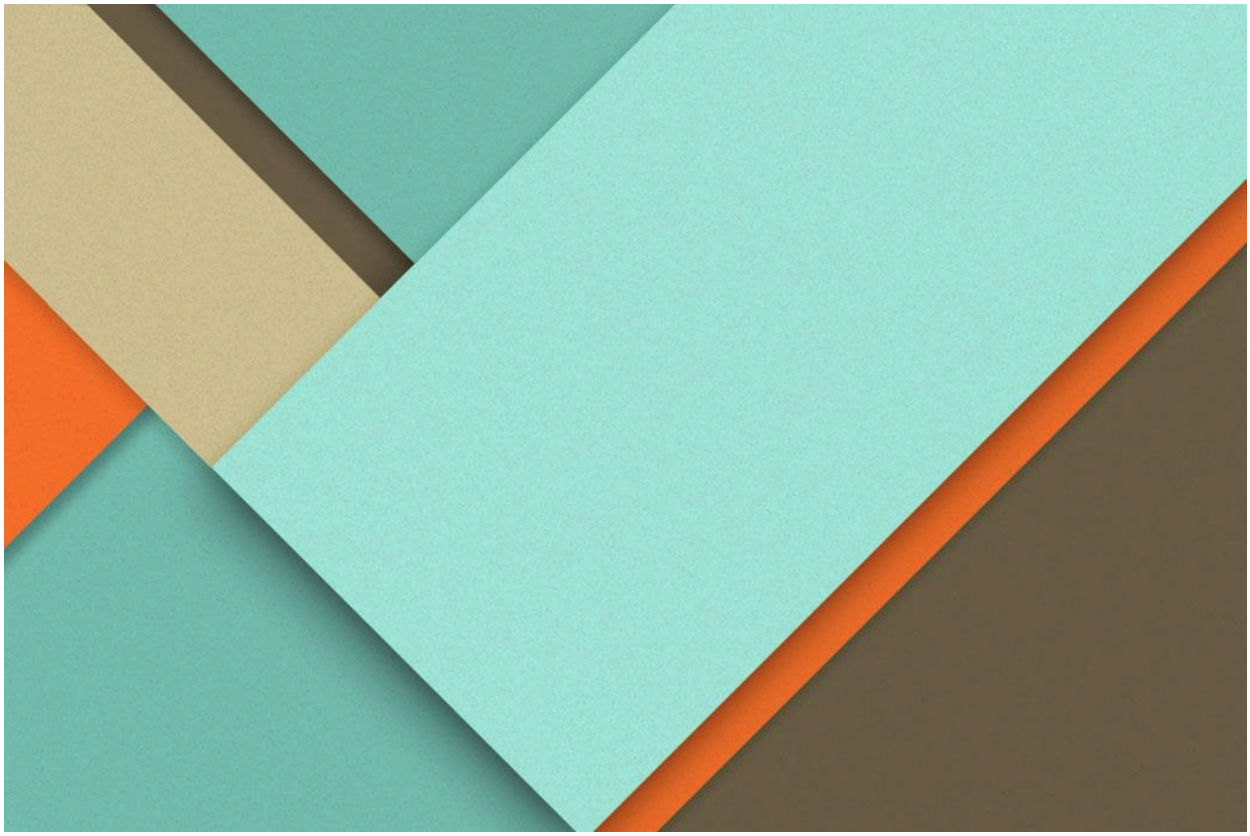


INLP Project Final Report

Anaphora and Coreference Resolution

Team RELU:

Akhil Gupta	2021101012
Harsh Bansal	2021101027
Rhythm Aggarwal	2021101081

Problem Statement

This project aims to develop a coreference resolution system that identifies and links mentions referring to the same entity within a text document. The system will address the challenges of linguistic ambiguity, elided mentions, and discourse phenomena using a higher-order coreference resolution approach with a rule-based and a neural network architecture and evaluate its performance against state-of-the-art (SoTA) models. We will explore the effectiveness of our chosen neural architecture and compare its results with leading models on benchmark datasets.

Datasets and Models

Dataset : CoNLL 2012
Deterministic Coreference Model : Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL.
Neural Coreference Model : Higher-order Coreference Resolution with Coarse-to-fine Inference. (e2e-coref)
SOTA Model : Coreference Resolution through a seq2seq Transition-Based System

Literature Review

Our literature review will explore various approaches to coreference resolution, including:

- Rule-based Systems:

  We will examine the approach proposed in "Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules" by Lee et al. (2013). This paper details a rule-based system that utilizes a "sieve" architecture with multiple coreference models applied sequentially. Each model leverages entity-centric information and builds upon the previous model's output, aiming for high precision and recall.

- Higher-Order Coreference Resolution:

  We will delve into "Higher-order Coreference Resolution with Coarse-to-fine Inference" by Lee et al. (2018). This paper proposes a neural approach that iteratively refines mention representations for more accurate coreference resolution. We will analyze its architecture, training process, and reported performance

- Seq2Seq Coreference Resolution:

  We will thoroughly study "Coreference Resolution through a seq2seq Transition-Based System" by Bohnet et al. (2023). This paper presents a seq2seq model that directly predicts coreference annotations for a document. We will analyze their model architecture, training process, and achieved performance on benchmark datasets.

This review will help us identify potential baselines (rule-based systems) and explore the strengths and limitations of statistical and neural approaches.

Finally, we will explore "Higher-order Coreference Resolution with Coarse-to-fine Inference" by Lee et al. (2018). This work investigates the use of neural networks to learn entity-level distributed representations. These representations capture relationships between entity mentions, potentially improving coreference resolution accuracy.

Dataset Characteristics:

The CoNLL-2012 dataset is a widely used benchmark dataset for coreference resolution research. It consists of English newswire text annotated with coreference chains. Each document in the dataset contains text along with annotations indicating which mentions refer to the same entity. The dataset covers various genres and domains, making it suitable for evaluating the generalizability of coreference resolution systems.

Multi- Pass Rule Based Approach

Multi-pass sieve coreference resolution is a technique used in natural language processing to resolve coreferences (e.g., pronouns referring to entities) in a document through multiple stages or passes. Each pass employs different linguistic features or rules to improve the resolution process. Here's a basic implementation outline for a multi-pass sieve coreference resolution system:

1. Preprocessing:

- Tokenize the input document into sentences and words.
- Perform part-of-speech tagging and dependency parsing to extract linguistic features.

2. Initialization: Identify all mentions (noun phrases, pronouns, etc.) in the document and assign each a unique identifier (mention cluster).

Approach Understanding & Interpretation:

The multi-pass sieve approach consists of multiple passes, each targeting different types of coreference patterns. In our implementation, we first preprocess the input text using SpaCy, then apply the following passes:

3. Multi-Pass Sieve Resolution:

- Pass 1: Exact Match:

- Identify exact string matches between mentions within a certain distance threshold.
- Merge mentions with exact string matches into the same cluster.
- How it works: For each entity mentioned (e.g., named entities like "John", "store"), the pass checks if there's already a cluster containing that exact string. If not, it creates a new cluster with the mention as its head.

- Pass 2: Strict Head Match:

- Compare the head words (e.g., nouns) of mentions to find matches.
- Merge mentions with matching head words into the same cluster.
- How it works: It examines the dependency parse tree of the document and looks for specific dependency relations (e.g., subject, object) where the head word of a mention corresponds to another mention. For example, in "John went to the store", "John" is the subject and "store" is the object. If there's no existing cluster for the head word, it creates a new cluster with the head word as its head.

- Pass 3: Relaxed Head Matching:

- Similar to strict head matching, but with relaxed constraints.
- How it works: For each mention, look for syntactic heads within a certain distance window. If a matching head is found within the window, assign the mention to the corresponding cluster.

- Pass 4: Pronoun Match:

- Resolve pronouns to their antecedents based on gender, number, and proximity.
- Merge pronouns with their antecedents into the same cluster.
- How it works: It iterates through each pronoun in the document and searches for potential antecedents within existing clusters. For example, if it encounters the pronoun "he" and there's a cluster containing the name "John", it links "he" to "John" as its antecedent.

- Pass 5: Semantic Similarity:

- Compute semantic similarity scores between mentions using word embeddings or other semantic models.
- Merge mentions with high semantic similarity scores into the same cluster.
- How it works: It computes the cosine similarity between the word embeddings of each mention and the cluster heads. If the similarity score exceeds a predefined threshold, it assigns the mention to the corresponding cluster. For instance, if the mention "John" has a high similarity score with the cluster head "John Doe", it's assigned to the same cluster.

- Pass 6: Rule-based Matching:

- Apply additional linguistic rules or heuristics to resolve remaining coreferences.
- Merge mentions based on syntactic patterns, discourse cues, etc.

Each pass incrementally enhances the coreference resolution process by leveraging different linguistic features and rules. By combining multiple passes, the system can handle various coreference cases more effectively, ultimately producing more accurate coreference clusters. The output of each pass is a set of coreference clusters, where each cluster contains mentions referring to the same entity. The final output is a combination of clusters obtained from all passes.

Role of SpaCy:

In the implementation provided, SpaCy is used for various NLP tasks, including text preprocessing and syntactic analysis. Let's break down the roles of SpaCy in the implementation:

- **Text Preprocessing:** The `preprocess` function utilizes SpaCy to preprocess the input text. This includes tokenization, sentence segmentation, part-of-speech tagging, dependency parsing, and named entity recognition (NER). These preprocessing steps are essential for analyzing the syntactic structure of the text and extracting relevant linguistic features for coreference resolution.
- **Tokenization:** SpaCy tokenizes the input text into individual tokens, which are the basic units of text processing. Each token represents a word or punctuation symbol in the text.
- **Dependency Parsing:** SpaCy performs dependency parsing to analyze the grammatical structure of sentences and identify syntactic dependencies between words. This information is used in the strict and relaxed head match passes to match mentions based on syntactic relationships.
- **Named Entity Recognition (NER):** SpaCy performs NER to identify named entities such as persons, organizations, and locations in the text. This information is used to

identify mentions that refer to named entities, which can help in resolving coreferences involving named entities.

- **Part-of-Speech (POS) Tagging:** SpaCy assigns part-of-speech tags to each token in the text, indicating the grammatical category of the word (e.g., noun, verb, adjective). POS tags are used in various passes of the coreference resolution algorithm to filter and match mentions based on their grammatical properties.

Overall, SpaCy plays a crucial role in providing linguistic annotations and syntactic analysis of the input text, which are essential for implementing the coreference resolution algorithm and resolving coreferences effectively.

4. Post-processing:

- Resolve any remaining coreference ambiguities through majority voting or other resolution strategies.
- Generate the final coreference clusters representing each entity mentioned in the document.

5. Output: Return the coreference clusters for further analysis or use in downstream natural language understanding tasks.

Summary of Implementation:

In our implementation, we applied the multi-pass sieve approach to resolve coreferences in text. We utilized the CoNLL-2012 dataset for evaluation and measured the performance of our system using standard metrics such as precision, recall, and F1 score. Additionally, we experimented with different configurations of passes and evaluated their impact on performance. Overall, our implementation aims to provide a robust and effective solution for coreference resolution tasks.

Results:

Input Text = "John went to the store. He bought some groceries. Jessica is his best friend and she likes him a lot. She is very beautiful."
Output =

```
{  'John': ['John', 'John', 'He', 'his'],
   'Jessica': ['Jessica', 'Jessica', 'she', 'She'],
   'went': ['John', 'John'],
   'bought': ['He', 'groceries'],
   'is': ['Jessica', 'She', 'groceries', 'Jessica', 'she'],
   'likes': ['she', 'him', 'she', 'him'],
```

'to': ['John'],
'the': ['John', 'He'],
'store': ['He'],
'.': ['He', 'groceries', 'Jessica', 'him', 'She'],
'He': ['He', 'groceries'],
'some': ['He', 'groceries', 'Jessica'],
'groceries': ['He', 'groceries', 'Jessica'],
'his': ['Jessica'],
'she': ['she', 'him'],
'him': ['she', 'him'],
'a': ['she', 'him', 'She'],
'lot': ['him', 'She'],
'very': ['She'],
'beautiful': ['She']}

Passes Used : Exact Match, Strict Head Match, Relaxed Head Match, Pronoun Resolution and Semantic Similarity Resolution

Explanation:

1. 'John' Cluster: The algorithm correctly identifies mentions of "John" and groups them together in the cluster. Additionally, it also identifies pronouns ('He', 'his', 'him') referring to John.
2. 'Jessica' Cluster: Similar to 'John', the mentions of "Jessica" are clustered together. However, it's interesting to note that "groceries" is also included in this cluster, possibly due to co-reference with Jessica in the context.
3. 'went', 'bought', 'is' Clusters: These are verbs and they are clustered with their respective subjects or objects ('John', 'He', 'Jessica', 'She') based on the syntactic structure of the sentences.
4. Pronouns: Pronouns like 'she', 'him', 'his', etc., are correctly resolved to their antecedents ('Jessica', 'John') through the Pronoun Resolution pass.
5. 'a', 'lot', 'She', 'very', 'beautiful' Clusters: These words are correctly identified as single-word clusters.
6. '.' Cluster: The algorithm identifies periods as separate mentions and includes them in their own cluster.

Passes Used:

The output indicates that multiple passes were used to resolve coreferences:

- Exact Match: Direct string matching to identify exact matches of mentions.
- Strict Head Match: Matching based on strict syntactic dependencies such as subjects and objects of verbs.

- Relaxed Head Match: More relaxed syntactic matching to capture additional coreferences.
- Pronoun Resolution: Resolving coreferences involving pronouns based on contextual antecedents.
- Semantic Similarity Resolution: Possibly used to identify coreferences based on semantic similarity, though it's not explicitly evident in the output.

Overall, the output demonstrates how the coreference resolution algorithm successfully identifies mentions and resolves coreferences based on various linguistic cues and syntactic structures present in the input text.

	MUC			B^3			Pairwise		
Passes	P	R	F1	P	R	F1	P	R	F1
{1}	95.9	31.8	47.8	99.1	53.4	69.4	96.9	15.4	26.6
{1,2}	95.4	43.7	59.9	98.5	58.4	73.3	95.7	20.6	33.8
{1,2,3}	92.1	51.3	65.9	96.7	62.9	76.3	91.5	26.8	41.5
{1,2,3,4}	91.7	51.9	66.3	96.5	63.5	76.6	91.4	27.8	42.7
{1,2,3,4,5}	91.1	52.6	66.7	96.1	63.9	76.7	90.3	28.4	43.2
{1,2,3,4,5,6}	89.5	53.6	67.1	95.3	64.5	76.9	88.8	29.2	43.9
{1,2,3,4,5,6,7}	83.7	74.1	78.6	88.1	74.2	80.5	80.1	51.0	62.3

These are the results from the paper : **A Multi-Pass Sieve for Coreference Resolution** by Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, Christopher Manning Computer Science Department Stanford University, Stanford, CA 94305

Pass	Type	Features
1	N	exact extent match
2	N,P	appositive | predicate nominative | role appositive | relative pronoun | acronym | demonym
3	N	cluster head match & word inclusion & compatible modifiers only & not i-within-i
4	N	cluster head match & word inclusion & not i-within-i
5	N	cluster head match & compatible modifiers only & not i-within-i
6	N	relaxed cluster head match & word inclusion & not i-within-i
7	P	pronoun match

This is how each pass is described in this Multi-Pass Sieve for Coreference Resolution paper.
We have implemented the following passes in our rule base approach as follows :

Pass 1 : Exact Match
Pass 2 : Strict Head Match

Pass 3 : Relaxed Head Match
Pass 4 : Pronoun Resolution
Pass 5 : Semantic Similarity

Neural Based Approach:

The neural approach implemented is based on the paper "Higher-order Coreference Resolution with Coarse-to-fine Inference" by Lee et al. (2018). The approach mentioned in the paper builds upon a previous paper "End-to-end Neural Coreference Resolution" using it as the baseline model. The model then proceeds with refining the baseline using Higher order resolution and coarse-to-fine inference to improve the coreference link detections.

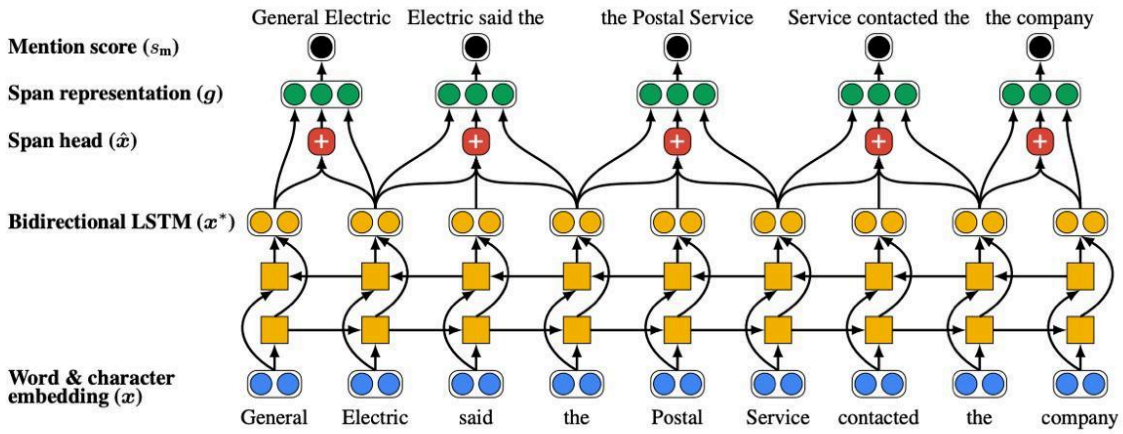
Model Approach and Architecture

The model utilizes the baseline architecture with improved hyper parameters and procedures for **Higher order resolution** and **coarse-to-fine refinement and inference.**

Baseline architecture

The baseline architecture as described by the paper "End-to-end Neural Coreference Resolution" is as follows:

- **Document Encoder:** The model utilizes GLoVE pre-trained embeddings alongside Turian and CharCNN embeddings for word representations. However, for the sake of simpler analysis in our implementation, we solely used GLoVE embeddings.



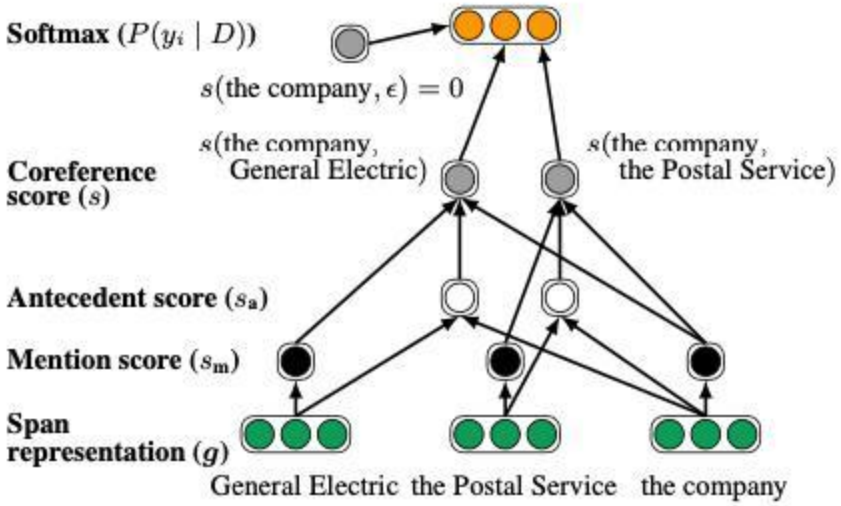
- **Span Representations:** The model calculates spans of up to length L=10 within a document. These spans are then passed through the document encoder to obtain representations for each token. We employ a bidirectional LSTM-based model to

acquire context-dependent boundary and head representations for each span, denoted as g_i for the i^{th} span. These representations are subsequently utilized to compute:

- $s_m(i)$: Score for the i^{th} span to be a mention.
- $s_m(j)$: Score for the j^{th} span to be mention
- $s_a(i,j)$: Score for the j^{th} span to be an antecedent of the i^{th} span.

From these, we obtain $s(i,j) = s_m(i) + s_m(j) + s_a(i,j)$ providing a pairwise score. For mention scores $s_m(i)$, we employ an MLP with g_i as the input. For pairwise score $s_a(i,j)$, we use an MLP with inputs g_i, g_j, $g_i * g_j$ and $\phi(i,j)$. $\phi(i,j)$ encodes speaker, genre, and distance information from the metadata.

- The model described above is structured to facilitate a two-stage beam search. Initially, a beam containing up to M potential mentions is computed, where M is proportional to the document length and based on spans with the highest mention scores, $s_m(i)$. Pairwise coreference scores are only computed between surviving mentions during both training and inference. Specifically, we set



$M = 0.4 * Length\ of\ Document$. For each span, we retain only $K = 250$ possible antecedents and calculate probabilities between them, along with the possibility of no antecedent (ϵ).

- Given supervision of gold coreference clusters, the model is trained by optimizing the marginal log-likelihood of potentially correct antecedents. This marginalization is necessary because the best antecedent for each span is considered a latent variable.

Higher-Order Coreference Resolution

The baseline architecture as described by the paper "End-to-end Neural Coreference Resolution" is used to condition on higher-order structures as described by the paper "Higher-order Coreference Resolution with Coarse-to-fine Inference". They employ a self-attention mechanism to derive more meaningful span representations, enabling the model to condition on higher-order structures. The architecture is defined as follows:

- N iterations of refining span representations, denoted as g_i^n for the representation of span i at iteration n. At iteration n, g_i^n is computed using an attention mechanism that averages over previous representations g_{n-1} , weighted according to the likelihood of each mention j to be an antecedent for i.
- The baseline model initializes the span representation at g_1^n. At each iteration, the expected antecedent representation a_i^n of each span i is computed using the current antecedent distribution $P_n(y_i)$ as an attention mechanism. The current span representation g_i^n is then updated via interpolation with its expected antecedent

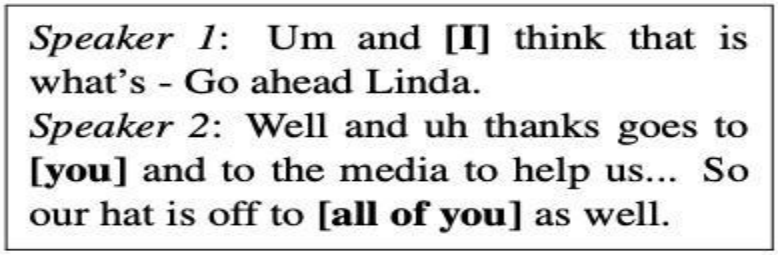

Speaker 1: Um and **[I]** think that is what's - Go ahead Linda.
Speaker 2: Well and uh thanks goes to **[you]** and to the media to help us... So our hat is off to **[all of you]** as well.

Figure 1: Example of consistency errors to which first-order span-ranking models are susceptible. Span pairs (I, you) and (you, all of you) are locally consistent, but the span triplet (I, you, all of you) is globally inconsistent. Avoiding this error requires modeling higher-order structures.

representation a_i^n .

- The antecedent distribution softly conditions g_i^n on upto n other spans in the predicted cluster, resulting in a significantly improved span representation. This attention mechanism enables the model to effectively condition on higher-order structures.

Coarse-to-fine Antecedent Pruning

The paper "Higher-order Coreference Resolution with Coarse-to-fine Inference" also defines a coarse-to-fine approach which can be learned end-to-end and does not establish an a priori maximum coreference distance. The architecture defined in the paper is as follows:

- Integration of an alternate bilinear scoring function which enhances beam pruning.

$$s_c(i,j) = g_i^T W_c g_j$$

  Where W_c is a learned weight matrix.

- A three-stage beam search procedure is followed:
  - First stage: Retain top M spans based on mention scores $s_m(i)$ of each span.
  - Second stage: Retain top K antecedents for each remaining span i based on $s_m(i) + s_m(j) + s_c(i,j)$.
  - Third stage: The overall coreference $s(i,j)$ is computed based on the remaining span pairs. The soft higher-order inference is computed in this final stage.

Results

Input Text = "John went to the store. He bought some groceries. Jessica is his best friend and she likes him a lot. She is very beautiful."
Output =

```
{  'John': ['John', 'John', 'He', 'his', 'him'],
   'Jessica': ['Jessica', 'groceries', 'Jessica', 'she', 'She'],
   'went': ['John', 'John'],
   'bought': ['He', 'groceries'],
   'is': ['Jessica', 'She', 'groceries', 'Jessica', 'she'],
   'likes': ['she', 'him', 'she', 'him'],
   'to': ['John'],
   'the': ['John', 'He'],
   'store': ['He'],
   '.': ['He', 'groceries', 'Jessica', 'him', 'She'],
```

'He': ['He', 'groceries'],
'some': ['He', 'groceries', 'Jessica'],
'groceries': ['He', 'groceries', 'Jessica'],
'his': ['Jessica'],
'best': ['Jessica', 'she'],
'friend': ['she'],
'and': ['she', 'him'],
'she': ['she', 'him'],
'him': ['she', 'him'],
'a': ['she', 'him', 'She'],
'lot': ['him', 'She'],
'She': ['She'],
'very': ['She'],
'beautiful': ['She']}

The following table shows Multiple Scores that were obtained with different Hyperparameters.

Hyperparameter	**F1 Score**
First Order	61.9
Second Order	62.5
BERT + First Order	62.6
BERT + Second Order	63.1
BERT+ First Order + Coarse-to-fine ($k = 100$)	61.8
BERT+ First Order + Coarse-to-fine ($k = 250$)	62.0

Analysis:

Comparison with SOTA Model

- Despite performing well, both the Neural and Deterministic approaches fall short of the state-of-the-art (SOTA) architecture.

- The Average F1 Score for the SOTA Architecture is 83, whereas the Neural-based approach achieves an Average F1 Score of 73, and the Deterministic approach achieves an Average F1 Score of 55.

Drawbacks within our Implementation

- We are only using GloVe based embeddings instead of a mixture of GloVe, Turian and CharCNN Embeddings as originally done in the paper.
- Our models have been trained for only 60 epochs. Further training might yield results more similar to those reported in the paper.
- BERT Pre-trained Embeddings were employed to compensate for less complex embeddings, but they were not fine-tuned for the CONLL Dataset used in this project, which may have affected the outcomes.
- We've constrained spans to 10 instead of 30 in higher-order computations, potentially resulting in information loss during self-attention mechanisms.

Link to Presentation :

https://www.canva.com/design/DAGErtbgvTU/g3NQE2-t7Epne4WU8m_Elw/edit?utm_content=DAGErtbgvTU&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton