Using Linguistic Features to Improve the Generalization Capability of Neural Coreference Resolvers

A Supplemental Material

A.1 Running Example for FP-Tree Construction

Assume D contains the following three samples:

X_1={ana-type=NAM, ant-type=NAM, head-match=F}, $C(X_1) = 0$

X_2={ana-type=NAM, ant-type=NAM, head-match=T}, $C(X_2) = 1$

X_3={ana-type=NAM, ant-type=NOM, head-match=F}, $C(X_3) = 0$

Based on these three samples

- A={ana-type=NAM, ant-type=NAM, head-match=F, head-match=T, ant-type=NOM},
- $support(a_i, 0)_{a_i \in A}$= {2,1,2,0,1}, e.g. "ana-type=NAM" appeared two times in non-coreferring ($C(X_i) = 0$) samples,
- and $support(a_i, 1)_{a_i \in A}$={1,1,0,1,0}.

If we sort A based on a_i's frequencies, i.e. $support(a_i, 0) + support(a_i, 1)$, the ordering of A's items will remain the same.

Now, we need to go through the samples again to build the tree. The FP-Tree construction steps after adding each of the above samples is demonstrated in Figure 1. ana-type=NAM, ant-type=NAM, head-match=F, head-match=T, and ant-type=NOM are abbreviated as ana=NAM, ant=NAM, head=F, head=T, and ant=NOM, respectively in Figures 1 and 2. Figure 2 shows the resulted FP-Tree in which corresponding support values for both classes, i.e. zero and one, are also included in each node.

Figure 3 and Figure 4 show the conditional FP-Trees that are built based on the FP-Tree of Figure 2 and for patterns p = {head=F} and p = {head=F, ant=NAM}, respectively.

A.2 Discriminative Pattern Mining vs. Feature Selection

In this paper, we used a discriminative pattern mining approach for determining feature-values that are informative for the coreference label when they are considered in combination.

An alternative approach would be to use a standard feature selection algorithm where each feature-value is considered as a feature. There are three feature selection models: *filter*, *wrapper* and *embedded*.

Wrapper models use a learning algorithm, i.e. coreference resolver in our scenario, in the loop, and therefore assess the performance of different feature subsets based on the performance of the learning algorithm on an evaluation set. Wrappers are, however, computationally expensive in our scenario since they require the coreference resolver to be executed in every iteration of the feature-value subset selection. For n feature-values, there exist 2^n possible combinations. n is around 500 in our data and deep-coref takes two days for training using GPUs.

Filter models, on the other hand, are solely data dependent and therefore are independent of the learning algorithm. The use of a discriminative pattern mining approach for informative feature-value selection, is equivalent to a filter model.

Finally, embedded models are incorporated into the learning algorithm itself. For instance, we can incorporate all possible feature-values in deep-coref and use various regularization methods, e.g. dropouts, l_1 and l_2 regularizations, instead of pre-selecting informative feature-values. We examined the above regularization methods in "top-pairs+linguistic" experiments. The use of each of the above regularizations on top of the feature layer in deep-coref results in significantly lower performance than either of "top-pairs" and "top-

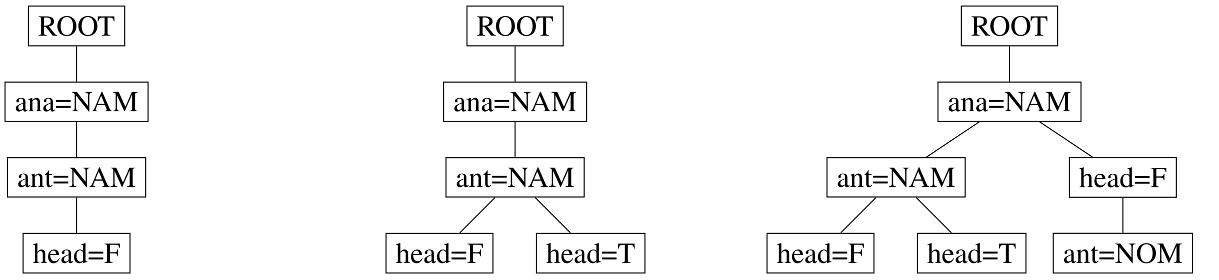


Figure 1: Left to right: (partial) constructed FP-Tree after adding each of the three given samples. The right-most tree is the final FP-Tree that represents all input samples.

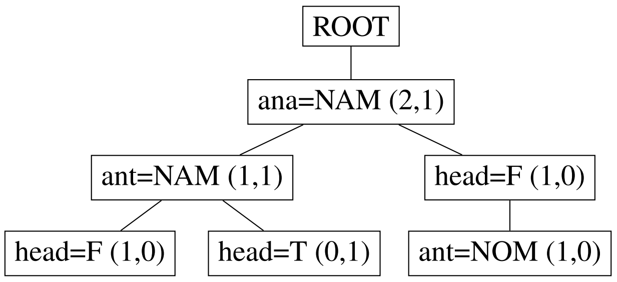


Figure 2: FP-Tree with corresponding support values of the nodes.

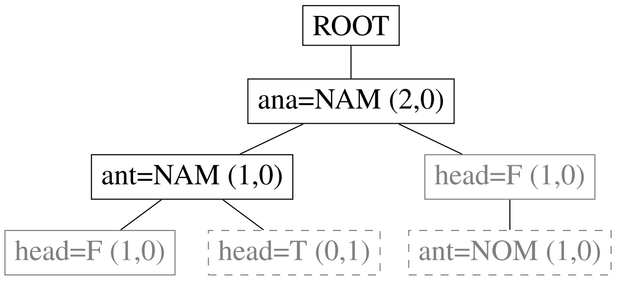


Figure 3: Conditional FP-Tree for the $p = \{\text{head=F}\}$ pattern.

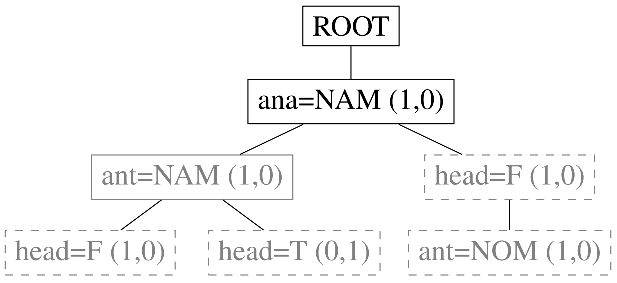
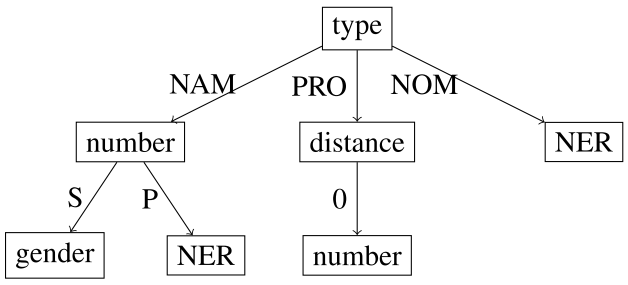


Figure 4: Conditional FP-Tree for the $p = \{\text{head=F, ant=NAM}\}$ pattern.

pairs+linguistic" results on the CoNLL development set. It is worth mentioning that we did not perform hyperparameter optimization for these experiments. We examine 0.2, 0.3 and 0.5 values for dropout and 0.01 as the regularization parameter. If we want to tune these parameters, the question would be the choice of the evaluation set since our main focus is to improve generalization. We leave this direction for future work.

Overall, it is worth noting that EPM uses an exhaustive search to explore all frequent combination of feature-values up to a certain length, unlike many existing feature selection algorithms that use heuristic algorithms for searching feature subsets.



type+number

type+number+gender

type+number+NER

type+distance

type+distance+number

type+NER

Figure 5: A sample decision tree and the list of all extracted feature conjunctions based on Fernandes et al.'s (2012) approach.

A.3 Example of Fernandes et al.'s (2012) Feature Templates

Figure 5 shows a sample decision tree the list of corresponding feature templates that are extracted from it.