COMPUTATIONAL LINGUISTICS-II
COURSE PROJECT
November, 2020

Coreference Resolution Model for Hindi and Malayalam

Mentor: Vandan Mujadia

Akshett Rai Jindal
2019114001
akshett.jindal@research.iiit.ac.in

Jerrin John Thomas
2019114012
jerrin.thomas@research.iiit.ac.in

Jayant Panwar
2019114013
jayant.panwar@research.iiit.ac.in

Aim

The aim of this project, as the title hints, is to develop a model to resolve coreference in Hindi and Malayalam domains. The intent is to use efficient algorithms as part of the model to detect and resolve coreference in both, written and spoken domains.

Background

Coreference resolution is a very famous yet complex area in the field of computational linguistics. There have been some great works regarding coreference resolution in English and other foreign languages, which is all the more motivation for us to carry out a small yet decent contribution on our behalf for two Indian languages. The whole process helps the team to apply basic understanding of language analysis, based on linguistic theories and to appreciate the challenges natural language processing possesses.
A better understanding of coreference and computational algorithms is also developed along the way. Such algorithmic models help in various fields and end up forming an essential part of computational

linguistic work many a times. Therefore, building a coreference resolution model for Hindi and Malayalam was the best way to get closer to the topic of Coreference resolution and at the same time building upon our knowledge of basic linguistic theories combined with computational algorithms.

Related Inspirational Works

Like any other good work in computational linguistics, it is important to get the theoretical base of our model correct before going ahead with its practical implementation. We have embraced the same approach. First and foremost we read some revolutionizing works in the field of Coreference Resolution in English like: Raghunathan et al. (2010), Heeyoung et al. (2011), Clark & Manning (2016), and Lappin & Leass (1994). The references for the same are provided at the end. Reading these works inspired us to come up with our own algorithmic models for coreference resolution in both Hindi and Malayalam.

Approach

Even though our model finds its inspiration in the previously listed works, our model is solely focused on the languages we are dealing with, i.e., Hindi and Malayalam. Since the languages are different, our model also differs a bit between this languages. Nevertheless, the idea remains the same. We term our approach as 'Step-down ladder approach' in which each step is treated as a new way to detect coreference in the domain and bring us closer to a known 'gold-base' where the model is able to find all the correct pairs of anaphoras and antecedents. Though the idea is to get closer and closer to the 'gold-base', it must be noted that our model is still far from reaching that base every time a piece of text is passed through it. As always, future improvements can be made on our model according to the needs, as we discuss them in the end.

Akshett and Jayant have handled the coreference model for Hindi while Jerrin has worked to establish a similar model for Malayalam coreferencing resolution.

Hindi model

Corpus

For our Hindi Corpus, we have selected 10 online news articles in Hindi. We have developed and tested our algorithmic model on the basis of these news articles. The articles serve as both written and spoken domain as some of them also contain interviews.

Annotation for Gold-Standard

The annotation scheme implemented for getting a gold standard is fairly simple. All the possible mentions are enclosed within [] brackets, followed by some Roman script acronym or abbreviation for the same. Sometimes the acronym/abbreviation may already be in use, then we opt to represent the major syllables of the word. All the mentions that link to the same entity are given the same abbreviation or acronym across the entire extent of article. For example, let's take a look at this annotation of Hindi text in roman script (the project of course has been developed to deal with Hindi in Devanagari script):

[kashmir]k jana hai [mujhe]m! [vaha]k bahut thand hoti hai!
Translation: [I]m want to go to [Kashmir]k! It is very cold [there]k.

Here, 'k' stands for kashmir and 'm' stands for the speaker. As you can see, other mentions of kashmir are also labelled with 'k'. Sometimes there might be child mentions within a mention as well:

[[kashmir]k ki janta]kkj bahut naraaz hai!
Translation: [People of [Kashmir]k]kkj are very angry.

Over here, 'k' is referring to kashmir but is also a part of the mention kkj, i.e., kashmir ki janta.
Same abbreviations can occur within different articles but only once for each article. Thus, different articles may share a single abbreviation but each article can only use it to represent a single entity.

The Step-down ladder model

As stated previously, the functioning of the model should resemble a step-down ladder. With each step, the system/model tries to come closer to a base, i.e., gold-standard in our case. The model tries its best to come close to that base but is still far off considering the time period within which this model was developed and improved upon.

We have tried to form seven such steps:
- Mention detection
- Clustering the mentions
- Exact String Matches
- Relaxed String Matches
- Proper Head Word Match
- Relaxed Head Word Match
- Pronoun match

To implement this model for Hindi, we have used the ISC parser, which in turn makes use of ISC POS tagger and ISC tokenizer. The reference for the same is provided at the end.

Mention Detection Step

The most important starting step in our model has to be detecting the possible mentions. Mentions represent real world entities in the sentences and therefore they are mostly noun phrases (NP) which are headed by a nominal or pronominal head. It is very much probable that various mentions refer to the same entity. For example,:

ex: [mai] [rohan] ke saath [cricket] khelta hu!
Translation: [I] play [cricket] with [Rohan].

Here, the mentions are the speaker himself, rohan, and cricket. This is how our model also finds mentions. It looks for words that are part of NPs and selects them. To take multiple word mention, the head word of the NP is also considered. We store two types of mention lists, one that has all the mentions, even repeating ones and the other one that just has a unique instance of mention, no duplicates. However, we do not include pronouns in our mentions list and we store them separately because we have come up with a different algorithm for pronoun resolution. Also, having pronoun in our mentions list with other mentions will cause problem and unwanted results in the Clustering and other string match steps.

Clustering Step

As we earlier stated, there is very much the possibility that a group of mentions might refer to the same real world entity. If that was not the case, the world would not need any coreferencing model. Therefore, to address the elephant in this room, we will group mentions that refer to the same entity and call it a single cluster. For example:

ex: kal [mai]m [rohan]r ke saath bahar gaya tha. [vo]r boht khush tha kyunki [usne]r daud jeeti thi.
Translation: Yesterday [I]m went out with [Rohan]r. [He]r was very happy as [he]r had won the race.

Here, the cluster will be {rohan, vo, usne} as they are referring to the single real world entity, i.e., rohan(r). For clustering, we are only writing the unique instance of the mention. Therefore, it is understood that even though we have not written the duplicates of mentions explicitly, they happen to be part of the same cluster. Due to earlier stated reasons, pronouns are also not part of clustering step.

Exact String Match

This part of our step-down ladder model is fairly simple and does not require any complex approach for it come to fruition. The mentions are cross-checked with each other to see if the exact same instance of any mention occurs at other places. The model basically makes use of the all mentions list here, i.e., the list of mentions which also has duplicate mentions in it. All in all, this step basically matches up the duplicate mentions and eliminates the possibility of having to check the duplicate mentions for coreference later again in the model.

Relaxed String Match

It is possible that the previous step may miss out on some important cases of coreference. For example,:

ex: [rohan sharma] boht tej bhagta hai. [rohan] padhaku bhi hai.
Translation: [Rohan Sharma] runs very fast. [Rohan] is studious as well.
Here both the marked phrases are referring to the same real world

entity: Rohan Sharma. But our previous ladder step will fail to identify it as a match because the content text is not 100% same. That is why we have introduced the concept of relaxed string matching. If either of the mentions happen to be the substring of each other, then it is implied that they are relaxed matches referring to the same entity.

Proper Head match
This step of the step-down ladder model is crucial as well. Sometimes, the mentions may not be substring of each other but may refer to the same proper noun or real world entity. Therefore, a way to overcome this is to check whether the head word of the mention is a proper noun. If it is, then we check if there exists another mention with the same proper noun. If there is, then both the mentions are said to be referring to the same entity.

Relaxed Head match
There might exist some mentions that do not have proper nouns as their head but do have the same head of their noun phrases. For example,

ex: [dilli ke high court] ne faisla de diya hai. [dilli court] ka manna hai ki ve doshi hain.
Translation: [Delhi's High court] has given its judgement. [Delhi court] has found them guilty.

Here, the phrases have the same head word: court and they don't have proper noun head word and neither are proper substrings of each other. Thus, to solve this scenario we bring in the step of relaxed head word matching, wherein if two mentions have the same head word, non-proper though, they are considered to be referring to the same real world entity.

Pronoun Matching
As stated earlier, we have decided to treat pronouns differently from the other mentions in the corpus. This is because they are either immune to the above listed steps or may end up giving unwanted coreference chain links when used in those steps. To treat the pronouns matching case with utmost care, we have decided to form a scoring algorithm, insipired from Lappin & Leass (1994). The

k1	100
k2	80
k3	60
k4	50
k5	50
k7	50
Sentence Recency	Deduct 10% from score with every passing sentence

TABLE 1: *Scoring weightage for pronoun resolution*

karaka relations will mostly affect the score. The respective karaka relation and its different forms (ex: k7, k7p, k7t) will also be awarded the same score as the base karaka relation. k1 has been given the most prominence because generally mentions that are the subject of the sentence have a higher probability to be referred with a pronoun later on. Then, there is a gradual decrease in scoring till k4, after which they have the same score.
The other major factor in pronoun resolution is the sentence recency. The mentions are stored with the sentence number in which they were encountered and their karaka score. Then if any pronoun is encountered in a sentence, the sentence distance is calculated, which is basically the difference between the current sentence number and the encountered sentence number of the mention in consideration. Then the difference is taken as the power of 0.9 and a reducing factor is calculated. This reducing factor is subtracted from the mention score to give the apt mention score. Then, the mentions are sorted and the mention with the highest apt score is selected. If multiple mentions have the same score, then all of them are provided as the possible candidates.

Observation and Challenges

The foremost challenge that we observed or encountered was the scarcity of good parsers and chunkers for Hindi except the LTRC Hindi shallow parser. The problem with incorporating LTRC parser with our project was that we needed to be on IIIT VPN in order to incorporate the shallow parser with our code. We tried installing the parser on our systems, but the installation failed (We had informed the mentor about this). This meant that our virtual meetings would end automatically due to network fluctuations. A good amount of time was spent in order to combat this problem. Finally, we decided to go with the ISC parser available online and the link for same is given in the references. This parser is not very accurate as such, as we noticed in some of the incorrect parsing cases, but it certainly was better than nothing. Also, this parse did not form the chunks. This made writing the model even more difficult for us. Due to this the model's accuracy decreased significantly at the mention detection level itself. Moreover, this parser was written in Python mainly which is the same language we have used to implement our model. This gave us all the more reason to go ahead with this parser.

Another important challenge that we faced was regarding listing mentions. We could not have all the mentions in a single list, as that would be problematic for steps like relaxed string matching that do not need duplicate mentions, otherwise it will start functioning as the exact string matching step. Also, calculating sentence recency factor was a bit problematic as the step runs within the mention list and we cannot determine the best candidate for the mention of pronoun as listing pronouns is a different feature altogether. Therefore, we came up with the solution of having two different mention lists. One that only had unique mentions and the other that had all the mentions, including duplicates. With the mentions we also stored them with the sentence number in which they occurred, so the sentence recency factor can be calculated with ease when dealing with pronouns later on in the model. We also stored the score the mention had according to the karaka role it played in the sentence.

Performance and Evaluations of the Hindi model

We evaluate the following parameters for the model:

1. The Precision basically is the percentage of correct markings in the output of the data.

$$\textbf{Precision (P)} = \frac{\text{Sum of all prediction scores}}{\text{Number of attempts of the system}}$$

2. The Recall is the percentage of the data in the gold-data which is correctly identified by the system.

$$\textbf{Recall (R)} = \frac{\text{Sum of all prediction scores}}{\text{Number of actual annotations in gold-standard}}$$

3. The F-measure is the combined parameter derived from P and R by giving them both different or same weights. We are calculating F1, which gives them both same weight because we want both the factors to be high.

$$\textbf{F-Measure (F)} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

We give the following numbering to the steps for representing them in the table.

Number	Step
1	Mention detection
2	Exact String Matches
3	Relaxed String Matches
4	Proper Head Word Match
5	Relaxed Head Word Match
6	Pronoun match

TABLE 2: *Numbering for steps*

Since the clustering step of the algorithm does not contribute to the results directly, we have not evaluated it in the way we have evaluated the other steps. The clustering step acts more like a base for our other steps to run and the metrics to validate clustering are not so clear either. Thus, we will skip clustering step and evaluate other steps of our algorithm.

Step	P	R	F
1	50.21	66.94	57.37
2	64.25	64.45	64.34
3	64.66	55.66	59.82
4	70.61	50.04	58.57
5	69.60	49.75	58.02
6	69.52	49.82	58.044

TABLE 3: *Various Parameters for Hindi Model*

The F1 score for all of our steps is well above the 50 mark. That means our system was able to detect good amount of coreferences correctly for a starter system that was built in a limited time period.

Our Mention detection and exact string match had the best recalls while proper and relaxed head matching as well as our pronoun matching algorithm gave us the best precision.

Malayalam Model

For the Malayalam corpus, the Malayalam treebank data has been chosen. This is because no suitable parsers were found and annotated data was required. 10 lines of 4 documents have been annotated with coreference information. The model and the annotation scheme is exactly similar to the Hindi counterparts described above.

Performance and Evaluations of the Malayalam model

The evaluation method is similar to that followed for the Hindi model.

Step	**P**	**R**	**F**
1	23.36	40	29.49
2	50	7.69	13.33
3	4.76	57.14	8.79
4	-	-	-
5	7.14	53.84	12.61
6	7.69	6.67	7.14

TABLE 4: *Various Parameters for Malayalam Model*

Conclusion & Future Improvements

In this project work, we demonstrated how a starter yet competitive coreference resolution model can be built using step by step approach. These steps of the ladder approach certainly help us to identify specific lexical, semantic, and syntactic information in the concerned discourse. As stated earlier this approach does bring us closer to a known 'gold-base' with every step of the model, however the final base is still pretty far off from the 'gold-base'. Nevertheless, we are grateful to our professor, mentor, and the related works for guiding us and giving us important knowledge about the coreference resolution topic in the field of computational linguistics.

As with any other work, there is always scope for improvement. For starters, we should try to refine our clustering information after every step. For example, currently relaxed string matching and relaxed head word matching give a lot of same results. We can try to refine them in such a way that the clustering information is modified once the antecedent for a mention has been found. This will also help to increase the modularity in our model, meaning we would be able to add new steps in certain places in future which will impact the overall performance in a positive way. Secondly, we can try to implement other steps like Acronym/Abbreviation matching step and Precise Construct step. These will also have a good impact on the performance of our model. We should also try and consider other karaka roles in our Hindi model apart from just k1,k2,k3,k4,k5 and k7. Roles like r6 should also be considered which relate to the genetive/possessive case and can help make our pronoun resolution algorithm more productive. In our Hindi model, we should also try to give a single mention as an antecedent of a pronoun, instead of listing all the mentions with the same score. Lastly, we should try and read more important works in the field of coreference resolution which will in turn inspire us build a more robust and better model. These improvements will certainly bring us a lot closer to the 'gold-standard base' our model aspires to reach.

References

Clark, Kevin & Manning, Christopher D. 2016. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 643–653. Berlin, Germany: Association for Computational Linguistics. https://doi.org/10.18653/v1/P16-1061. https://www.aclweb.org/anthology/P16-1061.

Corpus and Annotated Corpus. N.d. *For the Hindi Model.* https://github.com/akshettrj-iiith/Coreference-resolution-M2020/tree/master/Hindi.

ISC NLP. N.d. *ISC parser, ISC POS tagger, ISC tokenizer.* https://rb.gy/duyjjw.

Lappin, Shalom & Leass, Herbert J. 1994. An Algorithm for Pronominal Anaphora Resolution. *Computational Linguistics* 20(4). 535–561. `https://www.aclweb.org/anthology/J94-4002`.

Lee, Heeyoung et al. 2011. Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, 28–34. Portland, Oregon, USA: Association for Computational Linguistics. `https://www.aclweb.org/anthology/W11-1902`.

Raghunathan, Karthik et al. 2010. A Multi-Pass Sieve for Coreference Resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 492–501. Cambridge, MA: Association for Computational Linguistics. `https://www.aclweb.org/anthology/D10-1048`.