From Information Retrieval to Information Extraction

David Milward and James Thomas
SRI International
Suite 23 Millers Yard
Mill Lane
Cambridge
CB2 1RQ
milward@cam.sri.com, jrt1003@cam.sri.com

Abstract

This paper describes a system which enables users to create on-the-fly queries which involve not just keywords, but also sortal constraints and linguistic constraints. The user can specify how the results should be presented e.g. in terms of links to documents, or as table entries. The aim is to bridge the gap between keyword based Information Retrieval and pattern based Information Extraction.

1 Introduction

The amount of information available electronically in a free-text format is ever increasing. People require tools that give them the best possible answer to their queries even when a full answer may not be available.

Current web Information Retrieval (IR) engines standardly retrieve URLs to whole documents, and typical user queries are just an unordered set of keywords. This is robust and allows unrestricted queries, but precision can be poor, and the output is not particularly convenient for many queries. For example, if we are interested in e.g. companies associated with Mr. Jones, we are likely to prefer an output in terms of an alphabetically ordered list of companies (with links to the sentences in which they appear) rather than a list of URLs.

Information Extraction (IE) systems provide better presentation of results (e.g. tables, database records etc.), and tend to include more precise searches which depend upon not just finding keywords but finding them in particular positions or in particular grammatical relationships. However, current IE systems typically require queries to be programmed beforehand, so are suitable only for cases where the same query is to be used time and again e.g. in the processing of a newsfeed.

Customisation of an IE system involves two major tasks:

- adapting the information extraction system to a new domain, e.g. adding vocabulary, new grammatical constructions and domain ontology.
- creating new patterns for extraction in the new domain.

Customising current IE systems is thus expensive. A domain expert and a linguist with knowledge of the IE system will generally be required to specify the task, manually mark-up or categorise domain data, perhaps build a domain ontology, write extraction patterns and fine-tune them for the best compromise of precision and recall.

There are thus pressing reasons for automating the customisation task and reducing the involvement of linguists as much as possible. Several methods of automatically inferring patterns and semantic dictionaries from annotated texts have been suggested, e.g. (Riloff, 1993; Riloff, 1996; Sonderland et al., 1995; Yangarber and Grishman, 1997). Pierce and Cardie (Cardie and Pierce, 1998) suggest that the use of annotated texts can be replaced by an interactive approach in which an end-user assists the system in bootstrapping patterns from a set of example results. However even this method is ill suited to one-off queries, where the user is unlikely to start from example results, but needs to be able to interactively home in on a set of answers.

In this paper we describe a system which enables users to build advanced on-the-fly queries. The interface is intended to be intuitive for users already familiar with keyword IR but adds the extra functionality of sortal, linguistic and positional constraints that are more common in IE. Adding extra linguistic constraints may improve precision, and cause a drop in recall, in exactly the same way as adding an extra non-optional keyword.

The approach is less ambitious than providing full Natural Language querying, but allows

professional users of search technology (e.g. analysts or research scientists) to get most of the advantages of IE without having to program patterns. We assume that retraining of linguistic components (such as the tokeniser and tagger) for a new domain will still be performed by a linguist/programmer, and that end-users will be supplied with libraries of useful sorts such as person names, companies, locations, protein names etc. which are appropriate for their domain.

2 IR and NLP

Our key interest in this work was to provide a system which allowed users to get answers: not just documents or sub-documents. We have not addressed the question of whether or not these techniques would also be useful for more traditional IR in the sense of finding the most relevant document for a particular query. There is some potential since there are extra options to refine or expand a query e.g. using sortal constraints such as *company* and *location*, and restrictive constraints such as *subject_of* or *same sentence*. Since the linguistic constraints are under user control the query is more likely to be accurate than in systems where linguistic constraints are derived from a natural language query (though at the expense of usability).

The system was designed to deal with multiple answer queries such as "which protein interacts with TAF-2?". This differs somewhat from the TREC question answering track ((TREC), 2000), where the emphasis is on questions which have a single answer, and systems attempt to provide the most relevant sub-document. To attempt the TREC task we would need to extend the system with a relevance weighting mechanism, and provide further techniques for query expansion. We would then expect the system to do well, since (Srihari and Li, 1999) show that the use of sortal constraints such as *company, location* and *time* plus constraints such as *same sentence* give good results, even with a relatively simple ranking mechanism.

How much extra cost is involved in using linguistic information? There is obviously some initial cost in parsing and analysing the texts, however this can largely be hidden by preprocessing the documents. In addition there is a space cost in having a much larger index: this is necessary since we are keeping more information about the document's structure. Finally, there is the cost of evaluating more complex constraints. The cost of using sortal constraints is negligible: we can index them in exactly the same way as words. However,

```
i1: P
i2: a
i3: b
i4:i1(i2,i3)
```

Figure 1: Distributed representation of $P(a, b)$

relational constraints such as *same sentence* do introduce extra processing. The figures we present at the end of this paper show that this first implementation of the system is fast enough to be usable for some real applications (and very fast by Information Extraction standards), but is not yet in the same league as standard IR engines.

3 Highlight

Highlight (Thomas et al., 2000) is a general-purpose IE engine for use in commercial applications. The work described in the current paper extends Highlight by building an interface on top of it and replacing the internal representation of linguistically analysed texts with a representation based on distributed representations c.f. (Milward, 2000). Demos of the Highlight System are accessible from the SRI Cambridge web site at http://www.cam.sri.com

3.1 Distributed Representation

The inspiration for the approach taken here was the work in (Milward, 2000) on interpretation of dialogue utterances. This uses distributed semantic representations (based on indexing individual parts of a semantic structure) to encode both full and partial semantic analyses. For example, the logical form P(a,b) is represented by the set of constraints given in Figure 1.

This approach aims to combine the advantages of shallow pattern-matching approaches (which are fast and robust) with those of deeper analysis which tends to do better for phenomena which involve scope. For example, consider the utterance "The 2.10 flight to Boston will not stop at New York". A simple pattern matching approach looking for *[flight] to [location] via [location]* would extract the wrong information since it has no notion of the scope of the negation.

In this work we take the same idea, but apply it to Information Extraction. Sets of indexed constraints (which are themselves partial descriptions of semantic structure) are used directly as search expressions. Since the indexed constraints can encode full semantic structures, the search terms can be more or less specific: we can start with just constraints on lexical items, and then build up to a full semantic structure by adding structural con-

```
obligatory          optional
----------          --------
 I:J(K,L),           (1) K>H, H:M(N)
 J:at,               (2) K:M(N)        => departure_time(T)
 M:leave,
 L:T,
 time(T)
```

Figure 2: Extracting a departure time

straints.

Search is now a question of expressing the appropriate constraints on the information to be extracted. For example, Figure 2 is a sketch rule to extract departure time from a sentence such as "I leave at 3pm." The rule subsumes a range of rules which vary in the specificity with which they apply. For example, applying only the obligatory conditions will result in the extraction of 3pm as departure time from sentences such as "I leave Cambridge to arrive at 3pm" since in the obligatory component there is no restriction on M with respect to T, i.e. the *leave* event and the time. Adding the optional constraints in (1) gives us the restriction that the leaving must be dominated by the preposition *at*, e.g. "I leave from the station at 3pm," and (2) requires immediate dominance: "I leave at 3pm." This ranges from pattern matching to almost full semantics. Any particular application of a rule can be scored according to the number of optional conditions which are fulfilled, thereby using linguistic constraints if they are available.

In practice, the system described here departs from this approach in several respects. Firstly, we were working with the pre-existing Highlight system which uses shallow syntactic processing based on cascaded patterns (similar to (Hobbs et al., 1996)). Deriving semantic relationships from this is not particularly reliable, so it is preferable to use search terms which rely more on positional clues. We therefore used a distributed syntactic representation to provide more reliable syntactic constraints (e.g. head word of, subject of, same sentence) augmented by positional constraints (e.g. precedes, immediately precedes). Secondly, we wanted an intuitive user interface so constraints such as 'dominates' were not appropriate.

3.2 User Interface

Given the underlying representation described above, the user's task in building a query is to propose a set of constraints which can be matched against the representation of a text or set of texts. The interface attempts to preserve the convenience of keyword based IR but also enable more refined searches, and control over the presentation of results. Keyword based search is a special case where the user specifies one or more keywords which they want to find in a document. However, users can also specify that they want to find a class of item (e.g. companies) and refine the search for items within the same sentence, not just the same document.

For example, to find 'Esso' and a location in the same sentence we need the set of constraints given below:

```
J:Esso
location(T)
same_sentence(J,T)
```

The interface emphasises the items the user wants to search for. Consider Figure 3. The user is looking for two items, the first including the word Esso (this could be e.g. "Esso Corp.", "Esso Holdings" etc.), and the second item of sort 'location'. The effect of pressing 'Add' is to include the positional constraint that the two items must appear in the same sentence. In later parts of this paper we provide examples of more sophisticated queries which imitate what is currently achieved by pattern matching in more standard IE systems (e.g. Fastus (Hobbs et al., 1996)).

In our approach IE is a seamless extension of IR. This can be contrasted with some more typical ways of tying together IE and IR by performing an IR search followed by IE. In that approach, IR is used first to select a limited set of documents, then IE is applied. This is fine if the queries and the keywords are fixed (and appropriate for each other). It is not ideal otherwise. For example, suppose you are interested in Esso's profits. To achieve your query you might use IR to find documents containing Esso, then use an IE system which has been customised to look for company profits. However, some of the results are likely to be unexpected, for example, you would obtain Shell's profits if there were a document describing Shell's profits which just happens to mention Esso in passing.

Items can be constrained to have a particular head word, to include a particular word or to be of

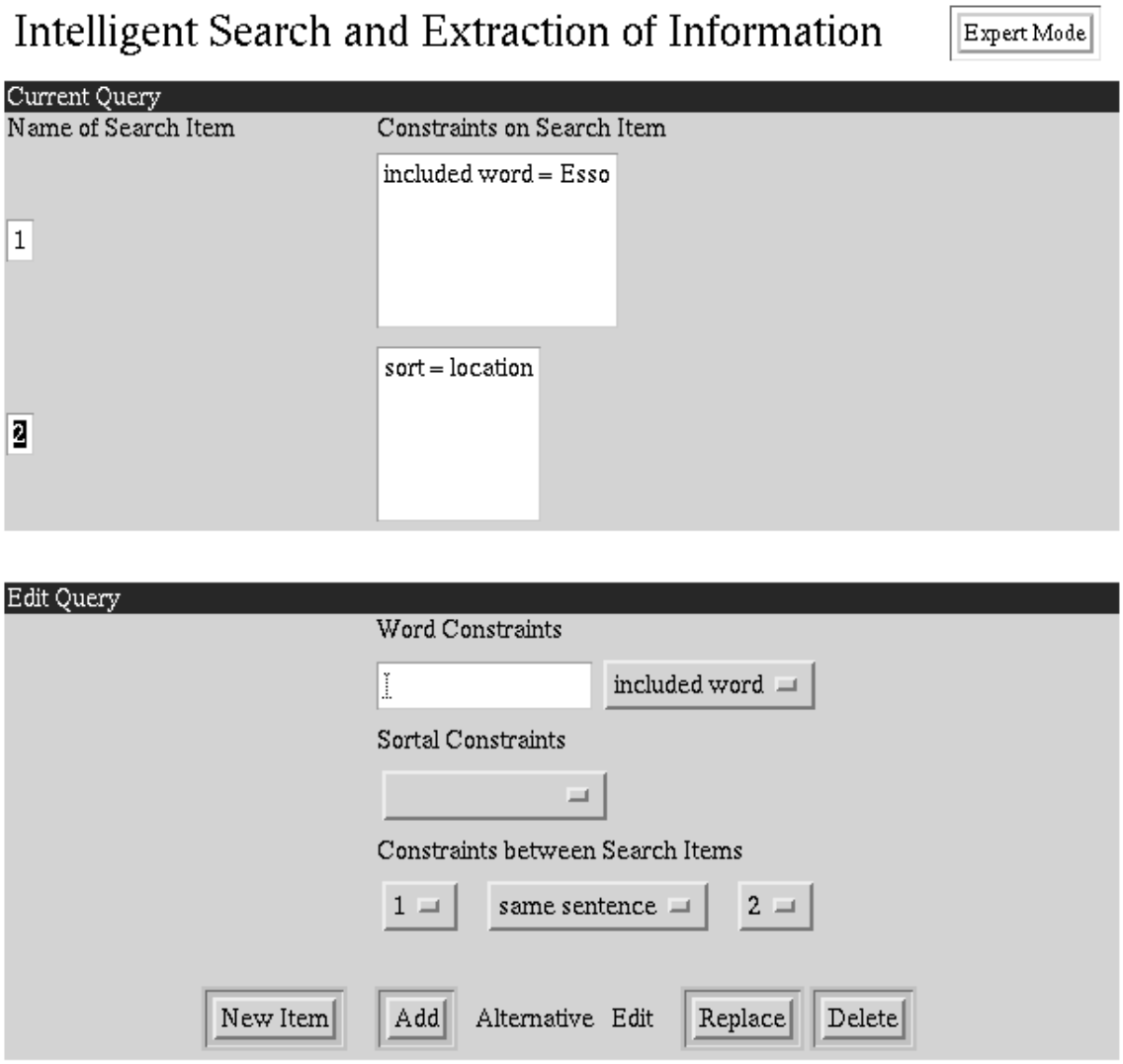

Figure 3: User interface

a particular sort. Multiple constraints on a single item are possible, as shown in Figure 5. Positional constraints can include any kind of inter-item constraints e.g. *precedes, same sentence*. There are further option buttons which concern the files to be queried and the layout of the output. The default is to provide a table which includes each item found plus the sentence in which it appears in the document.

Two levels of user expertise (*novice* and *expert*) allow more or less control over the technical details of the query. Expert mode (accessed by clicking the button at the top right) looks much the same but has facilities such as altering the template output, naming the items and more options on the pull-down menus. For instance, in expert mode the user may specify which items are optional in the query, and what their syntactic class might be. There are also additional extra parameters for the expert user for the output templates.

A typical user query is given in Figure 4. Here the user is looking for appositives in the pattern *Person Noun of Company*, for example *John Smith, chairman of X-Corp.*[1] Note that the query is not particularly linguistically sophisticated: the Position item is glossed as a noun group and the single preposition *of* is specified when a looser restriction (perhaps to be any preposition) would certainly turn up more results. However, this query is quick and simple to construct and can be used as a diagnostic for a more detailed query.

A more complex query is shown in Figure 5 for a protein interaction task. The sort *interaction* in Figure 5 could be defined as a disjunction of the constraints *head word = interact, head word = bind, head word = associate*).

3.3 One-off Query vs Pattern Base

Highlight can operate in two distinct modes: interactive and batch. The interactive mode suits one-off queries as seen but can also be used to prototype queries on a subset of a corpus. Once a user is satisfied with the accuracy of a query, this user-defined 'pattern' can be stored for later use. Batch mode is used when a query is to be run over a large amount of text and requires no intervention from the user other than the initial set-up.

3.4 Preprocessing Files and Scalability

If a user makes two queries over the same set of documents it does not make sense to do all the linguistic processing twice. To avoid this, documents can be preprocessed. Preprocessing involves tagging, chunking, recognition of sorts, and conversion of the results into a set of constraints. At query time, there is no further linguistic processing to be done, just constraint satisfaction.

The system has a relatively simple outer loop which considers a query for each document in turn (rather than e.g. using a single cross-document index). If a document has not been preprocessed, it is processed, then tested against the query. If a document has been preprocessed, the results of preprocessing are loaded, and tested against the query. Preprocessing produces a worthwhile increase in speed. Loading the preprocessed files ready for constraint satisfaction takes (on average) less than a tenth of the time it takes to process the files to get to the same stage. However, preprocessed files do take around 60 times more filespace than the source files from which they are derived[2].

Although loading a preprocessed file is much faster than processing from scratch, the loading is still a significant factor in the total processing time. In fact for simple queries the loading accounts for over 90% of processing time. We were therefore keen to load only those files where there was a chance of success for a query. One way to do this is to split the query into an IR and an IE component and to use IR to pre filter the set of documents. If we only have to load one tenth of the documents then again we can expect a 10 times speed up (assuming the time to do IR is relatively trivial relative to the time as a whole).

A simple way to achieve IR filtering is to extract out any non-optional keywords from the query, and then only process those documents that contain the keywords. However, many of the queries which we use do not contain keywords at all, only sorts. In these cases, we cannot run IR over the source documents, since these do not contain the sortal information. Instead we search over a summary file which is created during preprocessing. This contains a set of all the sorts and words found in the file. The IR stage consists of selecting just those files which match the sorts and keywords in the query. This set is then passed to the IE component which deals with relational constraints such as *same sentence*, and interactions between constraints which have to be calculated on the fly such as *precedes* which are not indexed during preprocessing.

The IE component satifies the constraints in the

[1] We currently do not index commas but a positional constraint representing separation by a comma could easily be added.

[2] This is worse than it need be: we have not yet attempted to rationalise the preprocessed files, or use encoding schemes to reduce redundancy.

Figure 4: A typical one-off query for case: *John Smith, chairman of X-Corp*

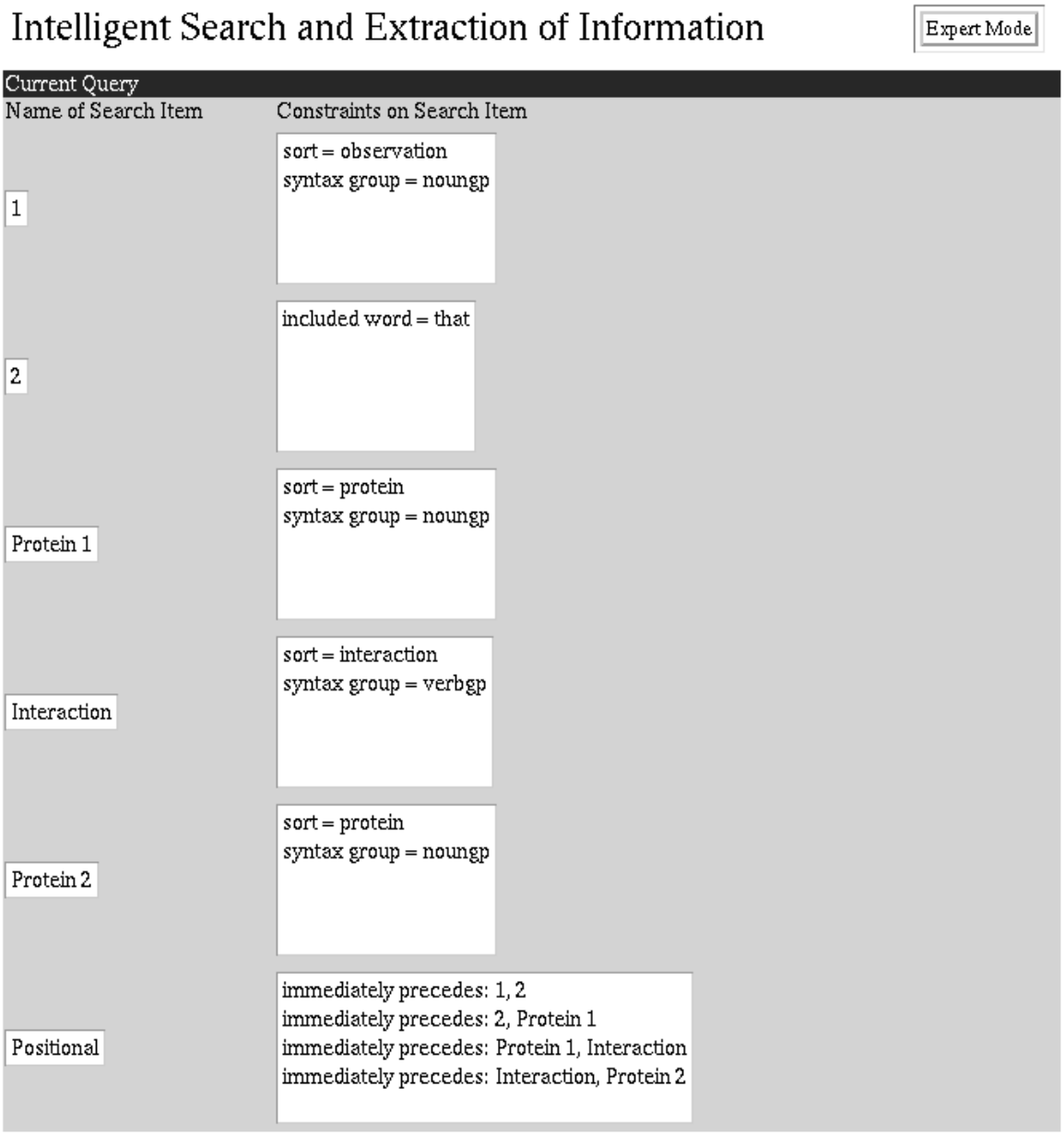

Figure 5: Query for protein interactions e.g. *observe that Taf-1 binds TBP.*

query against the constraints in the preprocessed file. The constraint solver tries to satisfy the most specific constraints in the query first. Constraints are indexed and reverse indexed for efficiency.

The processing times we currently achieve are very good by the standards of Information Extraction, and are adequate for the applications for which we have been using the system. The current approach is similar to that of e.g. Molla and Hess (Aliod and Hess, 1999), who first partition the index space into separate documents, and use the IR component of queries as a filter.

Table 1 shows the difference in processing times on two queries for two different datasets. Times for processing each query on each dataset are labelled Old for Highlight with no IR and no preprocessed files, New for Highlight with preprocessed files and NewIR for Highlight with both an IR stage and preprocessed files.[3]

New% and NewIR% give New/Old and NewIR/Old respectively. From the table we can see that (for these queries) adding the preprocessed files reduces total processing time by around 75%. Adding the IR stage reduces it by a further couple of percent in the case of the FT files and by 10% for the WSJ files. The performance increase on FT data is less dramatic because the data provides more hits (i.e. we extract more templates per input file) but note that for both datasets these improvements are at the worse end of the spectrum: both the WSJ and FT files stand a very good chance of containing both person and company (the sorts in our test queries) and so the IR component will propose a large number of each set for IE treatment, and we can also expect several hits per file, which tends to slow query processing. In other queries, e.g. a search for a company name, we would expect things to be much quicker as borne out by the results in Table 2.

4 Performance

Figures 6, 7 correspond to the queries in Figures 3 and 4 respectively. Figure 8 is the result of searching for protein interactions (a more general version of the query in Figure 5.)[4] Table 2 contrasts these results. FilesIR and FilesIE refer to the number of files input to each stage of the system, i.e. the query in Figure 6 was run over 500 news articles, only one of them was selected by our IR component and was thus input to the IE component. Hits denotes how many of the files passed to IE actually had at least one template in them and Templates shows how many templates were extracted as a result of the query. Time is the total time for the query in seconds.

The query in Figure 6 looks for documents containing *Esso* and a location.[5] Because *Esso* is such a good discriminator in this document set, appearing in only one out of 500 documents, the IE query only considers one document and the whole query succeeds in 0.5sec.

Figure 7 takes longer than this because 218 of 500 files contain the common sorts *person* and *company*. Figure 8 is similar but has the additional overhead of around 2/3 of all files having hits where Figure 7 has only 1/20.

Factors affect the total amount of time that a query takes include:

- specificity of the query
- complexity of the query
- size of the source file
- whether or not there is a hit in the file

With the addition of the IR stage, the specificity of the query has the potential to impact greatly on total query time as already seen. Having a hit in the file can also significantly affect timings since there is no chance of an early failure of constraint satisfaction. For example, taking a file from each of the queries in Figures 7 and 8 which each take 270msec to load (i.e. are of the same size) we find that the one in which there is no hit takes 10msec to process but the one is which there is a hit takes 6.5sec (of which just 110msec is spent writing results to file.)

The lessons from these results are not particularly surprising: queries should use the most specific words and sorts possible to get good value from the IR component. If there are many solutions—and remember that this system aims to extract specific information rather than a relevant document or document passage—then there *will* be a time penalty.

[3] The WSJ dataset consists of 100 Wall Street Journal articles, the FT set comes from 60 Financial Times documents. Query 1 looks for a person and company in the same sentence while Query 2 is for Person, X of/at/in Company. All tests were carried out on a Sun Ultra 2200 (200MHz, 256Mb RAM) running Sicstus Prolog 3.7.1 under Solaris 7.

[4] For space reasons we have only presented a sample of our query results.

[5] Because there is no sentential restriction on these two items, they do not appear on the same row in the output. This is an option available to the user which means that in queries of this kind, the output is a summary of the "true" output (a cross product of each item with each other item) which can obviously result in very large output tables.

Dataset	Query	Old (msec)	New (msec)	New%	NewIR (msec)	NewIR%
WSJ	1	133880	32890	25	19120	14
WSJ	2	135840	33610	25	19540	14
FT	1	94830	21300	22	18850	20
FT	2	92240	21590	23	18960	21

Table 1: Comparison of IR and non-IR preprocessing

Figure	FilesIR	FilesIE	Hits	Templates	Time (s)
6	500	1	1	19	0.5
7	500	218	11	11	27
8	100	59	42	44	66

Table 2: Comparison of query times

Our expectation is that complex queries which also involve a lot of hits are more likely for batch mode operation, so the time penalty will not be so crucial. One-off queries will tend to be simpler, involving searches for less frequent information. However, we are also investigating further ways to improve processing speed, in particular during constraint satisfaction.

5 Evaluation

We have not yet performed a large-scale evaluation of the user-customisable version of Highlight. The earlier version of Highlight was evaluated for the task of extracting protein interactions, obtaining a recall of 55-58% and precision of 77%. For this particular task we would not expect the results for the new system to differ significantly: the underlying engine is the same and we can use the interface to generate sets of constraints which are equivalent to the previous patterns. For example, Figures 5 (above) and 9 show how an old pattern can be generated in the new interface. Both look for phrases such as *...observed that TAF binds TBP*, the old pattern places an `interaction` tag around the relevant text while in the new interface the user selects which items will be output (in this case, the ones which have names rather than numbers.)

For other tasks, there are some factors which might affect the relative recall and precision figures between the original Highlight system and the customisable version. Firstly, the original Highlight system can use cascades of pattern matching rules to perform 'blocking tactics' i.e. one rule fires for the sole purpose of preventing another from firing. To achieve a similar ability in customisable Highlight we need to introduce a way of choosing between alternative queries when more than one can apply e.g. by always choosing the most specific. Secondly, the original Highlight sometimes resorted to external program calls which is no longer possible.

6 Applications

The system described in this paper is still under development but is already being used in pilot projects with commercial clients. Areas in which it has been deployed include:

- *expertise database*: extracting relationships between employees and the projects in which they are involved from internal company documents. The relationships degrade from those which represent certain involvement (e.g. writing a report) to those that represent possible involvement (e.g. mentioned in the same sentence).
- *competitor database*: extracting information about companies from Financial Times documents. Here, we were after in particular the chairman, chief executive and so on of any companies in the texts. Again, sets of relationships showed the degree of certainty with which a person was related to a company.
- *conference database*: extracting information such as conference title, location, date and cost from conference calls downloaded from the web. This task was particularly interesting because it required the gathering of information from a variety of locations in each document rather than the more usual single sentence.

In order to test the flexibility of the new system, we have also been recoding previous applications of the Highlight system.

Query results

Source	1	2	Sentence	Link
wsj_0473	Esso Resources Canada Ltd.	–	The producers include Shell Canada Ltd., a unit of Royal Dutch / Shell Group; Esso Resources Canada Ltd., a unit of Imperial Oil Ltd., which is 71 %–owned by Exxon Corp.; and Gulf Canada Resources Ltd., a unit of Olympia & York Developments Ltd.	link
wsj_0473	–	the Arctic	" Foothills wants to make it clear to other pipeline companies that it's on first insofar as transporting gas from the Arctic to southern markets ," Mr. Hillary said.	link
wsj_0473	–	Canada	At least two rival applications are expected to emerge in coming months, including one from TransCanada PipeLines Ltd., Canada's largest natural gas pipeline operator.	link
wsj_0473	–	Houston	The Toronto–based company, together with Tenneco Inc. of Houston, has had an incomplete proposal filed with Canadian regulators since 1984 that it is now updating.	link
wsj_0473	–	the Mackenzie River delta	Like Foothills, TransCanada's Polar Gas consortium plans to build a pipeline directly south from the Mackenzie River delta in Canada's western Arctic with an initial capacity to transport 1. 2 billion cubic feet of gas daily.	link
wsj_0473	–	Alberta	Industry sources said they expect a fierce battle to emerge between TransCanada, which has a monopoly on Canadian gas transportation east of Alberta, and Nova and Westcoast, which control the pipelines within and running west of Alberta, respectively.	link
wsj_0473	–	the Prudhoe Bay area	U. S. gas buyers must also decide whether they want to enter firm contracts for Mackenzie delta gas or develop Alaskan reserves in the Prudhoe Bay area first, a project that has been on hold for more than a decade.	link
wsj_0473	–	Alaska	But Foothills said it plans to seek regulatory approval to build an alternative line, the Alaska Natural Gas Transportation System further north toward Alaska.	link

Figure 6: Sample of results from query for keyword *Esso* and sort *location*

Query results

Source	Person	Position	Company	4	Sentence	Link
wsj_0111	Robert L. Bernstein	chairman and president	Random House Inc.	of	START Robert L. Bernstein, chairman and president of Random House Inc., announced his resignation from the publishing house he has run for 23 years.	link
wsj_0209	Murray Robinson	president	Delta & Pine Land Co.	of	" The development could have a dramatic effect on farm production, especially cotton ," said Murray Robinson, president of Delta & Pine Land Co., a Southwide Inc. subsidiary that is one of the largest cotton seed producers in the U. S.	link
wsj_0276	Michael Blair	former president and chief executive officer	Enfield Corp.	of	START Michael Blair, former president and chief executive officer of Enfield Corp., failed to win election to the company's board at a special shareholder meeting.	link
wsj_0391	David Rockefeller	chairman	Rockefeller Group	of	David Rockefeller, chairman of Rockefeller Group, said the company talked with many potential investors in Japan, the United States and Europe.	link
wsj_0509	Stephen Wolf	chairman	UAL Inc.	of	Among the other alumni are Stephen Wolf, now chairman of UAL Inc., and Thomas Plaskett, president of Pan Am Corp	link
wsj_0593	Joseph L. Dionne	chairman and chief executive officer	McGraw–Hill Inc.	of	START Joseph L. Dionne, chairman and chief executive officer of McGraw–Hill Inc., was elected to the board of directors of this electronics manufacturer.	link

Figure 7: Sample of results from query for *Person, X of Company*

Query results

Source	Protein 1	Interaction	Protein 2	Sentence	Link
bio_10004	TRRAP	interacts	the E2F1 transactivation domain	TRRAP also interacts specifically with the E2F1 transactivation domain.	link
bio_1014	Cdi1	interacts	cyclin–dependent kinases	In yeast, Cdi1 interacts with cyclin–dependent kinases, including human CDC2 (116940), CDK2 (116953), and CDK3 (123828), but not with CDK4 (123829).	link
bio_10187	Ptrf	interacts	both TTF1 and Pol I	Jansa et al. (1998) demonstrated that Ptrf interacts with both TTF1 and Pol I, and binds to transcripts containing the 3–prime end of pre–rRNA in vitro.	link
bio_10261	CNK	interacts	RAF	CNK physically interacts with RAF and appears to localize to cell–cell contact regions.	link
bio_10408	The SGT protein	interacts	the parvovirus nonstructural protein NS1	record (603419) The SGT protein interacts with the parvovirus nonstructural protein NS1.	link
bio_10444	RICK	interacts	CLARP	Inohara et al. (1998) demonstrated that RICK physically interacts with CLARP, a caspase–like molecule known to bind to FADD and caspase–8.	link
bio_10876	human TIM	interacts	Drosophila	Sangoram et al. (1998) demonstrated that human TIM interacts with Drosophila per, mouse PER1, and mouse PER2 (see 603426) in vitro.	link
bio_11073	MIZ1	interacts	MYC and NMYC	MIZ1 interacts specifically with MYC and NMYC (164840).	link

Figure 8: Sample of results from query for *X interacts with Y*

```
bio_relation :sp:

 [ VG1/tag(vg,VGType,Word1),                    [ VG1/tag(vg,VGType,Word1),
   that/ThatTag,                                  that/ThatTag,
   NP1/tag(np,N1,N1Sort,UL1,Id1),     ==>         [ NP1/tag(np,N1,N1Sort,UL1,Id1),
   VG/tag(vg,headed_vg,Word),                       VG/tag(vg,headed_vg,Word)],
   NP2/tag(np,N2,N2Sort,UL2,Id2)                  NP2/tag(np,N2,N2Sort,UL2,Id2)
 ]                                                ]/interaction(Id)
                                                ]

  if
    bio_transitive_relation(Word),
    observation_word(Word1),
    make_new_id(Id)
```

Figure 9: Pattern from basic Highlight for case: *observe that Taf-1 binds TBP.*

7 Summary

This paper has given a brief outline of a system which may be used for one-off queries in the style of current Information Retrieval systems but with the advantages of Information Extraction technology. The same interface can be used to build queries appropriate for more traditional batch mode Information Extraction tasks.

References

Diego Molla Aliod and Micheal Hess. 1999. On the scalability of the answer extraction system "extrans". In *Applications of Natural Language to Information Systems (NLDB'99)*, pages 219–224.

Claire Cardie and David Pierce. 1998. Proposal for an interactive environment for Information Extraction. Technical Report TR98-1702, Department of Computer Science, Cornell University.

Jerry R. Hobbs, Douglas Appelt, David Israel John Bear, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1996. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. In E. Roche and Y. Schabes, editors, *Finite State Devices for Natural Language Processing*. MIT Press.

David Milward. 2000. Distributing representation for robust interpretation of dialogue utterances. In *Proceedings of the 38th ACL Conference*.

Ellen Riloff. 1993. Automatically constructing a dictionary for Information Extraction tasks. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 811–816. AAAI/MIT Press.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1044–1049.

Stephen Sonderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. 1995. CRYSTAL:inducing a conceptual dictionary. In *Proceedings of the 14th International Joinct Conference on Artificial Intelligence*.

Rohini Srihari and Wei Li. 1999. Information extraction supported question answering. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*.

James Thomas, David Milward, Christos Ouzounis, Stephen Pulman, and Mark Carroll. 2000. Automatic extraction of protein interactions from scientific abstracts. In *Pacific Symposium on Biocomputing*.

Text Retrieval Conferences (TREC). 2000. `http://trec.nist.gov/`.

Roman Yangarber and Ralph Grishman. 1997. Customization of Information Extraction systems. In *Proceedings of the International Workshop on Lexically Driven Information Extraction*.