4	28.3
patterns ($f \geq 2$)	59.7	2.9	264.9	12.9
pair-patterns ($f \geq 2$)	20.0	3.6	96.6	22.2

As mutual information scores for low frequency events tend to be inaccurate, we only considered instance pairs, dependency patterns, and combinations of these, with a minimum frequency of 2. Pantel & Pennacchiotti (2006) use a discounting factor to correct for the overestimation of infrequent events by pmi. Using a discounting factor did not improve accuracy over using a frequency cut-off in our experiments.

The amount of data we have at our disposal exceeds the amount of data used by other researchers who have explored parsed data for related tasks, such as paraphrase learning or acquisition of taxonomic information. Lin & Pantel (2001), for instance, use 1 Gb of text parsed with Minipar (Lin 2003) from which they extract 7M dependency paths and 200K unique paths, for learning paraphrases. Snow, Jurafsky & Ng (2005) use a newswire corpus of 7M sentences, from which they extract 700K unique noun pairs, for learning hypernyms. McCarthy, Koeling, Weeds & Carroll (2007) use 90M words from the written portion of the British National Corpus, parsed with rasp (Briscoe & Carroll 2002) to construct a thesaurus for learning predominant word senses. Padó & Lapata (2007), finally, use all of the 100M words from the bnc parsed with Minipar for a range of lexical semantic acquisition tasks.

4. Learning Part-Whole Pairs

In our first experiment, we used the *Espresso* algorithm to extract pairs instantiating the *part-whole* relation from the parsed version of Dutch Wikipedia. Automatic extraction of *part-whole* pairs for English is well-studied (Berland & Charniak 1999; Girju, Badulescu & Moldovan 2006), and is also used by Pantel & Pennacchiotti (2006) for evaluating their algorithm. The part-whole relation is actually quite heterogeneous (Keet 2006), and covers at least the following subcases: *contained-in, located-in, member-of, structural part-of* and *subquantity-of*. We were particularly interested in the question how the presence of instance pairs representing the different subrelations influences the accuracy of results, and more in general, in the question how the choice of seeds influences results.
The "general" seed list for learning the part-whole relation contains instance pairs representing each of the subtypes. Examples of the seeds we used are given in Table 2. In addition, we constructed seed lists where all instance pairs were chosen from one subtype of the part-whole relation only. All seed lists contained 20 instance pairs.

Table 2 Sample seeds used for learning the part-whole relation, and their frequency in the corpus

Part	Whole	Frequency	Type
beeld 'statue'	*kerk* 'church'	120	contain
abdij 'abbey'	*gemeente* 'community'	36	located
club 'club'	*voetbal_bond* 'soccer league'	178	member
geheugen 'memory'	*computer* 'computer'	14	structural
alcohol 'alcohol'	*bier* 'beer'	28	subquantity

The results of learning the part-whole relation on the Wikipedia corpus are given in Table 3. *Espresso* parameters were set as in Pantel & Pennacchiotti (2006), i.e. initially, the 10 most reliable patterns are selected, and one pattern is added per iteration. The instance threshold (i.e. the number of instances preserved for the next round) is incremented by

100 in each round. We evaluated after each iteration, until the 5th round (i.e. approximately 500 instances).

Table 3 Accuracy (per iteration) for learning the part-whole relation using a seed list composed of all types (general), and seed lists representing each of the subtypes.

	general	member	subquantity	contains	structural	location
1	0.705	0.627	0.571	*0.645*	*0.598*	0.723
2	0.758	0.623	0.608	0.624	0.608	0.752
3	0.739	0.650	0.632	0.635	0.633	0.739
4	0.723	0.662	0.621	0.623	0.624	0.722
5	0.710	0.680	0.601	0.602	0.600	0.704

Some examples (translated to English) of instance pairs found by the *general* seeds are: *island - lake, protein - membrane, recommendation - report, actor - movie, picture - cover, descendant - family, altar - chapel, bacteria - digestive system, base player - band* and *batallion - brigade*.

The results in Table 3 suggest that the highest accuracy is achieved when the seed list is mixed, but also that the *member-of* and *located-in* seeds give rise to almost equally high accuracy figures. Note that for evaluating the results obtained by using a specialized seed list, all part-whole instances where counted as correct, not only instances corresponding to the relation represented by the seed list. In fact, all seed lists lead to results in which all subtypes are represented, although sometimes there is a strong bias towards specific subtypes.

Closer inspection of these results showed that after 5 iterations, the runs initialized with seeds representing the *sub-quantity-of*, *contained-in*, and *structural part-of* relation, respectively, were highly similar. That is, 490 pairs were present in all three runs, and were ranked in almost the same order (leading to a Spearman rank correlation in the range of ρ = 0.89-0.93 between the respective outputs). These three seed lists also led to discovery of a substantial number of common and prototypical part-whole dependency patterns such as *W bevat P* (‘W contains P’), *W omvat P* (‘W comprises P’) and *P is onderdeel van W* (‘P is part of W’). The most distinct results were obtained by the *located-in* and *member-of* seeds, with hardly any overlap in instances with the other results. The patterns learned by bootstrapping from the *located-in* and *member-of* seeds are more or less characteristic for these relations only. Examples of such patterns for the *located-in* relation include: *P bevindt zich in W* (‘P is located in W’), *P ligt in W* (‘P lies in W’), *P staat in W* (‘P stands in W’), P bouwt op W (‘P builds on W’) and for the *member-of* relation: *P is lid van W* (‘P is member of W’), *P richt W op* (‘P founds W’), and *P verlaat* W (‘P leaves W’).

It has been observed that the results of experiments involving bootstrapping from seeds depend heavily on the choice of seeds (McIntosh & Curran 2009). We therefore also compared the output of runs initialized with different general seed lists with the results for specific subtypes. To this end, we created five sets of general seed lists for Dutch, each time picking four seeds from each of the subtypes. In two cases, the resulting output correlated very strongly with that of the run for *located-in* (Spearman rank correlation of ρ=0.93), in the three other cases the output correlated with the output of the three runs for *contained-in, subquantity-of*, and *structural part-of* (rank correlations of ρ=0.89-0.93). We conclude from these findings that *member-of* and *located-in* are (linguistically) clearly different from the other part-whole relations. Furthermore, when starting from a mixed seed list, it is unpredictable in which direction the outcome converges. This could be seen as a subtle form of *semantic drift*, where it is not the case that the accuracy of results decreases strongly, but where there nevertheless is a strong bias towards patterns and instances representative for only a certain subtype of a given relation.

5. Learning Relations between Named Entities

Frequent question types for QA systems often ask for a named entity in a specific relation to some other named entity, i.e. *what is the capital of Togo?, for which club does David Beckham play?, which company is owned by Ted Turner?*, or *in which city does one find the Centre Pompidou?* Some QA systems (Soubbotin & Soubbotin 2002; Fleischman, Hovy & Echihabi 2003; Mur 2008) have used techniques for mining all potential instantiations of such relations from a corpus beforehand, using hand crafted extraction patterns or ie techniques similar to *Espresso*. In a second experiment, we concentrated on learning a number of such relations between named entities. As named entities are more diverse than nouns, and only a few pairs are expected to be highly frequent, we used the corpus composed of Wikipedia and a large collection of newspaper text described in section 3.1 (above).

We created seed lists (with 9-14 instance pairs) for the (*Dutch*) *politician - (Dutch) political party, soccer player -club, company - owner* and *institution - city* relations. The frequency of seed instances varied strongly, from 5 (*Carnegie Mellon Universiteit - Pittsburgh*) to 672 (*Concertgebouw - Amsterdam*). Results for the different relations are given in Table 4.

Table 4 Accuracy (per iteration) for learning various relationships between named entities: politician-party, soccer player - club, company - owner, and institute - city.

	politics	soccer	owner	institute
1	0.971	0.358	0.355	0.732
2	1.000	0.299	0.286	0.698
3	0.977	0.247	0.299	0.698
4	0.978	0.274	0.315	0.490
5	0.938	0.325	0.337	0.321

The *politics* relation leads to very accurate results. The reason for this appears to be that there are a number of frequent, and non-ambiguous dependency patterns associated with this relation involving function names (i.e. *parliamentary group leader, member of parliament, opposition leader*, and *(vice) minister*).

For the *soccer* relation, results are much less accurate. Initially, the system acquires patterns that appear to be typical for the soccer relation, but which also admit for a good deal of ambiguity: *C(lub), club of P(layer), P plays for C, P is missing in C, P scores for C, P returns in C, P knows, from his period with C* but also the very general *P (C)* (where C is analyzed as a modifier of P). After a number of rounds, patterns are added that are clearly of lower quality: *P, trainer of C, C, the organisation of P, C, the thinktank of P* and very general: *P at C, P of C*. It should be noted that in some settings, even low accuracy results can be useful. Mur (2008), shows, for instance, that even an ie system that is tuned only for recall, and which achieves a meager accuracy of 1% for learning the *soccer player - club* relation, can contribute positively to the performance of a QA system. The reason for this somewhat surprising outcome is, we think, the fact that in a QA system one of the arguments of the relation is always given in the question, and second, that a QA system like *Joost* (Bouma et al. 2005) uses additional heuristics, such as the frequency with which an answer is found, to pick the most promising answer.

For the *owner* relation, the system learns patterns like *O(wner)* is owner of *C(company), O is the mother/holding company of C, O takes over C*, and *O controls C*. Some of the international instance pairs found by the system are *Stelios Haji-Ioannou - EasyJet, Mohammed Al Fayed - Harrods, Charles Saatchi - Saatchi & Saatchi, Ted Turner - CNN* and *Richard Branson - Virgin Atlantic*. However, the system also finds many pairs in which one of the arguments is a common noun, such as *Al-Fayed - department store* and *cable firm - MTV*. A similar situation arises with institutions. The system finds a reasonable number of correct instances. For the location *Paris*, for instance, no fewer than 37 institutions are found, some of which are: *Opéra Bastille, Musée du Louvre, Théâtre de l'Odéon, Centre Pompidou, Jeu de Paume, Palais des Congrès, Institut du Monde Arabe* and

Maison Européenne de la Photographie. However, many erroneous pairs involve a predicate, such as *consulate - Rio de Janeiro* or *cultural heritage - Bonaire*.

We experimented with two methods for improving accuracy. In both cases, we filter the results of the *Espresso* algorithm by imposing additional constraints on what counts as a reliable instance pair. For the *owner* and *institutions* relation it seems most important to ensure that both arguments are proper names. As we did not preserve part of speech tags in our dependency patterns (so as to avoid spurious variation), the system has no means to learn that patterns apply to proper nouns only. As a simple remedy, we require that both arguments of an instance pair must start with an upper case letter.
For the soccer relation, the problem is that the quality of the learned patterns decreases, which leads to more incorrect instance pairs being ranked high, which in turn leads to even lower quality patterns. McIntosh & Curran (2009) observe a similar problem when trying to learn biomedical terms within a given semantic class. They propose to use distributional similarity to reduce the effect of *semantic drift*. New candidates are ranked higher if they are distributionally more similar to terms learned early (i.e. terms that are more likely to be correct) than to terms learned later.

We applied distributional similarity to filter unlikely instance pairs. We used the thesaurus described in van der Plas and Bouma (2005) and van der Plas (2008) to find distributionally similar terms. The thesaurus was built using the same 700M word parsed corpus we used in our ie experiments. For each noun and proper name, the syntactic context (the lexical head on which the nominal is dependent, and its syntactic relation) was stored in a feature-vector. Counts were weighted using pointwise mutual information (Church & Hanks 1990). Two nominals are distributionally similar if the distance between their vectors is small, according to the cosine-metric. The twenty most similar terms for keywords *Bayern München* and *David Beckham*, for instance, are:

(7) a. Bayern München: Borussia Dortmund, AC Milan, Juventus, Real Madrid, Manchester United, Chelsea, Lazio Roma, Celtic, Arsenal, AS Roma, Glasgow Rangers, Bayer Leverkusen, Olympique Marseille, Anderlecht, Inter, Lazio, Liverpool, Werder Bremen, Galatasaray

b. David Beckham: Roy Keane, Zinedine Zidane, Michael Owen, Ryan Giggs, Alan Shearer, Paul Scholes, Beckham, Raúl, Luis Figo, Diego Maradona, Andy Cole, Eric Cantona, Figo, Zidane, Ronaldo, Raul, Rivaldo, Jaap Stam, Romario

Van der Plas (2008) reports results for several alternative methods, and shows that combining mutual information and cosine gives the best results in terms of coverage and accuracy when evaluating against Dutch WordNet (Vossen 1998).

For filtering, we used the 100 most similar words for each noun or proper name that was found at least 10 times in the corpus. The filter works by accepting only new instance pairs for which each argument is distributionally similar to the corresponding argument of at least one of seeds or a previously found instance pair. In particular, a pair *Figo* - *Barcelona* is only accepted if the list of similar names for *Figo* contains at least one name which is also the first argument of a previously seen instance pair, or when the list of similar items for a first argument in a previously seen instance pair contains the name *Figo*, and similarly for the second argument *Barcelona*. Note that if a term *T* is among the *N* most similar terms of *T'*, it is not necessarily the case that *T'* is among the *N* most similar terms of *T*.

Table 5 Accuracy per iteration for learning the owner and institution relation using the upper case filter, and the soccer relation using the distributional similarity filter.

	Owner				Institution				Soccer			
	no filter		filtered		no filtered		filtered		no filter		filtered	
	N	Acc	N	Acc	N	Acc	N	Acc	N	Acc	N	Acc
1	107	0.355	45	0.844	112	0.732	93	0.882	109	0.358	40	0.650
2	210	0.286	70	0.829	212	0.698	168	0.881	211	0.299	74	0.527
3	311	0.299	108	0.824	311	0.698	242	0.897	312	0.247	88	0.375
4	409	0.315	150	0.853	412	0.490	278	0.723	412	0.274	176	0.409
5	501	0.872	195	0.872	514	0.321	291	0.560	511	0.325	290	0.452

Table 5 shows the results for the system with and without filtering. We give both the accuracy and the number of instances that remained after filtering (*N*). The upper case filter leads to an increase in accuracy of around 50% for the *owner* relation and 20% for the *institution* relation. One might argue that this increase in accuracy is only due to the fact that fewer elements are acquired per iteration. However, if we compare the accuracy of the second round without filtering (210 pairs) with that of the 5th round with filtering (195 pairs),[12] we still see that accuracy has gone up by almost 60%. The situation is a bit less clear for the *institution* relation, where the third iteration with filtering (242 pairs) is 20% more accurate than the second iteration without filtering (211 pairs), but where the

[12] The number of pairs per iteration in the system without filtering bootstrapped with *S* seeds is *100N + S*. We evaluate only on pairs not present in the seed list. This is why the number of instances given per iteration does not always go up with 100 exactly.

fifth iteration with filtering (291 instances) is 13% less accurate than the third iteration without filtering (311 pairs). This suggests that the upper case filter by itself may not be sufficient, if the quality of the patterns has deteriorated too much.

For the *soccer* relation, we used distributional similarity as a filter. Accuracies per iteration increase by 13-30%. If we compare the 5th iteration of the system with filter (290 pairs) with the 2nd (211 pairs) or 3rd (312 pairs) iteration of the unfiltered results, we still see an increase in accuracy of 16 and 21%, respectively.

6. Conclusion

In this paper, we have shown that the *Espresso* algorithm can be used to perform ie on parsed corpora, using dependency paths instead of surface strings as extraction patterns. In particular, we argue that available corpora are large enough to obtain interesting and accurate results. In two experiments, we investigated both the possibility of learning a very general, taxonomic, relation (*part-whole*) and the possibility of learning narrowly defined relations between named entities. The first type of relation is representative for work that aims at (semi-)automatically extending wordnets or other taxonomic resources, while the second type of relation can be used as a component in automatic QA systems.

Except for normalizing patterns involving an embedded conjunction, we extracted dependency paths from dependency trees without applying any rules that might be useful for the ie process. In our QA system for Dutch, the set of dependency triples (i.e. dependency paths of length 1) for a given sentence is automatically expanded with additional triples that deal with appositions, coordination, relative clauses, passives (Bouma, Mur & van Noord 2005), adjectival forms of geographical names, and compounds such as *Fiat-topman (Fiat director)* (i.e., where the first element is a name) are decomposed. Xu, Uszkoreit & Li (2007) observe that many ie methods tend to ignore the information in non-verbal patterns, such as *the 2005 Nobel Peace Prize*. The example above illustrates that for compounding languages such as Dutch and German, one also needs to take into account patterns that arise from decompounding to learn, for instance, the *director - company* relation.

While filtering on the basis of distributional similarity worked well for the *soccer player - club* relation, it turned out to be too restrictive for the other relations between named entities. This suggests that other methods for combining distributional similarity and the scores assigned by *Espresso* should be explored. Another option would be to cluster similar names and to filter on the basis of (reasonably large) clusters of names.

7. References

Berland, M., & Charniak, E. (1999). Finding parts in very large corpora. *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, 57-64.

Bouma, G., Fahmi, I., Mur, J. van Noord, G. van der Plas, L. & Tiedeman, J. (2005). Linguistic knowledge and question answering. *Traitement Automatique des Langues*, 46(2), 15-39.

Bouma, G., Mur, J. & van Noord, G. (2005). Reasoning over dependency relations for QA. In: F. Benamara and P. Saint-Dizier (eds.), *Proceedings of the IJCAI workshop on Knowledge and Reasoning for Answering Questions* (KRAQ), Edinburgh, 15-21.

Briscoe, T. & Carroll, J. (2002). Robust accurate statistical annotation of general text. *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, 1499-1504.

Church, K. W. & Hanks, P. (1990). Word association norms, mutual information and lexicography. Computational Linguistics, 16(1), 22-29.

Fahmi, I. (2009). Automatic Term and Relation Extraction for Medical Question Answering Systems. Ph.D. thesis, University of Groningen.

Fleischman, M., Hovy, E. & Echihabi, A. (2003). Off-line strategies for online question answering: Answering questions before they are asked. *Proc. 41st Annual Meeting of the Association for Computational Linguistics*, 1-7, Sapporo, Japan.

Girju, R., Badulescu, A. & Moldovan, D. (2006). Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1), 83-135.

Jijkoun, V., Mur, J. & de Rijke, M. (2004). Information extraction for question answering: Improving recall through syntactic patterns. *Proceedings of the 20th International Conference on Computational Linguistics*, 1284-1290, Geneva.

Keet, C.M. (2006). Part-whole relations in object-role models. In *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, Berlin & Heidelberg: Springer Lecture Notes in Computer Science. 1118-1127.

Kizito, J., Fahmi, I., Tjong Kim Sang, E., Nerbonne, J. & Bouma, G. (2009) Computational Linguistics and History of Science. In: Dibattista, L. (ed.) *Storia della Scienza e Linguistica Computationale* (History of Science and Computational Linguistics), Milan: FrancoAngeli, 55-73

Lin, D. (2003). Dependency-based evaluation of MINIPAR. In: Abeille, A. (ed.) *Treebanks: building and using parsed corpora*, 317-329.

Lin, D. & Pantel, P. (2001). Discovery of inference rules for question answering. *Natural Language Engineering*, 7, 343-360.

McCarthy, D., Koeling, R., Weeds, J. & Carroll, J. (2007). Unsupervised acquisition of predominant word senses. *Computational Linguistics*, 33(4), 553-590.

McIntosh, T. & Curran, J.R. (2009). Reducing semantic drift with bagging and distributional similarity. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 396-404.

Mur, J. (2008). *Off-line Answer Extraction for Question Answering*. Ph.D. thesis, University of Groningen.

Ordelman, R., deJong, F., van Hessen, A. & Hondorp, H. (2007). TWNC: a multifaceted Dutch news corpus. *ELRA Newsletter*, 12(3/4), 4-7.

Padó, S., & Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics 33*(2), 161-199.

Pantel, P., Ravichandran, D., & Hovy, E. (2004). Towards terascale knowledge acquisition. *Proceedings of the 20th international conference on Computational Linguistics*, 771-777.

Pantel, P., & Pennacchiotti, M. (2006). Espresso: Leveraging generic patterns for automatically harvesting semantic relations. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 113-120.

Pollard, C. & Sag, I.A. (1994). *Head-driven phrase structure grammar*. Stanford, USA: Center for the Study of Language and Information.

Prins, R., & Van Noord, G. (2001). Unsupervised POS-Tagging Improves Parsing Accuracy and Parsing Efficiency. In *IWPT: International Workshop on Parsing Technologies*.

Snow, R., Jurafsky, D. & Ng., A.Y. (2005). Learning syntactic patterns for automatic hypernym discovery. *Proc. of the 17th Annual Conference on Neutral Information Processing Systems*. 1297-1304.

Soubbotin, M. M., & Soubbotin, S. M. (2002). Use of Patterns for Detection of Likely Answer Strings: A Systematic Approach. *TREC* 11.

Stevenson, M., & Greenwood, M. A. (2009). Dependency pattern models for information extraction. Research on Language and Computation 7(1), 13-39.

Tjong Kim Sang, E., Bouma, G., & de Rijke, M. (2005). Developing offline strategies for answering medical questions. *Proceedings of the AAAI-05 Workshop on Question Answering in Restricted Domains, Pittsburgh, PA, USA.* 41-45.

Van der Plas, L. (2008). *Automatic lexico-semantic acquisition for question answering*. Ph.D. thesis, University of Groningen.

Van der Plas, L., & Bouma, G. (2005). Automatic acquisition of lexico-semantic knowledge for QA. *Proceedings of the IJCNLP workshop on Ontologies and Lexical Resources*, 76-84.

Van Noord, G. (2004). Error mining for wide-coverage grammar engineering. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics,* 446-453.

Van Noord, G. (2006). At last parsing is now operational. *TALN '06. Verbum Ex Machina. Actes de la 13e conference sur le traitement automatique des langues naturelles*, 20-42.

Van Noord, G. (2009). Learning efficient parsing. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 817-825.

Vossen, P. (1998). *A multilingual database with lexical semantic networks*. Dordrecht: Kluwer.

Xu, F., Uszkoreit, H. & Li, H. (2007). A seed-driven bottom-up machine learning framework for extracting relations of various complexity. *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 584-591.

Zwarts, F. (1983). Determiners: a relational perspective. *Studies in model-theoretic semantics*, 37-62.