CS626: Information Extraction

Week of 20th August

Information Extraction (IE)

20-SEPT-2021

Sachin Pawar, TCS Research
(sachin7.p@tcs.com)

Outline

- Information Extraction
- Entity Extraction
  - Named Entity Recognition (NER)
  - Techniques: Conditional Random Fields (CRF), Long Short-Term Memory Networks (LSTM)
- Entity Extraction using Python libraries
- Other Tasks in Information Extraction
  - Relation Extraction
  - Co-reference Resolution

Information Extraction: Introduction

- A lot of information is hidden in unstructured form in text documents
- Goal of Information Extraction (IE): To convert unstructured textual information to some structured form
  - CSV, XML, Relational databases
- Motivation: Structured information can be easily stored, indexed, searched, analyzed to derive more insights
- Applications: Fine-grained Information Retrieval, Question Answering, Summarization, Knowledge graph creation

Entity Types and Mentions

- The most basic units of information are entities
- An entity is an object or a set of objects in the real-world
- General entity types: PERSON, ORGANIZATION, LOCATION
- Domain-specific entity types:
  - **Biomedical**: PROTEIN, DRUG, DISEASE, CELL_TYPE
  - **Resumes**: EMPLOYER, DEGREE, EDUCATIONAL_INSTITUTE, DESIGNATION
  - **Legal**: APPELANT, RESPONDENT, LAWYER, WITENESS, WEAPON
- Entity mentions: Entities are referenced in text through entity mentions

Entity Mentions

- Entity mentions are generally of 3 types
  - **Named mentions**: Names of people, organizations; often expressed through proper nouns
    - `Sachin Tendulkar, Reliance Industries Ltd., New Delhi`
  - **Nominal mentions**: Entity mentions expressed through common nouns
    - `batsman, company, city`
  - **Pronoun mentions**: Entity mentions expressed through pronouns
    - `he, they, her, it`
- Named Entity Recognition (NER) task generally focuses on named mentions
  - Although, the techniques are fairly general to be applicable to all entity mention types

[Russia]$_{GPE}$ produces first batch of covid vaccines, to be given to doctors. [Russia]$_{GPE}$ on Saturday said it has produced the first batch of its corona virus vaccine, just days after President [Vladimir Putin]$_{PER}$ announced it had been first in the world to approve a vaccine. "The first batch of the novel corona virus vaccine developed by the [Gamaleya research institute]$_{ORG}$ has been produced," the health ministry said in a statement quoted by [Russian]$_{GPE}$ news agencies. [Russia]$_{GPE}$ has named the covid vaccine "Sputnik V" after the Soviet-era satellite that was the first launched into space in 1957. On Tuesday, [Putin]$_{PER}$ announced that [Russia]$_{GPE}$ has approved a vaccine against corona virus even though clinical trials were not yet complete. He said that the vaccine was safe and one of his daughters had been inoculated.

[Russia]$_{GPE}$ produces first batch of covid vaccines, to be given to [doctors]$_{PER}$. [Russia]$_{GPE}$ on Saturday said [it]$_{GPE}$ has produced the first batch of [its]$_{GPE}$ coronavirus vaccine, just days after [President]$_{PER}$ [Vladimir Putin]$_{PER}$ announced [it]$_{GPE}$ had been first in the world to approve a vaccine. "The first batch of the novel coronavirus vaccine developed by the [Gamaleya research institute]$_{ORG}$ has been produced," the [health ministry]$_{ORG}$ said in a statement quoted by [Russian]$_{GPE}$ [news agencies]$_{ORG}$. [Russia]$_{GPE}$ has named the covid vaccine "Sputnik V" after the Soviet-era satellite that was the first launched into space in 1957. On Tuesday, [Putin]$_{PER}$ announced that [Russia]$_{GPE}$ has approved a vaccine against coroanvirus even though clinical trials were not yet complete. [He]$_{PER}$ said that the vaccine was safe and one of [his]$_{PER}$ [daughters]$_{PER}$ had been inoculated.

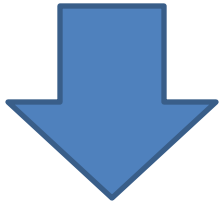
Effects of dehydration on endocrine regulation of the electrolyte and fluid balance and **[atrial natriuretic peptide]**$_{\text{Protein}}$ - binding sites in perinatally malnourished adult male rats. OBJECTIVE: The first aim of this work was to investigate, under basal conditions in adult male rats, the long - term consequences of perinatal maternal food restriction on the plasma concentrations of **[vasopressin]**$_{\text{Protein}}$ (**[VP]**$_{\text{Protein}}$), aldosterone and **[atrial natriuretic peptide]**$_{\text{Protein}}$ (**[ANP]**$_{\text{Protein}}$) and on plasma **[renin]**$_{\text{Protein}}$ activity (PRA). Furthermore, under these same conditions, the hypothalamic **[VP]**$_{\text{Protein}}$ gene expression as well as the density (B(max)), affinity (K(d)) and gene expression of **[ANP receptors]**$_{\text{ProteinFamilyOrGroup}}$ were determined in kidneys and adrenals.

Entity Extraction - Solution

- The task of Entity Extraction is to identity all the entity mentions in a given sentence, along with entity type for each mention
- Several techniques have been proposed for solving this task
  - **Rule-based techniques** – hand crafted lexical, syntactic or semantic rules
  - **Unsupervised or semi-supervised** - learning of gazetteers
  - **Supervised** – features-based, deep learning
- Most common supervised techniques model the entity extraction as a sequence labeling task.
  - An appropriate label is assigned to each word in a sentence so as to identify all the named entity mentions in that sentence
  - Label for each word should model – (i) whether the word is part of any entity mention or not, and (ii) entity type of the entity mention

Entity Extraction - Example

Air$_{1}$ India$_{2}$ women$_{3}$ pilots$_{4}$ to$_{5}$ fly$_{6}$ over$_{7}$ North$_{8}$ Pole$_{9}$ on$_{10}$ world$_{11}$'s$_{12}$ longest$_{13}$ air$_{14}$ route$_{15}$ from$_{16}$ Bengaluru$_{17}$ to$_{18}$ San$_{19}$ Fransisco$_{20}$.$_{21}$



Air India, ORG, words: 1-2
North Pole, LOC, words: 8-9
Bengaluru, GPE, words: 17
San Fransisco, GPE, words: 19-20

Multiple Label Encoding Strategies (1/2)

- IO : **I**nside / **O**utside
  - **Air/I-ORG India/I-ORG** women/O pilots/O to/O fly/O over/O **North/I-LOC Pole/I-LOC** on/O world/O 's/O longest/O air/O route/O from/O **Bengaluru/I-GPE** to/O **San/I-GPE Francisco/I-GPE** ./O

- BIO : **B**egin / **I**nside / **O**utside
  - **Air/B-ORG India/I-ORG** women/O pilots/O to/O fly/O over/O **North/B-LOC Pole/I-LOC** on/O world/O 's/O longest/O air/O route/O from/O **Bengaluru/B-GPE** to/O **San/B-GPE Francisco/I-GPE** ./O

Multiple Label Encoding Strategies (2/2)

- BILOU : **B**egin / **I**nside / **L**ast / **O**utside / **U**nit
  - **Air/B-ORG India/L-ORG** women/O pilots/O to/O fly/O over/O **North/B-LOC Pole/L-LOC** on/O world/O 's/O longest/O air/O route/O from/O **Bengaluru/U-GPE** to/O **San/B-GPE Francisco/L-GPE** ./O

  - **Tata/B-ORG Consultancy/I-ORG Services/L-ORG** , /O **India/U-GPE** ’s/O largest/O IT/O services/O firm/O posted/O its/O strongest/O third-quarter/O result/O in/O nine/O years/O . /O

Background: Multinomial Logistic Regression (Maximum Entropy classifier)

Point classification vs Sequence classification

- Point Classification: Predicting a category (class label) y for a single instance x
  - Classifying an email into – SPAM or NOT-SPAM
  - Classifying a news article into – BUSINESS, SPORTS, ENTERTAINMENT, POLITICAL

- Sequence Classification: Predicting a sequence of class labels $y_1, y_2, \cdots, y_n$ for a sequence of instances $x_1, x_2, \cdots, x_n$
  - POS Tagging: predicting a sequence of POS tags for a sentence which is a sequence of words
  - Hidden Markov Model (HMM) is used for this sequence classification
  - HMM computes probability distribution of possible sequences of POS tags
  - The sequence of POS tags which has the highest probability is chosen as the final prediction

Generative vs Discriminative Classifiers (1/2)

- Point Classification: Predicting a category (class label) y for a single instance x
  - Two frameworks of building Machine Learning models: Generative and Discriminative
- Generative models: Try to model the joint probability of an instance x and a class label y
  - Naïve Bayes classifier, a generative classifier predicts a class label as follows:

$$y^* = \underset{y}{\operatorname{argmax}}\, p(x, y) = \underset{y}{\operatorname{argmax}}\, p(x|y) \cdot p(y)$$

  - Naïve conditional independence: x characterized by multiple "features" such that, given these features are independent of each other given the class label

$$x = [x_1, x_2, \cdots, x_n]$$

  - **HMM is a s**
  - **HMM is als**

$$p(x|y) = p(x_1|y) \cdot p(x_2|y) \cdots p(x_n|y) = \prod_{i=1}^{n} p(x_i|y)$$

Generative vs Discriminative Classifiers (2/2)

- Discriminative models: Try to model the conditional probability of a class label y conditioned on an given instance x
  - A discriminative classifier predicts a class label as follows:

$$y^* = \underset{y}{\operatorname{argmax}}\, p(y|x)$$

- **Multinomial Logistic Regression (Maximum Entropy classifier / MaxEnt classifier) is a discriminative classifier**
  - Any instance is characterized by a set of feature functions.
  - A feature function is a function of an instance x as well as a candidate class label $y \in C$
  - A weight is associated with each feature, indicating a positive or negative vote for class label

$$p(y|x) = \frac{\exp \sum_i w_{iy} \cdot f_i(x, y)}{\sum_{z \in C} \exp \sum_i w_{iz} \cdot f_i(x, z)}$$

Features (1/2)

- Features functions are designed to capture important characteristics of an instance x which are potentially useful for predicting its class label y

E.g., we want to build a classifier to predict the POS tag of a word in a sentence

- Here, a word is an instance x and the corresponding POS tag is y
- $f_1(x, y)$ = 1 if *x* ends in the suffix "`ing`" and *y* = VBG
- $f_2(x, y)$ = 1 if *x* begins with a capital letter and *y* = NNP
- $f_3(x, y)$ = 1 if *x* is preceded by the word "`to`" and *y* = VB
- $f_4(x, y)$ = 1 if *x* is preceded by the word "`the`" and is succeeded by the word; the suffix "`ness`" and *y* = JJ
- $f_5(x, y)$ = 1 if *x* is a first word in the sentence and *y* = NN
- $f_6(x, y)$ = 1 if *x* is the word "`back`" and *y* = RB

Features (2/2)

- A different weight is learned for each combination of a Feature Function and a class label
  - Intuitively, a positive weight can be thought of as a vote for the class label
  - A negative weight can be thought of a vote against for the class label
  - Magnitude of the weight (positive or negative) corresponds to "discriminatory" power of the features as learned from the training data
- E.g., consider the feature function which is 1 if *x* is preceded by the word "`to`" and *y* corresponds to certain POS tag
  - Weight corresponding to $f_3(x, y)$
  - Weight corresponding to $f_3(x, VB)$ = *1.2*
  - Weight corresponding to $f_3(x, NNP)$ = *0.8*
  - Weight corresponding to $f_3(x, NN)$ *-4.5*
    $f_3(x, IN)$

Multinomial Logistic Regression: Training

- Consider a set of labelled (annotated) training instances

$$D = \{\langle x^1, y^1\rangle, \langle x^2, y^2\rangle, \cdots \langle x^N, y^N\rangle\}$$

- Goal: To choose the parameters (w_{iy}'s) which maximize the conditional likelihood of the training instances

$$LL(D) = \log \prod_{j=1}^{N} p(y^j|x^j)$$

$$= \log \prod_{j=1}^{N} \frac{\exp \sum_i w_{iy^j} \cdot f_i(x^j, y^j)}{\sum_{z \in C} \exp \sum_i w_{iz} \cdot f_i(x^j, z)}$$

$$= \underbrace{\sum_{j=1}^{N} \left(\sum_i w_{iy^j} \cdot f_i(x^j, y^j) \right)}_{= Nr(w)} - \underbrace{\sum_{j=1}^{N} \log \left(\sum_{z \in C} \exp \sum_i w_{iz} \cdot f_i(x^j, z) \right)}_{Dn(w)}$$

Derivative of the Numerator

$$Nr(w) = \sum_{j=1}^{N} \left(\sum_{i} w_{iy^j} \cdot f_i(x^j, y^j) \right)$$

- Differentiating with respect to w_{iy}

$$\frac{\partial Nr(w)}{\partial w_{iy}} = \sum_{j=1}^{N} 1\{y = y^j\} \cdot f_i(x^j, y^j)$$

- This is equal to the number of instances in the training data for which the i^{th} feature function is evaluated to 1 having the true class label as y

Derivative of the Denominator

$$Dn(w) = \sum_{j=1}^{N} \log\left(\sum_{z \in C} \exp \sum_{i} w_{iz} \cdot f_i(x^j, z)\right)$$

- Differentiating w.r.t. w_{iy}

$$\frac{\partial Dn(w)}{\partial w_{iy}} = \sum_{j=1}^{N} \frac{\exp\left(\sum_i w_{iy} \cdot f_i(x^j, y)\right) \cdot f_i(x^j, y)}{\sum_{z \in C} \exp \sum_i w_{iz} \cdot f_i(x^j, z)}$$

- This is equal to the model ssociated with the class label as per

$$\frac{\partial Dn(w)}{\partial w_{iy}} = \sum_{j=1}^{N} f_i(x^j, y) \cdot p(y|x^j)$$

y

Overall derivative of the log likelihood

- After differentiating with respect to w_{iy}

$$\frac{\partial LL(D)}{\partial w_{iy}} = Empirical\ count\ of\ f_i(x,y)\ -Predicted\ count\ of\ f_i(x,y)$$

- The optimal weight parameters are those for which each feature's predicted expectation is equal to its empirical expectation
- These models are also called Maximum Entropy models because the predicted class distribution is the most uniform one, given the constraints on expected values of the features.
- Any numerical optimization package can be used to find the optimal weight parameters, using the gradient computed above.
  - Gradient descent, Stochastic gradient descent (SGD), L-BFGS

Multinomial Logistic Regression: Summary

- A discriminative classifier – also known as Maximum Entropy classifier

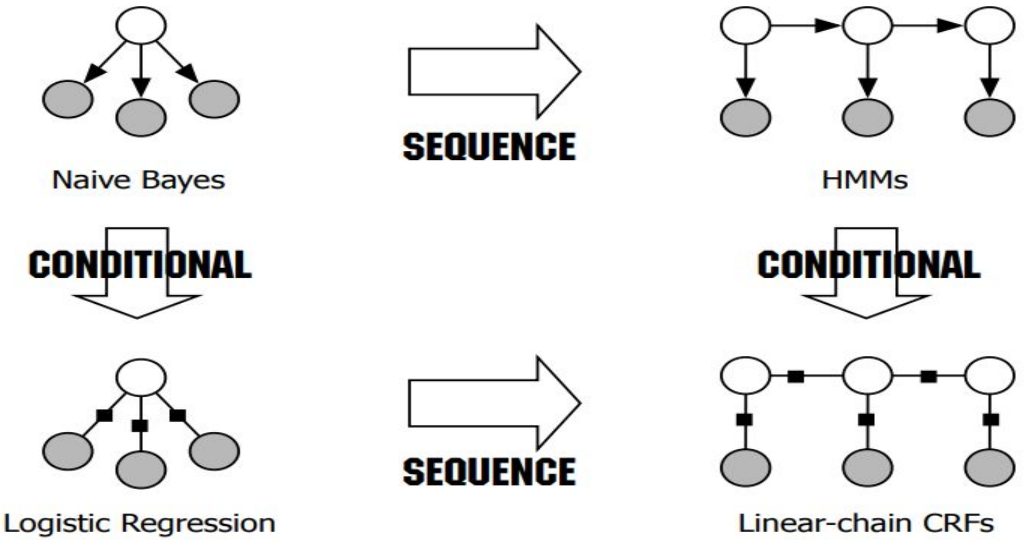
- Advantages over generative classifiers such as Naïve Bayes:
  - Several overlapping features can be designed
  - Handles multiple correlated features well – does not double-count evidences from such features

- Interpretability of the learned model:
  - In real life applications, more than just the correct class label prediction, we also need to know why the classifier has predicted a particular class label
  - Features in Multinomial Logistic Regression are often human-designed
  - Investigating weights associated with features is helpful for understanding why a certain prediction was made

Conditional Random Fields (CRF)

Introduction (1/2)

- Multinomial Logistic Regression is a **point classifier**
- **Sequence labelling** extension of Multinomial Logistic Regression
  - Maximum Entropy Markov Models (MEMM)
  - Conditional Random Fields (CRF)



CRF Tutorial by Sutton and McCullum: https://people.cs.umass.edu/~mccallum/papers/crf-tutorial.pdf

Introduction (2/2)

- Formally, Conditional Random Fields (CRF) are undirected graphical models which model probability distributions of the form $p(\mathbf{y}|\mathbf{x})$ where -
  - $\mathbf{y}$ is a finite set of output variables with arbitrary dependence structure $\mathbf{y} = \{y_1, y_2, \cdots, y_n\}$
  - $\mathbf{x}$ is a finite set of observed variables on which $\mathbf{y}$ is conditioned on $\mathbf{x} = \{x_1, x_2, \cdots, x_m\}$

- Conditional probability modelled by a general CRF model can be expressed as:

$$p(\mathbf{y}|\mathbf{x}) = p(y_1, y_2, \cdots, y_n | x_1, x_2, \cdots, x_m)$$
$$= \frac{p(y_1, y_2, \cdots, y_n, x_1, x_2, \cdots, x_m)}{p(x_1, x_2, \cdots, x_m)}$$

- Linear-chain CRFs are generally useful for NLP tasks, which are a specialization of general CRFs, where the structure of $\mathbf{y}$ is a linear chain

Joint Probability Distribution for Undirected Graphical Models

- A joint probability distribution expressed by the undirected graphical models is decomposed as a product of potential functions
- Each potential function corresponds to a clique in the underlying graph
- The conditional distribution can then be represented as,

$$p(\mathbf{y}|\mathbf{x}) = \frac{\prod_{c \in C} \varphi_c(\mathbf{x}_c, \mathbf{y}_c)}{\sum_{\mathbf{y}'} \prod_{c \in C} \varphi_c(\mathbf{x}_c, \mathbf{y}'_c)}$$

  - Where, *C* is the set of cliques in the underlying graph
  - is the potential functi
  - are the subset of
  - The denominator is only

φ_c

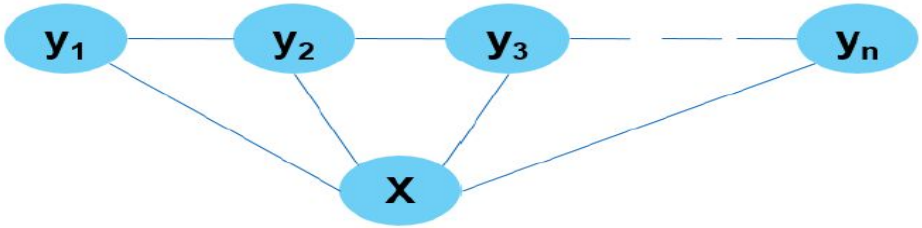
$\mathbf{x}_c, \mathbf{y}_c$

$\mathbf{x}$

$Z(\mathbf{x})$

Probability Distribution for Linear Chain CRFs (1/2)

- Linear chain CRFs: The graph structure of output variables is a simple chain



- Any clique in this graph will always involve two consecutive output variables and all the input variables

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=2}^{n} \varphi_t(y_t, y_{t-1}, \mathbf{x})$$

Probability Distribution for Linear Chain CRFs (2/2)

- As a sequence extension for the Maximum Entropy classifier, the potential functions can be represented using the log-linear form based on feature functions and their corresponding weights

- Hence, th

$$\varphi_t(y_t, y_{t-1}, \mathbf{x}) = \exp\left(\sum_i w_{i,y_t,y_{t-1}} \cdot f_i(y_t, y_{t-1}, \mathbf{x})\right)$$

where

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=2}^{n} \sum_i w_{i,y_t,y_{t-1}} \cdot f_i(y_t, y_{t-1}, \mathbf{x})\right)$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}'} \exp\left(\sum_{t=2}^{n} \sum_i w_{i,y'_t,y'_{t-1}} \cdot f_i(y'_t, y'_{t-1}, \mathbf{x})\right)$$

Examples of feature functions

- $f_1(y_t, y_{t-1}, \mathbf{x}) = 1$ if the word at t^{th} position (i.e. x_t) is `an` and the label at t^{th} position $y_t = DT$ and the label at $(t-1)^{th}$ position $y_{t-1} = VBD$
- $f_2(y_t, y_{t-1}, \mathbf{x}) = 1$ if the word x_t starts with a capital letter and the labels are such that $y_t = NNP$ and $y_{t-1} = NNP$
- $f_3(y_t, y_{t-1}, \mathbf{x}) = 1$ if the word x_{t-1} starts with a capital letter and the word x_t and the labels $y_t = VBZ$ $y_{t-1} = NNP$
- $f_4(y_t, y_{t-1}, \mathbf{x}) = 1$ if the word x_t has a suffix `s` and the word x_{t+1} is `the` and $y_t = VBZ$ and $y_{t-1} = NN$

Linear-chain CRFs: Training and Inference

- Training:
  - Training data $D = \{\langle \mathbf{x}^1, \mathbf{y}^1 \rangle, \langle \mathbf{x}^2, \mathbf{y}^2 \rangle, \cdots \langle \mathbf{x}^N, \mathbf{y}^N \rangle\}$ where for each i^{th} sequence of observed variables $\mathbf{x}^i = \{x_1^i, x_2^i, \cdots, x_{n_i}^i\}$ with the corresponding sequence of output variables $\mathbf{y}^i = \{y_1^i, y_2^i, \cdots, y_{n_i}^i\}$
  - **Goal:** Similar to Logistic Regression, choose the model parameters which maximize the likelihood of the training data
  - Techniques: Gradient descent, Stochastic gradient descent (SGD), L-BFGS
- Inference:
  - **Goal:** Given a sequence of observed variables, choose the most probable sequence of output variables
  - **The Viterbi Algorithm**, which is used in HMMs, can also be used for Linear-chain CRFs with little revision.

CRF for Entity Extraction

Entity Extraction using CRF

- CRF for indentifying entity mentions in a sentence
  - Sequence labeling problem similar to POS tagging
  - For each word, an appropriate label is predicted, as per the label encoding strategy
  - **Training** – A manually annotated dataset of sentences is required for training
    - Each word in each sentence of this dataset is annotated with a gold-standard label
    - The training process learns an optimal weight for each feature
  - **Inference** – Given any new sentence, the features are generated for it and using the learned feature weights, the most probable label sequence is predicted
- Feature engineering:
  - Each word is represented using a set of features
  - Each feature is supposed to capture a certain characteristic of the word w.r.t. its entity label
  - Various types of features – lexical, semantics, gazetteer-based

Examples of Features

- **Word** itself, next word, previous word, previous to previous word
- **Word structure** based features like whether the word consists of all capital characters, whether the word begins with a capital character, whether the word contains any special character or digits
- **POS tag** of the current word, POS tags of the previous and next words
- **WordNet** category of the word
- **Gazetteer-based features** – whether the word is a first word of any entry in the gazetteer, whether the word is part of any entry in the gazetteer etc.
- **Sentence or document position** – whether the word is first or last word in a sentence, whether the word is in the beginning of the document etc.

Examples of Features

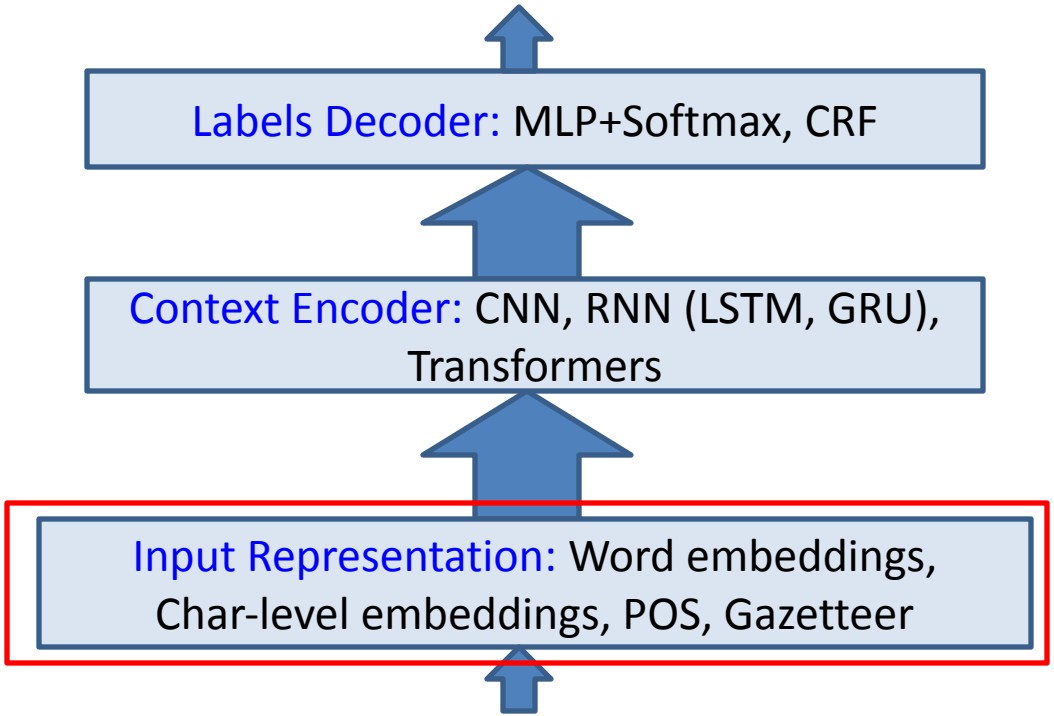
```
Air/B-ORG India/I-ORG women/O pilots/O to/O fly/O over/O North/B-LOC
Pole/I-LOC on/O world/O 's/O longest/O air/O route/O from/O
Bengaluru/B-GPE to/O San/B-GPE Francisco/I-GPE ./O
```

- $f_1(y_t, y_{t-1}, \mathbf{x}) = 1$ if word at t^{th} position is **Pole** and y_t is I-LOC and y_{t-1} is
- if word at position is **North** and is I-LOC and is B-LOC
- $f_2(y_t, y_{t-1}, \mathbf{x}) = 1$ if POS tag of the word at $(t-1)^{th}$ position is NNP and y_t is I-LOC and y_{t-1} is B-LOC
- $f_3(y_t, y_{t-1}, \mathbf{x}) = 1$ if POS tag of the word at t^{th} position is NNP and y_t is I-LOC and y_{t-1} position is **on** and is B-LOC and is O

$f_4(y_t, y_{t-1}, \mathbf{x}) = 1$ t^{th} $(t+1)^{th}$ y_t y_{t-1}

Entity Extraction using Deep Learning

Barack/B-PER Obama/I-PER was/O born/O in/O **Hawaii/B-GPE** ./O



Li et al., A Survey on Deep Learning for Named Entity Recognition, IEEE TKDE 2020

Word embeddings

- Continuous real-valued vector representations for words
- Pre-trained on a huge corpus for language modeling task
- Semantically similar words have “similar” vectors

```
print(embedding_weights[word_to_index['paris']])
```

```
tensor([ 0.9261, -0.2282, -0.2552,  0.7400,  0.5007,  0.2642,  0.4006,  0.5615,
         0.1791,  0.0505,  0.0241, -0.0648, -0.2549,  0.2966, -0.4760,  0.2424,
        -0.0670, -0.4603, -0.3841, -0.3654,  0.1575,  0.2112,  0.0182, -0.4206,
         0.8261, -1.2281, -0.1881, -1.1329,  0.9173, -0.4774, -0.7841, -0.8603,
        -0.3982, -0.1561, -0.5067,  1.0742,  0.9398, -0.0912, -0.7997,  0.1381,
        -0.1492, -0.6278, -0.2552, -0.5254,  0.2368,  0.1814, -0.5929, -0.0813,
         0.5192, -0.1956,  0.0715,  0.5752,  0.4363,  0.5570, -1.4944, -2.6694,
        -0.4988,  0.5934,  0.6980,  0.1371, -0.2494,  0.3421, -0.9035, -0.5169,
        -0.2329, -0.0410, -0.9352,  0.1660, -0.0390,  0.4233, -0.7212, -0.7250,
        -1.0439, -0.4638,  0.0822, -0.5714,  1.2457, -0.3014, -0.9908, -0.2526,
         0.5288,  0.1419, -0.0370,  0.0377, -1.0728, -0.4354, -0.3792,  0.2541,
        -0.4297, -0.4041,  0.1598, -0.5357,  0.9804,  0.6460, -0.7709,  1.1849,
        -0.2360, -0.4099,  0.3555,  0.0970])
```

Word embeddings

```
import torch
cos = torch.nn.CosineSimilarity(dim=0)
s1 = cos(embedding_weights[word_to_index['paris']], embedding_weights[word_to_index['london']])
s2 = cos(embedding_weights[word_to_index['paris']], embedding_weights[word_to_index['author']])
print(s1,s2)
```

```
tensor(0.7338) tensor(0.3109)
```

```
import torch
cos = torch.nn.CosineSimilarity(dim=0)
s1 = cos(embedding_weights[word_to_index['attacked']], embedding_weights[word_to_index['assaulted']])
s2 = cos(embedding_weights[word_to_index['attacked']], embedding_weights[word_to_index['jumped']])
s3 = cos(embedding_weights[word_to_index['attacked']], embedding_weights[word_to_index['enjoyed']])
print(s1, s2, s3)
```

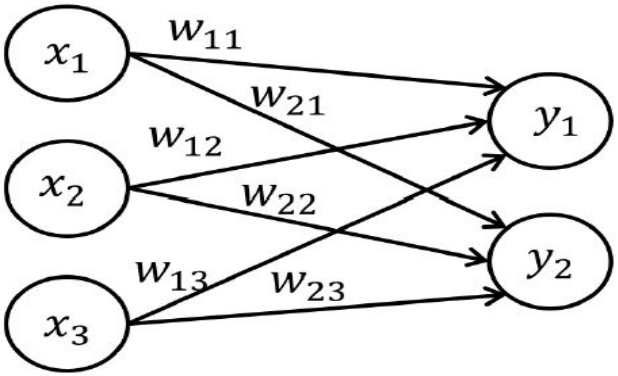
```
tensor(0.6909) tensor(0.3306) tensor(0.1977)
```

Context Encoder

- The job of context encoder is to capture context dependencies
  - Label for the current word may depend on the surrounding words and labels
  - E.g., `The flight was diverted to Entebbe.` Even if you haven't heard of `Entebbe`, you can guess it's some location due to its context in the sentence
- Several options:
  - Convolutional Neural Networks (CNN)
  - Recurrent Neural Networks (RNN)
  - Transformers (e.g., BERT)
- Most popularly used context encoders RNN
  - Long Short-Term Memory Networks (LSTM)
  - Gated Recurrent Units (GRU)

Background: LSTM

Basic NN : Multi-layer Perceptron



Compactly
f is any ac

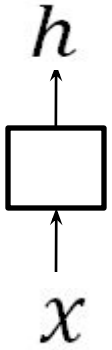
$$y_1 = f(w_{11}x_1 + w_{12}x_2 + \cdots + w_{1n}x_n + b_1)$$

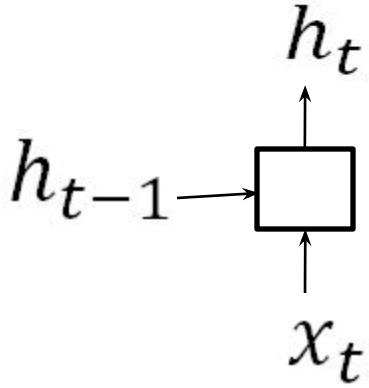
$$y_2 = f(w_{21}x_1 + w_{22}x_2 + \cdots + w_{2n}x_n + b_2)$$

$$\boldsymbol{y} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

$$S(x) = \frac{1}{1 + e^{-x}}$$

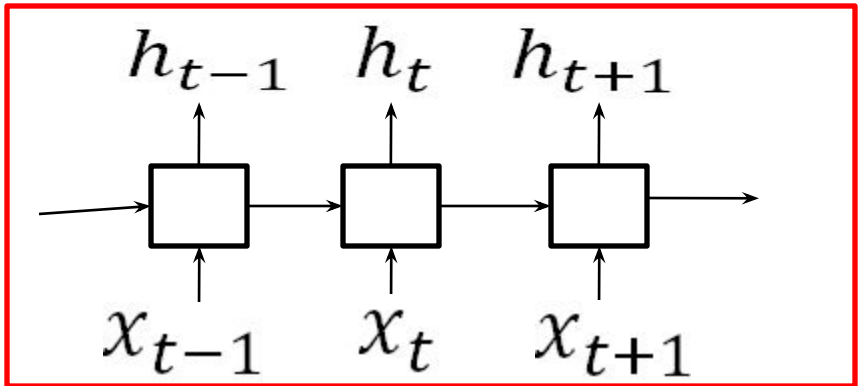
Basic NN vs Recurrent NN





$$h = f(Wx + b)$$

$$h_t = f(W[x_t; h_{t-1}] + b)$$



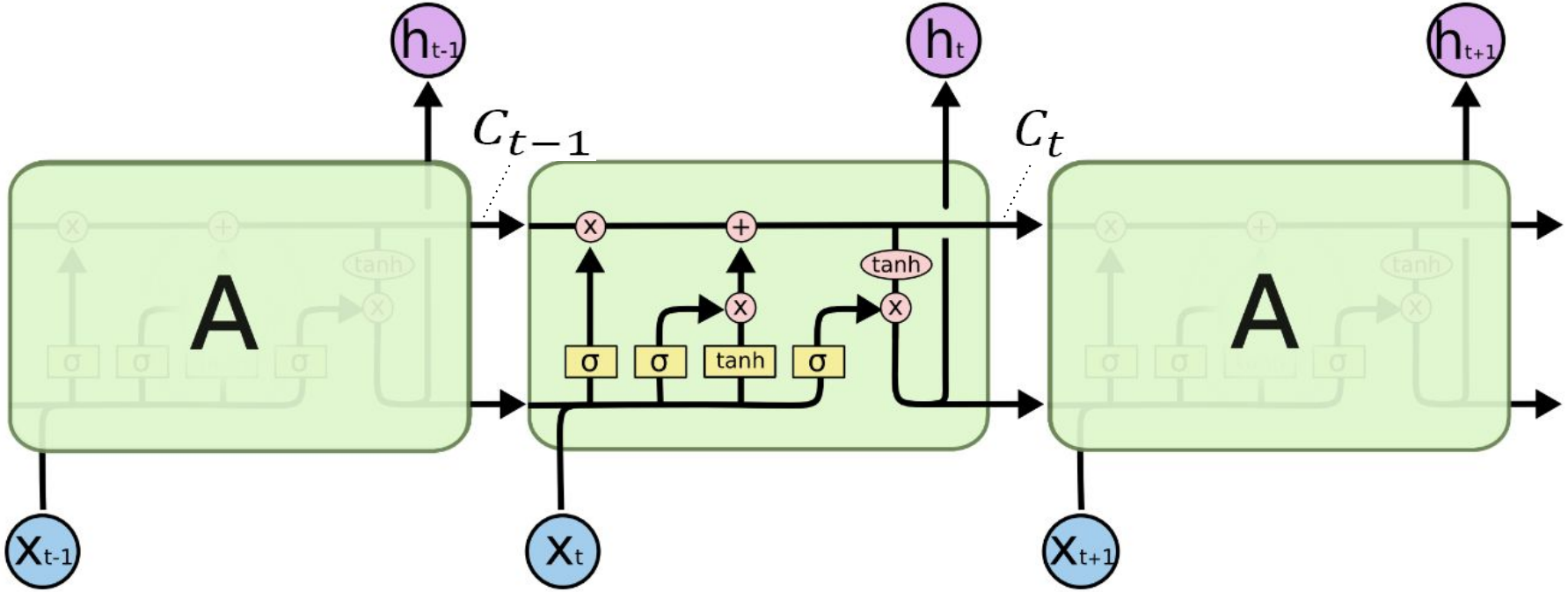
LSTM – A special type of RNN

- In theory, RNNs are capable of learning long range dependencies among the input symbols separated by large time gap
- In practice, RNNs don't seem to capture such dependencies well
  - Exploding and diminishing gradient problems

- Long Short-Term Memory Networks (LSTM) are explicitly designed to avoid long term memory problem
- LSTMs follow the same chain-like structure of basic RNN
  - But the repeating module has a more complex structure
  - Instead of having a single neural network layer, there are four layers interacting in a very special way.

Colah's Blog- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM – Structure



Colah's Blog- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM – Structure

- The key idea behind LSTMs is the cell state
  - Acts intuitively as memory of the network where vital information about the past time steps is stored
- Each repeating unit consists of multiple structures referred as gates
  - Gates are composed of a neural network layer followed by a pointwise multiplication operation

- Forget Gate - decides what information is to be erased from the cell state, by looking at the current input and the previous hidden state

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f)$$

Colah's Blog- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM – Structure

- A candidate cell state at the current time step is generated depending on the current input and the previous hidden state

$$\boldsymbol{C}'_t = tanh(\boldsymbol{W}_C[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_C)$$

- Input Gate - decides which values in the cell state are to be updated, by looking at the current input and previous hidden state.

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i)$$

Colah's Blog- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM – Structure

- To get new cell state:
  - Previous cell state is pointwise multiplied by f_t to forget certain values
  - The candidate cell state is pointwise multiplied by i_t to retain only certain values

$$C_t = f_t \odot C_{t-1} + i_t \odot C'_t$$

- Output Gate - decides which values in the cell state are to be output, by looking at the current input and the previous hidden state.

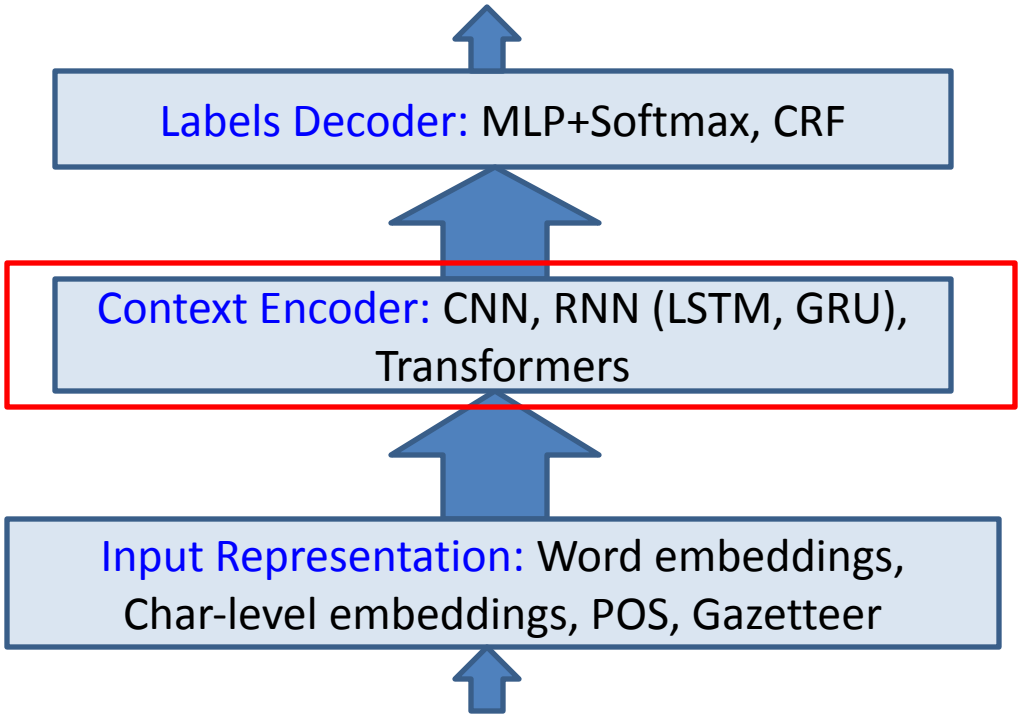
$$o_t = \sigma(W_o[x_t; h_{t-1}] + b_o)$$

$$h_t = o_t \odot tanh(C_t)$$

Colah's Blog- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Entity Extraction using Deep Learning

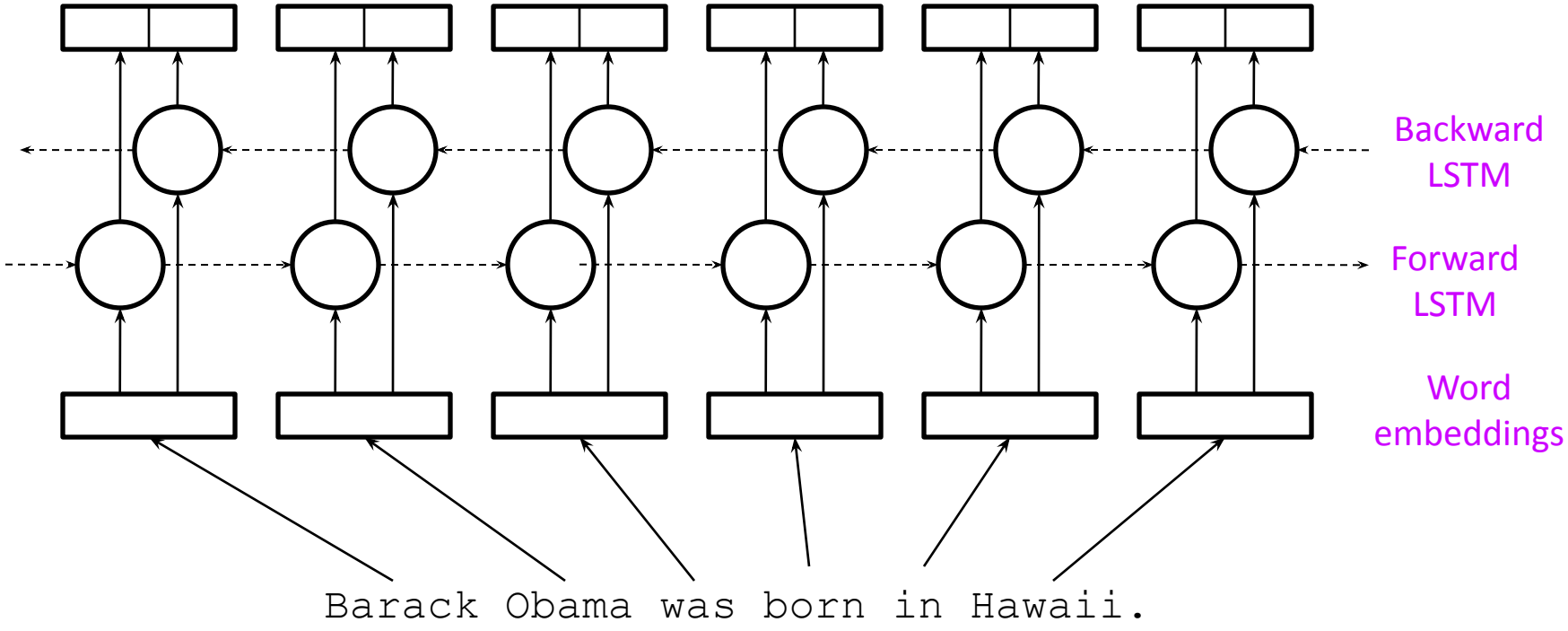
Barack/B-PER Obama/I-PER was/O born/O in/O **Hawaii/B-GPE** ./O



Li et al., A Survey on Deep Learning for Named Entity Recognition, IEEE TKDE 2020

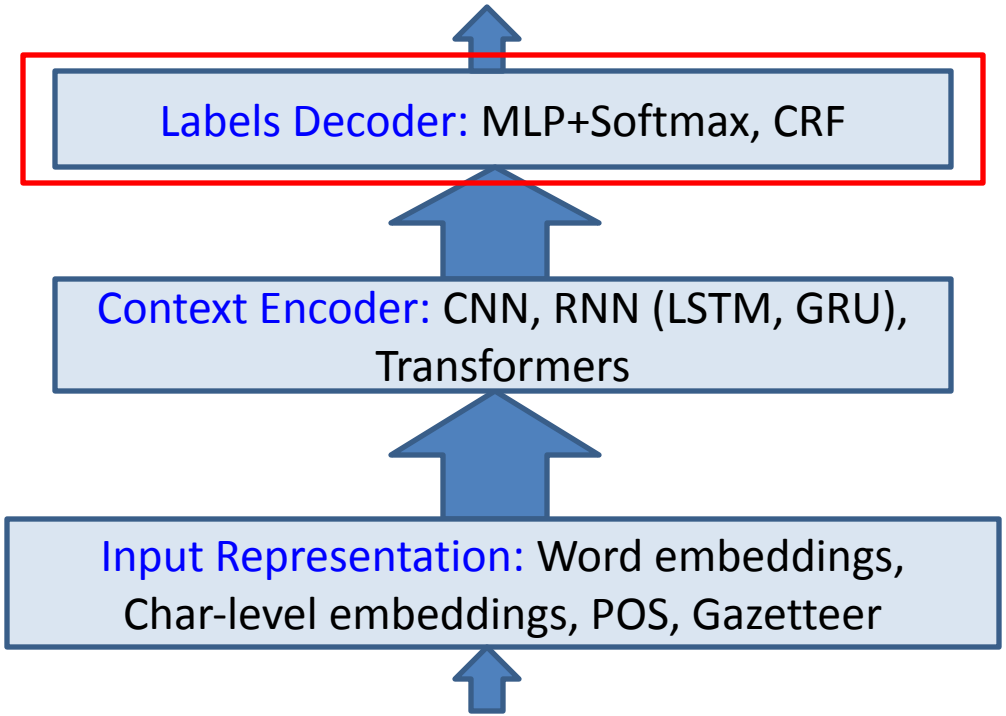
Context Encoding using Bidirectional LSTM

Forward & backward context representation for each word as outputted by the bidirectional LSTM layer



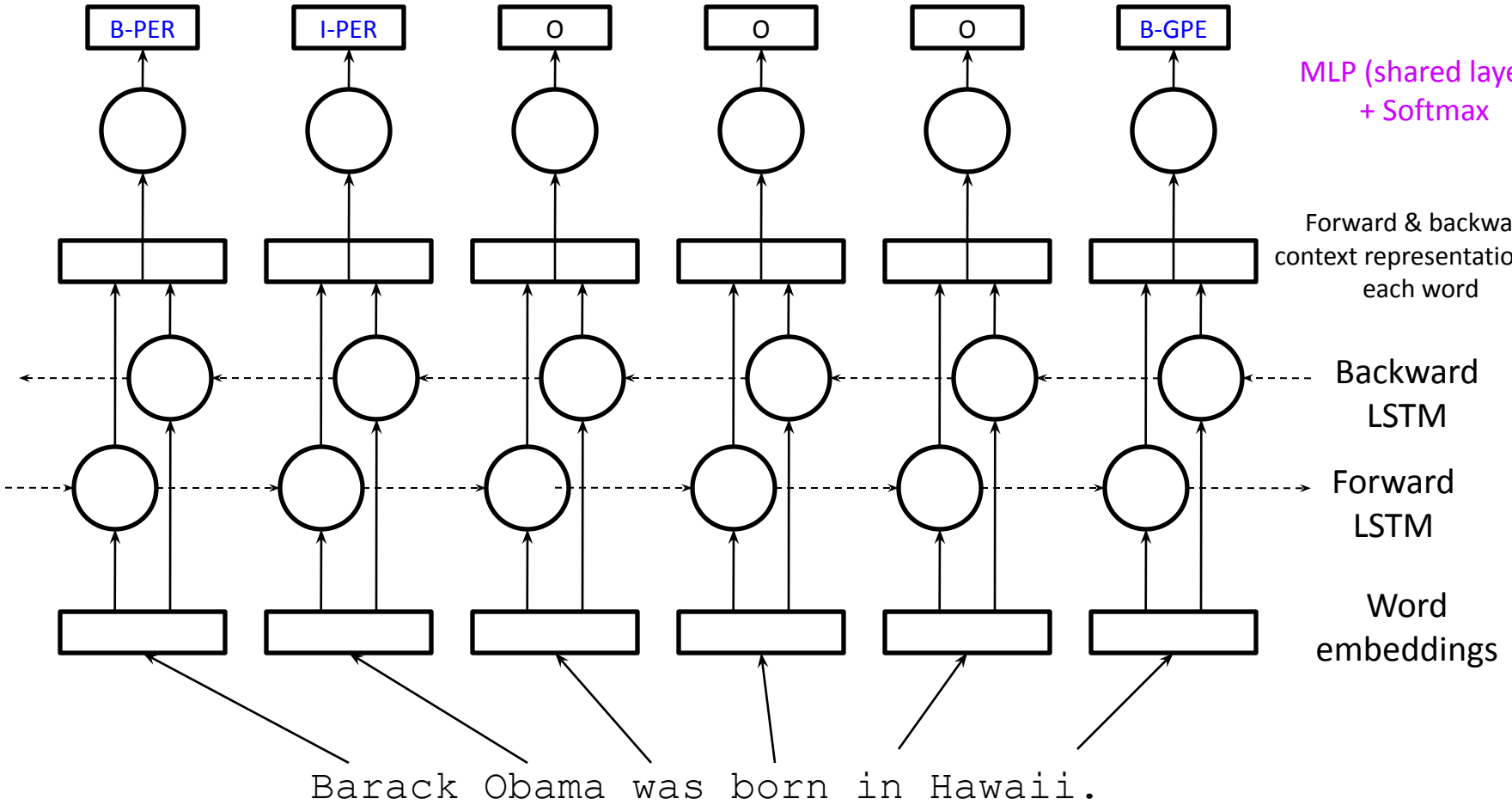
Entity Extraction using Deep Learning

Barack/B-PER Obama/I-PER was/O born/O in/O **Hawaii/B-GPE** ./O

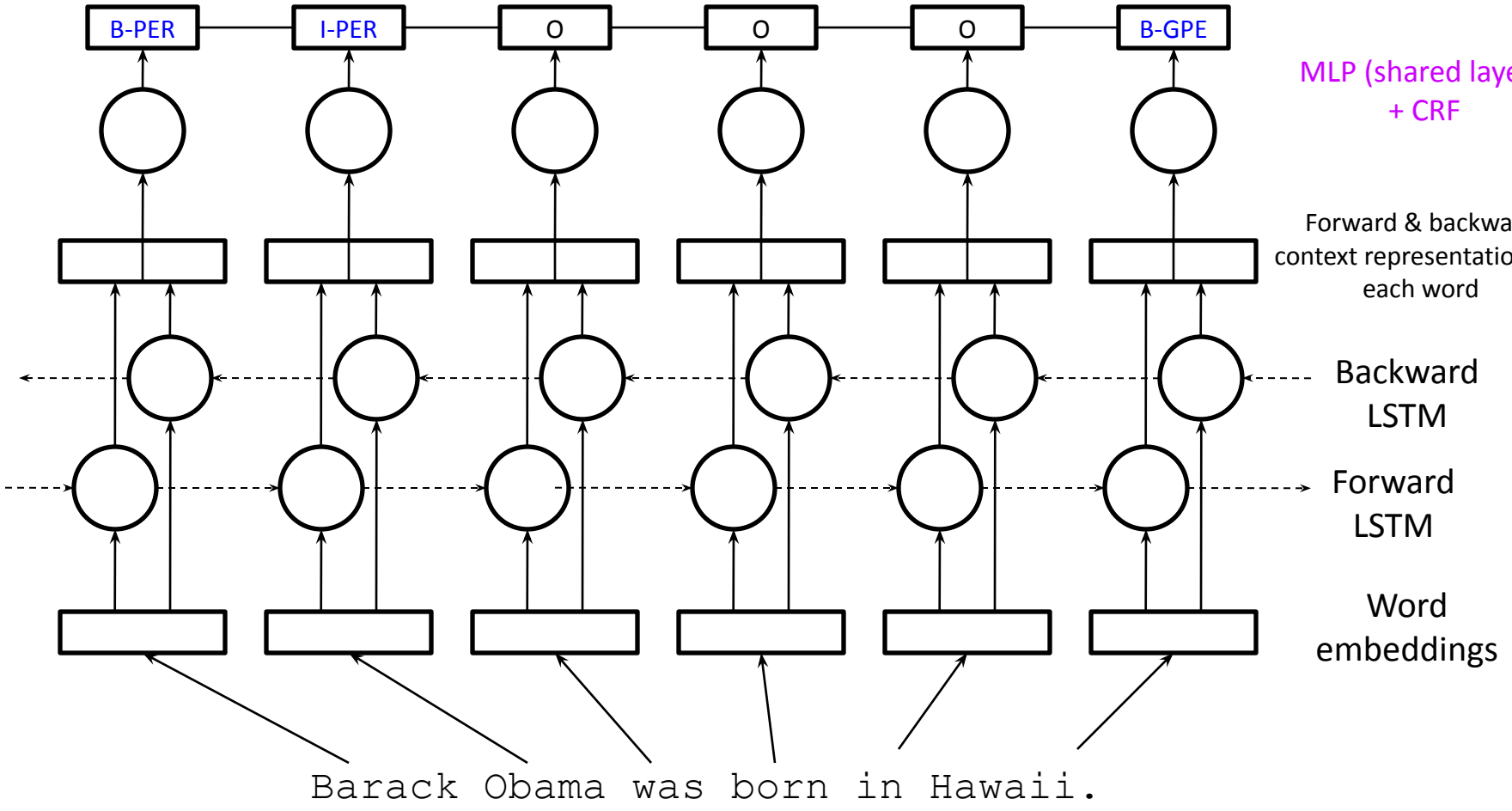


Li et al., A Survey on Deep Learning for Named Entity Recognition, IEEE TKDE 2020

Labels Decoder: MLP + Softmax



Labels Decoder: CRF



Evaluation of Entity Extraction

Evaluation

- For each entity type E, the following metrics are computed
  - True Positives (TP): Number of entity mentions of type E which were actually predicted as of type E
  - False Positives (FP): Number of predicted entity mentions of type E which are not actual / expected entity mentions for type E
  - False Negatives (FN): Number of actual entity mentions of type E which are not predicted as entity mentions of type E
- Then for each entity type E,
  - Pr = TP/(TP + FP); Re = TP/(TP + FN); F1 = 2*Pr*Re/(Pr + Re)
- Overall metrics
  - Macro Pr / Re / F1 = Average of the corresponding metrics across entity types
  - Micro Pr / Re / F1 = Weighted average of the corresponding metrics across entity types where weights correspond to support for each type
- Strict vs Lenient evaluation – Depending on the domain and the end application, lenient evaluation can be carried out. E.g., n-1 words match can be allowed for entity mentions having n > 3

Evaluation

- Expected: **Air/B-ORG India/I-ORG** women/O pilots/O to/O fly/O over/O **North/B-LOC Pole/I-LOC** on/O world/O 's/O longest/O air/O route/O from/O **Bengaluru/B-GPE** to/O **San/B-GPE Francisco/I-GPE** ./O
  - ORG: {Air India}; LOC: {North Pole}; GPE: {Bengaluru, San Fransisco}
- Predicted: **Air/B-PER India/I-PER** women/O pilots/O to/O fly/O over/O **North/B-LOC Pole/I-LOC** on/O world/O 's/O longest/O air/O route/O from/O **Bengaluru/B-ORG** to/O **San/B-GPE Francisco/I-GPE** ./O
  - PER: {Air India}; LOC: {North Pole}; GPE: {San Fransisco}; ORG: {Bengaluru}
- For ORG: TP = 0, FP = 1, FN = 1
- For PER: TP = 0, FP = 1, FN = 0
- For GPE: TP = 1, FP = 0, FN = 1
- For LOC: TP = 1, FP = 0, FN = 0
- Overall: TP = 2, FP = 2, FN = 2 => Pr = 0.5, Re = 0.5, F1 = 0.5 (Micro-averaged)

Entity Extraction using CRF in Python

CoNLL 2003 Dataset for NER

- Three partitions of the annotated dataset
  - Training set: 14041 sentences
  - Validation set: 3250 sentences
  - Test set: 3453 sentences
- Four types of entity labels
  - PER, ORG, LOC and MISC
- Each sentence contains the following information for each word
  - **Word itself**
  - POS tag of the word
  - Shallow parsing tag – NP, VP etc.
  - **Gold-standard entity label using the BIO encoding strategy**

```
-DOCSTART- -X- -X- O

EU NNP B-NP B-ORG
rejects VBZ B-VP O
German JJ B-NP B-MISC
call NN I-NP O
to TO B-VP O
boycott VB I-VP O
British JJ B-NP B-MISC
lamb NN I-NP O
. . O O

Peter NNP B-NP B-PER
Blackburn NNP I-NP I-PER
```

```python
def read_dataset(filepath):
    lst_sentences = []
    sentence = []
    f = open(filepath)
    for line in f:
        line = line.strip()
        if(line.startswith('-DOCSTART-')):
            continue
        if(len(line) == 0):
            if(len(sentence) > 0):
                lst_sentences.append(sentence)
                sentence = []
        else:
            parts = line.split(' ')
            sentence.append(parts)
    f.close()
    return lst_sentences

train_sentences = read_dataset('/kaggle/input/conll003-englishversion/train.txt')
test_sentences = read_dataset('/kaggle/input/conll003-englishversion/test.txt')
valid_sentences = read_dataset('/kaggle/input/conll003-englishversion/valid.txt')
```

```
print('Number of training sentences=', len(train_sentences))
print('Number of test sentences=', len(test_sentences))
print('Number of validation sentences=', len(valid_sentences))
pprint.pprint(train_sentences[0])
```

```
Number of training sentences= 14041
Number of test sentences= 3453
Number of validation sentences= 3250
[['EU', 'NNP', 'B-NP', 'B-ORG'],
 ['rejects', 'VBZ', 'B-VP', 'O'],
 ['German', 'JJ', 'B-NP', 'B-MISC'],
 ['call', 'NN', 'I-NP', 'O'],
 ['to', 'TO', 'B-VP', 'O'],
 ['boycott', 'VB', 'I-VP', 'O'],
 ['British', 'JJ', 'B-NP', 'B-MISC'],
 ['lamb', 'NN', 'I-NP', 'O'],
 ['.', '.', 'O', 'O']]
```

```python
def get_features_for_a_sentence(sentence):
    lst_features = []
    for i in range(len(sentence)):
        features = {} #Dictionary of features per word
        features['W'] = sentence[i][0].lower()
        features['POS'] = sentence[i][1]
        if(i > 0):
            features['PW'] = sentence[i-1][0].lower()
            features['PPOS'] = sentence[i-1][1]
        if(i < len(sentence)-1):
            features['NW'] = sentence[i+1][0].lower()
            features['NPOS'] = sentence[i+1][1]
        if(len(sentence[i][0]) >= 3):
            features['Suff3'] = sentence[i][0][-3:].lower()
        if(len(sentence[i][0]) >= 2):
            features['Suff2'] = sentence[i][0][-2:].lower()
        lst_features.append(features)
    return lst_features

def get_labels_for_a_sentence(sentence):
    lst_labels = []
    for i in range(len(sentence)):
        lst_labels.append(sentence[i][-1])
    return lst_labels
```

```
X_train = [get_features_for_a_sentence(s) for s in train_sentences]
y_train = [get_labels_for_a_sentence(s) for s in train_sentences]
X_test = [get_features_for_a_sentence(s) for s in test_sentences]
y_test = [get_labels_for_a_sentence(s) for s in test_sentences]
print(len(X_train), len(y_train), len(X_test), len(y_test))
pprint.pprint(train_sentences[3][:10])
pprint.pprint(X_train[3][2])
```

```
14041 14041 3453 3453
[['The', 'DT', 'B-NP', 'O'],
 ['European', 'NNP', 'I-NP', 'B-ORG'],
 ['Commission', 'NNP', 'I-NP', 'I-ORG'],
 ['said', 'VBD', 'B-VP', 'O'],
 ['on', 'IN', 'B-PP', 'O'],
 ['Thursday', 'NNP', 'B-NP', 'O'],
 ['it', 'PRP', 'B-NP', 'O'],
 ['disagreed', 'VBD', 'B-VP', 'O'],
 ['with', 'IN', 'B-PP', 'O'],
 ['German', 'JJ', 'B-NP', 'B-MISC']]
{'NPOS': 'VBD',
 'NW': 'said',
 'POS': 'NNP',
 'PPOS': 'NNP',
 'PW': 'european',
 'Suff2': 'on',
 'Suff3': 'ion',
 'W': 'commission'}
```

Training the CRF model

```
crf = sklearn_crfsuite.CRF(algorithm='lbfgs',
                           c1=0.1,c2=0.1,
                           max_iterations=100,
                           all_possible_transitions=True,
                          verbose=True)
crf.fit(X_train, y_train)
```

```
loading training data to CRFsuite: 100%|██████████| 14041/14041 [00:02<00:00, 6288.69it/s]
```

```
Feature generation
type: CRF1d
feature.minfreq: 0.000000
feature.possible_states: 0
feature.possible_transitions: 1
0....1....2....3....4....5....6....7....8....9....10
Number of features: 83618
Seconds required: 0.397
```

Training the CRF model

```
Iter 1   time=0.38  loss=294134.84 active=83334 feature_norm=1.00
Iter 2   time=0.58  loss=219282.96 active=82795 feature_norm=4.45
Iter 3   time=0.20  loss=167595.34 active=79036 feature_norm=3.75
Iter 4   time=0.58  loss=130654.99 active=80802 feature_norm=3.18
Iter 5   time=0.20  loss=113984.74 active=82282 feature_norm=3.77
Iter 6   time=0.20  loss=104194.81 active=82458 feature_norm=4.26
Iter 7   time=0.20  loss=78611.27 active=74289 feature_norm=7.38
Iter 8   time=0.20  loss=69648.84 active=67280 feature_norm=8.32
```

```
Iter 97  time=0.20  loss=7741.54  active=26789 feature_norm=163.35
Iter 98  time=0.23  loss=7740.02  active=26769 feature_norm=163.35
Iter 99  time=0.20  loss=7739.36  active=26754 feature_norm=163.33
Iter 100 time=0.21  loss=7737.82  active=26751 feature_norm=163.33
L-BFGS terminated with the maximum number of iterations
Total seconds required for training: 23.541

Storing the model
Number of active features: 26751 (83618)
Number of active attributes: 17134 (66253)
Number of active labels: 9 (9)
```

Inference using the trained CRF model

```
y_pred = crf.predict(X_test)
!pip install seqeval
from seqeval.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

         LOC       0.84      0.81      0.82      1668
        MISC       0.78      0.72      0.75       702
         ORG       0.74      0.68      0.71      1661
         PER       0.81      0.81      0.81      1617

   micro avg       0.79      0.76      0.78      5648
   macro avg       0.79      0.76      0.77      5648
weighted avg       0.79      0.76      0.78      5648
```

CRF model: Insights

```
from collections import Counter
pprint.pprint(Counter(crf.transition_features_).most_common(10))
```

```
[(('B-ORG', 'I-ORG'), 7.861978),
 (('B-PER', 'I-PER'), 6.515959),
 (('B-LOC', 'I-LOC'), 6.299926),
 (('I-ORG', 'I-ORG'), 6.249851),
 (('B-MISC', 'I-MISC'), 6.231935),
 (('I-MISC', 'I-MISC'), 5.752927),
 (('I-LOC', 'I-LOC'), 4.86833),
 (('O', 'O'), 3.689662),
 (('I-PER', 'I-PER'), 3.052938),
 (('O', 'B-MISC'), 2.129322)]
```

CRF model: Insights

```
pprint.pprint(Counter(crf.transition_features_).most_common()[-10:])
```

```
[(('B-PER', 'B-ORG'), -3.098907),
 (('I-ORG', 'B-PER'), -3.101531),
 (('I-PER', 'B-LOC'), -3.328572),
 (('O', 'I-PER'), -3.56472),
 (('I-ORG', 'B-ORG'), -3.796707),
 (('O', 'I-LOC'), -3.860906),
 (('O', 'I-MISC'), -3.865568),
 (('O', 'I-ORG'), -4.043245),
 (('I-PER', 'B-PER'), -4.165913),
 (('B-PER', 'B-PER'), -4.210622)]
```

CRF model: Insights

```
pprint.pprint(Counter(crf.state_features_).most_common(20))
```

```
[(('POS:CD', 'O'), 7.895185),
 (('W:$', 'O'), 6.844689),
 (('W:german', 'B-MISC'), 6.589871),
 (('W:stenning', 'B-PER'), 6.47218),
 (('W:dutch', 'B-MISC'), 6.191205),
 (('W:mideast', 'B-LOC'), 6.11303),
 (('PW:v', 'B-ORG'), 5.982092),
 (('W:attendance', 'O'), 5.956291),
 (('W:clinton', 'B-PER'), 5.876132),
 (('POS::', 'O'), 5.80823),
 (('Suff2:-1', 'O'), 5.805164),
 (('W:stansted', 'B-LOC'), 5.709132),
 (('W:to', 'O'), 5.572647),
 (('PW:wisc', 'I-LOC'), 5.44061),
 (('PW:colo', 'I-LOC'), 5.435837),
 (('W:france', 'B-LOC'), 5.393077),
 (('W:med', 'B-LOC'), 5.381044),
 (('Suff2:-0', 'O'), 5.363889),
 (('W:masters', 'I-MISC'), 5.297543),
 (('W:senate', 'B-ORG'), 5.261652)]
```

CRF model: Insights

```
pprint.pprint(Counter(crf.state_features_).most_common()[-20:])
```

```
[(('Suff2:da', 'O'), -2.528193),
 (('Suff3:ese', 'O'), -2.538075),
 (('PW:cdu', 'O'), -2.592426),
 (('Suff3:bay', 'O'), -2.617086),
 (('W:aces', 'O'), -2.618174),
 (('NW:oval', 'O'), -2.661009),
 (('Suff2:ni', 'O'), -2.701302),
 (('POS:RB', 'I-ORG'), -2.766404),
 (('POS:NNS', 'I-LOC'), -2.780267),
 (('PW:u.s.', 'I-LOC'), -2.820058),
 (('W:nations', 'O'), -2.866962),
 (('POS:NN', 'I-MISC'), -2.883792),
 (('W:nice', 'O'), -3.056933),
 (('PW:interior', 'O'), -3.069985),
 (('POS:NNPS', 'O'), -3.086432),
 (('W:french', 'O'), -3.103848),
 (('PW:moody', 'O'), -3.200683),
 (('POS:NN', 'I-LOC'), -3.448477),
 (('PW:beat', 'O'), -3.67361),
 (('PW:lloyd', 'O'), -3.98852)]
```

Entity Extraction using LSTM in Python

```
import numpy as np
import torch
def read_word_embeddings(word_embeddings_file):
    word_to_index = {}
    word_to_vector = {}
    DIM = -1
    word_id = 1
    f = open(word_embeddings_file)
    for line in f:
        parts = line.strip().split(' ')
        if(DIM == -1 and len(parts) > 1):
            DIM = len(parts)-1
            print('Word vectors dim = ', DIM)
        word = parts[0]
        if(word not in word_to_index):
            word_to_index[word] = word_id
            word_id = word_id + 1
            vec = torch.from_numpy(np.array(parts[1:]).astype(np.float32))
            word_to_vector[word] = vec
    f.close()
    embedding_weights = torch.zeros(len(word_to_index)+1,DIM,dtype=torch.float32)
    for word, index in word_to_index.items():
        embedding_weights[index] = word_to_vector[word]
    return(embedding_weights, word_to_index)
```

```
embedding_weights, word_to_index = read_word_embeddings(
    '../input/glove-global-vectors-for-word-representation/glove.6B.100d.txt')
print(embedding_weights.shape, len(word_to_index))
```

```
Word vectors dim =  100
torch.Size([400001, 100]) 400000
```

```
import torch
cos = torch.nn.CosineSimilarity(dim=0)
s1 = cos(embedding_weights[word_to_index['attacked']],
         embedding_weights[word_to_index['assaulted']])
print(s1)
s2 = cos(embedding_weights[word_to_index['attacked']],
         embedding_weights[word_to_index['jumped']])
print(s2)
s3 = cos(embedding_weights[word_to_index['attacked']],
         embedding_weights[word_to_index['enjoyed']])
print(s3)
```

```
tensor(0.6909)
tensor(0.3306)
tensor(0.1977)
```

```python
def get_label_indices(y_train):
    labels = set()
    for lst_labels in y_train:
        for label in lst_labels:
            labels.add(label)
    labels = list(labels)
    return labels

def get_pos_indices(train_sentences):
    pos_tags = set()
    for sentence in train_sentences:
        for i in range(len(sentence)):
            pos_tags.add(sentence[i][1])
    pos_tags = list(pos_tags)
    return pos_tags
```

```
labels = get_label_indices(y_train)
pos_tags = get_pos_indices(train_sentences)
pos_tags.append('X')
print(len(labels))
print(labels)
print(len(pos_tags))
print(pos_tags)
```

```
9
['I-PER', 'I-MISC', 'B-MISC', 'B-PER', 'B-ORG', 'B-LOC', 'O', 'I-ORG', 'I-LOC']
46
['UH', ')', '.', 'VBZ', 'WP', 'PDT', 'FW', 'NN|SYM', 'RBS', 'RBR', 'LS', 'WDT', 'TO', 'EX', 'JJS', 'NN
PS', 'VBN', 'PRP', 'VBG', '$', 'DT', 'JJ', 'WP$', '(', "''", 'CC', 'NN', 'VBP', 'RP', 'WRB', 'PRP$',
'CD', 'VB', 'VBD', ':', 'NNP', ',', '"', 'NNS', 'RB', 'JJR', 'MD', 'SYM', 'POS', 'IN', 'X']
```

```python
class NER_Model(torch.nn.Module):
    def __init__(self, embedding_weights, word_to_index, pos_tags, labels,
                 WORD_DIM, POS_DIM, LSTM_REPR_DIM):
        super().__init__()
        self.word_embedding_layer = torch.nn.Embedding.from_pretrained(embedding_weights)
        self.pos_embedding_layer = torch.nn.Embedding(len(pos_tags)+1, POS_DIM)
        self.word_to_index = word_to_index
        self.lstm_layer = torch.nn.LSTM(WORD_DIM+POS_DIM, LSTM_REPR_DIM,
                                        batch_first=True, bidirectional=True)
        self.linear_layer = torch.nn.Linear(LSTM_REPR_DIM*2, LSTM_REPR_DIM)
        self.final_linear_layer = torch.nn.Linear(LSTM_REPR_DIM, len(labels))
        self.dropout = torch.nn.Dropout(0.3)

    def forward(self, X_words, X_pos):
        X_words = self.word_embedding_layer(X_words)
        X_pos = self.pos_embedding_layer(X_pos)
        X_words_pos = torch.cat([X_words, X_pos], dim=-1)
        lstm_output, (_,_) = self.lstm_layer(X_words_pos)
        lstm_output = self.dropout(lstm_output)
        linear_output = self.linear_layer(lstm_output)
        classification_output = self.final_linear_layer(linear_output)
        return classification_output

ner_model = NER_Model(embedding_weights, word_to_index, pos_tags, labels, 100, 20, 200)
```

```
from torch.utils.data import Dataset, DataLoader
class NER_Dataset(Dataset):
    def __init__(self, sentences, word_to_index, pos_tags, labels, MAX_WORDS_IN_A_SENT):
        self.X_words = torch.zeros(len(sentences), MAX_WORDS_IN_A_SENT, dtype=torch.long)
        self.X_pos = torch.zeros(len(sentences), MAX_WORDS_IN_A_SENT, dtype=torch.long)
        self.y = torch.zeros(len(sentences), MAX_WORDS_IN_A_SENT, dtype=torch.long)
        self.sent_lengths = torch.zeros(len(sentences), dtype=torch.long)
        for i, sent in enumerate(sentences):
            for j in range(len(sent)):
                if(j >= MAX_WORDS_IN_A_SENT):
                    break
                word = sent[j][0].lower()
                if(word in word_to_index):
                    self.X_words[i][j] = word_to_index[word]
                pos = sent[j][1]
                if(pos in pos_tags):
                    self.X_pos[i][j] = pos_tags.index(pos)
                else:
                    self.X_pos[i][j] = pos_tags.index('X')
                label = sent[j][-1]
                if(label in labels):
                    self.y[i][j] = labels.index(label)
            for j in range(len(sent), MAX_WORDS_IN_A_SENT):
                self.X_pos[i][j] = pos_tags.index('X')
                self.y[i][j] = labels.index('O')
            if(len(sent) < MAX_WORDS_IN_A_SENT):
                self.sent_lengths[i] = len(sent)
            else:
                self.sent_lengths[i] = MAX_WORDS_IN_A_SENT
```

```
    def __len__(self):
        return self.X_words.shape[0]

    def __getitem__(self, idx):
        return (self.X_words[idx], self.X_pos[idx], self.y[idx], self.sent_lengths[idx])

train_dataset = NER_Dataset(train_sentences, word_to_index, pos_tags, labels, 40)
valid_dataset = NER_Dataset(valid_sentences, word_to_index, pos_tags, labels, 40)
test_dataset = NER_Dataset(test_sentences, word_to_index, pos_tags, labels, 40)
```

```
train_dataloader = DataLoader(train_dataset, batch_size=32, shuffle=True)
valid_dataloader = DataLoader(valid_dataset, batch_size=32)
test_dataloader = DataLoader(test_dataset, batch_size=32)
print(len(train_dataloader.dataset), len(train_dataloader))
print(len(valid_dataloader.dataset), len(valid_dataloader))
print(len(test_dataloader.dataset), len(test_dataloader))
```

```
14041 439
3250 102
3453 108
```

```
def get_predicted_labels(classification_op, sent_lengths, labels):
    pred_labels = []
    pred_label_indices = torch.argmax(classification_op, dim=-1)
    for i in range(pred_label_indices.shape[0]):
        pred_labels_sent = []
        for j in range(sent_lengths[i].item()):
            pred_labels_sent.append(labels[pred_label_indices[i][j].item()])
        pred_labels.append(pred_labels_sent)
    return pred_labels

def get_actual_labels(y, sent_lengths, labels):
    actual_labels = []
    for i in range(y.shape[0]):
        actual_labels_sent = []
        for j in range(sent_lengths[i].item()):
            actual_labels_sent.append(labels[y[i][j]])
        actual_labels.append(actual_labels_sent)
    return actual_labels

optimizer = torch.optim.Adam(ner_model.parameters(), lr=0.001)
loss_function = torch.nn.CrossEntropyLoss()
```

```
for epoch in range(4):
    actual_labels = []; predicted_labels = []
    actual_labels_validation = []; predicted_labels_validation = []
    avg_loss = 0; avg_loss_validation = 0
    ner_model.train()
    for i, data in enumerate(train_dataloader):
        if(i%100 == 0):
            print(i)
        X_words = data[0]
        X_pos = data[1]
        y = data[2]
        classification_op = ner_model(X_words, X_pos)
        loss = loss_function(classification_op.transpose(-2,-1), y)
        actual_labels.extend(
            get_actual_labels(y, data[3], labels))
        predicted_labels.extend(
            get_predicted_labels(classification_op, data[3], labels))
        avg_loss = avg_loss + loss.item()
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

```
ner_model.eval()
for i, data in enumerate(valid_dataloader):
    X_words = data[0]
    X_pos = data[1]
    y = data[2]
    classification_op = ner_model(X_words, X_pos)
    actual_labels_validation.extend(
        get_actual_labels(y, data[3], labels))
    predicted_labels_validation.extend(
        get_predicted_labels(classification_op, data[3], labels))
    loss = loss_function(classification_op.transpose(-2,-1), y)
    avg_loss_validation = avg_loss_validation + loss.item()

print('Epoch = ', epoch,
      'Training loss = ', avg_loss/len(train_dataloader.dataset),
      'Validation loss = ', avg_loss_validation/len(valid_dataloader.dataset))

report = classification_report(actual_labels, predicted_labels)
print('Training accuracy:')
print(report)
report = classification_report(actual_labels_validation, predicted_labels_validation)
print('Validation accuracy:')
print(report)
```

```
Epoch =  0 Training loss =  0.0034164670402709 85 Validation loss =  0.0013531871584351533
Training accuracy:
              precision    recall  f1-score   support

         LOC       0.70      0.70      0.70      7090
        MISC       0.66      0.38      0.48      3419
         ORG       0.57      0.50      0.53      6284
         PER       0.66      0.74      0.70      6550

   micro avg       0.65      0.61      0.63     23343
   macro avg       0.65      0.58      0.60     23343
weighted avg       0.65      0.61      0.62     23343

Validation accuracy:
              precision    recall  f1-score   support

         LOC       0.85      0.89      0.87      1807
        MISC       0.67      0.71      0.69       914
         ORG       0.73      0.67      0.70      1329
         PER       0.91      0.92      0.91      1770

   micro avg       0.81      0.82      0.82      5820
   macro avg       0.79      0.80      0.79      5820
weighted avg       0.81      0.82      0.82      5820
```

```
Epoch =  3 Training loss =  0.00060627459528581 44 Validation loss =  0.0008627226369930073
Training accuracy:
              precision    recall  f1-score   support

         LOC       0.93      0.94      0.94      7090
        MISC       0.84      0.82      0.83      3419
         ORG       0.84      0.87      0.86      6284
         PER       0.95      0.96      0.95      6550

   micro avg       0.90      0.91      0.90     23343
   macro avg       0.89      0.90      0.89     23343
weighted avg       0.90      0.91      0.90     23343

Validation accuracy:
              precision    recall  f1-score   support

         LOC       0.94      0.91      0.93      1807
        MISC       0.82      0.79      0.80       914
         ORG       0.82      0.86      0.84      1329
         PER       0.91      0.95      0.93      1770

   micro avg       0.89      0.89      0.89      5820
   macro avg       0.88      0.88      0.88      5820
weighted avg       0.89      0.89      0.89      5820
```

```
actual_labels_test = []; predicted_labels_test = []
ner_model.eval()
for i, data in enumerate(test_dataloader):
    X_words = data[0]
    X_pos = data[1]
    y = data[2]
    classification_op = ner_model(X_words, X_pos)
    actual_labels_test.extend(get_actual_labels(y, data[3], labels))
    predicted_labels_test.extend(
        get_predicted_labels(classification_op, data[3], labels))
report = classification_report(actual_labels_test, predicted_labels_test)
print('Test accuracy:')
print(report)
```

```
Test accuracy:
              precision    recall  f1-score   support

         LOC       0.90      0.89      0.90      1638
        MISC       0.71      0.74      0.72       694
         ORG       0.79      0.83      0.81      1646
         PER       0.91      0.92      0.92      1560

   micro avg       0.85      0.86      0.85      5538
   macro avg       0.83      0.84      0.84      5538
weighted avg       0.85      0.86      0.85      5538
```

Error Cases

```
print(len(test_sentences), len(actual_labels_test), len(predicted_labels_test))
incorrect_sent_indices = []
for i in range(len(test_sentences)):
    if(len(test_sentences[i]) >= 40):
        continue
    for j in range(len(test_sentences[i])):
        if(actual_labels_test[i][j] != predicted_labels_test[i][j]):
            incorrect_sent_indices.append(i)
print(len(test_sentences), len(incorrect_sent_indices))
import random
lst_random = random.sample(incorrect_sent_indices, 20)
print(lst_random)
for i in lst_random:
    actual_sentence = []; pred_sentence = []
    for j in range(len(test_sentences[i])):
        actual_sentence.append(test_sentences[i][j][0]
                               +'/'+actual_labels_test[i][j])
        pred_sentence.append(test_sentences[i][j][0]
                             +'/'+predicted_labels_test[i][j])
    print('\nActual: ', ' '.join(actual_sentence))
    print('\nPredicted: ', ' '.join(pred_sentence))
```

```
3453 3453 3453
3453 1214
[3306, 991, 154, 2367, 513, 3311, 2386, 2226, 806, 2164, 1171, 1940, 1582, 1867, 0, 3317, 3047, 2633,
1777, 3250]

Actual:  Results/O of/O National/B-ORG Basketball/I-ORG

Predicted:  Results/O of/O National/B-MISC Basketball/I-MISC

Actual:  One/O Global/O Depository/O Receipt/O represents/O 10/O common/O shares/O ./O

Predicted:  One/O Global/B-MISC Depository/I-ORG Receipt/I-ORG represents/O 10/O common/O shares/O ./O

Actual:  Leeds/B-ORG had/O already/O fined/O Bowyer/B-PER 4,000/O pounds/O (/O $/O 6,600/O )/O and/O w
arned/O him/O a/O repeat/O of/O his/O criminal/O behaviour/O could/O cost/O him/O his/O place/O in/O t
he/O side/O ./O

Predicted:  Leeds/B-ORG had/O already/O fined/O Bowyer/B-ORG 4,000/O pounds/O (/O $/O 6,600/O )/O and/
O warned/O him/O a/O repeat/O of/O his/O criminal/O behaviour/O could/O cost/O him/O his/O place/O in/
O the/O side/O ./O
```

```
Actual:  Japan/B-LOC NTT/B-ORG says/O hopes/O to/O start/O int'l/O business/O soon/O ./O

Predicted:  Japan/B-LOC NTT/I-ORG says/O hopes/O to/O start/O int'l/O business/O soon/O ./O
```

```
Actual:  He/O has/O offered/O to/O apologise/O if/O Costa/B-PER acknowledges/O the/O provocation/O ./O

Predicted:  He/O has/O offered/O to/O apologise/O if/O Costa/B-LOC acknowledges/O the/O provocation/O
./O

Actual:  Munich/B-ORG Re/I-ORG says/O to/O split/O stock/O ./O

Predicted:  Munich/B-ORG Re/O says/O to/O split/O stock/O ./O
```

```
Actual:  The/O Foreign/B-ORG Ministry/I-ORG said/O the/O 10/O Europeans/B-MISC and/O seven/O Africans/
B-MISC took/O a/O special/O flight/O from/O the/O Garamba/B-LOC national/O park/O in/O northern/O Zair
e/B-LOC to/O the/O Ugandan/B-MISC capital/O Kampala/B-LOC where/O they/O were/O being/O looked/O afte
r/O at/O the/O Italian/B-MISC embassy/O ./O

Predicted:  The/O Foreign/B-ORG Ministry/I-ORG said/O the/O 10/O Europeans/B-MISC and/O seven/O Africa
ns/I-MISC took/O a/O special/O flight/O from/O the/O Garamba/B-LOC national/I-LOC park/O in/O norther
n/O Zaire/B-LOC to/O the/O Ugandan/B-MISC capital/O Kampala/B-LOC where/O they/O were/O being/O looke
d/O after/O at/O the/O Italian/B-MISC embassy/O ./O
```

Other Advanced Information Extraction Tasks

Relation Extraction

- Usually, entity mentions often do not occur independently in text.
  - interplay or dependence among these entity mentions is one of the essential aspects of the meaning of the whole sentence
- The term relation is used to represent such well-defined semantic interaction among the entity mentions
  - As there are multiple types of entity mentions, various types of such relation types are possible
- Examples:
  - BornIn, ResidentOf relations can exist between a PER and a LOC
  - FounderOf, EmployedBy relations can exist between a PER and an ORG
  - In a specific domain like Agriculture, relation like Affects can exist between a DISEASE and a CROP.

Relation Extraction

- The task of Relation Extraction (RE) is :
  - to automatically identify whether any pre-defined semantic relation exists between two (or more) entity mentions, and
  - if some relation exists then to identify the type of that relation
- Global vs Mention-level Relations
  - Global-level relations hold between entities (and not entity mentions) and have global scope (i.e., not specific to any sentence or document); useful for Knowledge Base (KB) population
    - BornIn (`Napoleon Bonaparte`, `Corsica`)
  - Mention-level Relation Extraction focuses on determining whether the given sentence expresses any particular relation type or not.

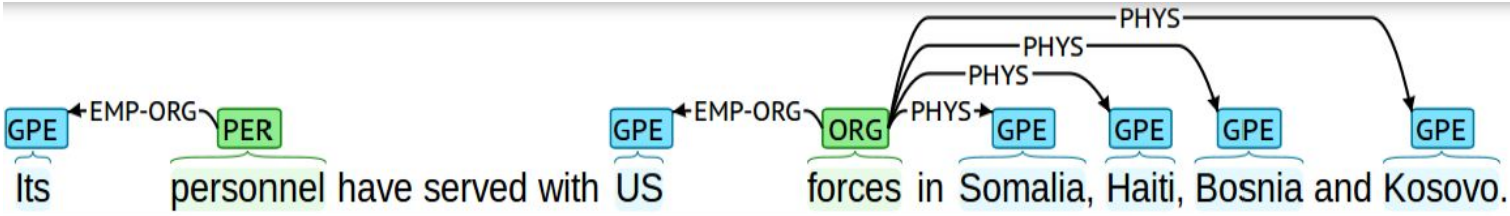
Relation Extraction (Example)

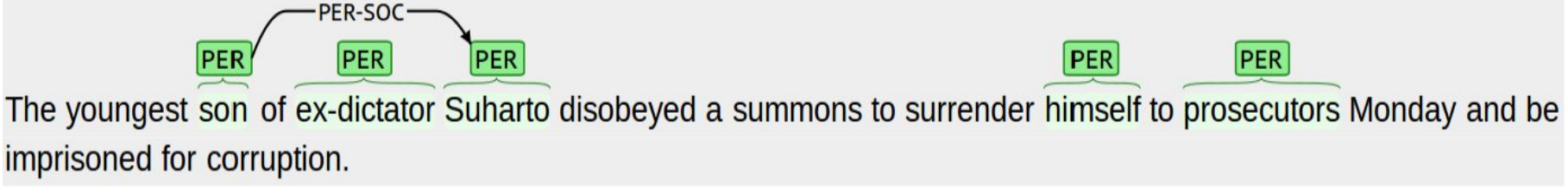
S1 Napoleon Bonaparte was born on the island of Corsica.

S2 Napoleon Bonaparte left Corsica for joining a military school in France.

- Both sentences contain the entity mentions Napoleon Bonaparte and Corsica, but only the sentence **S1** expresses the relation BornAt.
- Here, a mention level RE system should identify the BornAt relation between Napoleon Bonaparte and Corsica for **S1**,
- Whereas for **S2**, it should identify that **no relation exists** between these entity mentions in this particular sentence.

Relation Extraction (Example)





Relation Extraction: Techniques

- Rule-based techniques
- Supervised techniques:
  - Traditional features-based Machine Learning algorithms
  - Deep learning algorithms
- Unsupervised and semi-supervised techniques
  - Given a seed relation mentions, extracting more such relations
- Open Information Extraction (Open IE)
  - No pre-defined set of relation types
- Joint extraction of entity mentions and relations among them
- Extraction of complex relations - N-ary (more than two arguments), Cross-sentence
- Survey: https://arxiv.org/pdf/1712.05191.pdf

Co-reference Resolution

- The task of Co-reference Resolution is to identify all the entity mentions in a given text which correspond to the same entity
- Example:
  - John was suffering from Malaria. He was having a lot of fever. He was administered with chloroquine. It is one of the most commonly used drugs for the disease.
  - {John, He, He} all of these entity mentions correspond to only one real life entity – i.e., John
  - Similarly, {chloroquine, It} and {Malaria, the disease} and are sets of entity mentions which are co-references of each other.

Co-reference Resolution

- Difficult Example:
  - Creta won the Indian Car of the Year award in 2016. This car is manufactured by Hyundai Motor India Limited. **It** is a wholly owned subsidiary of the Hyundai Motor Company headquartered in South Korea.
  - **It** has two co-reference candidates from the previous sentence

- Survey: https://ojs.aaai.org/index.php/AAAI/article/view/11149/11008

Summary

- Introduction to Information Extraction
- Entity Extraction task and related techniques
  - CRF and Bi-LSTM based models
- Evaluation of Entity Extraction
- Practical implementation examples for the CoNLL 2003 dataset
- Other advanced Information Extraction tasks
  - Relation Extraction and Co-reference Resolution

Questions?

ADR Extraction (BMC Bioinformatics 2018)

Semi-Supervised Recurrent Neural Network for Adverse Drug Reaction mention extraction

S. Gupta, S. Pawar, N. Ramrakhiyani, G.K. Palshikar, V. Varma

BMC Bioinformatics 19, 212 (2018).

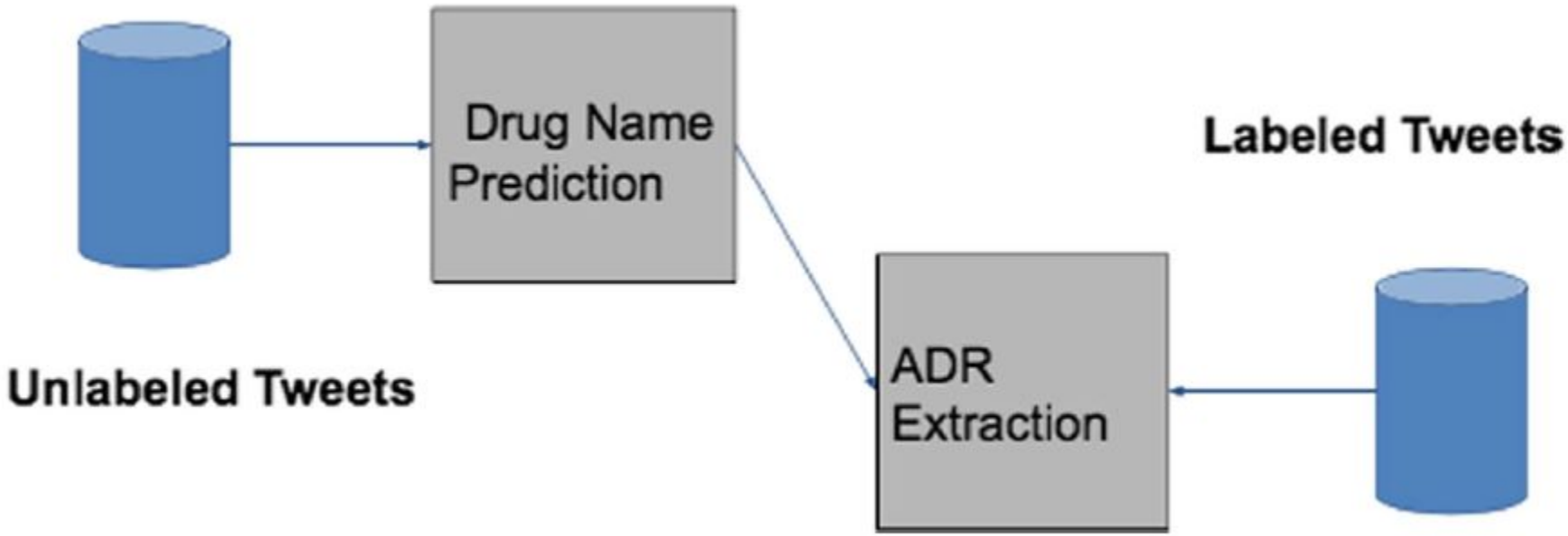
Background

- Goal: Social media is a good candidate for public-health monitoring tasks, specifically for *pharmacovigilance*.
- Problem: Extraction of Adverse-Drug-Reaction (ADR) mentions from social media, particularly from Twitter.
  - `headache, loss of appetite, pain at the injection site`
  - `Cymbalta, you're driving me insane`
- Challenges:
  - Short and highly informal nature of text, as compared to more technical and formal medical reports
  - Costly to create manually annotated training data

Overview of the proposed approach

- Modelled as a word-level sequence labelling problem:
  - IO-Encoding: Two labels I-ADR (Inside ADR) and O (Other)
  - `@BLENDOS/O Lamictal/O and/O trileptal/O and/O seroquel/O of/O course/O the/O seroquel/O I/O take/O in/O severe/O situations/O because/O weight/I-ADR gain/I-ADR is/O not/O cool/O ./O`
- Two-phase approach:
  - Unsupervised learning
    - Dummy task of Drug name prediction for a given tweet
    - Utilizing lots of easily available unlabelled data
    - Pre-training – incorporating context knowledge in the model
  - Supervised Sequence labelling
    - Pre-trained model is further trained with small training data

Overall System



Dummy Task of Drug Name Prediction

- Training data is created automatically
  - Given a tweet, identify the drug name mentions in it
  - Once drug names are identified in the tweet, replace all drug name mentions with a single dummy token (<DRUG>)
  - The context of the masked drug name in the tweet as input to predict the actual drug name
- A Bi-LSTM model is used to get context information of each word in a tweet
- Overall representation of the entire tweet is obtained by average pooling
- Finally, a softmax layer is used to generate a probability distribution over all drug names
- The model is trained using a categorical loss function

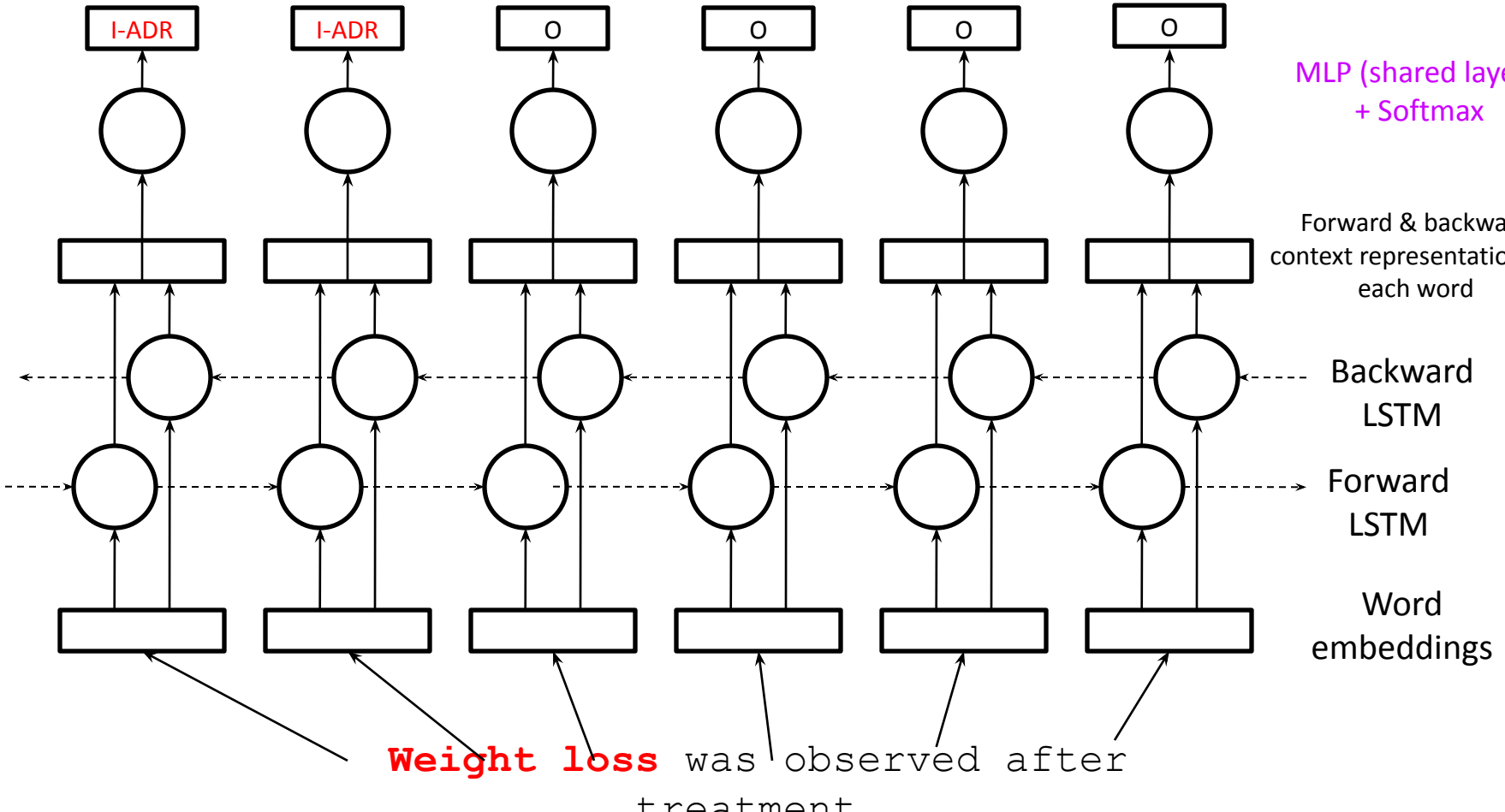
Supervised Sequence Labelling

- The same Bi-LSTM model is further trained using smaller training data annotated with word-level labels I-ADR and O

- Now, probability distribution over I-ADR and O labels is generated at each word (each time step)

- Overall loss is sum of categorical cross-entropy loss computed at each word (time step)

Supervised Sequence Labelling



Experiments: Dataset

- Phase-I: Unsupervised learning – Drug name prediction
  - 100,000 tweets containing exactly one drug mention
  - `humira`, `dronedarone`, `lamictal`, `pradaxa`, `paxil`, `zoledronic acid`, `trazodone`, `enbrel`, `cymbalta`, `quetiapine`
- Phase-II: Supervised sequence labelling
  - 645 tweets annotated with ADR mentions
  - Training: 470 tweets; Testing: 170 tweets

- **Text pre-processing**: normalizing HTML links and user mentions; removal of special characters, emoticons, and stop words

Experiments: Evaluation

Approach	Precision	Recall	F1
Baseline (Cocos et al., JAMIA, 2017)	0.695 ± 0.109	0.776 ± 0.121	0.729 ± 0.027
Our Bi-LSTM ADR Extraction (word2vec trained on Twitter data)	0.731 ± 0.035	0.774 ± 0.073	**0.751** ± **0.036**
More Analysis:			
Without drug name masking	0.723 ± 0.106	0.780 ± 0.108	0.747 ± 0.037
With labeled tweets dictionary only	0.727 ± 0.072	0.769 ± 0.097	0.745 ± 0.039
With GoogleNews vectors	0.708 ± 0.095	0.774 ± 0.118	0.736 ± 0.031
With medical embeddings	0.642 ± 0.089	0.716 ± 0.118	0.673 ± 0.021

Conclusions

- Proposed a novel semi-supervised Bi-directional LSTM based model for ADR mention extraction
- Leveraged a large unlabeled corpus using pre-training for a dummy task of drug name prediction
- Outperformed the state-of-the-art method by 3% in F1-score.
- Demonstrated that word embeddings trained on a large domain-agnostic Twitter corpus performs better than those trained on News Corpus
- In future, we plan to explore drug and ADR mention relation extraction along with ADR extraction in a multi-task learning setup

Retrieval of Prior Court Cases using Witness Testimonies

TATA CONSULTANCY SERVICES
Experience certainty.

TATA

Retrieval of Prior Court Cases using Witness Testimonies

Kripabandhu Ghosh[1], Sachin Pawar[2], Girish K. Palshikar[2], Pushpak Bhattacharyya[3], Vasudeva Varma[4]

[1]Indian Institute of Science Education and Research Kolkata

[2]TCS Research and Innovation

[3]Indian Institute of Technology Bombay

[4]International Institute of Information Technology

Hyderabad December 11th, 2020 (Friday)

TCS RESEARCH & INNOVATION

Motivation

- **Witness** testimony is a cornerstone of court decisions.
- **Prior cases** form the backbone of judicial systems following Common Law; e.g., in India.
- To our knowledge, no prior work attempts to retrieve prior cases from fine grained legal questions (e.g., Which are the cases where the appellant accepted bribe?).

Contributions

- We propose two NLP techniques (linguistic knowledge-based and distantly supervised) to identify sentences of class *Testimony*.
- We extract details of events mentioned in such witness *Testimony* sentences.
- We leverage witness testimony events for retrieval of *Prior Cases*.

Importance of Witness Testimonies

- Witnesses – prosecution or defence, lay or expert – are important in court cases.
- Witness testimonies and their cross-examinations by the counsels affect judges' decision.
- Court judgements contain the judges' summaries of the witness testimonies presented during the proceedings.
- Judges often comment in the judgement on
  - the correctness, quality, completeness and reliability of the testimonies of a witness;
  - the interrelationships between the testimonies of various witnesses (e.g., consistency or contradictions);
  - the impact ("weighing in") of various witness testimonies on their final decision.

TCS RESEARCH & INNOVATION

Identifying TESTIMONY sentences

Identifying TESTIMONY sentences (1/9)

Step 1:Identify those sentences from the corpus which contain candidate witness mentions in any of the following form:

- Explicit mentions, pronoun mentions, PERSON named entities

Explicit mentions:

- Dr. Gupta (PW 1) has stated that he found these injuries on the person of the appellant when he examined him on 7th April , 1986 at 4.10 p.m.

Pronoun mentions: he, she or they

- Hesaid they got into panic as they could be suspected as Tamilians.
- Hefurther stated that the portion of the ground on which the grass was cut was shown to the Police Inspector.

Identifying TESTIMONY sentences (2/9)

PERSON named entity mentions:

- ButDamodar Raoin his evidence denied having made the oral gift or having attested the sale deeds in favour of plaintiffs.
- Shindestated that on reaching the Police Station, he had reported "the matter" to S.I. Patil , who was incharge of the Police Station.

Common nouns having "Person" as their ancestor in WordNet hypernym tree:

- In arbitration proceedings theownersadmitted their liability to the charterers, but contended that payment should be made in cruzeiros.
- Thedoctorstated that the deceased died as a result of cumulative effect of injuries on the lungs and liver and the same were sufficient to cause death in the ordinary course of nature.

Identifying TESTIMONY sentences (3/9)

Step 2:For the sentences which have a mention of a witness in some form, filter out sentences which do not have any statement indicating verbs.

Statement indicating verbs:

- (stated|testifi(ed|es)|testify(ing)?|admitt?ed|mention(s|ed|ing)?|
  acknowledg(e|es|ed|ing)|depos(e|ed|es|ing)|certifi(ed|ed)|
  certify(ing)?|said|say(s|ing)?|narrat(e|ed|es|ing)|denied|deny(ing)?|
  reject(ed|ing)?|inform(ed|ing)?|corroborat(e|ed|ing)|accept(ed|ing)?|
  alleg(e|ed|ing)|disclos(e|ed|ing)|claim(ed|ing)?|narrat(e|ed|ing)|
  refus(e|ed|ing)|describ(e|ed|ing))

Identifying TESTIMONY sentences (4/9)

- Hesaidthey got into panic as they could be suspected as Tamilians.
- Hefurtherstatedthat the portion of the ground on which the grass was cut was shown to the Police Inspector.
- ButDamodar Raoin his evidencedeniedhaving made the oral gift or having attested the sale deeds in favour of plaintiffs.
- Shindestatedthat on reaching the Police Station, he had reported ‘‘the matter’’ to S.I. Patil , who was incharge of the Police Station.
- In arbitration proceedings theownersadmittedtheir liability to the charterers, but contended that payment should be made in cruzeiros.
- Thedoctorstatedthat the deceased died as a result of cumulative effect of injuries on the lungs and liver and the same were sufficient to cause death in the ordinary course of nature.

Identifying TESTIMONY sentences (5/9)

Step 3:For the sentences which are selected in the Step 1 and 2, are further checked to ensure that the statement verb contains within its dependency subtree at least one of the following:

- a clausal complement (ccomp) or
- open clausal complement (xcomp)

Clausal complements (ccomp):

- PW-7, Ganeshdeniedthat he had made any statement to the Police.
- Similarly,PW13alsoadmittedthat other rickshaws were standing at the stand.

Identifying TESTIMONY sentences (6/9)

Open Clausal complements (xcomp):

- PW 17the Investigating Officer initiallydeniedto have seen the intimation of the doctor (Ex. C1), but later on admitted that he had received it.
- PW13Dr.Kuldip Kanwarstatedto have medically examined PW 4 Gangawati who had received simple injurybut the report was not formally proved while recording the statement of the doctor.

Sentences with statement verb but **without any clausal complements**:

- Dilip Kumar,PW-4, Binay Mondal,PW-6, Anukul Chandra,PW-7and Prasanna Kumar,PW-8, alsodeposedto the same effect.
- This is whatPW 2hasstated.

Identifying TESTIMONY sentences (7/9)

Step 4:For the sentences which are selected in the first 3 steps, are further checked to ensure that the statement verb is not negated.

- The statement verb should not have any child in dependency tree with relation "neg"
- PW-2Aruna the victim has alsonotstatedthat she was sexually assaulted by the appellant.
- During investigationPW-1hadnotstatedthat he had seen the accused standing near the dead body of the deceasedor that on hearing her cries her son Venkanna who has not been examined came there and informed the incident to the police by phone.

Identifying TESTIMONY sentences (8/9)

Step 5:For the sentences which are selected in the first 4 steps, are further checked to ensure that:

- The statement verb should have at least one candidate witness mention within its "nsubj" or "agent" dependency subtree, and
- The statement verb should NOT have any "legal role" mention within its "nsubj" or "agent" dependency subtree

Legal roles:

- (counsels?|lawyers?|advocates?|judges?|magistrates?|attorneys?|solicitors?|prosecutors?|pleaders?|solicitors?|justice|bench|munsiff?|sir|court)

Identifying TESTIMONY sentences (9/9)

No witness mention within the “nsubj” / “agent” subtree of the statement verb:

- It issaidthat in between these days some correction slip was filed in the Court seeking 25 corrections in the statement ofPW 34.
- It isstatedthatthePWs 1 & 2were stated to be injured witnesses , but their evidence does not inspire confidence.

Legal role mention within the “nsubj” / “agent” subtree of the statement verb:

- The learnedcounselstatedthatPWs 1, 2 and 3must have come there to attack the appellantsand that the well spoken of by PW-1 in his evidence was the well close to the house of the appellants.
- The SessionsJudgealsostatedin his orderthat the reasons for examining him as a courtwitnesshad been elaborately recorded in the order-sheet dated 17.2.1982 and 22.3.1983.

Evaluation

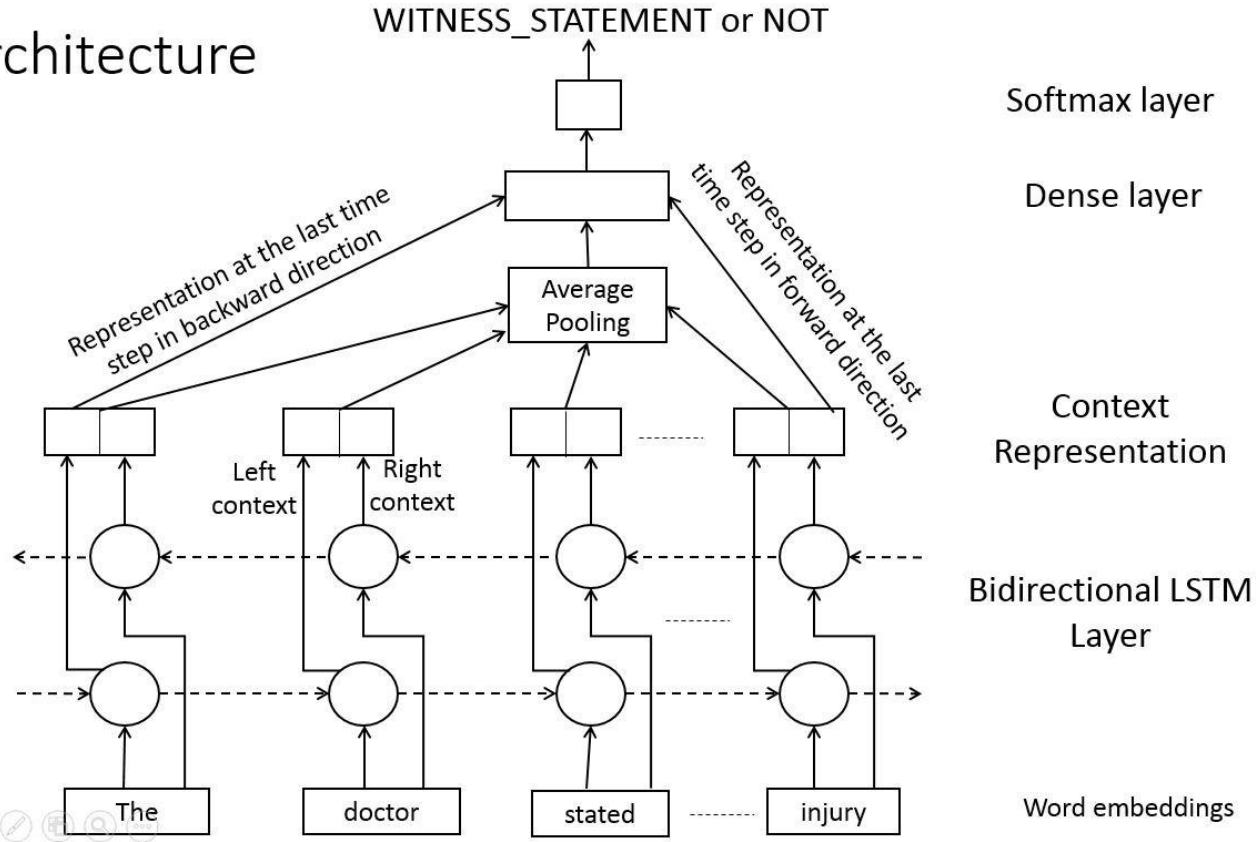
- Dataset: 30,035 Supreme Court judgements from 1950 to 2012
  - Available at http://liiofindia.org/in/cases/cen/INSC/
  - 4,634,075 sentences.
- Rules identified 37572 TESTIMONY sentences, 14382 non-TESTIMONY sentences
- Recall is difficult to estimate
- Precision of 85% is estimated by manually verifying the random samples

Distantly Supervised Sentence Classification (1/2)

- Created the training dataset automatically by using linguistic rules.
  - No. of positive instances = 37572 TESTIMONY sentences identified using linguistic rules
  - No. of negative instances = 14382 non-TESTIMONY sentences identified using linguistic rules + 23190 randomly selected sentences
- Trained a Bi-LSTM based sentence classifier which does not use any dependency information but uses only the sequence information of the words.
- After training, we classify all the remaining sentences in the corpus and select 10000 sentences with highest confidence as TESTIMONY sentences.
- Manually verified 200 random sentences out of these 10000 (P=0.75)
- This classifier clearly learns more patterns over the rule based method.

Distantly Supervised Sentence Classification (2/2)

Architecture



Extraction of Events from TESTIMONY Sentences

Extraction of Events from TESTIMONY Sentences

- Next, we extract events mentioned in TESTIMONY sentences.
  - A witness testimony provides factual or subjective details about various events, objects and persons.
  - We extract information provided by witnesses about various events, and not about the persons or objects, though the approach can be easily extended.
  - We restrict an event to mean a physical action or communication.
- We represent event information provided by a witness as an event frame, consisting of (i) the action verb, (ii) the agent who initiated the action, and (iii) the patient (or beneficiary) who experienced the action.
  - Other event details (e.g., time, location), can be easily extracted.
- We use MatePlus semantic role labeling (SRL) tool, to identify the predicate and associated A0, A1 arguments and fill up event frames.

Example: Semantic Roles

Table:Examples of predicate-argument structures in PropBank style. The A0 (Arg0) argument plays an agent semantic role and A1 (Arg1) plays a patient/theme semantic role.

S_1: P.W. 1 to 5 have stated that the appellant assaulted the deceased with a crow bar on his head. **Predicate**: assaulted **A0 (agent)**: the appellant, **A1 (patient)**: the deceased
Q_1: Which are the cases where the appellant has attacked the deceased? **Predicate**: attacked **A0 (agent)**: the appellant, **A1 (patient)**: the deceased

Prior Case Retrieval

Prior Case Retrieval using Events in Witness Testimonies

- We use events extracted from witness testimonies for improving retrieval of relevant past cases (prior cases) based on high-level English queries which might be asked by a lawyer or a lay person.
  - Prior cases form the backbone of judicial systems following Common Law; e.g., in India.
- For prior case retrieval, we propose two techniques.
- **M1**: *Exact Semantic Match* of the event frames (one from the query and another from a past court judgement).
- **M2**: *Semantic Match using Event Frame Representation* by learning a representation for event frames and then using a similarity measure over event frames.

Exact Semantic Match (M1)

- We find the similarity of a query, *Q* with each sentence, *S* (e.g., S1 in Table) in a candidate prior case document *D*
- We match the predicate-argument structure of *Q* with that of *S* , where the corresponding predicate and arguments are matched. That is, the Predicate in *Q* is matched with the Predicate in *S* , the A0 in *Q* is matched with the A0 in S and so on.
- The similarity between *Q* and *S* is $SIM_S(Q, S) = \frac{\sum_r match(Q_r, S)}{|Q|}$, where $r \in$ {Predicate, A0, A1} (semantic roles), match(.) = 1 if there is an exact match, 0 otherwise; $|Q|$ is the number of non-null arguments in *Q*.
- The document level similarity, i.e. between *Q* and *D* is calculated using $SIM(Q, D) = max_S\, SIM_S(Q, S), S \in D$.

Semantic Match using Event Frame Representation (M2)

- Learning a representation for complete event frame structure (predicate, A0, A1)
- Train a denoising autoencoder by masking either predicate, A0 or A1 of an event frame at a time and trying to reconstruct the complete frame.
- Input layer accepts a vector (of 900 dimensions) which is a concatenation of 300-dimensional pre-trained word vectors corresponding to predicate, A0 and A1, where any one of these is masked by using a zero vector.
- Next layer is a fully connected dense layer of 300 dimensions.
- Finally, the output layer is again a 900-dimensional layer reconstructing the original concatenated vector corresponding to the complete frame.
- Once this autoencoder is trained, its encoder part (i.e. first two layers) is used to obtain embedded 300-dim representation of any event frame.
- Similarity is calculated as $SIM(Q, D) = max_S \bar{cosine}\ sim(Repr(Q), Repr(S))$, where $Repr(x)$ is the representation of a frame x .

Experimental Setup

Baseline s

- **BM25 (B1)**[1]: popular term scoring models based on bag-of-words i.e. it does not consider the relative ordering of the words in the query and documents.
- **Doc2Vec (B2)**[2]: neural model that offers representations (embeddings) of a piece of text (sentence, paragraph and document). It overcomes the drawbacks of BoW models by incorporating the relative ordering of words in a text in the embeddings.
- **Sentence-BERT (B3)**[3]: A recent technique for obtaining sentence embeddings using Siamese-BERT networks.

---

[1]Robertson S.E., Walker S. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. SIGIR 1994

[2]Le Q., Mikolov T. Distributed representations of sentences and documents. ICML 2014

[3]Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP-IJCNLP 2019

Ground Truth Creation

- As there is no publicly available ground truth for our queries, we apply the *pooling technique* for selection of candidate documents for annotation.
- We run several ranking models (including our own techniques) on the collection (30,035 Supreme Court judgements from 1950 to 2012) and select top 20 documents for each model to form a pool which we annotate manually.

Evaluation Measures

- Average Precision (AP): incorporates the relative ranking order of relevant documents; combines the joint effect of Precision and Recall.
- R-Precision (R-Prec): Precision at R, the number of relevant documents

Results (1/2)

Query	R-Precision (R-Prec)				
	B1	B2	B3	M1	M2
q1: Which are the cases where a husband has set his wife on fire?	0.13	0.00	0.50	0.63	0.63
q2: Which are the cases where the appellant has attacked the deceased?	0.21	0.10	0.24	0.28	0.45
q3: Which are the cases where the respondent killed the deceased?	0.00	0.00	0.0	1.00	1.00
q4: Which are the cases where the appellant demanded money?	0.06	0.13	0.0	0.56	0.75
q5: Which are the cases where the respondent has forged signatures?	0.00	0.00	0.25	0.75	0.75
q6: Which are the cases where the appellant accepted bribe?	0.00	0.00	0.17	0.33	0.50
q7: Which are the cases where an appointment was challenged?	0.14	0.14	0.00	0.43	0.57
q8: Which are the cases where an election was challenged?	0.08	0.31	0.08	0.38	0.46
q9: Which are the cases where the complainant was beaten by wife?	0.00	0.00	1.00	1.00	1.00
q10: Which are the cases where the respondent has admitted the charge?	0.00	0.00	0.00	1.00	1.00
Average over all queries	0.06	0.07	0.22	0.64	**0.71**

Results (2/2)

Query	Average Precision (AP)				
	B1	B2	B3	M1	M2
q1: Which are the cases where a husband has set his wife on fire?	0.13	0.00	0.54	0.70	0.89
q2: Which are the cases where the appellant has attacked the deceased?	0.10	0.06	0.09	0.28	0.51
q3: Which are the cases where the respondent killed the deceased?	0.00	0.00	0.17	1.00	1.00
q4: Which are the cases where the appellant demanded money?	0.03	0.07	0.02	0.56	0.76
q5: Which are the cases where the respondent has forged signatures?	0.05	0.00	0.17	0.95	0.62
q6: Which are the cases where the appellant accepted bribe?	0.02	0.00	0.10	0.33	0.43
q7: Which are the cases where an appointment was challenged?	0.04	0.05	0.00	0.43	0.63
q8: Which are the cases where an election was challenged?	0.01	0.15	0.04	0.38	0.50
q9: Which are the cases where the complainant was beaten by wife?	0.00	0.00	1.00	1.00	1.00
q10: Which are the cases where the respondent has admitted the charge?	0.00	0.00	0.00	1.00	1.00
Average over all queries	0.04	0.03	0.21	0.66	**0.73**

Analysis (1/2)

- Exact Semantic Match (M1) and Semantic Match using Event Frame Representation (M2) outperform the baselines for all the queries and in both the evaluation measures, by a considerable margin.
- M2 outscores M1 on most queries.
- **Utility of Witness Testimony sentences**
  - We considered complete documents for BM25 as against only witness Testimony sentences.
  - BM25 could not find even a single relevant document within top 10 for all the queries.
  - Hence, we run all the experiments considering only the witness Testimony sentences.

Analysis (2/2)

- **Notion of interpretability**
  - In the query q1 (Which are the cases where a husband has set his wife on fire?), the predicate-arguments are: Predicate: set, A0: husband, A1: wife.
  - This semantically captures an event and matches it with a prior case where a similar event has occurred e.g., a husband has poured kerosene on his wife and set her on fire, based on the similarity of the semantic argument structure.
- **Semantic match**
  - For the query q2 (Which are the cases where the appellant has attacked the deceased?), M2 is able to retrieve the document containing the sentence P.W. 1 to 5 have stated that the appellant assaulted the deceased with a crow bar on his head.
- The variation in AP and R-Prec scores in queries is due to the fact that some queries are more general (e.g., q8) i.e., having higher number of relevant documents than some other queries which are specific (e.g., q9).

Future Work

- We look to increase the number of queries.
- We look to do experiment on more complex queries.
- We look to work on other court case collections.

Prior Case Retrieval using Evidence Extraction from Court Judgements

Basit Ali, Ravina More, Sachin Pawar and Girish K. Palshikar

TCS Research, Tata Consultancy Services Limited, Pune, India.

ASAIL @ ICAIL 2021

Link: http://ceur-ws.org/Vol-2888/paper1.pdf

25th June 2021

Motivation

- Evidence & Witness Testimony are the Cornerstone of court decisions.
- Prior Cases form the backbone of judicial systems following Common Law; e.g., in India.
  - `What are the cases where blood stains were found on clothes of the deceased?`
- Hence, prior case retrieval using evidence and witness testimony information is the focus of this work.
- Contributions:
  - Two NLP-based techniques used to identify Evidence and Testimony sentences – linguistic rules and weakly supervised Bi-LSTM sentence classifier.
  - Evidence Structure Instances – For representing events described in Evidence and Testimony sentences.
  - Leveraging this structured representation for retrieval of Prior Cases.

Examples Of EVIDENCE & TESTIMONY sentences

TCS RESEARCH & INNOVATION

EVIDENCE:

- A bullet which was marked as Art C was found in the body of the deceased.
- In the affidavit dated 3rd January, 1999 of Sri Ramji Patel filed on behalf of the petitioner , it has been stated that they have entered into an agreement with the Sunraj Construction Company for the construction of the bio-gas plant of 45 cubic meter capacity.
- The same blood group was found on the clothes recovered from the appellants .

TESTIMONY:

- The non seizure of blood from the cot is concerned, the investigating officer has stated that he found blood stained earth at the place of occurrence and had seized it.
- He stated in his evidence when he had seen the accused hurling bombs at him he requested the deceased to slow down the scooter.

Importance of Evidence Sentences

TCS RESEARCH & INNOVATION

- Evidence - A **document** (e.g., `letter, receipt, report`) or a **physical object** (e.g., `knife, guns, photos`) used by lawyers in their arguments in court cases.
- Observations made through these evidences may have a significant effect on the judge's final decision.
- Court Judgements represent the evidence object as `Exhibit 7`, `Evidence 23` or explicitly mentions it such as `post-mortem report`.

- Evidences can help in the court Judgements in the following ways:
  - Determining the strength & the weakness of the arguments.
  - Identifying relevant past cases.

Importance of Witness Testimony Sentences

- Our prior work Ghosh et al. in JURIX 2020 focused on Witness Testimonies.
- Witnesses – prosecution or defence, lay or expert – are important in court cases.
- Witness testimonies and their cross-examinations by the counsels affect judge's decision.
- Court judgements contain the judges' summaries of the witness testimonies presented during the proceedings.
- Judges often comment in the judgement on:
  - the correctness, quality, completeness, and reliability of the testimonies of a witness;
  - the interrelationships between the testimonies of various witnesses (e.g., consistency or contradictions);
  - the impact (``weighing in'') of various witness testimonies on their final decision.

Identifying EVIDENCE and TESTIMONY sentences

Overall Approach

- EVIDENCE and TESTIMONY sentences are identified in two steps.

- Step 1 – **Linguistic rules** to identify EVIDENCE and TESTIMONY sentences with high precision.

- Step 2 – **Weakly supervised sentence classifier**
  - Use the sentences identified in the first step for automatically creating training data for a sentence classifier.

Linguistic Rules for Identifying EVIDENCE sentences

- **Rule 1**: Identify sentences containing at least one *Evidence Object*.
  - Documents (`autopsy report`, `post-mortem report`, `affidavit`, `letter`, `cheque`, `agreement`, `petition`, `FIR`, `signature`)
  - Material objects (`gun`, `bullet`, `clothes`)
  - Substances (`poison`, `alcohol`, `kerosene`)

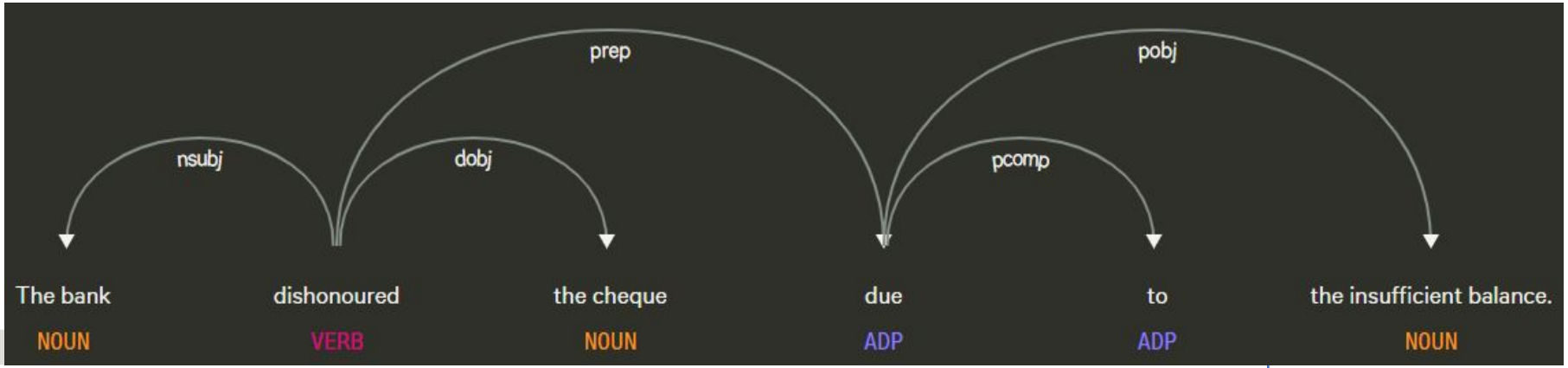
List of all words for which the following WordNet synsets are ancestors in hypernym tree – artifact, document, and substance.

S: (n) **bullet**, slug (a projectile that is fired from a gun)
- *direct hyponym* / *full hyponym*
- *direct hypernym* / ***inherited hypernym*** / *sister term*
  - S: (n) projectile, missile (a weapon that is forcibly thrown or projected at a targets but is not self-propelled)
    - S: (n) weapon, arm, weapon system (any instrument or instrumentality used in fighting or hunting) *"he was licensed to carry a weapon"*
      - S: (n) instrument (a device that requires skill for proper use)
        - S: (n) device (an instrumentality invented for a particular purpose) *"the device is small enough to wear on your wrist"; "a device intended to conserve water"*
          - S: (n) instrumentality, instrumentation (an artifact (or system of artifacts) that is instrumental in accomplishing some end)
            - S: (n) artifact, artefact (a man-made object taken as a whole)

S: (n) **kerosene**, kerosine, lamp oil, coal oil (a flammable hydrocarbon oil used as fuel in lamps and heaters)
- *direct hyponym* / *full hyponym*
- *direct hypernym* / ***inherited hypernym*** / *sister term*
  - S: (n) fuel (a substance that can be consumed to produce energy) *"more fuel is needed during the winter months"; "they developed alternative fuels for aircraft"*
    - S: (n) substance (a particular kind or species of matter with uniform properties) *"shigella is one of the most toxic substances known to man"*

Linguistic Rules for Identifying EVIDENCE sentences

- **Rule 2**: Sentences identified by Rule 1 should also contain:
  - At least one action verb (`tamper`, `kill`, `sustain`, `forge`, etc.) OR,
  - At least one observation verb (`report`, `show`, `find`, `discovered`).
- **Rule 3**: In the dependency tree of a sentence identified by both Rules 1 and 2, the evidence object (Rule 1) should occur within the subtree rooted at the action or observation verb (Rule 2). Moreover, there should not be any other verb (except aux verbs: `has`, `have`, `been`, `was`, `were` etc.) between the two.



Examples of EVIDENCE sentences

- The bank dishonoured **the cheque** due to insufficient balance.

- The evidence of PW 65 D.B.Bhagve reveals that one Ravinder Sharma had purchased **a bank draft** of Rs. 15,000 from the Bank of Baroda, Pune, on 25th January, 1986 in the name of Neelam Madan.

- **The FSL report Exhibit P77** had clearly established that the blood of group ` O ' was found on **the clothes** of the deceased and that was her blood group .

- The **report** revealed that **organo-phosphorus compound** was found in the stomach, small intestines, large intestines and brain of the deceased.

Linguistic Rules for Identifying TESTIMONY sentences

TCS RESEARCH & INNOVATION

- Detailed rules are covered in our prior work – Ghosh et al., JURIX 2020.

- **He** also denied that masons were working there .
- **Shinde** stated that on reaching the Police Station, he had reported "the matter'' to S.I. Patil , who was in charge of the Police Station.
- In answer to another question **he** stated that Mohinder Singh was empty handed .
- **He** said they got into panic as they could be suspected as Tamilians.
- It is proved that in her police Statement , **P.W.3** had stated that Vishnu was following while she was going to the shop of appellant No . 1 and had not referred to Shankar- P.W. 4 at all .

Linguistic Rules : Evaluation

TCS RESEARCH & INNOVATION

- Dataset: **30,032** Supreme Court judgements from 1950 to 2012.
  - Available at http://liiofindia.org/in/cases/cen/INSC/
  - **4,111,091** sentences, where the average sentence length is 31 words and standard deviation of 24.
- Linguistic Rules identified **62,310** EVIDENCE sentences.
- Linguistic Rules identified **37,572** TESTIMONY sentences.

- Evaluated by manually verifying random samples of 100 sentences.

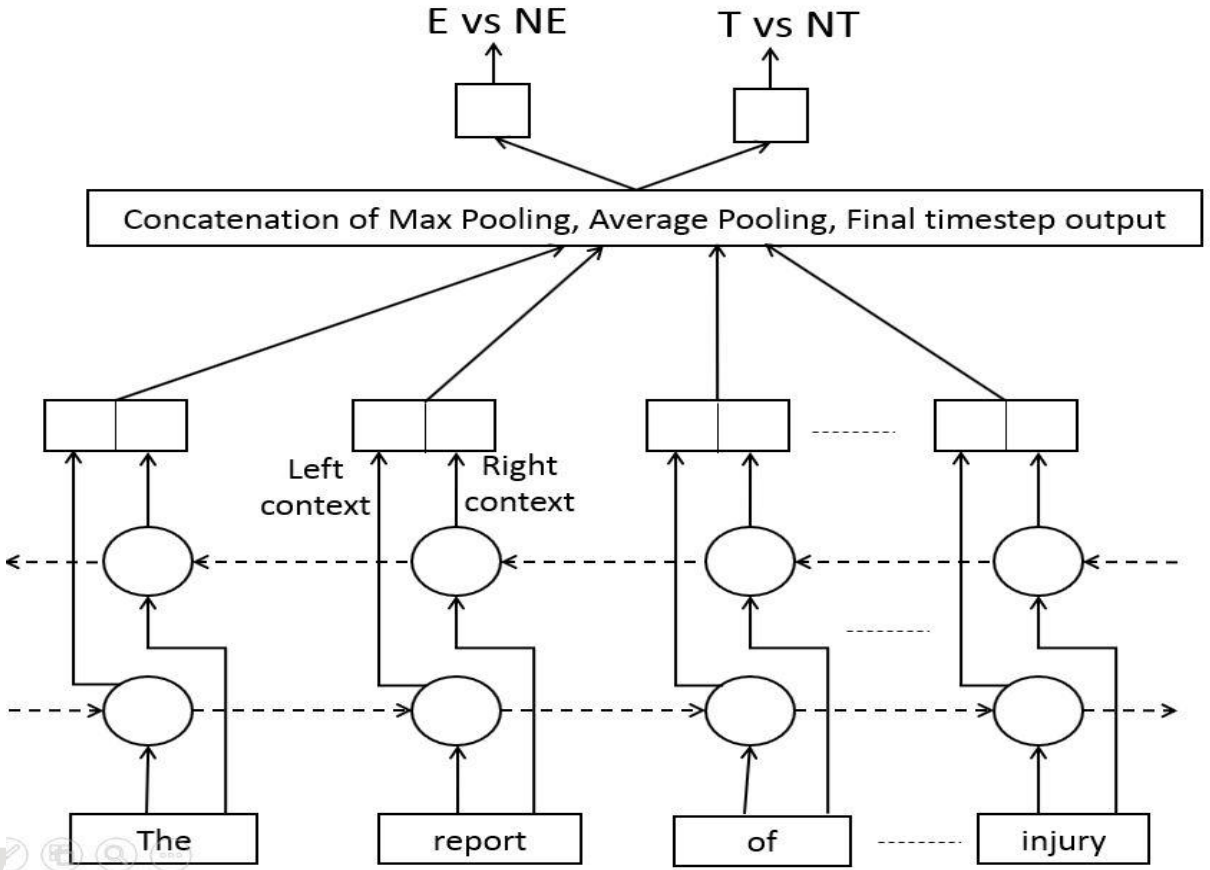
- The Precision of 85% is estimated for both EVIDENCE & TESTIMONY sentences.

Weakly supervised sentence classifier

- Training data is created automatically by using the linguistic rules.
  - 1824 sentences are labelled as Evidence & Testimony both
  - 60486 sentences are labelled as Evidence & Non-Testimony
  - 34649 sentences are labelled as Non-Evidence & Testimony
  - 14234 sentences are labelled as Non-Testimony & Non-Evidence
- Trained a BiLSTM Multi-Label Sentence Classifier, which does not use any dependency information but uses only a sequence of the words in a sentence.
- Then classified all the remaining sentences in the corpus and selected top 10000 EVIDENCE sentences and top 5000 TESTIMONY sentences.

- Evaluation using manually verifying random samples of 100 sentences.
  - Observed precision of 72% for EVIDENCE & 68% for TESTIMONY sentences.

Architecture of the sentence classifier



Softmax layer

Overall sentence representation

Word-level Context Representation

Bidirectional LSTM Layer

Word embeddings

Representation using Evidence Structure Instances

Evidence Information Model

- Goal: To define a suitable structure to represent evidence information mentioned in court judgements.
- The Evidence Information Model represents every EVIDENCE Sentence giving information about one or more Evidence Objects in an Evidence Structure .
  - Evidence structure consists of an optional *Observation Frame* and a mandatory *Evidence Frame* .
  - Each frame contains a set of arguments corresponding to **semantic roles** such as A0, A1, CAU, MNR, LOC.

- Evidence Structure Instances: Instantiations of Evidence Structure for a particular EVIDENCE sentence.
  - The Same structure is used for TESTIMONY sentences as well.

Examples of Evidence Structure Instances

- The autopsy report reveals that some poisonous compounds are found in the stomach of the deceased.
  - **OF** = **[OV** = reveals, **A0** = The autopsy report, **EO** = The autopsy report**]**
  - **EF** = **[EV** = found, **A1** = some poisonous compounds, **LOC** = in the stomach of the deceased**]**
- The bank dishonoured the cheque due to insufficient balance.
  - **EF** = **[EV** = dishonoured, **A1** = the cheque, **CAU** = due to insufficient balance**]**
- He has categorically stated that by reason of enmity , A1 and A2 together have murdered his brother-in-law .
  - **OF** = **[OV** = stated, **A0** = He**]**
  - **EF** = **[EV** = murdered, **A0** = A1 and A2 together, **A1** = his brother-in-law, **CAU** = by reason of enmity**]**

Prior Case Retrieval

Prior Case Retrieval

- Goal: To demonstrate effectiveness of the proposed Evidence Structure, we apply it for the task of prior case retrieval
- Task: To create a relevance-based ranked list of court judgements (documents) in our corpus for a query
  - Query is an English sentence which may be asked by a lawyer or a lay person

- Proposed technique: *SemMatch (SM)*
  - A query is represented using Evidence Structure Instance ($EvStruct_Q$)
  - It is matched against each Evidence Structure Instance ($EvStruct_D$) obtained from EVIDENCE and TESTIMONY sentences present in each document
  - The matching score indicates the extent of semantic similarity between two Evidence Structure Instances
  - The highest matching score within a document is considered as its relevance score with respect to the query

Example of matching using SemMatch (1/2)

- Query: The autopsy report reveals that some poisonous compounds are found in the stomach of the deceased.

  $EvStruct_Q$: **OF** = **[OV** = reveals, **EO** = The autopsy report**]**

  EF = **[EV** = found, **A1** = some poisonous compounds, **LOC** = in the stomach of the deceased**]**

- Sentence: The report of the Chemical Examiner showed that a heavy concentration of arsenic was found in the viscera.

  $EvStruct_D$: **OF** = **[OV** = showed, **EO** = The report of the Chemical Examiner**]**

  EF = **[EV** = found, **A1** = a heavy concentration of arsenic, **LOC** = in the viscera**]**

Example of matching using SemMatch (2/2)

- Similarity between main predicates
  - $Sim_E = CosineSim\big(WordVec(found), WordVec(found)\big) = 1.0$
- Similarity between arguments
  - $sim_{A1} = CosineSim\begin{pmatrix} PhraseVec(some\ poisonous\ compounds), \\ PhraseVec(a\ heavy\ concentration\ of\ arsenic) \end{pmatrix} = 0.5469$
  - $sim_{LOC} = CosineSim\begin{pmatrix} PhraseVec(in\ the\ stomach\ of\ the\ deceased), \\ PhraseVec(in\ the\ viscera) \end{pmatrix} = 0.3173$
  - $sim_{args} = (sim_{A1} + sim_{LOC})/2 = 0.4321$
- Similarity between Evidence Objects
  - $sim_{EO} = CosineSim\begin{pmatrix} PhraseVec(the\ autopsy\ report), \\ PhraseVec(the\ report\ of\ the\ Chemical\ Examiner) \end{pmatrix} = 0.8641$
- Final similarity
  - $sim_{final} = sim_E \times sim_{args} \times sim_{EO} \times sim_{SBERT} = 1.0 \times 0.4321 \times 0.8641 \times 0.607 = \mathbf{0.2266}$

Experimental Analysis

Experimental Analysis (1/3)

- **Baselines**:
  - BM25: TF-IDF based relevance computation technique. This does not consider the relative ordering of the words in the query & documents.
  - Sentence-BERT: Siamese-BERT based network to obtain more meaningful sentence embedding than vanilla BERT (Reimers and Gurevych, EMNLP-IJCNLP 2019).
- **Dataset**: 30,032 Supreme Court judgements from 1950 to 2012
- **Queries**: 10 queries which are diverse in nature in terms of the type of case (domestic violence, financial fraud, etc.) and the evidence object in focus.
- **Ground Truth:**
  - We apply the pooling technique for selection of candidate documents for annotation.
  - We run several ranking models (Baselines + our own technique SemMatch) on the collection and select top 10 documents from each model to form a pool which we annotate manually.

Experimental Analysis (2/3)

- **Evaluation Metrics**:
  - Average Precision (AP): This captures the joint effect of Precision & Recall. This incorporates the relative order of the relevant documents.
  - R-Precision (R-Prec): Precision at R, the number of relevant documents.
- **Experimental Results (Queries):**

What are the cases where···

Q_1: blood stains were found on clothes of the deceased.
Q_2: the deceased had attacked some person with sticks.
Q_3: the police has murdered the deceased.
Q_4: some evidence shows that the exhibited gun was not used.
Q_5: the autopsy report reveals that some poisonous compounds are found in the stomach of the deceased.
Q_6: the deceased is attacked with a knife.
Q_7: a letter by the deceased reveal that dowry was demanded.
Q_8: a cheque was dishonoured due to insufficient funds.
Q_9: bribe was demanded by police.
Q_{10}: a signature was forged on an affidavit.

Experimental Analysis (3/3)

- Evaluation results in the form (R-Prec, Avg. Precision).
- Subscripts all, T, E and TE denote using all the sentences, using only TESTIMONY sentences, using EVIDENCE sentences and using both types of sentences, respectively.
- SM: SemMatch, SB: Sentence-BERT

Query	$BM25_{all}$	$BM25_T$	$BM25_E$	$BM25_{TE}$	SB_T	SB_E	SB_{TE}	SM_T	SM_E	SM_{TE}
Q_1	0.24; 0.26	0.06; 0.02	0.59; 0.49	0.59; 0.52	0.00; 0.01	0.24; 0.15	0.18; 0.14	0.00; 0.01	0.24; 0.16	0.24; 0.14
Q_2	0.25; 0.43	0.00; 0.05	0.00; 0.04	0.00; 0.06	0.00; 0.01	0.00; 0.00	0.00; 0.00	0.25; 0.14	0.25; 0.25	0.50; 0.30
Q_3	0.00; 0.01	0.00; 0.03	0.33; 0.33	0.33; 0.35	0.33; 0.12	0.00; 0.00	0.00; 0.09	0.33; 0.12	0.00; 0.00	0.33; 0.12
Q_4	0.17; 0.06	0.00; 0.01	0.00; 0.02	0.00; 0.04	0.00; 0.01	0.42; 0.25	0.42; 0.22	0.08; 0.04	0.25; 0.27	0.33; 0.29
Q_5	0.30; 0.43	0.10; 0.05	0.40; 0.35	0.40; 0.37	0.20; 0.15	0.70; 0.80	0.70; 0.80	0.00; 0.02	0.40; 0.40	0.40; 0.40
Q_6	0.31; 0.42	0.33; 0.28	0.38; 0.35	0.46; 0.52	0.23; 0.14	0.33; 0.38	0.36; 0.40	0.20; 0.18	0.28; 0.27	0.41; 0.42
Q_7	0.25; 0.35	0.00; 0.08	0.50; 0.54	0.50; 0.33	0.00; 0.04	0.00; 0.12	0.00; 0.09	0.25; 0.06	0.00; 0.00	0.25; 0.06
Q_8	0.48; 0.46	0.01; 0.09	0.67; 0.71	0.71; 0.73	0.05; 0.02	0.62; 0.67	0.62; 0.67	0.00; 0.00	0.57; 0.63	0.57; 0.64
Q_9	0.20; 0.23	0.20; 0.17	0.20; 0.21	0.40; 0.31	0.40; 0.39	0.20; 0.21	0.50; 0.51	0.40; 0.41	0.10; 0.12	0.50; 0.48
Q_{10}	0.50; 0.52	0.00; 0.11	0.25; 0.16	0.25; 0.21	0.00; 0.01	0.00; 0.04	0.00; 0.03	0.25; 0.13	0.50; 0.61	0.50; 0.61
Avg	0.27; 0.32	0.08; 0.09	0.33; 0.32	0.36; 0.34	0.12; 0.09	0.25; 0.26	0.28; 0.30	0.18; 0.11	0.26; 0.27	0.40; 0.35

Conclusions and Future Work

TCS RESEARCH & INNOVATION

- We discussed several NLP techniques for:
  - identifying EVIDENCE and TESTIMONY sentences from court judgements.
  - representing them in the semantically rich Evidence Structure.
  - retrieving relevant prior cases exploiting this structure.

- The proposed techniques are weakly supervised
  - No dependence on manually annotated training data, except for the reliance on human expertise in designing the linguistic rules.

- In future, we plan to learn the embedded representation of an entire Evidence Structure Instance.
- We plan to explore an ensemble of multiple retrieval techniques for improving prior case retrieval performance.

Semantic Relation Extraction from Resumes

TATA CONSULTANCY SERVICES
Experience certainty.

TATA

Extraction of Complex Semantic Relations from Resumes

ASEA @ IJCAI 2021

Sachin Pawar, Devavrat Thosar, Nitin Ramrakhiyani,

Girish K. Palshikar, Anindita Sinha Banerjee, Rajiv Srivastava

TCS Research

21-AUG-2021

TCS RESEARCH & INNOVATION

Motivation

- Resumes often contain extensive details about the resume writer:
  - personal details, education, work history, skills, roles, projects, trainings, certifications, publications, patents, awards, achievements
- We model information extraction from resumes as a problem of extracting *complex* semantic relationswhere each relation type has the following characteristics:
  1. it may have more than 2 entity arguments (N-ary),
  2. argument entity mentions may span multiple sentences (cross-sentence), and 3. some argument entity mentions may be absent (partial mentions).
- **Challenge**: Lot of variation in the structure, contents, and styles of resumes
  - across languages, countries, functional areas (e.g., engineering, finance, marketing), and industrial domains (e.g, banking, IT, pharma)

Summary of the Proposed Technique

- Our approach is based on sectioning a document into meaningful chunks of consecutive sentences.
- Each such chunk is expected to capture all the argument entity mentions of a single N-ary cross-sentence relation mention.
- Moreover, we jointly model the following two tasks:
  - task of extraction of entity mentions (as word-level sequence labelling) and
  - task of identification of sentence chunks corresponding to a single relation mention (as sentence-level sequence labelling)

Relation Types

- CAREER – a ternary relation indicating that the candidate has worked for an Employer with a specific Designation for a specific Duration
- EDU – a 4-ary relation indicating that the candidate has obtained a Degree from an Institute in a specific YearOfPassing and with specific Marks

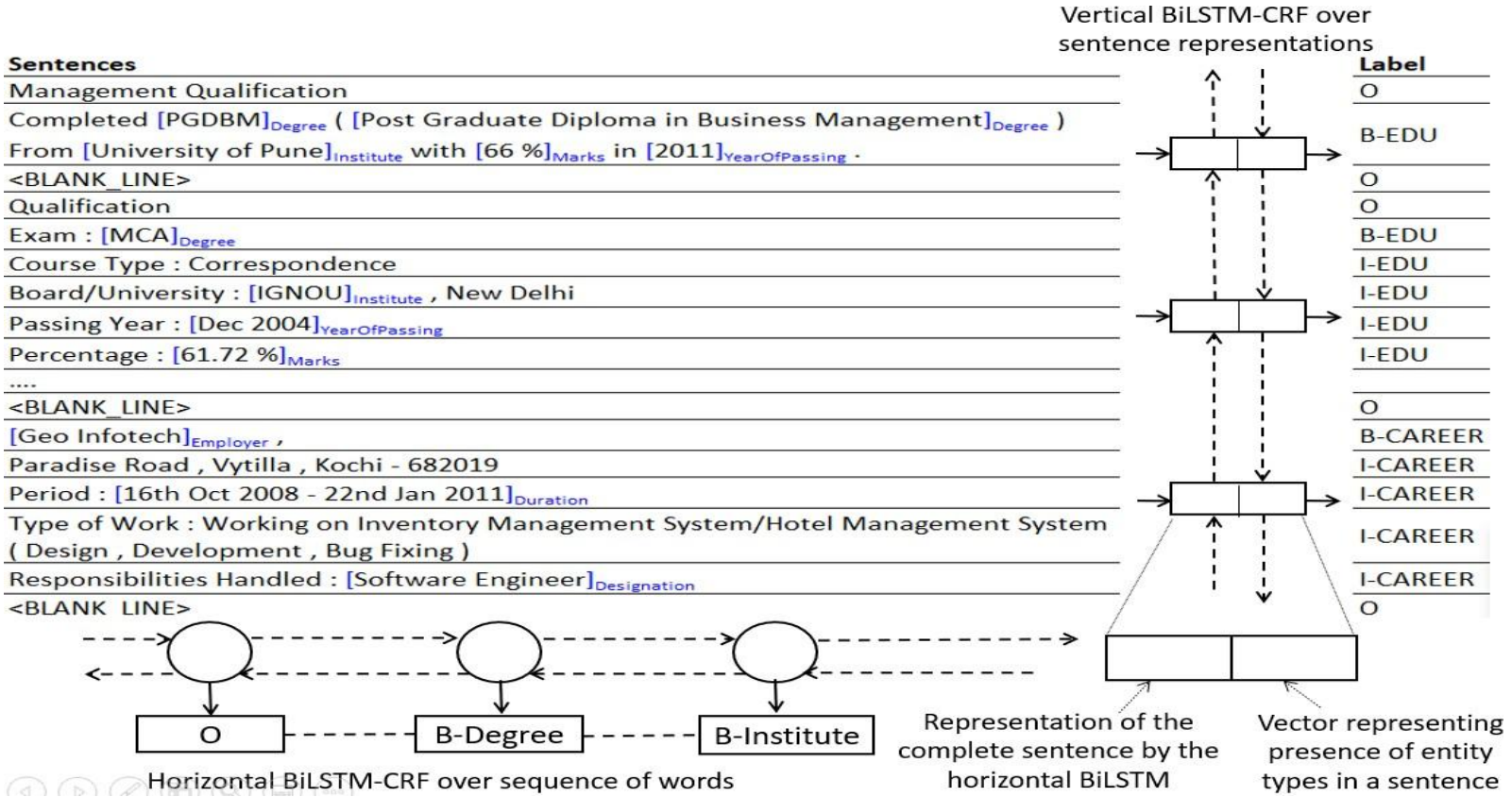
Relation	Relation Mention
EDU	(MCA, 61.72 %, IGNOU, Dec 2004)
EDU	(Post Graduate Diploma in Business Management, NA, University of Pune, 2011)
CAREER	(GXX Infotech, 16th Oct 2011 - 22nd Jan 2014, Software Engineer)

Table: Examples of relation mentions of the relation types EDU (Degree, Marks, Institute, YearOfPassing) and CAREER (Employer, Duration, Designation)

Problem Definition

- **Input:** Resume document X^{test}
- **Output:** List of entity mentions and relation mentions extracted from X^{test}
- **Training regime:** *n* training resumes $\{(X_1^{train}, L_{h_1}^{train}, L_{v_1}^{train}), \cdots (X_n^{train}, L_{h_n}^{train}, L_{v_n}^{train})\}$
  - L_{h_i} are the word-level labels (using BIO encoding, e.g., B-Employer, B-Degree, O) for each word in each sentence in X_i^{train}
  - $L_{v_i}^{train}$ are the sentence-level labels (e.g., B-CAREER, B-EDU, I-EDU, O) for each sentence in X_i^{train}
- **Optional Inputs:** We assume that two independent entity extraction techniques are available:
  - E_{rules} which identifies entity mentions using linguistic rules and gazetteers of known degrees, designations, employers and educational institutes.
  - E_{CRF} which is a traditional CRF-based entity extractor based on manually engineered features. It requires training data having word-level annotations similar to L_{h_i}.

Joint Model Architecture (1/3)



Joint Model Architecture (2/3)

- **Input representation:** Each word in each sentence of a resume is represented as a concatenation of
  - static pre-trained word embedding, and
  - corresponding POS tag and NER tag embeddings which are learnt while training
- **Hierarchical BiLSTM structure:** A horizontal BiLSTM layer over sequence of words feeds into a vertical BiLSTM over sequence of sentences.
- **Horizontal BiLSTM-CRF layer:** A sequence of word representations are fed as input and entity type labels are predicted for each word.
- **Vertical BiLSTM-CRF layer:** A sequence of sentence representations obtained from horizontal BiLSTM layer is fed as input and relation type labels are predicted for each sentence.

Joint Model Architecture (3/3)

- **Sequence Autoencoder:** Shares the same horizontal BiLSTM layer as its encoder layer
  - Encoder accepts a sequence of words as input and outputs a sentence represention
  - Decoder LSTM accepts this representation as input at each time step, and tries to reconstruct the original word representations
- **Pipeline model:** Sequential training
  - Only horizontal BiLSTM-CRF layer is first trained using word-level labels
  - Then only vertical BiLSTM-CRF layer is trained using sentence-level labels
- **Joint model:** Both the horizontal and vertical BiLSTM-CRF layers are trained jointly using both word and sentence level labels

Experimental Analysis: Dataset

- **Training Datasets:**
  - 2248 resumes for training oursequence auto-encoder
  - 1256 resumes for training E_{CRF} and pre-training ofhorizontal BiLSTM-CRF layerin our joint model, and
  - 642 resumes for training ourcomplete joint modelfor jointly identifying entity mentions and sentence chunks.

- **Evaluation Dataset:** A manually annotated dataset of 175 resumes containing 597 and 648 gold-standard relation mentions of EDU and CAREER, respectively.

- All the resumes in our dataset are originally in DOC, DOCX or PDF form which are converted to plain text format.

Experimental Analysis: Baseline and Evaluation

- **Rule-based baseline:**
  - Assumes that entity mentions have been already extracted
  - Starts from an entity mention which is a pivot entity argument for a relation type and then attaches entity mentions of other entity arguments in the vicinity (±4 sentences) to construct a relation mention
  - Several constraints and exceptions incorporated in this attachment decision
  - Similar to an expert system with hand-coded rules
- **Evaluation:**
  - True Positive: A gold-standard relation mention of type *r* for which there is a "matching" predicted relation mention of type *r*
  - False Negative: A gold-standard relation mention of type *r* for which there is no "matching" predicted relation mention of type *r*
  - True Positive: A predicted relation mention of type *r* which is not a true positive
  - Two relation mentions are considered to be **matching** only if ALL of their corresponding entity mention arguments are matching with at least 80% string similarity between them

Evaluation Results: Relation Extraction

	EDU			CAREER			Overall
	P	**R**	**F1**	**P**	**R**	**F1**	**Macro-F1**
Baseline	0.633	0.566	0.598	0.520	0.511	0.516	0.557
Baseline w/o E_{rules} and E_{CRF}	0.612	0.514	0.559	0.569	0.444	0.499	0.529
Pipeline model	0.707	0.672	**0.689**	0.673	0.582	0.624	0.657
Pipeline model w/o E_{rules} & E_{CRF}	0.620	0.533	0.573	0.622	0.478	0.541	0.557
Joint model	0.714	0.656	0.684	0.706	0.585	**0.640**	**0.662**
Joint model w/o E_{rules}	0.708	0.62	0.661	0.693	0.542	0.608	0.635
Joint model w/o E_{CRF}	0.709	0.648	0.677	0.695	0.556	0.618	0.648
Joint model w/o E_{rules} & E_{CRF}	0.648	0.533	0.585	0.641	0.442	0.522	0.554

Table: Relation extraction performance on the test dataset of 175 resumes (averaged over 3 runs)

Evaluation Results: Entity Extraction

Entity type	Precision	Recall	F1
Degree	0.874	0.805	0.838
Marks	0.940	0.853	0.894
Institute	0.890	0.827	0.857
YearOfPassing	0.959	0.894	0.925
Employer	0.937	0.813	0.871
Duration	0.922	0.753	0.829
Designation	0.877	0.720	0.791

Table:Entity extraction performance on the test dataset for the "Joint model" setting (only considering entity mentions which are part of at least one gold or predicted relation mention)

Conclusions and Future Work

- Proposed to model education and career details mentioned in resumes in the form of well-defined semantic relations.
- These relations are complex; they can be N-ary,cross-sentence and may allow partial relation mentions with empty entity arguments.
- Proposed a joint neural model based technique for extracting mentions (tuples) of such complex semantic relations, along with entity mentions.
- Our technique is embedded in a larger resume information extraction system of our organization which is currently in use by several customers.
- In future, we wish extend our technique for extraction of similar complex semantic relations for domains other than resumes.

Thank you! (Questions ?)