Coreferences in REL

Flavio Hafner

January 13, 2023

Contents

1 Introduction

REL is a software for entity linking. Entity linking consists of three steps: mention detectio, candidate selection, and entity linking. Before entity linking, the current version of REL looks for coreferences in the detected mentions. Consider the example text: "This sentence is about Jimi Hendrix. The music of Hendrix is popular." Here, the second mention "Hendrix" refers to the entity "Jimi Hendrix" in the preceding sentence. Now, for each mention m_i, REL checks whether there is another mention m_j that contains mention m_i as a separate word. If this is the case, REL replaces the original candidate entities of m_i with the candidate entities of m_j. Moreover, if there are multiple mentions m_j that contain the same candidate entities, their $p(e|m)$ scores are aggregated for the mention m_i. Coreference search can make entity disambiguation more accurate if the original candidates for mention m_i are more noisy than the selected candidates for

m_j. But this coreference search also slows down REL for large data sets because it has quadratic time complexity.

I extend REL so that the user can choose from three options on how to deal with coreferences. The first option is "all" which corresponds to the current default of comparing all mentions with all other mentions. The second option is "off" that does not look for coreferences at all. It should be the most efficient. The third option is "lsh" that first applies locality-sensitive hashing (LSH) to all mentions, and then only searches for coreferences among mentions that are clustered together by LSH. If LSH is efficiently implemented, it could strike a balance between efficiency and effectiveness for datasets with many mentions.

To assess the performance (efficiency and effectiveness) of the options, I run REL with the three options on three data sets: on the AIDA data to check effectiveness, and on synthetic AIDA data—stacking the mentions from AIDA multiple times—to check the time complexity with larger datasets. The purpose of the last run with the msmarco data is to check time complexity on real-world data. I try to check the effectiveness on these data by checking the fraction of mentions that are different between the two approaches.

In what follows, I define coreferring mentions as mentions like the mention “Hendrix” above, and non-coreferring mentions as all other mentions.

The calculations are run on a Dell XPS 13 laptop, using Flair and Wikipedia 2019 for mention detection. I set the limit number of documents from AIDA to 500, which processes all documents. The settings for LSH are such that they have high precision and recall on the coreferring mentions in the AIDA ground truth data.

2 Results

2.1 Performance on AIDA data

Table 1 shows precision, recall and F1 scores on the AIDA data sets (all documents) with the three options. The performance is a little lower for the “lsh” option than for the others. Regarding timing for ED, “off” is the fastest and “lsh” is the slowest option. To compare to Erik’s results (table 1 in his document), I ran REL also only with 50 documents. The effectiveness results are very similar to his results. The efficiency results, however, differ—in Erik’s case, ED takes more than twice as long than in my case. I am not sure why this is.

Because only a minority of the mentions in the AIDA data are coreferences, I check

coref option	Mention detection				Entity linking		
	Precision	Recall	F1	Time	Precision	Recall	F1
“all”	97.9%	62.9%	76.6%	2.97 sec.	62.1%	39.9%	48.6%
“lsh”	97.9%	62.9%	76.6%	4.82 sec.	62.0%	39.8%	48.5%
“off”	97.9%	62.9%	76.6%	2.43 sec.	62.1%	39.9%	48.6%

Table 1: Performance on AIDA data—all mentions. Setup: processing all documents in AIDA data set with wikipedia 2019.

coref option	Mention detection				Entity linking		
	Precision	Recall	F1	Time	Precision	Recall	F1
“all”	90.7%	36.7%	52.3%	—	78.8%	31.9%	45.4%
“lsh”	90.7%	36.7%	52.3%	—	76.2%	31.0%	44.1%
“off”	90.7%	36.7%	52.3%	—	76.8%	31.3%	44.4%

Table 2: Performance on AIDA data—coreferring mentions. Setup: processing all documents in AIDA data set with wikipedia 2019. The timing is not reported because the results are taken from the same run as the results in table 1.

the performance only coreferring mentions. To do this, I classify the gold mentions by whether they are coreferences or not using the same method as in REL, and then compare precision, recall and F1 scores only on these mentions. The results are in table 2.

First, we can compare the results from this restricted sample with the results from the full sample of mentions. We see that recall for mention detection for coreferring mentions is much worse than for non-coreferring mentions—only 40 percent of coreferring mentions are detected, while 60 percent of all gold mentions are detected. This could be (1) because coreferences are harder to detect, or (2) because some detected mentions are not labelled as coreferences because the mention they are referring to is not detected. I think case (1) is more likely. The tables also indicate that in the ED step, REL has a higher precision but a lower recall for coreferring mentions than non-coreferring mentions.

Second, comparing the effectiveness of ED for the three options, we see that “lsh” performs worse than either of the alternatives.

2.2 Time complexity

To assess time complexity of the different options, I create a synthetic AIDA data set: For each document, I repeat the detected mentions multiple times to increase the size of the input, and then compare the running time for entity linking as a function of the size of the input. Figure 1 shows the results. Comparing option "off" with option "all", we see that the quadratic time complexity of "all" only starts to impact the overall running time when the number of mentions is higher than around 1500 mentions. This is because the `__predict` function in REL has a relatively large overhead that dominates the running time of coreference search when the number of mentions is small (further results are available if necessary). Comparing option “all” with option “lsh”, we see that LSH currently has worse time complexity than the option “all”. This is because LSH hashing is implemented inefficiently.

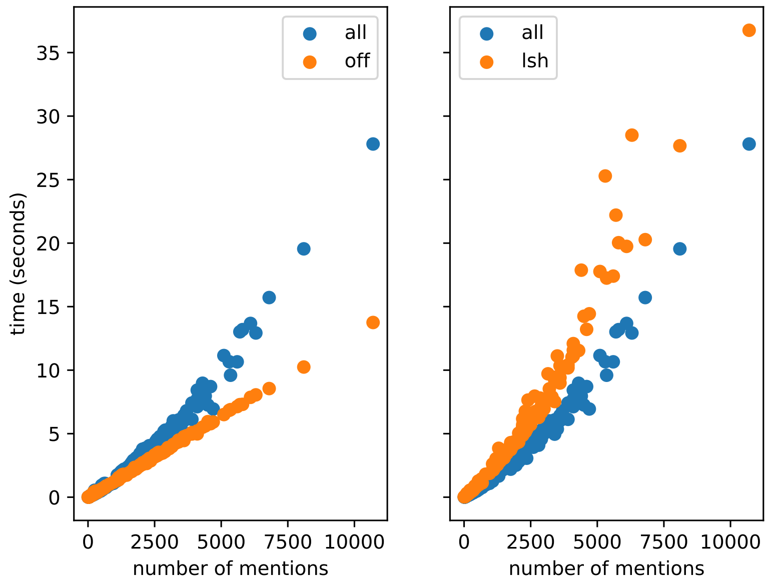

Figure 1: Time complexity on synthetic AIDA data, with different coref options

I also profile the execution time of each function in entity disambiguation for the synthetic data. Figure 2 shows the the time taken by the function `with_coref`, which is the main function for coreference resolution. As before, the inefficient implementation of LSH slows down `with_coref`. But for the one trial with more than 10000 mentions, the time appears to be better for “lsh” than for “all”. The current implementation of “lsh” creates large time overhead to deal with very many mentions. It is possible that for input with more than 10000 mentions, the quadratic time taken by “all” starts to dominate that overhead in “lsh”. But further tests with larger datasets would be necesary to confirm this possibility.

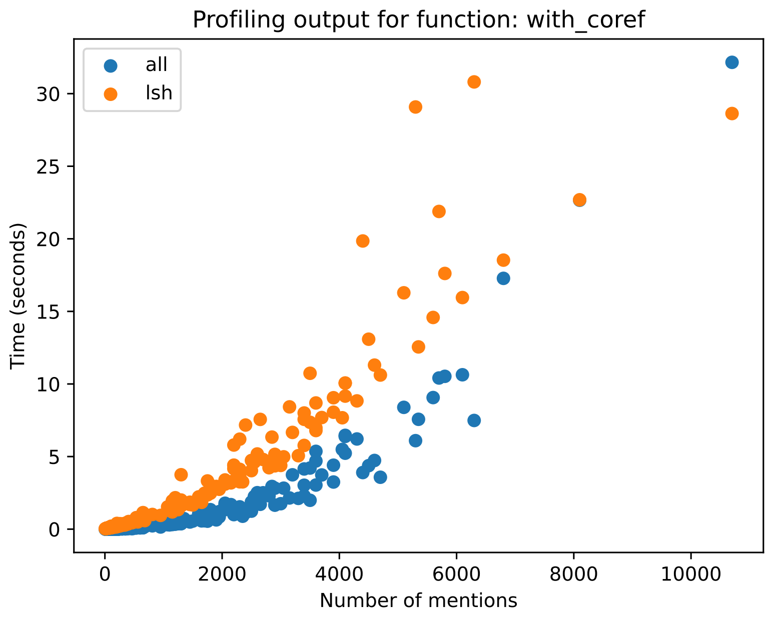

Figure 2: Profiling the function *with_coref* with synthetic AIDA data.

2.3 Performance on msmarco data

As there is no ground truth for the msmarco data set, I check the overlap of the predicted entities for the same mention when using either the option "off" or "on". I exclude the option "lsh"—it does not run out of memory anymore, but the solution cannot compete on running time with the other options.

I start by comparing the fraction of mentions that have the same predicted entity by aggregating across all documents. Having labelled each mention as a coreferring mention or not, I do this separately by coreferring mentions and non-coreferring mentions. Table 3 shows the results: nearly all mentions that are not coreferring have the same linked entity, but only 40 percent of coreferring mentions have the same linked entity. The lack of a ground truth prevents from drawing conclusions about which of the option is performing better. But the low overlap for coreferring mentions suggests that the choice of coref option can matter for documents with many coreferring mentions.

Now I compare the overlap of predicted entities by document. Figure 3 plots the fraction of mentions with the same predicted entity separately for all mentions, coreferring mentions, and non-coreferring mentions (panel b). The x-axis is the total number of linked mentions in a document. To abstract from cases with very few coreferring

coreferring mention	count	mean	std
no	247754	0.99	0.056
yes	8767	0.402	0.490

Table 3: Fraction of mentions with the same linked entity

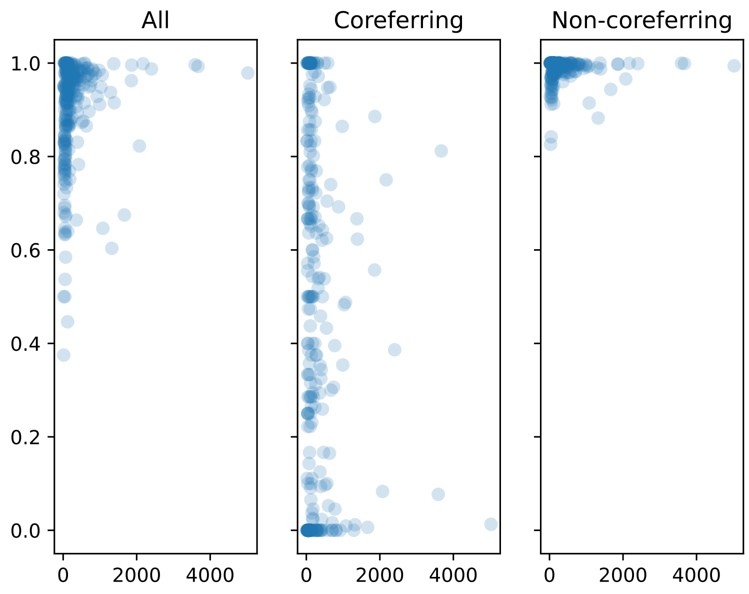

Figure 3: Overlap in predicted entities, by mention type. One dot refers to the average overlap of one mention type in one document.

mentions, I focus on documents with at least five such mentions.

The figure shows considerable variation in the overlap of the predicted entities overall, stemming from variation in the overlap for the coreferring mentions. The data do not allow to draw conclusions about whether there is a relation between the number of linked mentions in the document and the overlap in the predicted entities for coreferring mentions.

Lastly, figure 4 show the timing of entity disambiguation with the two options; it confirms the findings on the time complexity with the synthetic AIDA data. There is one outlier for for the option “off” that I do not know where it comes from.

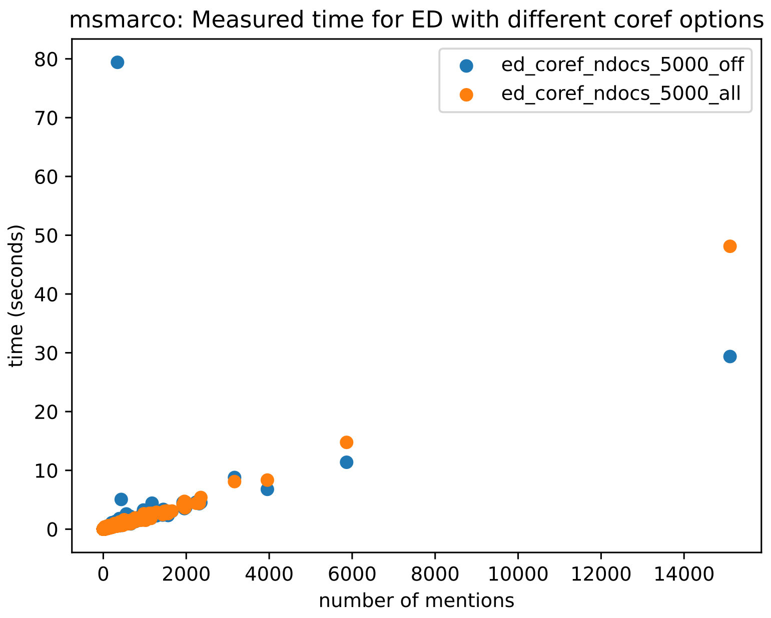

Figure 4: Time complexity on msmarco data, with different coref options

3 Conclusion

3.1 Summary

1. For small input data, the coreference search is inconsequential for running time. Effectiveness is highest when using option “all”.

2. For large input data, efficiency is a problem when doing a full coreference search. Particularly for documents with many coreferring mentions the choice of the option could matter.

3. For large input data, LSH could be a solution, but precisely in this case a time- and memory-efficient solution is key. Implementing a time-efficient LSH from scratch could take some time. There are external alternatives such as faiss (link).

4. An important reason for low recall for coreferring mentions is the mention detection, not the disambiguation, but these mentions are a small part of the total number of mentions in the AIDA data. And some of this may be resolved with the update for MD from Erik (which I have not yet integrated in my version of REL).

3.2 Implications

1. Including the coref options “off” and “all” in REL should not take too much time. Perhaps one can turn it to “off” as a default when the number of mentions is sufficiently large, with a respective warning/notification to the user.

2. LSH may be useful for a use case with large documents with possibly many coreferring mentions (but I do not know what kind of documents these could be). The options are either an external dependency, or I can consult with other people at the eScience Center to find out options to write an efficient solution from scratch.