CSE538 Natural Language Processing Fall 18

Prof. Niranjan Balasubramanian
Stony Brook University

Mid-Term 2
1st December, 2018

Name - Rushikesh Nalla
ID - 111971011

1 Dependency Parsing:

1.1 Five Headlines that fail to parse correctly with explanation:

- **Sentence -** Labour union strikes factories remain idle: In this sentence strikes is parsed as a verb and we get an incorrect meaning for the sentence. This is noun-verb ambiguity.
- **Modified Sentence -** factories remain idle as labour union go on strike: Strike should actually be parsed as a noun to get the correct meaning of the sentence. Refer 1 and 2.

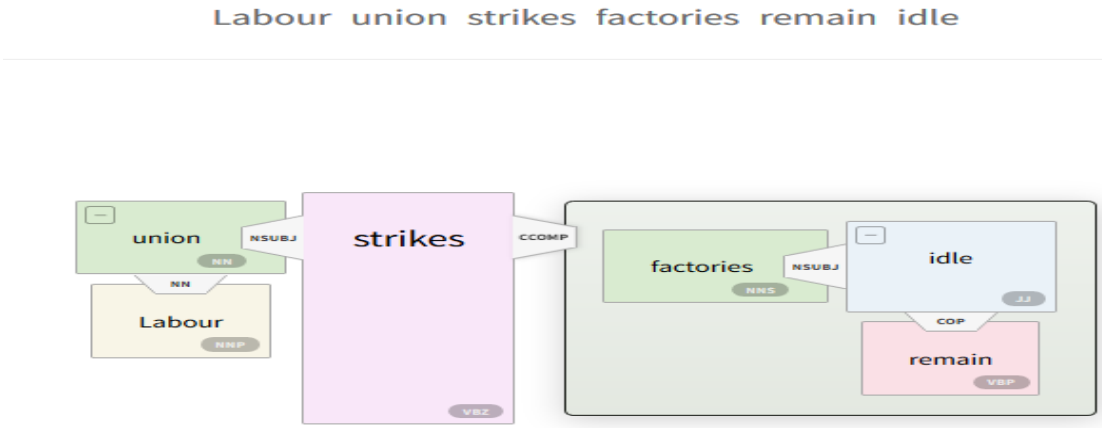

Figure 1: Sentence 1 Incorrect Parsing

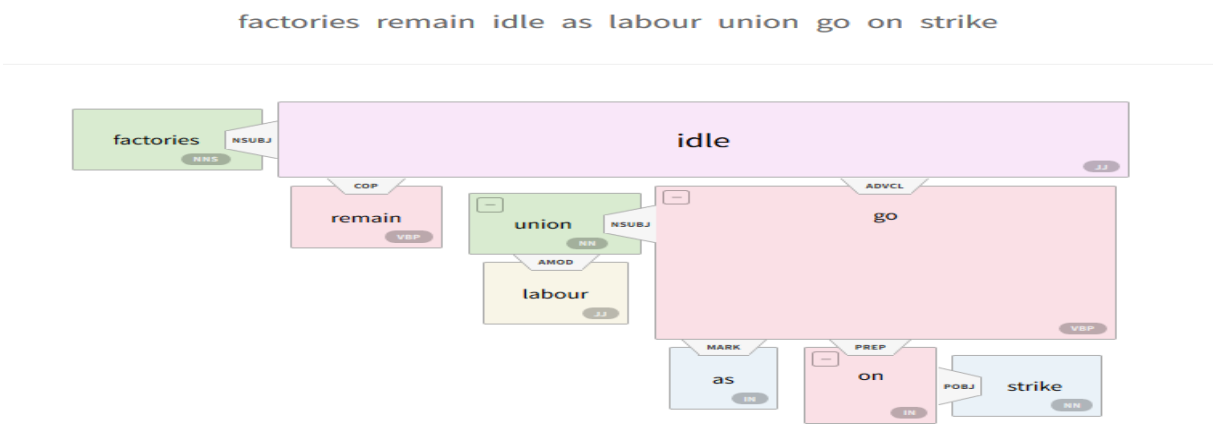

Figure 2: Sentence 1 Correct Parsing

- **Sentence -** senator banks on republican funds: Here the parser treats banks as a noun plural and gives incorrect meaning. This is a noun-verb ambiguity.
- **Modified Sentence -** senator relies on republican funds: We can modify the sentence and use a verb that means the same. Refer 3 and 4.

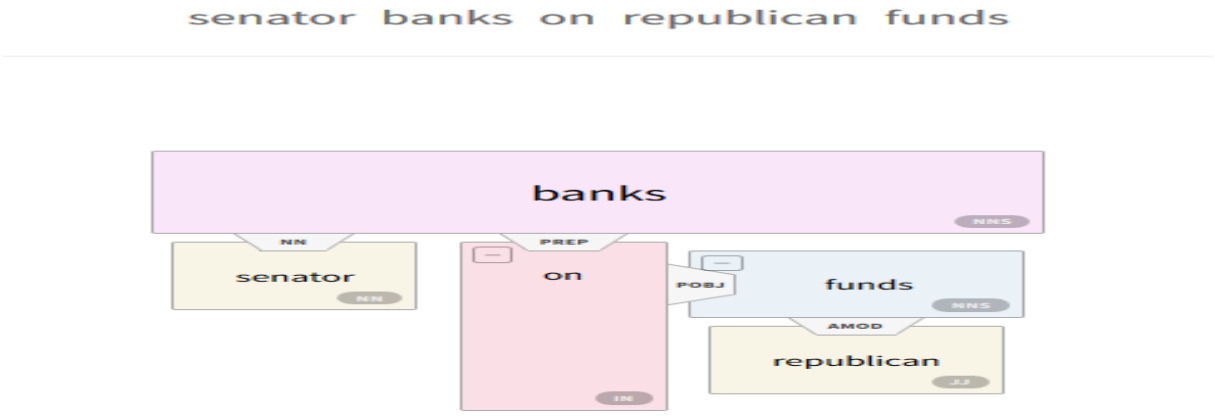

Figure 3: Sentence 2 Incorrect Parsing

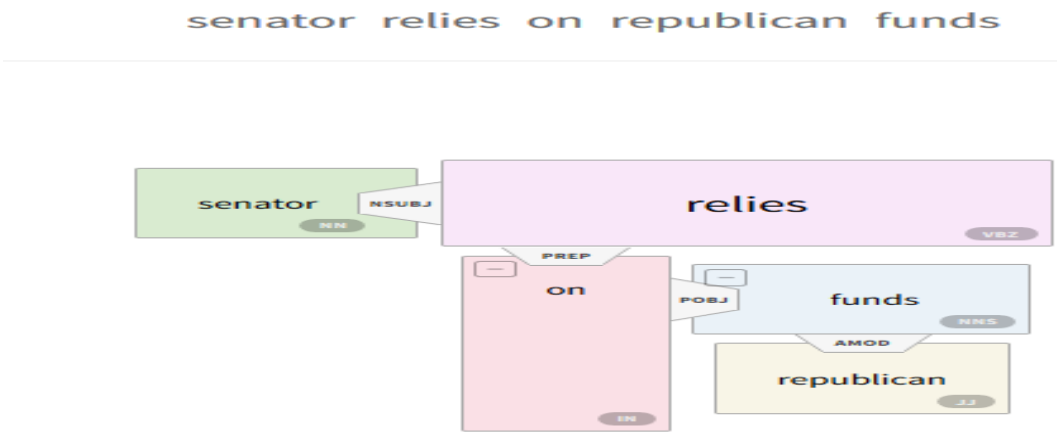

Figure 4: Sentence 2 Correct Parsing

- **Sentence -** new era blooms in united states: Here blooms is parsed as a noun as we get incorrect meaning. This is again a verb-noun ambiguity.
- **Modified Sentence -** new era begins in united states: We can modify the sentence by using begin which means the same. Refer 5 and 6.

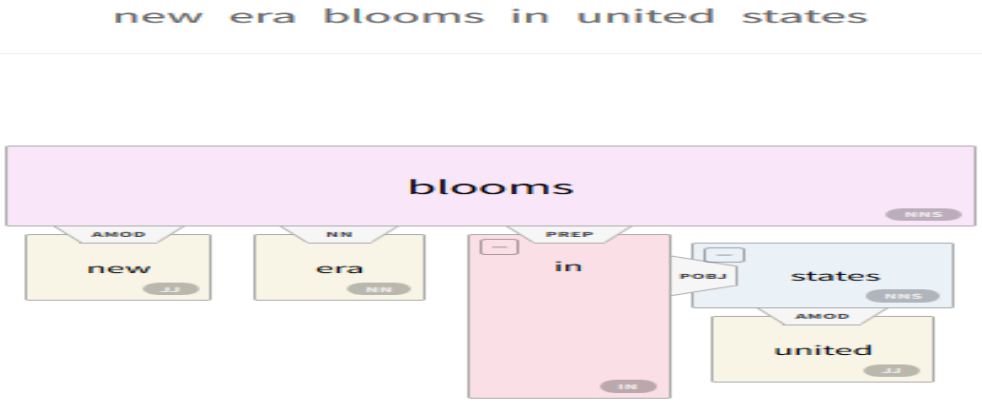

Figure 5: Sentence 3 Incorrect Parsing

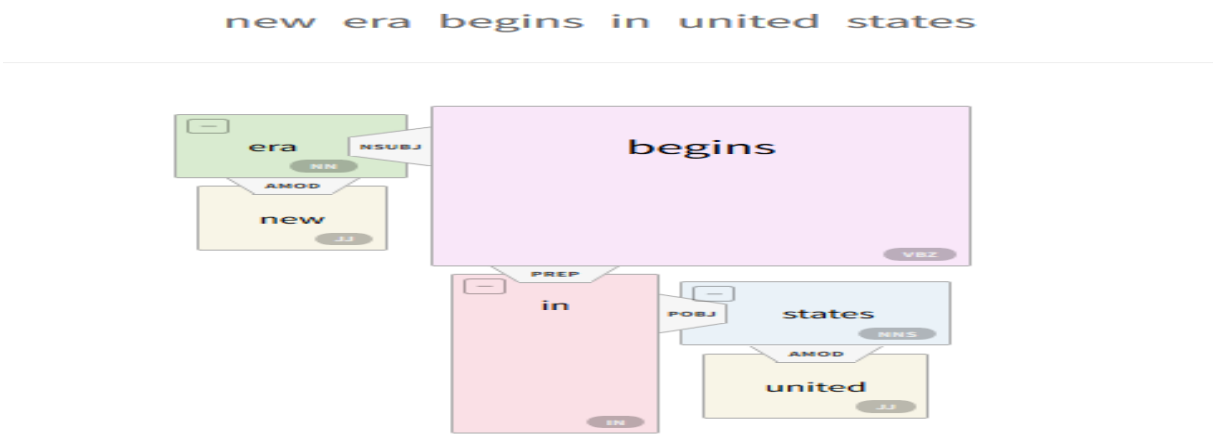

Figure 6: Sentence 3 Correct Parsing

- **Sentence -** president was robbed by thief in white house: Here the sentence is parsed to mean that thief was in white house already and he robbed the president which is incorrect.
- **Modified Sentence -** president was robbed in white house by a thief. The actual meaning of the sentence is that the president is in white house and he was robbed. Refer 7 and 8.

Figure 7: Sentence 4 Incorrect Parsing

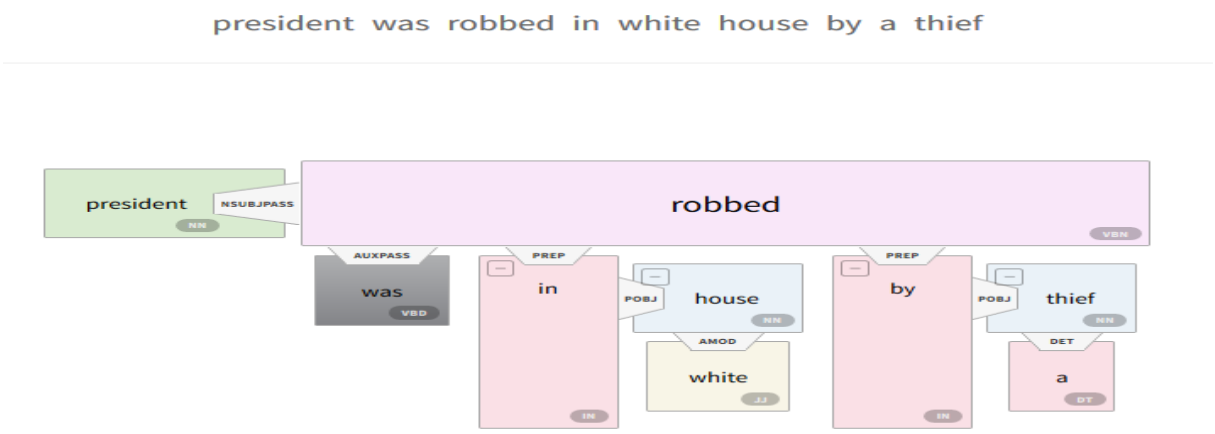

Figure 8: Sentence 4 Correct Parsing

- **Sentence -** Reagan wins on budget, but more lies ahead Here the parser takes lies a noun plural which gives incorrect meaning. This is a noun-verb ambiguity.
- **Modified Sentence -** Reagan wins on budget, but more needs to be known ahead. This modified sentence removes the ambiguity and known parses as a verb. Refer 9 and 10.

Figure 9: Sentence 5 Incorrect Parsing

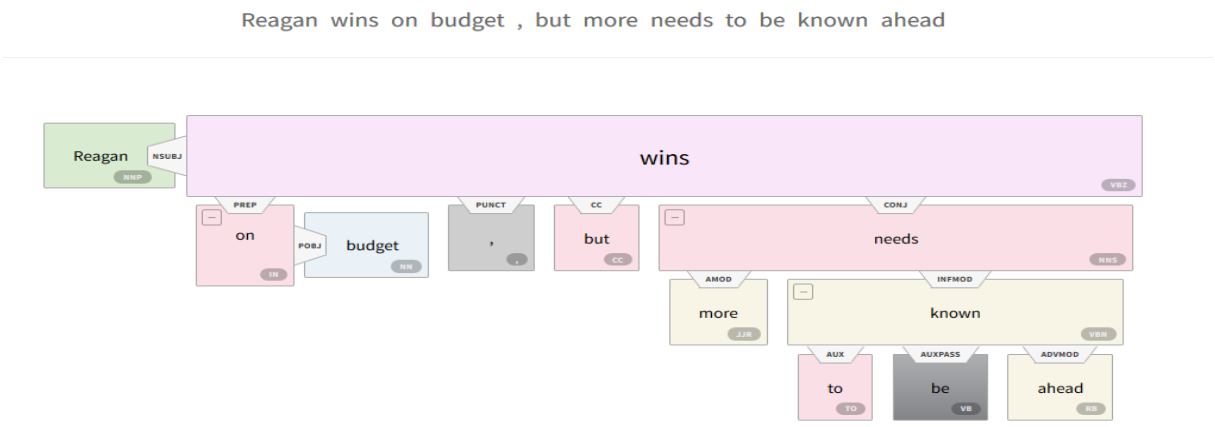

Figure 10: Sentence 5 Correct Parsing

Some reasons for failure -

- We are giving headlines of the sentences as input to the parser which are a compact version of the actual news. So this is one of the main reasons for obtaining ambiguous parses.
- There are a lot of words in a language which have the same word but different meaning. These are difficult to parse.
- Parsers generally face with the following problem of structural ambiguity, semantic ambiguity or lexical ambiguity.

1.2 Using first line of story -

It is given that the first line of the news story is often a repeat of the headline. The headline is generally made compact so it results in being ambiguous and hence it is difficult to parse correctly.

As we are given access to the first sentence of the story we can use this sentence and the headline to generate features for removing the ambiguity in the headline sentence.

Features to be included -

- **Sentence level Features:** We extract features like verbs, nouns, adjectives and adverbs. This is done for the first line of the story and the headline.
- **Features between headline and first line:** We extract VerbSim (cosine similarity of verbs), NounSim (cosine similarity of nouns), AdjSim (cosine similarity of adjectives), AdvSim (cosine similarity of adverbs) etc.
- **Ambiguity Features:** An ambiguous part extracted from the headline of length M is represented as $[z_1, z_2, ..., z_M]$, where $z_n \epsilon R$ d is the (pretrained) d-dimensional vector representation of the n-th word in the sequence. We again use the average vector $\bar{z}$ to obtain the representation of ambiguity information:

$$\bar{z} = \frac{1}{M} \sum_{n=1}^{M} z_n$$

- The model first builds local interactions between first line of the story and the headline and obtain a matching histogram. It then employs a feed-forward matching network to learn hierarchical matching patterns and produce a matching score for each term of the headline. Finally, the overall matching

score is obtained by aggregating the scores from each single term with a term gating network. The model optimizes the hinge loss with Adagrad.

Using the above method we can remove some ambiguities from headlines using the first line of the story. This way the modified sentences can then be used by the dependency parser. This will result in better parsers.

Reference to the paper:
`https://www.ijcai.org/proceedings/2018/0588.pdf`

1.3 Improving the parser -

The model described in this paper `https://web.stanford.edu/~tdozat/files/TDozat-CoNLL2017-Paper.pdf` has been built on top of deep biaffine neural dependency parser (used in https://demo/allennlp.org) `https://arxiv.org/abs/1611.01734` which uses vector representations in novel biaffine classifiers to predict the head token of each dependent and the class of the resulting edge. The two extensions are -

- Used character level representations to better represent rare/infrequent words
- Used their own pos-taggers

In general it is a simple parser trained only words and tags. This system uses BiLSTM networks for tagging and parsing, and includes character-level word representations in addition to token-level ones.

Model overview -

- The input to the model is a sequence of tokens and their part of speech tags, which is then put through a multilayer bidirectional LSTM network.
- The output state of the final LSTM layer is then fed through four separate ReLU layers, producing four specialized vector representations:
  - one for the word as a dependent seeking its head
  - one for the word as a head seeking all its dependents
  - another for the word as a dependent deciding on its label
  - fourth for the word as head deciding on the labels of its dependents.
- These vectors are then used in two biaffine classifiers:
  - The first computes a score for each pair of tokens, with the highest score for a given token indicating that tokens most probable head
  - the second computes a score for each label for a given token/head pair, with the highest score representing the most probable label for the arc from the head to the dependent

The major advantages of this parser over existing ones are the extensions that they have done. They have shown in their results that this method produces a tagger with higher accuracy.

Most of the examples sentences that have been incorrectly parsed (mentioned in previous section) was mainly due to the words receiving incorrect tags based on the context. Using this parser might actually resolve some of the mistakes.

2 Medical Relation Extraction:

2.1 Feature-based Extractor Solutions:

2.1.1 Case 1: Handful of examples

- We are given a bunch of data sentences (medical abstracts) but we are given only few instances for each known relation.
- Our job is to find out more instances of each relation.
- **For obtaining training data**, we use the technique of bootstrapping.
  - We initially have a seed set of relations/examples.
  - For each relation, we find sentences from articles which contains these entities and extract patterns from them.
  - The above extracted patterns can then be used to find more instances having that relation.
  - We can repeat the above two steps until no new instances can be added.
- In this way we can generate training data. Refer Fig. 11
- **For identifying target entities**, we can design another relation extractor which contains relations of type IS-A to find target entities. From these given relations:
  - **treats(drug, disease)**
  - **side-effect(drug, disease)**
  - **causes(drug, disease)**

  we can create the following relations which are of types
  - **IS-A(M, drug)**
  - **IS-A(M, disease)**
  - **IS-A(M, organism)**

  As we don't have enough training data, we generate them using the technique of bootstrapping as used previously.
  - We have an initial set of relations/examples constructed from given instances.
  - For each relation, we find sentences from articles which contains these entities and extract patterns from them.
  - The above extracted patterns can then be used to find more instances having that relation.
  - We can repeat the above two steps until no new instances can be added.
- In this way we can identify the target entities.

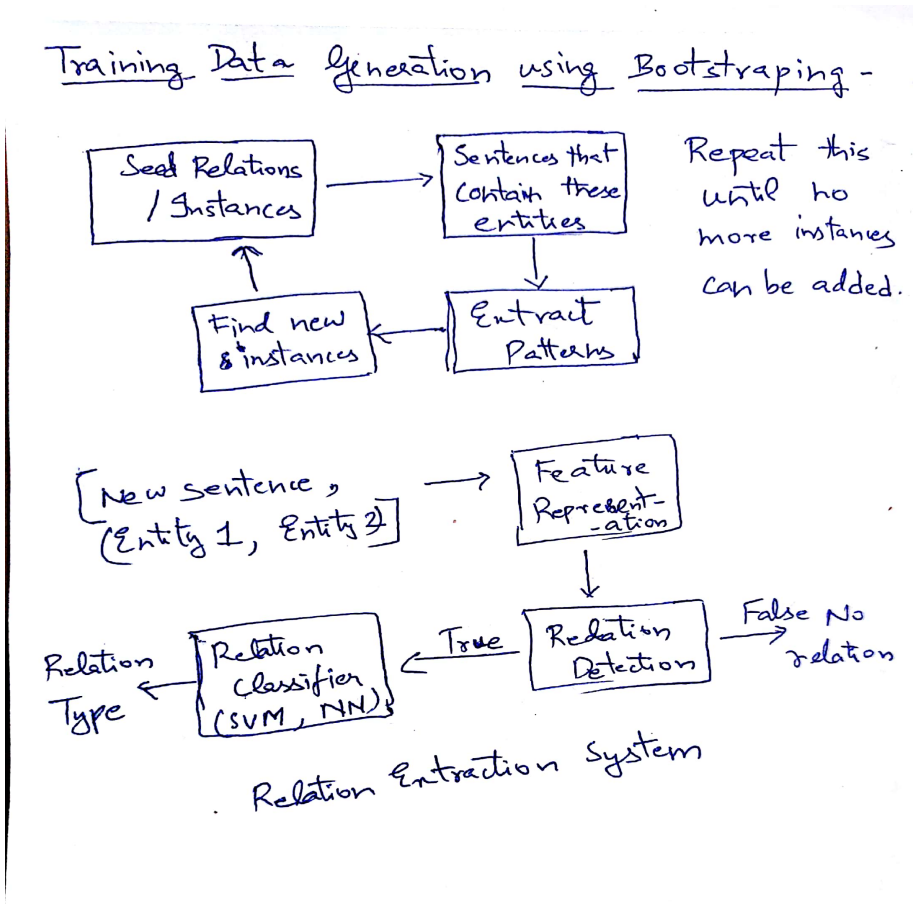

Figure 11: Relation Extraction Model Overview

Example Sentence - *Thus, taurine, as a potential therapeutic agent, may hold promise in* ***preventing*** *oxidative and ER stress mediated* ***diabetic*** *testicular complications in rats.*

Here in this example we have "taurine" which may "prevent" "diabetic testicular" complications in rats. We can have synonyms of the word "treat" such as prevent/cure/mitigate/eradicate as features for identifying the relation "treat".

Reference Sentence Link: `https://www.ncbi.nlm.nih.gov/pubmed/30496779`

Features that can be included -

- Using the stemmed version of the words of the sentence
- Including the entity types

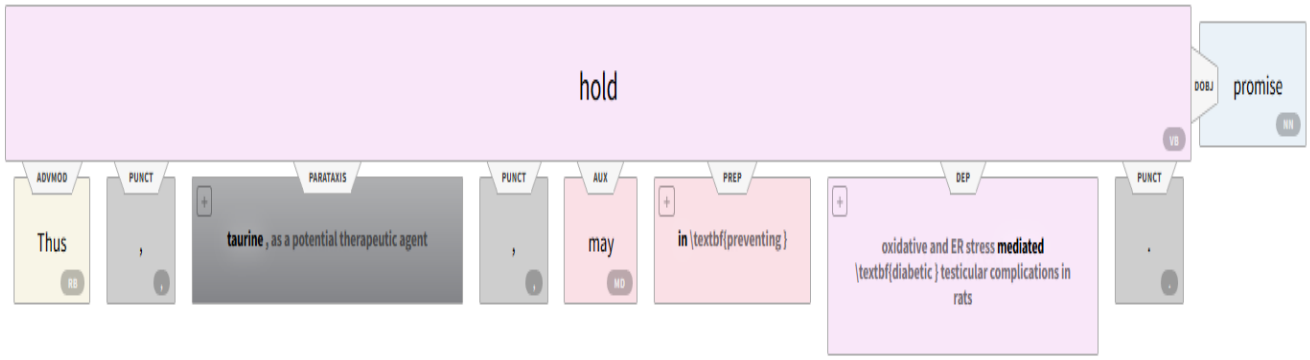

Figure 12: Dependency parse of sentence (Source: `https://demo.allennlp.org/dependency-parsing`)

- Base phrase chunks
- Head word for the above sentence is "hold" which can be seen in the dependency parse tree Fig. 15.
- Using before and after words as features.
- Using POS Tagging to help understand the syntactic nature of the sentence.
- Dependency Parse tree paths can be used to understand the distance and dependence between entities.

Challenges anticipated -

- While doing bootstrapping we sort of assume that the entities extracted for a relation in the initial iterations to be true and if we make a mistake in finding those, the error propagates through next iterations and gives us incorrect instances.
- If we do not stop the process after a couple of iterations it can lead to semantic drift (stated in class) which can give incorrect results.
- It is also assumed that the initial instances given for each relation as true (there could be some error).

2.1.2 Case 2: Large Database of examples

1. **Obtaining Training Data-** We are given tens of thousands of examples for each relation type. We have a separate table for each relation. We can use the entire web as training data (a large unlabeled corpus).
2. **Identifying target entities-** The entities in a sentence can be identified using Stanford NER Tagger which labels persons, locations and organizations. We consider only those entities that are a part of one of the relation table. The assumption in distant supervision is that any sentence containing these entities might represent that relation. The features from this sentence contribute to the feature representation of the relation.
3. **Features used -**
   - **Lexical Features -** The word sequence between the entities, parts-of-speech tags for the words
   - **Syntactic Features -** The dependency path between the entities

- **NER Tagger** We can add the NER tag for each word.

4. **Training -** Features for each tuple (relation, entity1, entity2) from different sentences are combined and fed into a multi-class logistic classifier to train it.

5. **Testing -** We have identify the entities using NER for the test sentences. Features are extracted from those sentences and added for each entity pair. When we pass these entity pair features through the classifier we get the relation type.

6. **Challenges anticipated -** Feature generation takes a lot of time.

Reference to the paper:
`https://www.aclweb.org/anthology/P09-1113`

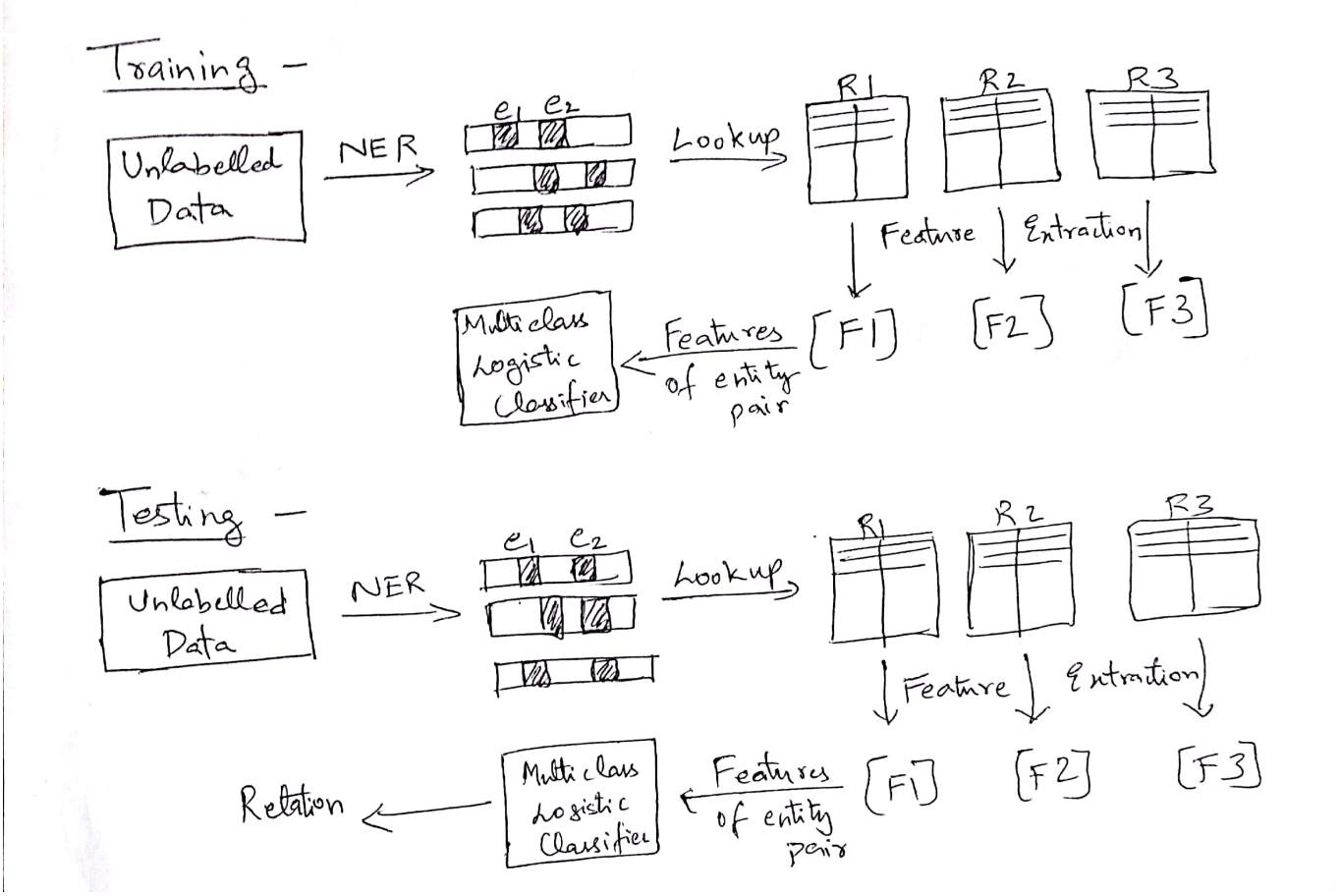

Figure 13: Model Overview

2.2 Extraction across sentences:

We need to develop a model which tackles the case where the entities of a relation belong to different sentences.

We can use NER to first identify the entities.

- The model that can be used for relation extraction is inter-sentential dependency-based neural networks (iDepNN).

- This models the shortest and augmented dependency paths using RNN to extract relationships within and across different sentences.

Some definitions -
iDepNN: the inter-sentential Dependency-based Neural Network, an NN that models relationships between entity pairs spanning sentences, i.e., inter-sentential within a document.
iDepNN-SDP: iDepNN that only models the shortest dependency path (SDP) spanning sentence boundary.
iDepNN-ADP: iDepNN that only models the augmented dependency paths (ADPs).

- We can compute the inter-sentential Shortest Dependency Path (iSDP) between entities from different sentences for a relation.
- The shortest path spanning sentence boundary is a sequence of words between two entities. We also look at the dependency subtree for more information.
- For constructing subtree embedding c_w, we traverse in a bottom-up fashion from its leaf words to the root for entities from different sentences.
- For leaf node (no subtree) we take the embedding as c_l. Each word has a dependency relation r, we learn a transformation matrix W.
- The subtree embedding is computed as:

$$c_w = f(\sum_{q \in Children(w)} W_{R_{w,q}} + p_q + b + and p_q = [x_q][c_q])$$

  where $R_{w,q}$ is the dependency relation between word w and its child q and b is the bias. We do this recursively up to the root word.
- We can combine both iSDP and dependency subtrees to form inter-sentential Augmented Dependency Path (iDepNN-ADP).
- We use a biRNN for iDepNN-SDP and iDepNN-ADP and pass the last hidden layer h_N to a softmax layer whose output is the probability distribution y over relation labels R,

$$y = softmax(U.h_n + b_y)$$

  where $U \in R_{RxH}$ is the weight matrix connecting hidden vector of dimension H to output of dimension R and $b_y \in R_R$ is the bias. h_N is the last hidden vector of biRNN.
- To compute semantic representation h_w for each word w, we combine the forward and backward pass by adding their hidden layers (h_{f_t} and h_{b_t}) at each time step t and also adds a weighted connection to the previous combined hidden layer h_{t-1} to include all intermediate hidden layers into the final decision.

$$h_{f_t} = f(Vi_t + Wh_{ft})$$
$$h_{b_t} = f(Vi_{N-t+1} + Wh_{bt+1})$$
$$h_t = f(h_{ft} + h_{bt} + Wh_{t-1})$$

  where $V \in R^{Hx|i|}$, N is the total number of words on iSDP and i_t the input vector at t, defined by: iDepNN-SDP : $i_t = [x_t; L_t]$ iDepNN-ADP : $i_t : [p_t; L_t]$

- The optimization objective is to minimize the cross-entropy error between the ground-truth label and softmax output. The parameters are learned using backpropagation.

Features that are used -

- Add positions of the entities in the vocabulary.
- Assign the entity type.
- Part of speech tagging for the words
- Dependency path

Reference to the paper:
`https://arxiv.org/abs/1810.05102`

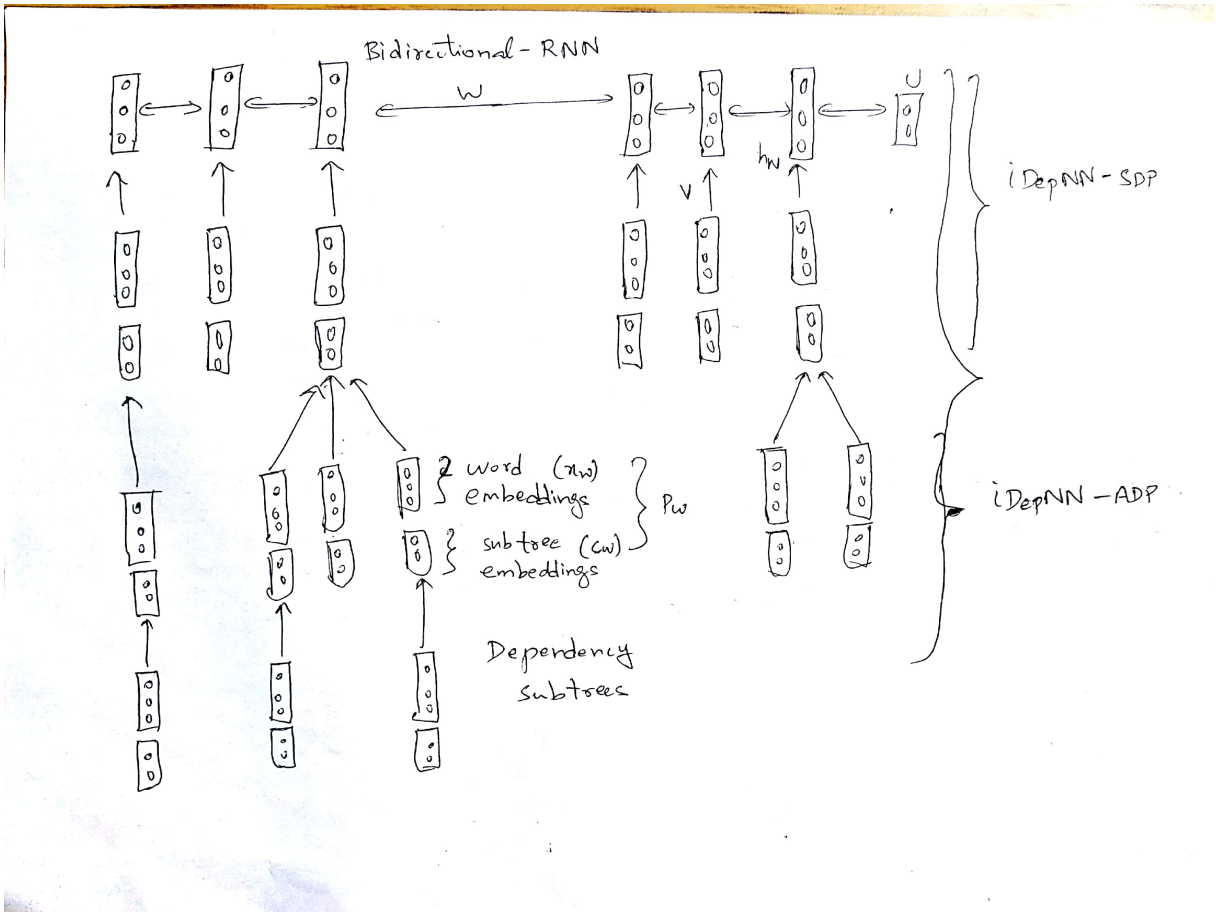

Figure 14: Model Overview

3 Machine Translation:

3.1 A third language:

3.1.1 Noisy Channel Model including German Language:

- One of the effective ways in which we can design this is to use German as an intermediate language during translation.

- This implies that we are translating from French (source) to English (target) using German (intermediate) language.
- This model can perform better than direct translation because there could be some instances in French that have good translations to German and similarly some instances in German that have good translations to English.
- The alignment can improve by including German language.

Generative Story -

- Select an arbitrary length for the intermediate language.
- Select an arbitrary alignment from source to intermediate language based on the lengths.
- Any intermediate word can be obtained by translating the source word based on the alignment.
- We can do the same process for translation between intermediate to target language.

3.1.2 Modified EM Algorithm:

We can't use Maximum Likelihood Estimation (MLE) because we don't have the alignments so we use the EM algorithm. We can modify the EM Algorithm to work for both source to intermediate and intermediate to target translations simultaneously.

The EM algorithm has two important steps:

- **Expectation Step:** In this step we estimate the alignment probabilities using the translation probabilities for both source to intermediate and intermediate to target words.

$$Pr(a|f,e) = \frac{Pr(f,a|e)}{\sum_a Pr(f,a|e)}$$

- **Maximization Step:** In this step we estimate the new translation probabilities using the alignment probabilities for both source to intermediate and intermediate to target words as found in previous expectation step.

$$Pr(f_i|e_j) = \frac{\sum_{f_i \leftrightarrow e_j} Pr(a,f|e)}{\sum_{e_j} \sum_{f_i \leftrightarrow e_j} Pr(a,f|e)}$$

We do the above steps one after the other until the values converge.

The model equation is:

$$E = argmax_e Pr(e)Pr(f|e) \tag{1}$$

$$Pr(f|e) = Pr(f|g)Pr(g|e) \tag{2}$$

Here we are making independence assumption for translation from French to German and German to English. We can calculate $Pr(f|g)$ using the following equations: m - length of the source sentence p - length of the intermediate sentence t - length of the target sentence

$$Pr(m|p) = \epsilon$$

$$Pr(a|m,p) = Pr(a) = \prod_{i=1}^{m} Pr(a_i) = \prod_{i=1}^{m} \frac{1}{p+1}$$

$Pr(f|g_{a_i})$ is the translation probability

$$Pr(f,a|g) = Pr(m|p)Pr(a|m,p)Pr(f|g,a)$$

$$= \epsilon \times \frac{1}{(p+1)^m} \times Pr(f|g_{a_i})$$

$$Pr(f|g) = \sum_{a \in align} Pr(f,a|g) \tag{3}$$

Similarly we can calculate for

$$Pr(g|e) \tag{4}$$

Substituting equations 3 and 4 in 2 and using this obtained result in 1 gives us the model. The above equations have been adopted from the slides and lectures.

3.1.3 Justification:

- There can be cases where the translations from source to target languages can be incorrect which is generally due to misalignment between the languages that the model has learnt.
- By including an intermediate language there is a good possibility that the alignments are better because there can be some instance/words where the translations from source to intermediate are better and similarly for some other words translations from intermediate to destination are good.
- We can get a more generalized model using this approach. The semantics of the source language can be learnt in a better way using the intermediate language rather than target language.

3.2 Translating headlines:

I have translated the following sentences from English to Telugu and here are my observations -

1. **Key GOP lawmakers flip on health care after meeting.**
   The word 'GOP' when translated to Telugu, remained as it is ('GOP') and the word 'key' meaning importance in this context was literally translated to 'ki' in Telugu.
2. **Spicer says part of spending bill will go to border wall.**
   The word 'Spicer' when translated to Telugu, remained as it is ('Spicer') but the syntactic structure was not right in Telugu.
3. **Ridge foster dad found not guilty of sex abuse charges.**
   The entire sentence got translated almost correctly except the word 'sex' which got translated to 'sexual' in Telugu.
4. **DWI suspect four times the legal limit with kids in car.**
   The word 'DWI' when translated to Telugu, remained as it is ('DWI') but the semantic structure was completely lost in Telugu.
5. **Confident politician says he wants to prove them wrong and get a Mideast peace deal.**
   The sentence got translated almost perfectly except the word 'Mideast'. The meaning is properly conveyed. A word to word translation would also be a good understandable translation in Telugu.

Hypothesis for explaining the errors -

1. One of the areas where Google Translate fails is while considering the abbreviations because they vary from language to language. This is evident from the above Sentences 1 and 4. **"Give me your SSN"**, in this sentence SSN is not translated properly (remains same). It should actually translate to 'es es yen".
2. Google Translate will also fail when dealing with complex sentence. **"The complex houses married and single soldiers and their families."** is one such sentence. It can be observed in Sentence 4 as well.
3. Sometimes google translate just gives the literal translation of the sentence. This behaviour was seen in Sentence 1. **"Hello, where is the key to my room"** is another such example where 'hello' and 'key' are literally translated.

Suggestions -

- To translate abbreviations from one language to another we can be fed them as separate entities and do translations using an existed translation database.
- As google translate fails on complex sentences we should come up with better neural architectures like variations of Bi-directional Tree LSTMs which could capture both the semantic and syntactic structure of the sentences.

3.3 Knowledge-backed Machine Translation:

- We are given access to knowledge bases that contain relations about entities.
- We use Stanford NER Tagger for finding the entities.
- We can directly find the relations between the entities from the relations table and create such triples <entity1, relation, entity2>.
- We use an encoder-decoder framework. This framework consists of three components:
  - RDF (Relation Triplet) pre-processor.
  - A target text pre-processor.
  - An encoder-decoder module.
- We can give each relation triplet to the above framework to get the corresponding sentence.
- From all the obtained sentences we select the most important sentences closely related (using cosine similarity) to the given sentence.
- From these selected sentences we use abstractive text summerizer to obtain the target sentence.
- We can finally use a seq2seq encoder-decoder framework to translate the sentence (from English to Telugu)

Reference to the paper - `http://aclweb.org/anthology/P18-1151`

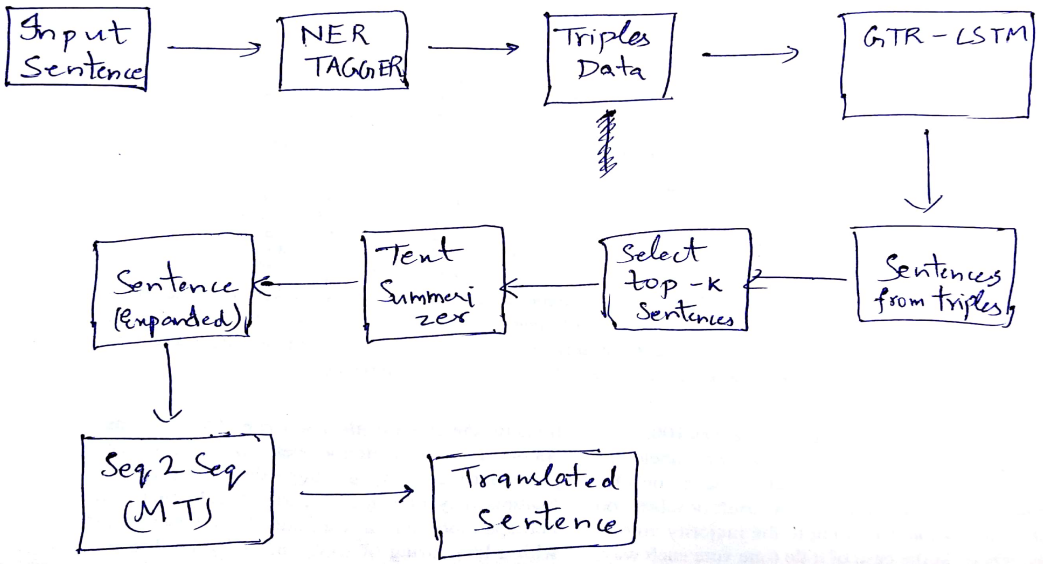

Figure 15: Model Overview

Components in my model -

- Named Entity Recognizer (NER)
- GTR-LSTM Model
- Cosine Similarity Module
- Text Summerizer
- Sequence to sequence translator