Genic Interaction Extraction with Semantic and Syntactic Chains

Sebastian Riedel S.R.RIEDEL@SMS.ED.AC.UK
Ewan Klein EWAN@INF.ED.AC.UK
Institute for Communicating and Collaborative Systems, School of Informatics, The University of Edinburgh, Edinburgh EH8 9LW, Scotland UK

Abstract

This paper describes the system that we submitted to the "Learning Language in Logic" Challenge of extracting directed genic interactions from sentences in Medline abstracts. The system uses Markov Logic, a framework that combines log-linear models and First Order Logic, to create a set of weighted clauses which can classify pairs of gene named entities as genic interactions. These clauses are based on chains of syntactic and semantic relations in the parse or Discourse Representation Structure (DRS) of a sentence, respectively. Our submitted results achieved 52.6% F-Measure on the dataset without and 54.3% on the dataset with coreferences. After adding explicit clauses which model non-interaction we were able to improve these numbers to 68.4% and 64.7%, respectively.

1. Introduction

This paper describes the system that we submitted to the "Learning Language in Logic" (LLL) Challenge of extracting directed genic interactions from sentences in Medline abstracts. As an illustratation of the extraction task, given a sentence like (i), we wish to extract information of the form in (ii):

(i) Localization of SpoIIE was shown to be dependent on the essential cell division protein FtsZ.

(ii) *genic_interaction(FtsZ, SpoIIE)*

The LLL Challenge organizers provided training and test sets containing sentences with and without coreferential expressions. For all data sets, 'enriched' versions were available which contained lemmas and syntactic parses for each sentence.

Our initial goal was to explore whether relation extraction could be effectively carried out over semantic representations, rather than surface strings or parse structures. In particular, we decided to take as our starting point the logical forms produced by ccg2sem (Bos, 2005), a tool whose input is a CCG (Steedman, 2001) dependency tree built by a wide coverage statistical parser (Clark & Curran, 2004) and whose output is a Discourse Representation Structure (DRS) (Kamp & Reyle, 1993). This approach was based on the hope that target relations between entities would be easier to recover once the whole sentence had been converted into a set of semantic relations. Although we still believe this approach has considerable potential, we found that for the LLL Challenge dataset, it was less successful than we had hoped, primarily due to the effect of parser errors and to problems in processing the semantic output that ccg2sem yielded for coordinate structures. However, it turned out that using syntactic information from the enriched data sets improved performance significantly.

We had a strong bias in terms of the clauses to be learned: they had to connect both genes transitively. With this requirement in mind and the observation that our initial attempts to apply ILP systems (FOIL and Aleph) did not yield such clauses, we decided instead to generate a set of clauses based on *chains* of relations between the two genes. Clause candidates that were automatically extracted from the training corpus were fed to a Markov Logic (Domingos & Richardson, 2004) system; this in turn learned probabilistic weights that reflected how often the clause candidates were actually observed in the training data.

Appearing in *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

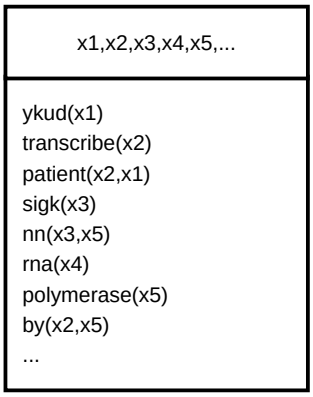

Figure 1. A simple DRS in box representation

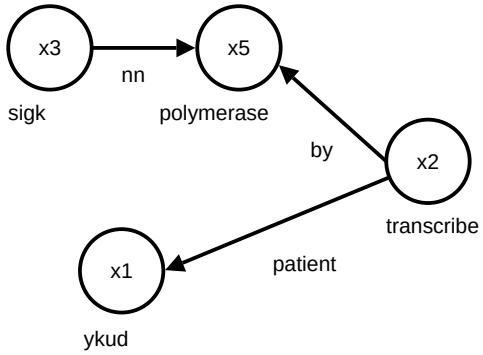

Figure 2. A semantic chain between two discourse referents

2. Data Preprocessing

2.1. Tokenization, Part-of-speech Tagging and Parsing

In order for ccg2sem to create logical forms, the training sentences had to be tokenized, part-of-speech tagged and parsed into a CCG tree. Tokenization was straightforward: the gene terms (retrieved from a list) were mapped into single tokens even when containing whitespace, hyphens or parentheses. part-of-speech tagging and parsing were both handled by the CCG parser(Clark & Curran, 2004).

2.2. DRS Construction

After parsing, the CCG tree was transformed by ccg2sem into a Discourse Representation Structure (DRS), complete with resolved coreferences.

DRSs are defined as ordered pairs of a set of discourse referents and a set of DRS-conditions, where conditions can be n-ary relations and equations over discourse referents (DRs), together with implications, disjunctions and negations of sub-DRSs. For a sentence like

> ykuD was transcribed by SigK RNA polymerase from T4 of sporulation

ccg2sem produced, among others, a discourse referent $x1$ referring to the ykuD gene, $x2$ which refers to a transcription event and $x3$, $x4$ and $x5$ which are associated with *SigK RNA polymerase*. Figure 1 shows a graphical representation of the DRS the upper section of the box contains the discourse referents, and the main box contains the conditions.[1]

[1]Note that *nn* indicates an underspecified compound noun relation.

2.3. Chains

For every gene pair in a sentence, the shortest semantic chain between the corresponding pair of Discourse Referents (DRs) was extracted. To find a DR given a gene token we used the token-to-DR mapping recovered from the DRS file.

Given two DRs a, b corresponding to gene terms, we define a *semantic chain between a and b*, $sem_{a,b}$, as a sequence of DRs $dr_1 = a, dr_2, \ldots, dr_n = b$ together with a sequence of edges $e_1, e_2, \ldots, e_{n-1}$, where each edge $e_i = (rel_i, dir_i)$ consists of a relation rel_i (in the set of relations produced by ccg2sem) between dr_i and dr_{i+1} and a direction $dir_i \in \{\rightarrow, \leftarrow\}$ which indicates whether $rel_i(dr_i, dr_{i+1})$ or $rel_i(dr_{i+1}, dr_i)$ holds. Figure 2 illustrates a semantic chain extracted from the DRS in Figure 1. As can be seen, the circled nodes correspond to DRs, unary relations label the nodes, and binary relations label the arcs between the nodes. Informally, Figure 2 says that there is a transcription event $x2$, the patient of $x2$ is ykuD, the *by* agent of $x2$ is a polymerase $x5$, and SigK is in an underspecified relation to $x5$.

Syntactic chains were calculated in the same fashion, but based on the parse representations supplied in the enriched LLL Challenge training data rather than on the output of the CCG parser. Given a phrase such as *the essential cell division protein*, the set of parse relations would include the clause `relation('mod_att:N-ADJ',13,10)`, where `13` and `10` are the word indices of *protein* and *essential* respectively. More generally, then, syntactic relations were of the form $relation(rel_i, w_i, w_j)$, where rel_i is one of a fixed set of syntactic relations assigned by the LLL parser. We define a *syntactic chain between word indices i and j*, $syn_{i,j}$, as a sequence of word indices $w_1 = i, w_2, \ldots, w_n = j$ and a sequence of edges $e_1, e_2, \ldots, e_{n-1}$, where each edge $e_i = (rel_i, dir_i)$ con-

sists of a syntactic relation rel_i between w_i and w_{i+1} and a direction $dir_i \in \{\rightarrow, \leftarrow\}$ which indicates whether $relation(rel_i, w_i, w_j)$ or $relation(rel_i, w_j, w_i)$ holds.

3. Machine Learning

Initially we tried to apply Inductive Logic Programming (ILP) directly to the conditions of our DRSs. This yielded clauses which were very dependent on the actual gene names due to the small size of the training set. Moreover, it was not possible to extract clauses which included the generalization that interacting genes must have a connecting semantic chain. This can be attributed to two reasons. First, since the chains in question can be rather long, they would induce correspondingly large clauses; however, the latter are filtered out by the ILP algorithms because they failed to cover enough examples relative to their complexity. Second, the chains can have varying lengths and might match only in small subchains, which makes it difficult for the ILP method to find generalizations.

In the light of this observation we decided to directly extract a set of candidate clauses which capture certain subparts of the semantic chains. Instead of discarding clauses which don't hold in all cases we used Markov Logic[2](Domingos & Richardson, 2004) to attach and learn weights for them. In Markov Logic the weight of a formula (or clause in our case) corresponds to the difference in log-likelihood between a world where the formula holds and a world where it doesn't, all other things being equal.

A *Markov Logic Network* L is a set of formula-weight pairs (F_i, w_i). Together with a set of constants it can be mapped to a log-linear joint probability distribution

$$(1) \quad p_w(X = x) = \frac{1}{Z} exp\left(\sum_{f \in Features_L} w_f f(x)\right)$$

over a sequence $X = (X_1, X_2, \ldots)$ of binary variables, one for each possible ground predicate (with respect to the available constants). The value of such a node is 1 if the ground predicate is true, 0 otherwise. The set of feature functions $Features_L$ is created by adding a feature function f for each possible grounding of each formula F_i in L. A function f returns 1 if its corresponding ground formula is true given x, 0 otherwise. w_f is the weight w_i of the original formula F_i defined in L. Z is a normalisation factor.

[2]See (Riedel & Meza-Ruiz, 2005) for a publicly available implementation.

3.1. Candidate Clause Generation

As mentioned above, we automatically extracted a set of candidate clauses from the given data. All these clauses were based on subchains within the shortest semantic and syntactic chains of the gene pair to be classified. Every time we observed a particular subchain we added a clause that entails $genic\_interaction(A, B)$ or $\neg genic\_interaction(A, B)$ depending on the label of the observed gene pair.

For instance, after seeing a $(agent, \leftarrow), (patient, \rightarrow)$ subchain in a genic interaction chain we generated:

$$genic\_interaction(A, B) :-$$
$$contains(sem_{A,B}, (agent, \leftarrow), (patient, \rightarrow))$$

This clause roughly[3] reads "if gene term A is related to some DR which is the agent of an event and something related to gene term B is the patient of this event, then there is a genic interaction between A and B". Note that 'related' here means reflexively and transitively related with respect to the binary relations in the DRS.

In addition, *typed* subchain clauses are extracted. A typed clause requires certain DRs in the chain to be members of a specified predicate:

$$genic\_interaction(A, B) :-$$
$$contains(sem_{A,B}, (agent, \leftarrow), activate)$$

This clause reads "if something related to gene term A activates something that is related to gene term B, then genic_interaction(A, B) holds" because *activate* is the predicate of the DR followed by the agent relation.

In the case of syntactic chains, clauses look as follows:

$$genic\_interaction(A, B) :-$$
$$contains(syn_{A,B}, (subj, \rightarrow), (comp\_of, \rightarrow))$$

Typed syntactic clauses used words to type the relation members; an example is the following (negative) clause:

$$\neg genic\_interaction(A, B) :-$$
$$contains(syn_{A,B}, (comp\_of, \leftarrow), activity)$$

3.2. Weight Estimation

To find the weights, we maximized the logarithm of the conditional likelihood of the training data

$$(2) \quad \sum_{(x_h, x_o) \in T} log\left(p_w(X_h = x_h | X_o = x_o)\right)$$

[3]We say 'roughly' because one also has to bear in mind that only shortest paths are extracted.

where X_h is a list of possible hidden variables and x_h are the corresponding values in a concrete observation. In this case X_h contains all variables refering to possible grounded *genic_interaction* atoms. X_o is the set of observed variables and corresponds to all possible instantiations of the *contains* predicates. T is the set of all training observations (x_h, x_o). In our current setting, each *genic_interaction* atom only depends on *contains* atoms, which are fully observed. This gives rise to

$$(3)\quad p_w\left(X_h = x_h | X_o = x_o\right) = \prod_{Gene\ pairs(a,b)} p_w\left(X_{g(a,b)} = x_{g(a,b)} | X_{c(a,b)} = x_{c(a,b)}\right)$$

where the variable $X_{g(a,b)}$ corresponds to the ground atom $genic\_interaction(a,b)$, and $X_{c(a,b)}$ is a list of all variables corresponding to *contains* atoms that relate to the syntactic and semantic chains between a and b.

With (3), the conditional in (2) simplifies to

$$(4)\quad \sum_{(x_h,x_o)\in T} \sum_{Gene\ pairs(a,b)} log\left(p_w\left(x_{g(a,b)} | x_{c(a,b)}\right)\right).$$

where $p(x|y)$ is an abbreviation for $p(X = x|Y = y)$

To calculate the conditional in (3), Bayes Rule yields

(5)

$$p_w\left(x_{g(a,b)} | x_{c(a,b)}\right) = \frac{p_w\left(x_{g(a,b)}, x_{c(a,b)}\right)}{p_w\left(1, x_{c(a,b)}\right) + p_w\left(0, x_{c(a,b)}\right)}$$

where the joint probabilities can be calculated using a version of (1) that only contains features of ground clauses related to the gene pair (a,b). This is sound due to the independence of genic interaction labels we are assuming in our clauses.

Using the gradient of (4), we run several iterations of L-BFGS (Liu & Nocedal, 1989), a gradient descent implementation, until convergence.

3.3. Inference

To infer the truth value of an actual genic interaction candidate pair, one could run an inference algorithm such as Belief Propagation or Gibbs Sampling in a Markov network equivalent to (1). However, as all hidden atoms only depend on a set of fully observed atoms this is not necessary; we simply calculate

$$(6)\quad p_g\left(a,b\right) = p_w\left(x_{g(a,b)} | x_{c(a,b)}\right)$$

for each possible gene pair (a,b) using (2). In the case of the gene pair $ykuD, SigK$ in our example in section 2.2 $x_{g(a,b)}$ would correspond to a truth value for the atom

$$genic\_interaction(ykuD, SigK)$$

and $x_{c(a,b)}$ would contain a sequence of truth values for atoms such as

$$contains(sem_{ykuD,SigK}, (patient, \leftarrow), (by, \rightarrow))$$

(true) or

$$contains(syn_{ykuD,SigK}, (comp\_of, \leftarrow), activity)$$

(false).

We classified a pair a, b as participating in a genic interaction if $p_g(a,b) > 0.5$. If both (a,b) and (b,a) are classified as genic interactions we only accept the one with the highest probability. However, in future versions this could be done more soundly using a clause such as

$$\neg genic\_interaction(A,B) :-$$
$$genic\_interaction(B,A)$$

on the assumption that genic interaction is always asymmetric.

Table 1. Clause sets

No.	Clause sets	Neg. clauses
1	syn2-0, syn2-1	yes
2	sem2-0, sem2-1	yes
3	1, 2	yes
4	syn2-0, syn2-1	no
5	3, sem1-1l, sem1-1r	no

4. Results

We tested different sets of clauses as candidates for the weight estimation. Table 1 enumerates the combinations of clause sets we used during the experiments. *sem1-0, sem2-0* and *syn2-0* refer to (untyped) clause sets with semantic or syntactic chains of length one and two, respectively. *sem1-1l*, *sem1-1r, syn1-1l* and *syn1-1r* represent sets where either the left or right node of all clauses is typed with a predicate or word, respectively. *sem2-1* and *syn2-1* type the middle node of subchains with length two.

4.1. Without Coreferences

Table 2 shows results for the test data without coreferences. Note that the submitted result was clause

Table 2. Results on the test data without coreferences

Clause set No.	Precision	Recall	F-M
1	65.0	72.2	68.4
2	68.5	44.4	53.9
3	64.8	64.8	64.8
4	60.8	51.8	55.9
5	60.9	46.2	52.6

Table 3. Results on the test data with coreferences using the clause sets 1, 2 and 5 from Table 1

Clause set No.	Precision	Recall	F-M
1	63.2	66.2	64.7
2	50.0	33.7	40.2
5	55.6	53.0	54.3

set 5, since this was the clause set that yielded the best results on the cross-validated training set (65% F-Measure). On the test data, it only achieved 52.6% F-measure, as shown. Clause set 5 did not use negative clauses — this functionality was not implemented by the submission date. It combined semantic and syntactic chains which seemed to help only when no negative clauses were added. Finally, it used chains of length one — this only helped during cross-validation, maybe due to the similarity in genic interaction pairs in training and test folds when sentences were drawn from the same abstract.

4.1.1. Syntactic vs. Semantic Chains

Results for the clause sets 1, 2 and 3 in Table 2 show that using syntactic chains alone (i.e., clause set 1) yielded a significantly higher F-Measure compared with using semantic chains or a combination of semantic and syntactic chains.

The main weakness of the semantic approach is its low recall. It can be attributed to two factors. First, incorrect CCG parses were responsible for missing or incorrect semantic chains within the DRSs. Second, ccg2sem generates multiple domain referents for indefinite objects of coordinated expressions.[4] For instance, in the DRS for the sentence

> Most cot genes, and the gerE gene, are transcribed by sigmaK RNA polymerase.

there are two discourse referents for *sigmaK RNA polymerase*, due to the fact that the part-of-speech tagger integrated into the CCG parser failed to label the term as a proper name. Although it might be possible to modify the chain extraction component to deal with this, it would be preferable to change the output of the tagger prior to syntactic analysis.

[4]Semantically, this is indeed the correct result.

4.1.2. Negative clauses

Clause sets 1 and 4 contain the same positive clauses, but clause set 1 benefits from the addition of negative clauses. As shown in Table 2, the performance of clause set 1 is significantly superior to that of clause set 4 in both recall and precision. Having a model of what it means to have no interaction can improve *precision* whenever positives clauses would "vote" for a genic interaction with low confidence, while negative clauses "vote" against a genic interaction with high confidence. Without negative clauses, this scenario would result in a false positive. Negative clauses can increase *recall* when they are highly confident that a gene pair is "not a non-interaction" while positive clauses are uncertain.

4.2. With Coreferences

As we mentioned briefly earlier, an attractive feature of ccg2sem is its built-in coreference resolver. Thus, for testing and training on data with coreferences[5] we expected better results with semantic chains compared to syntactic chains. However, even in this case syntactic chains outperform semantic chains, as shown in Table 3. This can be explained partly by the fact that many coreferences were appositions which the parser could extract. Furthermore, most sentences in the test and training set lacked coreferences, and this also contributed to syntactic chains performing better.

The submitted clause set was 5, since again this was the best performing set under cross-validation on the training data.

5. Conclusion

Perhaps the most striking observation is the dramatic effect of adding negative clauses to the rule base. It seems clear that clauses modeling non-interaction are essential for good performance, increasing both recall and precision as explained above.

[5]This corresponds to the "With and Without Coreferences" data set of the LLL Challenge.

When comparing syntactic and semantic chains, syntactic chains appear to be the clear winner. However, this conclusion has to be tempered by the fact that the syntactic chains were based on manually corrected parses, whereas the semantic chains were based on a completely automatic statistical parser (trained on the Penn Treebank). Moreover, there is also potential in exploiting semantic chains to incorporate domain knowledge in future experiments.

It also turned out that we were unable to apply ILP directly to the problem, due to the small size of the training set and to the varying length and structure of the chains we were looking for. Instead of carefully biasing ILP towards the clauses we had in mind, we decided to extract a set of chain-based clauses and learn probabilistic weights. However, the two approaches are in fact orthogonal — it would make perfect sense to first generate a set of candidate rules using an ILP system and then learn their weights using the Markov Logic approach.

Acknowledgments

Many thanks to Malvina Nissim and Claire Grover for their helpful input during the course of the project.

References

Bos, J. (2005). Towards wide-coverage semantic interpretation. *Proceedings of IWCS-6, Tilburg, The Netherlands*.

Clark, S., & Curran, J. R. (2004). Parsing the WSJ using CCG and log-linear models. *Proceedings of ACL* (pp. 103–110).

Domingos, P., & Richardson, M. (2004). Markov Logic: A unifying framework for statistical relational learning. *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields, Banff, Canada: IMLS.* (pp. 49–54).

Kamp, H., & Reyle, U. (1993). *From discourse to logic*. Studies in Linguistics and Philosophy. Kluwer Academic.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Math. Program.*, *45*, 503–528.

Riedel, S., & Meza-Ruiz, I. (2005). Markov The Beast - Markov Logic software platform **url**: `http://homepages.inf.ed.ac.uk/s0349492/thebeast`.

Steedman, M. (2001). *The syntactic process*. The MIT Press.