Event Coreference Resolution in Social Media Text
Deliverable - 2 ; Group 4

Aim

Goal of this project is to implement the Event Coreference algorithm proposed by Chao et al. in the paper "Selective Expression for Event Coreference Resolution on Twitter".

Abstract

Event coreference is the problem of identifying and connecting mentions of the same events in different contexts. It is a fundamental problem in NLP with wide-spread applications.The given paper is the state-of-the-art for event coreference in Twitter. With the growth in popularity and size of social media, there is an urgent need for systems that can recognize the coreference relation between two event mentions in texts from social media. Approach till now basically depend upon NLP features which restricts domain scalability and leads to propagation error.In this paper a novel selective expression approach based on event trigger to explore the coreferential relationship in high-volume Twitter texts is proposed. Firstly a bidirectional Long Short Term Memory (Bi-LSTM) is exploited to extract the sentence level and mention level features. Then, to selectively express the essential parts of generated features, a selective gate is applied on sentence level features. Next, to integrate the time information of event mention pairs, an auxiliary feature is designed based on triggers and time attributes of the two event mentions. Finally, all these features are concatenated and fed into a classifier to predict the binary coreference relationship between the event mention pair. They also released a new dataset called EventCoreOnTweet(ECT) dataset on which they evaluated their methods.It annotates the coreferential relationship between event mentions and event trigger of each event mention. The experimental results demonstrate that the approach achieves significant performance in the ECT dataset.

Related Work

1. Liu et al., Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation, EMNLP 2018
2. Choubey and Huang, Improving Event Coreference Resolution by Modeling Correlations between Event Coreference Chains and Document Topic Structures, ACL 2018

Both of these papers are discussed in detail in the appendix.

Baseline Methodologies tried

Since the aim is to implement a paper most of the methodologies or the way to get the result has already been tried by the author. Our task is to replicate the results of the author. First we had to decide upon the framework to use. We decided to use Tensorflow since the author also used the same(and it was suggested by our mentor also).

1. **Data Collection:**

   To reproduce the results we need to use the dataset used by the authors. However since they did not have permission to distribute the tweets themselves, they removed the terms, replacing them with *"WORD"* and left with instructions to create their dataset. Thus to create their dataset we followed these instructions. First we extracted tweet ids from *corpus.txt* which they uploaded. Script to do so was created on the fly. Then we downloaded tweets corresponding to these tweet ids using api provided by *tweepy*. In total we got around 1994 tweets out of total of 2994 tweets that were mentioned. ***Since the collected data is only 67% of the actual dataset we do not expect results to same as the results of the author.*** Tweets which were missing were due to one of the following reasons:
   1) User no longer exists
   2) Tweet no longer exists
   3) We didn't have the permission to view the tweet.
   After getting tweet we tokenized and pre-processed the tweets using the scripts provided by them and then we wrote a script to insert actual tweets using tweet ids in orignal *corpus.txt* file provided by the author. In case tweet is not available, we left it blank.

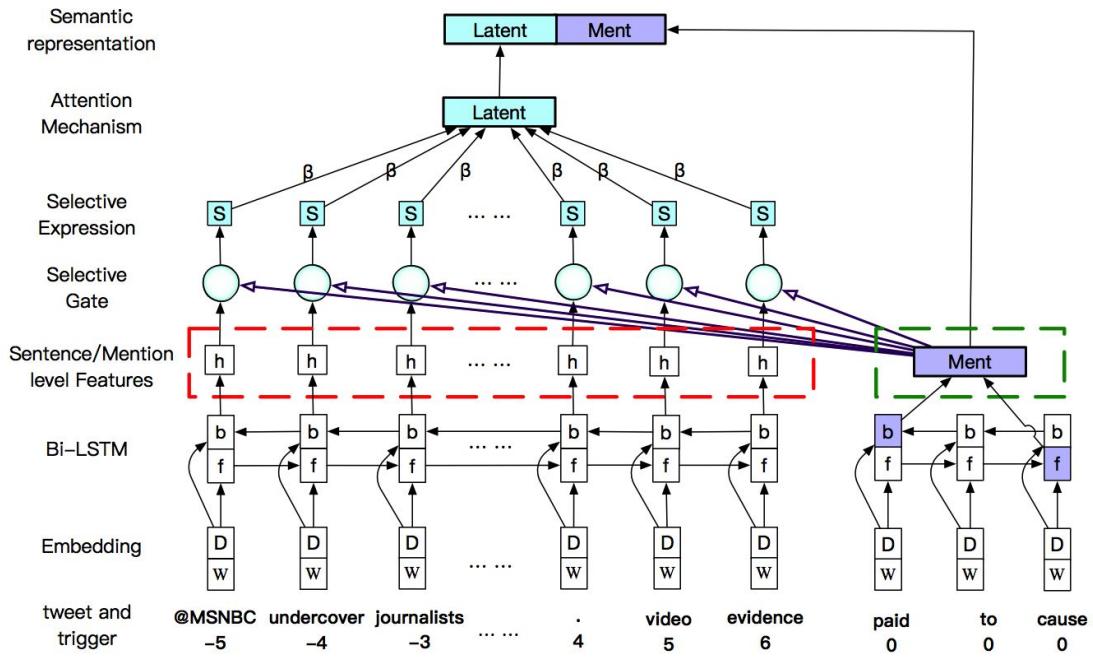
2. **Dataset creation:**

   a. From the data obtained using tweepy, we created the dataset for training(sentence pairs and label), validation and testing using the algorithm mentioned in the paper. We merged the training data and validation data as the data for training is less.
   b. For each event mention e, we paired it with all other event mentions which were tweeted in a time period of less than 7 days after e.
   c. For each pair, we stored the event_mention_id for EM1, tweet_text for EM1, trigger_text for EM1, event_mention_id for EM2, tweet_text for EM2, trigger_text for EM2, label and timestamp difference between the pair.
   d. The event-mentions in a pair are considered co-referential(i.e. label=1), iff they have the same event_instance_id.

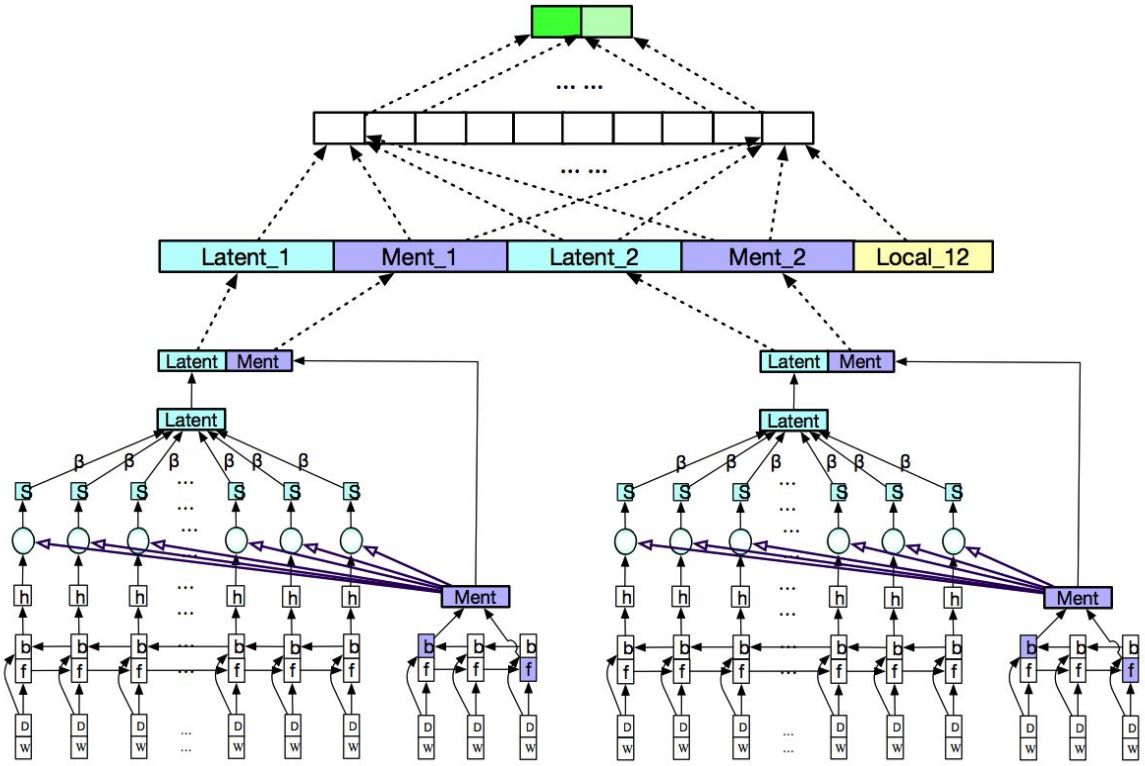
3. Implementation:

a. We indexed the unique words in the tweet_text and trigger_text for both train and test data.
b. We used Glove twitter 100d embeddings to assign embeddings to words. If any word is not present in Glove, we assigned a [0]*100 embedding value to it.
c. We also calculated relative distance of each word from event trigger and sent it as input to model. The model initializes random distance embeddings and trains them along with others.
d. We appended the word embedding and distance embedding and created final embedding for each word and passed it as input to BiLSTM for both sentence pairs and event trigger pairs.
e. We coded the next layers in the network as specified in the paper.
f. We divided the training data into batches of size 128. The number of epochs required to train the data is not mentioned in the paper. Hence, for the time being, we set the number of epochs to 5. We are trying to plot a graph of accuracy (or loss) obtained vs the number of epochs. We are planning to do this for number of epochs = 5 to 15 with an interval > 3.

Architecture



Semantic representation for a single event mention on Twitter.



coreferential decision of two event mention

1. **Sentence and Mention Level Features**

   After pre-processing each segmented token in a tweet will be transformed into an embedding vector. The embedding vector consists of two parts: word embedding and distance embedding. Word2vec was used to pre-train word embedding, and label *<unknown>* to represent the out of vocabulary (OOV) words. Words that are close to event trigger have more semantic relevance which is captured by distance embeddings. Bi-LSTMs contains two parts $LSTM_f$ and $LSTM_b$. $LSTM_f$ captures left context and $LSTM_b$ captures right context. Output of both of them are concatenated and fed to selective gate. The sentence level feature (dashed rectangle in the left of figure above) is a sequence of each word's context feature in tweet when tweet as input where n is the length of tweet.

$$h_{f,i} = LSTM_f[x_i, h_{f,i-1}]$$
$$h_{b,i} = LSTM_b[x_i, h_{b,i+1}]$$
$$h_i = [h_{f,i}, h_{b,i}]$$
$$Sent_{level} = (h_0 , h_1 , h_2 , ..., h_n)$$

   And the mention level feature (dashed rectangle in the right of the figure above) is the concatenation of the last output of forward and backward LSTM cell when event trigger as input where m is the length of event trigger.

$$Ment_{level} = [h_{f,m} , h_{b,0}]$$

2. **Selective Expression**

Each word plays a different role for one specific event trigger in the same sentence. Some are core words, while another portion of words is semantically confusing. Therefore selective expression mechanisms were employed to achieve more accurate latent features of event mentions by limiting the semantic expression of unimportant or irrelevant words according to the event trigger. 'Select' represents the selective representation of event mentions.

$$R_c = h_i * Ment_{level}$$
$$\alpha_i = \tanh(W_s \cdot R_c + b_s)$$
$$Select = \alpha * Sent_{level}$$

3. **Attention Mechanism**

If there are different event mentions in a tweet, the semantic contribution of each word in the tweet to an event mention's semantic representation is varied. Therefore attention mechanisms is used to compute the importance score for each word's selective expression to reflect the different contributions to semantic representation of each event mention. Then the importance scores are normalized to obtain the latent feature by the weighted sum.

$$u_i = V_a^T \tanh(W_a\ Select_i + b_a)$$
$$\beta_i = \exp(u_i)/ (\Sigma \exp(u_i))$$
$$latent = \Sigma \beta_i\ Select_i$$
$$V_{em} = [latent, Ment_{level}]$$

4. **Coreference Decision**

To determine coreference between two event mentions the second part of the architecture receives the semantic representation of two event mentions V_{em}^1 and V_{em}^2 along with $V_{local}^{1,2}$ where $V_{local}^{1,2}$ captures the number of words in the overlap between event triggers and number of days between two event mention.

$$V_{pair} = (V_{em}^1, V_{em}^2, V_{local}^{1,2})$$
$$V_{local}^{1,2} = (V_w, V_d)$$

V_W : The number of words in the overlap between event triggers.

V_D : The number of days between two event mention.

The pairwise features are processed by a simple neural network layer to calculate the distributed similarity

$$V_{ds} = relu(W_{ds} \cdot V_{pair} + b_{ds})$$

A softmax layer is applied finally to calculate the probability of two categories (coreferential and not coreferential)

$$Score = Softmax(W_{pro} \cdot V_{ds} + b_{pro})$$

Evaluation Mechanism & Results

Hyperparameter settings

Batch size	128
LSTM size	128
Attention size	128
Co-reference size	128
Word embedding size+Distance embedding size	100+14
Learning rate	0.01
Number of epochs	5

We divided the test dataset into batches of size 128
We will report results in terms of Accuracy(A), Precision (P), Recall (R) and F1 -score (F1) using commonly-used coreference scoring algorithms given by the CoNLL scorer to evaluate the event coreference resolution system.
We are still in the process of generating results since training is taking a large amount of time. We will update the results in our GitHub page once its done.

Analysis

1. We see that our system is able to perform the task of coreference resolution but the results are not that great. The reason is the dataset size. Since a lot of tweets are missing, the performance of the system is adversely affected.
2. For two tweets which has co-reference, even if one of the tweets goes missing, the other is of no use to us and thus training example size is greatly reduced affecting the performance of the system.
3. Also, we observed that in pre-processing stage of raw twitter data that, some event triggers are not a part of their corresponding tweets. Example: ‘call’ is event trigger whereas ‘calls’ is present in event mention (The code for pre-processing twitter data is made available by the authors. We used that as it is)
4. We would also like to add that using Glove embeddings instead of word2vec embeddings trained on the given training data increased the efficiency of model.
5. Using BiLSTM produced better results than using LSTM even for event triggers(though they are small in length and don’t have heavy long-distance interdependencies)
6. Overall, this paper uses a really efficient network to solve coreference resolution problem.

7. We also looked at the two papers mentioned in ‘Related Work’ section and are thinking of ways to combine those ideas with our current approach to produce a more efficient model.

Code Link

https://github.com/him-mah10/IRE-Major-Project

Link to webpage

https://him-mah10.github.io/group4ire/

Appendix

1. Liu et al., Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation, EMNLP 2018

The paper talks about multiple events extraction from a sentence. A lot of work used sequential modelling methods which suffered because long-range dependencies were not captured. In this paper multiple events are extracted by introducing syntactic shortcut arcs to enhance information flow and attention-based graph convolutional networks to model graph information.

Approach

The approach used by the paper is join approach which extracts event triggers and arguments simultaneously as a structured prediction problem. Also it is important to understand why the paper goes for sentence level event extraction and not document level event extraction. The reasons are:
1) It was found that the document-level co-occurrence distributions of 33 types of events in the ACE 2005 dataset are relatively similar to the sentence-level co-occurrence distributions.
2) There are many off-the-shelf sentence-level linguistic resources in the NLP community which can offer analytical information about the shortcut paths of some structures, like dependency parsing trees, AMR parsing graphs, and semantic role labeling structures.
3) It was observed that events within the same sentences have more explicit relationships with each other than events in different sentences of a document.

Now coming to JMEE framework. First they used BIO annotation schema to assign trigger label t_i to each token w_i , as there are triggers that consist of multiple tokens. This was then fed to the module as discussed. The JMEE framework consists of four modules:

1. Word representation module

In a nutshell, it's job is to represent the sentence with a vector. Each token w_i is transformed to a real-valued vector x_i by concatenating following.

a. Word embedding vector.(Used Glove embedding)
b. POS-tagging label embedding vector of w_i. This is generated by looking up the randomly initialized POS-tagging label embedding table.

c. The positional embedding vector of w_i. Similar to POS-tagging label vector it is generated by looking up the randomly initialized POS-tagging label embedding table.
d. The entity type label embedding vector of w_i. Similar to POS-tagging label vector it is generated by looking up the randomly initialized POS-tagging label embedding table.

2. Syntactic graph convolution network module

In a nutshell, it's job is to perform convolution operations by introducing shortcut arcs from syntactic structures. In syntactic graph each node representing token w_i. Each edge is directed syntactic arc from token w_i to token w_j , with the type label $K(w_i , w_j)$. Additionally, to allow information to flow against the direction, we also add reversed edge (v_j , v_i) with the type label $K' (w_i , w_j)$. Self loops can also be there. In the k^{th} layer the convolution vector is calculated as:

$$h_v^{(k+1)} = f(\sum_{u \in \mathcal{N}(v)} (W_{K(u,v)}^{(k)} h_u^{(k)} + b_{K(u,v)}^{(k)}))$$

K(u,v) is type label of edge between u and v. N(v) is neighbours of v. The definition of $K(w_i,w_j)$ is modified to:

$$K(w_i, w_j) = \begin{cases} along, & (v_i, v_j) \in \mathcal{E} \\ rev, & i! = j \& (v_j, v_i) \in \mathcal{E} \\ loop, & i == j \end{cases}$$

Since all types of edges are not equally informative gates were used to weight their individual importance. With these gating mechanism final syntactic GCN computation is formulated as:

$$h_v^{(k+1)} = f(\sum_{u \in \mathcal{N}(v)} g_{u,v}^{(k)} (W_{K(u,v)}^{(k)} h_u^{(k)} + b_{K(u,v)}^{(k)}))$$

Highway units were added to prevent information over-propagating if shortcut path between two triggers is less than k. They use Bi-LSTM to encode the word representation X(concatenation of forward LSTM and backward LSTM) which is then fed into initial layer of GCN.

3. Self-attention trigger classification module

In a nutshell, it's job is to capture the associations between multiple events in a sentence. Given a current token w_i, the self-attention score vector and context vector at position i are calculated as:

$$score = norm(exp(W_2 f(W_1 D + b_1) + b_2))$$

$$C_i = [\sum_{j=1, j!=i}^{n} score_j * D_j, D_i]$$

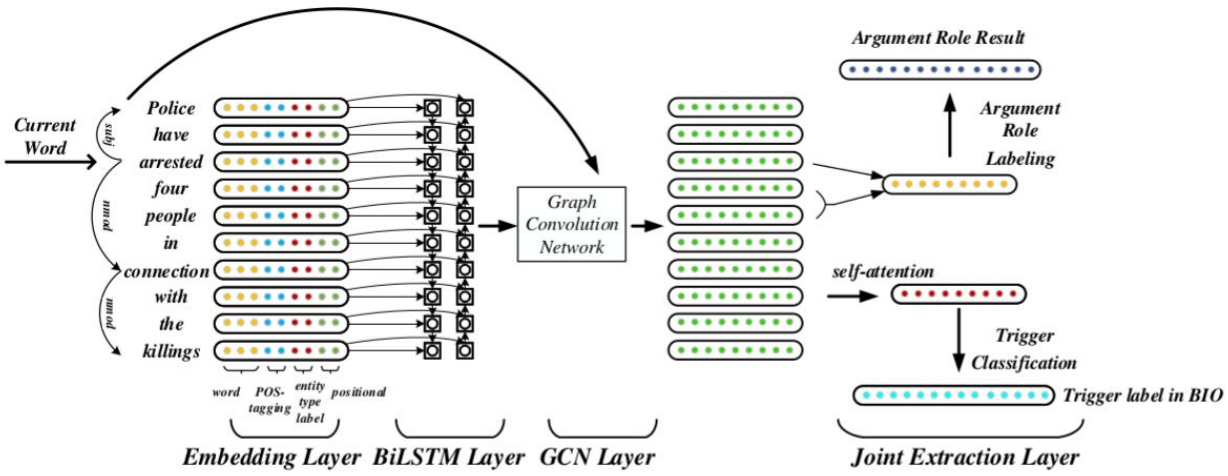
Where norm is normalization operation. Then we feed the context vector C i into a fully-connected network to predict the trigger label in BIO annotation schema using softmax.

4. Argument classification

In a nutshell, it's job is to predict the roles each entity mention e_j plays in the event candidates of specific types. For each entity-trigger pair, as both the entity and the trigger candidate are likely to be a subsequence of tokens, we aggregate the context vectors of subsequences to trigger candidate vector T_i and entity vector E_j by average pooling along the sequence length dimension. Then we concatenate them together and feed into a fully-connected network to predict the argument role using softmax.

Loss Function

In order to train the networks, they minimize the joint negative log-likelihood loss function. Due to the data sparsity in the ACE 2005 dataset, they adapt our joint negative log-likelihood loss function by adding a bias item.



(Architecture)

Dataset:

They evaluated JMEE framework on the ACE 2005 dataset. The ACE 2005 dataset annotate 33 event subtypes and 36 role classes, along with the NONE class and BIO annotation schema,

they classified each token into 67 categories in event detection and 37 categories in argument extraction. Stanford CoreNLP toolkit was used to pre-process the data.

Learning:
Though this paper talks about events extraction and not event coreference resolution, there is an important thing to be learnt from it. It uses graph convolutional network for learning and not traditional neural networks or RNN or CNN. Graph convolutional network is not much used for NLP tasks and thus reading about the same is indeed a new thing for us. Similarly the idea of using syntactic structure of the sentence and idea of bias loss is also new to us. Thus, this paper introduces a lot of new ideas to us.

2. Choubey and Huang, Improving Event Coreference Resolution by Modeling Correlations between Event Coreference Chains and Document Topic Structures, ACL 2018

This paper proposes a novel approach for event coreference resolution that models correlations between event coreference chains and document topical structures
through an Integer Linear Programming formulation(ILP formulation).
Compared to entities, coreferential event mentions are fewer in a document and much more sparsely scattered across sentences. Referring back to the same entity serves a different purpose than referring to the same event. Entities will be generally mentioned
when describing various events it was involved in, whereas most events only
appear once in a text, and there is less motivation to repeat them.

1. The same event is referred back only when a new aspect or further information of the event has to be described, and
2. Repetitions of the same events are mainly used for content organization purposes and, consequently, correlate well with topic structures.

This paper models four aspects of correlation

1. **Correlations between Main Event Chains and Topic Transition Sentences**:
   The main events usually have multiple coreferent event mentions that span over a large portion of the document. Topic transition sentences often overlap in content and can be identified by calculating sentence similarities.
2. **Correlations across Semantically Associated Event Chains:**
   Semantically associated events often co-occur in the same sentence. Therefore, coreference links are created between event mentions in sentences that contain other already known coreferent event mentions.
3. **Genre-specific Distributional Patterns:**
   Document level distributional patterns of coreferent event mentions that may be specific to a genre in ILP are modelled(In news articles, a majority of event coreference chains tend to be initiated in the early sections of the document. Inspired by this observation, the authors modified the objective function of ILP to

encourage more event coreference links in early sections of a document.)

4. **Subevents:**
   Subevents may share the same lexical form as the parent event and cause spurious event coreference links. It is observed that subevents referring to specific actions were seldom referred back in a document and are often singleton events.
   Therefore, constraints are specified in ILP to discourage coreference links between a specific action event and other event mentions.

Approach:

Baseline ILP system is defined over pairwise scores between event mentions obtained from a pairwise neural network-based coreference resolution classifier.

1. **The Local Pairwise Coreference Resolution Classifier**
   There are 3 inner layers and an output layer
   a. The first layer is a common layer with 347 neurons shared between two event mentions to generate embeddings corresponding to word lemmas (300) and parts-of-speech (POS) tags (47).
   b. The second layer contains 380 neurons to embed suffix and prefix of event words, distances (euclidean, absolute and cosine) between word embeddings of two event lemmas and common arguments between two event mentions.
   c. The output from the second layer is concatenated and fed into the third neural layer with 10 neurons.
   d. The output embedding from the third layer is finally fed into an output layer with 1 neuron that generates a score indicating the confidence of assigning the given event pair to the same coreference cluster.
2. **The ILP model for Event Coreference Resolution**
   ILP inference for coreference resolution was performed by optimizing a global objective function(Θ) where
   $\Theta = K_B \Theta_B + K_T \Theta_T + K_G \Theta_G + K_C \Theta_C + K_D \Theta_D + K_S \Theta_S$
   $K_B = K_T = 1.0$ and $K_C = K_G = 0.5$ and $K_D = 2.5$
   $K_S = 10.0$
   K_G (K_C) and K_D through 2D grid search in range [0, 5.0] at an interval of 0.5 on a held out training data.
3. **Basic ILP**

$$\Theta_B = \sum_{i,j \in \Lambda} -log(p_{ij})x_{ij} - log(1 - p_{ij})(\neg x_{ij}) \quad (1)$$
$$s.t.\ x_{ij} \in \{0, 1\}$$

$$\neg x_{ij} + \neg x_{jk} \geq \neg x_{ik} \quad (2)$$

λ represents the set of all event mentions in a document, Λ denotes the set of all event mention pairs. $p_{ij} = p_{cls}$ (coref | i, j)

4. **Correlation between Main Event Chains and Topic Transition Sentences**

$$\Theta_T = \sum_{m,n \in \Omega} -log(s_{mn}) w_{mn} - log(1 - s_{mn})(\neg w_{mn})$$

$$s.t.\ w_{mn} \in \{0, 1\}$$
$$(n - m) \geq |S|/\theta_s \quad (3)$$

$$\sum_{i' \in \xi_m, j' \in \xi_n} x_{i'j'} \geq w_{mn} \quad (4)$$

ω represents the set of sentences in a document and Ω denotes the set of sentence pairs. $s_{ij} = p_{sim}$ (simscore|m, n),

5. **Identifying Topic Transition Sentences Using Sentence Similarities**
First, a weighted average of words' embeddings in a sentence is computed, where the weight of a word w is given by a/(a + p(w)). Here, p(w) represents the estimated word frequency obtained from English Wikipedia and a is a small constant (1e-5). Then the first principal component of averaged word embeddings corresponding to sentences in a document is computed and the projection on the first principal component is removed from each averaged word embedding for each sentence to get the sentence embedding. Then, similarity between two sentences is computed as cosine similarity between their embeddings.

6. **Constraints for Avoiding Fragmented Partial Event Chains**
The above equations (3-4) consider a pair of sentences and encourage two coreferent event mentions to appear in a pair of topic transition sentences. But the local nature of these constraints can lead to fragmented main event chains. Therefore, the main event chains are encouraged to have a large number of coreferential mentions and a long stretch, to avoid creating partial chains.

In equation 7, variable σ_{ij} is used to identify main event chains as those chains which are extended to at least 75% of the document. M is a large positive number and y_{ij} represents a slack variable that takes the value 0 if the event chain represented by σ_{ij} is a global chain.

$$\Theta_G = -\sum_{i,j\in\mu} \gamma_{ij} \tag{5}$$

$$\sigma_{ij} = \sum_{k<i} \neg x_{ki} \wedge \sum_{j<l} \neg x_{jl} \wedge x_{ij} \qquad \sigma_{ij} \in \{0,1\} \tag{6}$$

$$\begin{aligned} \Gamma_i &= \sum_{k,i\in\Lambda} x_{ki} + \sum_{i,j\in\Lambda} x_{ij} \\ M(1-y_{ij}) &\geq (\varphi[j]-\varphi[i]).\sigma_{ij} - \lceil 0.75\ (|S|)\rceil \\ \gamma_{ij} - \Gamma_i - \Gamma_j &\geq M.y_{ij} \\ \Gamma_i, \Gamma_j, \gamma_{ij} \in Z;\ \Gamma_i, \Gamma_j, \gamma_{ij} &\geq 0;\ y_{ij} \in \{0,1\} \end{aligned} \tag{7}$$

7. Cross-chain Inferences

$$\Theta_C = -\sum_{m,n\in\Omega} \Phi_{mn} \tag{8}$$

$$\Phi_{mn} = \sum_{i\in\xi_m, j\in\xi_n} x_{ij} \qquad |\xi_m| > 1;\ |\xi_n| > 1;\ \Phi_{mn} \in Z;\ \Phi_{mn} \geq 0 \tag{9}$$

φ_{mn} equals the number of coreferent event pairs in a sentence pair, with each sentence having more than one event mention.ξ_m represents the event mentions in sentence m.

8. Segment-wise Distributional Patterns

$$\begin{aligned} \Theta_D = -\sum_{i\in\xi_m, j\in\xi_n} x_{ij} + \sum_{k\in\xi_p, l\in\xi_q} x_{kl} \\ s.t.\ m,n < \lfloor\alpha|S|\rfloor;\ p,q > \lceil\beta|S|\rceil \\ \alpha \in [0,1];\ \beta \in [0,1] \end{aligned} \tag{10}$$

For the event pairs that belong to the first α (or the last β) sentences in a document, negative (positive) sum of their indicator variables (x) is added to the objective function Θ_D .

9. **Restraining Subevents from being included in Coreference Chains**

$$\Theta_S = \sum_{s \in S} \Gamma_s \qquad (11)$$

where S represents the set of subevents in a document and Γ_s equals the number of mentions that are coreferent to s.

Team Members(Group-4)

Pulle Sri Satya Sravya (20161079)
Himanshu Maheshwari (20171033)
Rushitkumar Jasani (2018201034)
Nitish Kumar Dwivedi (2018201068)