Index Expansion for Machine Reading and Question Answering

Giuseppe Attardi, Luca Atzori, Maria Simi

Dipartimento di Informatica, Università di Pisa, Italy
{attardi, atzoril, simi}@di.unipi.it

Keywords: index expansion, question answering, machine reading, information retrieval

Abstract. The paper reports our experiments in tackling the CLEF 2012 Pilot Task on Machine Reading for Question Answering. We introduce the technique of index expansion, which relies on building a search index enriched with information gathered from a linguistic analysis of texts. The index provides a highly tangled representation of the sentences where each word is directly connected to others representing both meaning and relations. Instead of keeping the knowledge base separate, the relevant knowledge gets embedded within the text. We can hence use efficient indexing techniques to represent such knowledge and query it very effectively. We explain how index expansion was used in the task and describe the experiments that we performed. The results achieved are quite positive and a final error analysis shows how the technique can be further improved.

1 Introduction

The paper reports our experiments in tackling the CLEF 2012 Pilot Task on Machine Reading for Question Answering. The task aimed at exploring the ability of a machine reading system to answer questions about a scientific topic, in particular about the Alzheimer disease, using knowledge extracted from biomedical texts.

The evaluation has the format of traditional Multiple Choice Reading Comprehension tests: it involves reading a single scientific article at a time and answering a set of questions regarding information that is stated or implied in the text. Multiple choice answers are provided for each question, each having five options with only one being correct. A background collection of reference articles about the topic is provided by the organizers, which may be exploited for learning knowledge useful in answering the questions. Nonetheless the principal answer is to be found among the facts presented in the given test document.

The rationale of the task [13] is to concentrate on the step of Answer Validation which is the last one in a traditional QA pipeline (Question Analysis, Retrieval, Answer Extraction, Answer Selection/Validation).

We approach the task using a technique that we call “index expansion”, which is the dual to query expansion. Query expansion adds terms to the query in order to achieve better recall, but this often results in poorer precision, since, because of term polysemy, the terms introduced may have also quite different meaning and may match irrelevant documents, introducing a lot of noise in the results.

Index expansion instead keeps the original query terms but adds variants of terms to the index, for example synonyms and hyperrnyms. This introduces much less noise, since if the variant term is used in the query, then it is relevant and the match will be successful. On the other hand, inappropriate terms will not be used in the queries and hence their presence in the index will not affect negatively the results. The only drawback of the approach is that the size of the index increases, but this is a minor issue, since the index is compressed and disk space is cheap. In other words we trade space for speed and accuracy, which is akin to the technique of “database denormalization”.

The approach extends the one that we proposed and applied successfully to the task of Blog Opinion Mining at TREC 2006 [3].

The index is enriched with information extracted from linguistic analysis of the documents. The index in fact can be seen as a multilayer index, where each layer represents one kind of annotation. The layers are:

— form	the original text
— lemma	the lemma of each word
— POS	the POS of each word
— head	the governor of each word
— deprel	the dependency relation of each word with its governor
— lemma	the lemmas of each word
— synonyms	list of synonyms for each word
— hypernyms	list of hypernyms for each word

The process of answering a question relies on a similar analysis of text: questions are parsed and from the parse tree a basic query in the DeepSearch query language is generated. To each of these queries one of the multiple possible answers is added. Those queries which obtain an answer with the highest score are considered to include the correct answer.

2 Document Analysis

The collection of background documents about Alzheimer's Disease provided for the task included: around 66,000 abstracts from PubMed[1]; around 8,000 Open Access full articles from Central[2]; full articles about the key hypotheses in Alzheimer Disease published by Elsevier.

The documents were provided also in a preprocessed format, split into sentences and tokens, each one annotated with lemma, POS, dependency parsing annotations

[1] http://www.ncbi.nlm.nih.gov/pubmed/
[2] http://www.ncbi.nlm.nih.gov/pmc/

from the dependency parser GDep [15], and two types of Named Entities, one from a UMLS-based NE tagger developed at CLiPS, and one from the ABNER NE tagger [16].

3 Index Expansion

The idea of index expansion is to create a highly tangled representation of the sentences where each word is directly connected to others representing both meaning and relations. Instead of keeping the knowledge base separate, the relevant knowledge gets embedded within the text. We can hence use efficient indexing techniques to represent such knowledge and query it very effectively with suitably modified techniques of information retrieval.

The first step of the process is to analyze the sentences and annotate them with syntactic and semantic tags. Despite the fact that documents were provided preprocessed with the GDep parser, we preferred to parse them with a parser that could produce Stanford Dependencies, which provide more refined analysis with respect to standard parsers and whose annotation is closer to represent semantic roles. In particular we will make use of the ability to distinguish apposition.

Besides syntactic information, we looked for some kind of semantic information, in particular grouping syntactic variants and identifying synonyms and hypernyms.

3.1 Synonym expansion

We explored using the Unsupervised Semantic Parser by [14], in order to obtain a set of synonyms or related terms to annotate the documents. USP transforms dependency trees in the Stanford Dependencies notation [9] into quasi-logical forms and clusters them to abstract away syntactic variations of the same meaning. The approach seemed promising, in particular since USP had been trained on the Genia Corpus. Results on the Genia corpus showed the USP ability to discover similar terms, albeit sometimes with opposite polarity, and alternative phrasing for the same concept. For example, here is a sample of the top clusters from the Genia corpus:

1. overexpression, over-expression
2. sustain, have
3. expression, co-expression, accumulation
4. receptor-alpha, receptor
5. low, highest
6. display, exhibit
7. htlv-i-infected, human, lipopolysaccharide-stimulated, uninfected
8. greater, higher, lower
9. emphasize, support
10. alteration, change
11. novel, many, other, new, known, latter, various, multiple, individual, several, certain, respective, additional
12. previously, originally

13. srf, e2f, nfkappab, nf-kappab, hgata-3, nf-kb
14. susceptibility, sensitivity
15. govern, effect, control, specify, regulate, modulate, mediate

Items no. 5, 7 and 9 clearly include opposites, but the remaining ones are indeed either syntactic variants or have similar meaning.

USP requires the input annotated with Stanford Dependencies [9]. These dependencies are typically produced using the Stanford Parser which produces constituency trees that are then converted to dependency trees.

This process turned out to be too slow to handle large collections. Therefore we decided to train the DeSR dependency parser [1] on a version of the Penn TreeBank annotated with basic Stanford Dependencies. This produced a native parser for Stanford dependencies which outperforms the Stanford parser itself in accuracy[3] (91.18 % of Unlabeled Attachment Score), but most importantly in performance, reducing significantly the parsing. This is due to the fact that DeSR algorithm has linear complexity and parsing a sentence takes in the order of hundredth of a seconds, while the Stanford parser is cubic and the time grows to dozen of minutes for long sentences.

We applied USP to the Elsevier corpus. We had to overcome a number of problems with the implementation, in particular:

- the handling of apostrophes, which the parser introduces to denote copy nodes and therefore confuse USP
- the handling of underscores, whose presence in words also confused USP
- infinite loops due to the presence of cycles in the dependencies

After fixing these problems, we had to partition the Elsevier collection into four smaller subsets because the parser was too slow to handle it as a whole. Unfortunately the clusters that we were able to obtain were not very significant. Here are the top clusters from the first subset:

- furthermore, however
- rinse, wash
- moreover, thus
- recent, previous
- feature, manifestation
- nacl, hepes
- experimental, procedures
- the (dep body), the (amod body), the (nn body), the (num body), the (nsubj body), the (advmod body), the (poss body), the (nsubjpass body), …
- phosphorylation (prep_at s45), phosphorylation (prep_of s45), phosphorylation (amod s45), …

Except for the first few, all the remaining ones were cases of alternative parses of the same phrase, e.g. ‘body’ with different dependency relations: ‘amod’, ‘num’ or ‘nsubj’. Therefore the clusters turned out not to be very useful for our purposes.

---

3 https://sites.google.com/site/desrparser/Announcements/stanforddependencies

Hence as an alternative we resorted to use WordNet [9] to extract synonyms and hypernyms. Two layers of annotation were added to represent them for each term.

3.2 Appositions and Acronyms

We exploit the dependency annotations as appositions, to add the apposition at the position of the head. For example, “Abeta” occurs as an apposition for “amyloid beta”: the term “Abeta” is then added as a synonym in correspondence of “beta” and hence it will inherit any relations that the phrase has.

Similarly the expansion of acronyms can be added as variants of terms.

3.3 Syntactic Variants

Syntactic variants are also added to the expanded index. For example to deal with alternative passive forms, when a term has the dependency “nsubj_pass”, to the index is also added the same term with a tag as “dobj”, while the corresponding “agent “ is annotated also as “nsubj”.

3.4 Indexing

The search engine that we used in our experiments is called DeepSearch and is built by means of IXE [1], an Open Source search engine library in C++ that we have been developing along several years.

DeepSearch provides facilities for passage retrieval, so that it can return individual passages, in our case sentences, matching queries. Scoring is based on a relevance metric for the match within one sentence or within a number of adjacent sentences.

For this task we exploited the capability of IXE of dealing with multiple layers in documents. A layer is an overlay of different terms on the same document. Each layer has its own full-text index and can be queried independently. However the layers can be considered as stacked and querying performed across layers, for instance one can find a word with a given POS by searching for the word in the text layer that has the given POS in the same position in the POS layer. Also proximity and phrase searches that rely on term positions can exploit this.

Besides having multiple layers, a layer can have multiple terms in the same position. This feature will be exploited to deal with variants, like synonyms and hypernyms.

Notice that in among both synonyms and hypernyms we include the lemma itself, hence there is no loss in generality when querying on these layers.

Finally, the column for dependencies is dealt specially in order to enable searching for terms that are syntactically related. IXE exploits the fact that heads of terms represents positions, and list of positions are already represented in an inverted index. Therefore a special posting list with position is created for the heads column. The technique used for search is based on the Small Adaptive Set Intersection [1, 8], which relies on cursors, which are scanned in parallel, until a match is found at the same document position.

The SASI algorithm has been extended to handle dependencies queries, which look for two related terms *d* and *h*, where *h* is the head of *d*. A special dependency cursor uses three inner cursors, cursor C_d on the posting list of *d*, cursor C_h for that of *h* and a cursor H_d on the postings of the heads of *d*. It first uses C_d and C_h to find matches for *d* and *h*. Whenever such a match is found, it checks that the value stored in the posting list of *H* at the position of *d* corresponds to the position of *h*. Having a single posting list for all terms, rather than having one for each term, allows repeating the last step and checking whether there is a transitive dependency. Despite the fact that the posting lists for heads becomes very long, the algorithm is still quite fast since it exploits skip lists to quickly scan such list.

To illustrate what goes into the index, we show here the annotated parse tree for one sentence from one of the reading documents (with slight corrections of parser mistakes) that we will also use as an example for the process of answer selection:

form	*lemma*	*POS*	*H*	*dep*	*NE*	*synonym*	*hypernym*
the	the	DT	4	det	O		
γ-secretase	γ-secretase	JJ	3	amod	B-protein		
inhibitor	inhibitor	NN	4	appos	O	inhibitor	substance drug
Semagacestat	Semagacestat	NN	11	nsubj_pass dobj	O		
tested	test	VBN	4	nmod	O	essay run exam screen examine prove try trial test examination	examine evaluate judge submit check experiment attempt effort endeavor check see ascertain watch ...
in	in	IN	5	VMOD	O		
phase	phase	NN	10	nmod	O	phase form stage	point arrange appearance visual_aspect time_period period synchronise state
III	III	CD	10	nn	O		
clinical	clinical	JJ	10	amod	O	clinical	
trials	trial	NNS	6	prep_in	O	run visitation trial tribulation tryout test	experiment attempt effort endeavour try affliction proceeding contest competition

Each column in the table represents a layer, which we represent here vertically for easier readability. Most layers contain a single term. The terms in the last two layers are to be considered as variants; hence they appear in the index as if they had the same position. For eample the sentence will match a query for both form "Semagates-tac" and synonym "check". Notice that the presence of a possibly irrelevant synonym like "run" would not affect any search that does not use it explicitly, which will happen instead if one performs query expansion with synonyms.

3.5 Query Language

The DeepSearch query language allows specifying conditions occurring simultaneously at the same position in different layers. Each layer is identified by its name, so for example:

```
ne:protein
```

matches the term protein in the "ne" (Named Entity) layer.

```
dep:nsubj
```

matches the term "nsubj" in the "dep" (Dependency Relation) layer. The co-occurrence of these condition at the same position, i.e. looking for a term which is both a protein and the subject of a verb, can be specified using the align operator "|":

```
ne:protein|dep:nsubj
```

Moreover one can specify the presence of a syntactic dependency, either direct or indirect, between two terms, like in this example:

```
(ne:protein|dep:nsubj <- lemma:test)
```

Indirect dependencies are useful since quite often terms are connected though intermediate prepositions.

Dependencies can be chained, as in:

```
(phase <- lemma:trial <- lemma:test)
```

These queries can be tested in our online demo accessible at: http://semawiki.di.unipi.it/alzheimer/

4 Question Answering

4.1 Query generation

Questions are processed similarly to documents, using the parsers DeSR and then USP, in order to obtain a parse tree with semantic annotations.

A script analyzes the annotated parse tree and generates a query for the enriched index.

Here is an example of the processing of one of the questions in the evaluation:

What candidate drug that blocks the γ-secretase is now tested in clinical trials?

The sentence is parsed with Stanford dependencies and then expanded with synonyms, hypernyms and syntactic variants, obtaining the following layers (omitting some of the synonyms, hypernyms):

form	*lemma*	*POS*	*H*	*dep*	*NE*	*synonym*	*Hypernym*
What	What	WP	2	nsubj	O		
candidate	candidate	NN	0	root	O	prospect candidate campaigner nominee	person individual someone somebody mortal soul politician politico political_leader
drug	drug	NN	2	dobj	O	do_drugs drug dose	consume ingest take_in take havemedicate medicineagent
that	that	WDT	3	NMOD	O		
blocks	block	VBZ	3	rcmod	O	obstruct freeze obturate immobilise auction_block pulley_block mental_block stymie stoppage forget jam …	computer_memory_unit collection aggregation accumulation assemblage shape form artifact inability cast casting obstruction machine obstruction obstructor impediment hide conceal prevent forestall foreclose preclude forbid …
the	the	DT	7	det	O		
γ-secretase	γ-secretase	NN	10	nsubjpass dobj	B-protein		
is	be	VBZ	10	auxpass	O		
now	now	RB	10	advmod	O		
tested	test	VBN	5	ccomp	O	essay run exam screen examine prove quiz try trial tryout test examination …	examine evaluate judge take submit check core communication communicating experiment experimentation attempt effort endeavor try determine check find_out see ascertain watch learn covering cover …
in	in	IN	10	VMOD	O		
clinical	clinical	JJ	13	amod	O	clinical	
trials	trial	NNS	10	prep_in	O		
?	?	.	8	P	O		

Among the hypernyms for the word “candidate” appear terms like “politician”. It is fairly clear that if these terms were used for expanding the query, quite confusing results might be retrieved, unless one does some sophisticated kind of word sense disambiguation to discard those terms. In our case though, the last three columns of the analyzed sentence are discarded during query generation.

The query generator creates a basic DeepSearch query, which includes relevant terms and both syntactic and semantic features from the query. A list of clauses is produce which are combined in a Boolean disjunctive query. A few heuristics are used in producing such clauses, for instance the possible answer is not included if it is already present in another clause from the question.

For the above example the query generator produces this base DeepSearch query:

candidate OR syn:candidate OR drug OR syn:drug OR γ-secretase OR syn:γ-secretase OR clinical OR syn:clinical OR lemma:candidate OR (ne:protein <- lemma:test) OR syn:drug

Five variants of this query are submitted by adding to each, one of the possible multiple answers. The one with the addition of “Semagacestat” returns two results, with the highest score of 19.35. The query with the addition of “LPR1” returns one result with a score of -1.64; the addition of “biochemical” returns 21 results with the best score of 4.24; the addition of “AD” obtains 9 results, with the best score of 9.18; the addition of “PSEN1” returns 21 results, the highest with a score of 5.51. Hence “Semagacestat” is selected as the correct answer, as it is indeed.

The sentence retrieved with the highest score is the one presented in Section 3.4.

4.2 Answer Selection

Most Question Answering system perform sophisticated processing on the candidate answers, in order to determine which one is the most appropriate one. This processing sometimes involves complex reasoning based on theorem proving techniques [11]: the answer is transformed in some form of first order logic formula and an attempt is made to prove that the formula entails the question either directly or by abduction.

Our system instead relies only on the ranking provided by the DeepSearch engine. If the engine does not return any answer for a given query, the candidate solution is discarded. When more than one query has answers, the one is chosen whose first answer has the highest rank.

One limitation of the approach is due to the fact we employ a passage retrieval engine, which splits documents at the sentence level and returns sentences that match the query. We had planned to apply the anaphora resolution tool [5] that we had developed for SemEval 2011 to add an anaphora layer to our expanded index, but time limitation prevented us from doing it. The problem is somewhat alleviated by exploiting feature of the passage retrieval engine, which considers matches occurring also in adjacent sentences, albeit with a lower score. This is controlled by parameter `ConsecutivePassage` in DeepSearch, which was set to 2 in our experiments.

5 Evaluation

The metric for the evaluation is c@1 [12], which is based on the number of correct single answers but takes into account the option of not answering certain questions. We submitted a single run for evaluation, which obtained a cumulative c@1 score of 0.55, with the following breakdown on the four documents:

Reading Test	*n*	n_R	n_U	*c@1*
1	10	4	0	0.40
2	10	5	0	0.50
3	10	6	0	0.60
4	10	7	0	0.70

5.1 Error Analysis

Here are the answers provided by the system for the fourth reading document. Correct answers are highlighted in bold.

Reading Document 4.

1. What effect can be observed when when γ-secretase is blocked?
   System: **APP-CTF accumulation** Correct: APP-CTF accumulation
2. When APH1 genes are overexpressed in MEF KO what happens with the Aβ?
   System: **They are longer** Correct: They are longer
3. In which gene are mutations associated to many cases of early-onset familial forms of Alzheimer's disease?
   System: **PSEN1** Correct: PSEN1
4. What experimental technique was used specifically to purify the γ-secretase complex?
   System: lysate Correct: affinity chromatography
5. What peptide is able to control the expression of the ApoE gene?
   System: **AICD** Correct: AICD
6. Which amino acid is critical for the activity of the PS1 protein?
   System: **aspartate** Correct: aspartate
7. What experimental technique was used to determine the structure of γ-secretase?
   System: immunostaining Correct: EM
8. What candidate drug that blocks the γ-secretase is now tested in clinical trials?
   System: **Semagacestat** Correct: Semagacestat
9. What mutation of the PS1 protein causes γ-secretase activity almost to disappear?
   System: wild-type Correct: P436Q
10. How many mutations relevant for familial forms of Alzheimer's disease have been detected for the PSEN1 gene?
    System: **185** Correct: 185

We briefly investigate the reason for the failures.

For the 8th question the query generated contained the phrase "affinity chromatography", but neither the term “affinity” nor “chromatography” appear in the document.

For question 7, the query with “EM” did not have the highest score. The answer was hard to find, because the relevant sentence was twisted, saying that “protocols for the purification of … γ-secretase … allowed the reconstitution of 3D structures … by EM”. Some form of splitting and rewriting of the sentence might help with cases like this.

Finally, in question 9 “wild-type” prevailed over “P436Q”, because the generated query did not constrain the term “wild-type” to be connected with “mutation”.

These errors appear that could be reduced by improving some aspects of index expansion and query generation.

6 Conclusions

The approach of index expansion relies on enriching the index with information which is gathered from a linguistic analysis of texts. Differently from traditional approaches to Question Answering, where an abstract separate formal representation of texts is produced and then queried through some logical reasoning process, our approach keeps the information in the text itself.

The idea is to create a highly tangled representation of the sentences where each word is directly connected to others representing both meaning and relations. Instead of keeping the knowledge base separate, the relevant knowledge gets embedded within the text. We can hence use efficient indexing techniques to represent such knowledge and query it very effectively with suitably modified techniques of information retrieval.

The approach proved fairly effective for the Pilot task of Machine Reading for Question Answering. In the experiments we incurred into some limitations of the tools we had planned to use. In particular it is important for the approach to be able to identify alternative forms of expressions, in particular synonyms and hypernyms, which are specific to the domain. We had some success in doing this by using dependency relations produced by a statistical dependency parser. In particular the apposition relation allowed identifying synonyms and the recognition of passive forms allowed normalizing them. We also tried some recent tools for semantic analysis that in principle could have provided additional versions of linguistic variants, more domain specific. This did not work as expected and we had to resort to general linguistic knowledge provided by WordNet. We also did not have time to incorporate our tool for coreference resolution.

We hope that exploiting better or tuned versions of the tools for syntactic and semantic analysis of text, the index expansion approach can provide an effective solution to answer validation in the context of question answering.

Acknowledgments.
Mihai Surdeanu provided us access to a version of the Penn Treebank annotated with basic Stanford Dependencies and assisted us in the training of the DeSR parser.

References.

1. Attardi, G.: IXE at the TREC Terabyte Task. In: *Proceedings of The Forteenth Text Retrieval Conference* (*TREC 2005*), NIST, Gaithersburg (MD) (2005)
2. Attardi, G.: Experiments with a Multilanguage Non-Projective Dependency Parser. In: *Proc. of the Tenth Conference on Natural Language Learning*, New York, (NY) (2006)
3. Attardi,G., Simi, M.: Blog Mining Through Opinionated Words, *Proceedings of The Fifteenth Text Retrieval Conference (TREC 2006)*, NIST, Gaithersburg (MD), (2006)
4. Attardi, G., Dei Rossi, S., Simi, M.: The Tanl Pipeline. In: *Proc. of Workshop on Web Services and Processing Pipelines in HLT*, Malta (2010)
5. Attardi, G., Dei Rossi, S., Simi, M.: TANL-1: Coreference Resolution by Parse Analysis and Similarity Clustering. In: *Proc. of SemEval 2010*, Uppsala (2010)
6. Attardi, G., Simi, M., Zanelli, A.: Tuning DeSR for Dependency Parsing of Italian, In: *Proc. of Evalita 2011*, Springer *LNCS* (to appear) (2012)
7. Bird, S., Looper, E.: Natural Language Processing with Python. O'Reilly Media (2009)

8. Demaine, E.D., López-Ortiz, A., Munro, J.I.: Adaptive set intersections, unions, and differences. In: *Proc. of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 743–752 (2000)
9. de Marneffe, M.-C., Manning, C.D.: The Stanford typed dependencies representation. In: *COLING 2008 Workshop on Cross-framework and Cross-domain Parser Evaluation* (2008)
10. Miller, G.A.: WordNet: A Lexical Database for English. *Communications of the ACM,* Vol. 38, No. 11: 39-41 (1995)
11. Moldovan, D., et al.: COGEX: A Logic Prover for Question Answering. In: *Proceedings of HLT-NAACL* 2003, Edmonton, pp. 87-93 (2003)
12. Peñas. A., et al.: Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation. In: C. Peters et al.(Eds.), *Multilingual Information Access Evaluation. Vol. I, Text Retrieval Experiments*. Workshop of the Cross-Language Evaluation Forum. CLEF 2009, *LNCS* 6241, Springer-Verlag, Corfu, Greece (2010)
13. Peñas, A., et al.: Overview of QA4MRE at CLEF 2011: Question Answering for Machine Reading Evaluation. In *Proc. of CLEF 2011*, Amsterdam (2011)
14. Pon, H., Domingo, P.: Unsupervised semantic parsing. In *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing*, Volume 1, 1–10, Singapore (2009)
15. Sagae, K., Tsujii, J.: Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proc. of the CoNLL 2007 Shared Task*. Prague, Czech Republic (2007)
16. Settles, B.: ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192 (2005)