Yufei Lin

Final Project

May 13th 2019

Computational Linguistics

Information Extraction

This document is a note on my reading of the textbook *Speech and Language Processing*. Therefore, most example tables and graphs comes from the book, unless otherwise stated.

I. Named Entity Recognition

i. Definition

1. **Named Entity:** Anything that can be referred to with a proper name.
2. **Named Entity Recognition(NER):** The combined task of finding spans of text that constitute proper names and then classifying the entities being referred to according to their type.
3. **Temporal Expressions:** Dates, times, named events
4. **Numerical Expressions:** Measurements, counts, prices
5. **Long Short-Term Memory (LSTM):** It divides the context management problem into two sub-problems: removing information no longer needed from the context, and adding information likely to be needed for later decision making.
6. **Conditional Random Field(CRF):** A class of statistical modeling method often applied in pattern recognition and machine learning and used for structured prediction.

ii. Concept

(1)Ambiguity

1. One word have different meaning in one category.
2. One word have being in different categories.

(2) NER as Sequence Labelling

IOB Labelling: Label text with inside a name or entity, beginning of a word and outside a name or entity.

iii. Algorithm

(1)A Feature-Based Algorithm for NER

The chart below shows the most frequent used features in a feature-based NER algorithm:

Feature	Explanation
Lexical Items	The token to be labeled
Stemmed Lexical Items	Stemmed version of the target token
Shape	The orthographic pattern of the target word
Character Affixes	Character level affixes of the target and surrounding words
Part of Speech	Part of speech of the word
Syntactic Chunk Labels	Base phrase chunk label
Gazetteer or Name List	Presence of the word in one or more named entity lists
Predictive Tokens	Presence of predictive words in surrounding text
Bag of Words/Bags of N-Grams	Words and/or N-Grams occurring in the surrounding texts

Table 1: Features in an NER System

Shapes are usually defined by cases, punctuations, numbers, etc. The instances are shown as follows:

Shape	Example	Regex (Written by Me)
Lower	cummings	`^[a-z]+$`
Capitalized	Washington	`^[A-Z][a-z]+$`
All caps	IRA	`^[A-Z]+$`
Mixed Case	eBay	`^(?=.*[a-z])(?=.*[A-Z])[a-zA-Z]*$`
Capitalized Initial with Period	U.S.A.	`^([A-Z]\.){1,}$`
Ends in Digit	A9	`^[a-z]+\d+$`
Contains Hyphen	H-9	`^[a-zA-Z0-9]+\-[a-zA-Z0-9]+$`

Table 2: Shape Feature, Explained with Regex

The process of building a feature-based NER is as follows:

1. Obtain the most effective name-list (for instance, gazetteers - a list of location names)

2. Mark the sentence with IOB Labelling

3. Offer a context score to judge the type of the entity

4. Tag the words and output

(2) A Neural Algorithm for NER

The standard neural algorithm for NER is the bi-LSTM. In this model, word and character embeddings are computed for input word w_i. These input words are passed through a left-to-right LSTM and a right-to-left LSTM in order to obtain a single output layer at position i. This single output layer is then passed onto a softmax that creates a probability distribution over all tags. Then, we would choose the most likely tag t_i.

One architectural example is named as "Bidirectional LSTM-CRF Models for Sequence Tagging":
First, use two LSTMs, one reading each word in a sentence from beginning to end and another reading the same but from end to beginning, producing for each word, a vector representation made from both the un-folded LSTM (i.e., forward and backward) read up to that word. There is this intuition that the vector for each word will take into account the words read/seen before, on both directions.
This bidirectional-LSTM architecture is then combined with a CRF layer at the top. A Conditional Random Field (CRF) layer has a state transition matrix as parameters, which can be used to efficiently use past attributed tags in predicting the current tag.

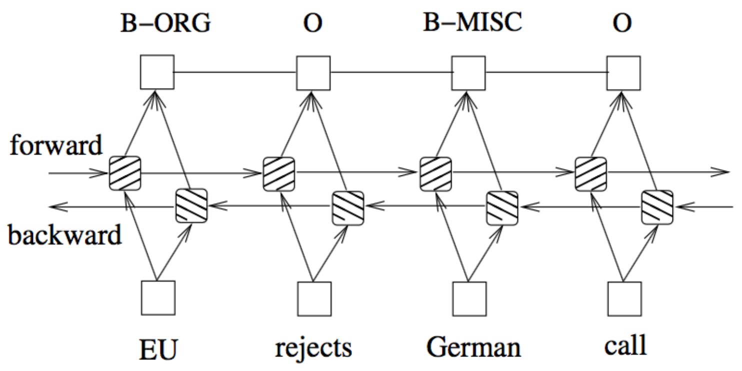


Figure 1: Neural Algorithm for NER

(3) Rule-Based NER

This approach is more suitable for commercial use of NER and the procedure is as follows:

1. First, use high-precision rules to tag unambiguous entity mentions.
2. Then, search for substring matches of the previously detected names.
3. Consult application-specific name lists to identify likely name entity mentions from the given domain.

4. Finally, apply probabilistic sequence labeling techniques that make use of the tags from previous stages as additional features.

iv. Evaluation of NER

The following methods are used to evaluate different perspectives of an NER system:

- **Recall:** The ratio of the number correctly labeled responses to the total that should have been labeled
- **Precision:** The ratio of the number of correctly labeled responses to the total labeled
- **F-Measure** Harmonic mean of recall and precision

Thus, by looking at the output of these three methods, we could know that whether we are having an accurate output from our program.

v. Implementation

Packages involved: bf4, request, re.

(1) input

This implementation takes in an input from an online source. Then, by using beautifulsoup, we could obtain a string and the string is our input. Url is given in a .txt. And we only takes in one url for this demo.

(2) code

../code/getOnlineText.py
../code/namedEntityRecognition.py
../code/nerOnAnArticle.py

(3) output

A list of count of entities in .txt.

(4) Improvement

If I have more time, I would make a more precise model, especially for the IOB system. Also, I need to improve my beautifulsoup algorithm so that I can get a proper string. It sometimes gives me some phrases that are not suppose to be together. Also, I can't identify entities if they are separated by comma(i.e. John, Peter, Mary). I would recognize those as one entity. I still haven't figure out a way, maybe because of my regex or there should be a whole other system identifying them.

Here is an online implementation of NER system with packages named NLTK and SpaCy: https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

II. Relation Extraction

i. Definition

1. **Relation Description Framework (RDF):** A framework that stores the information of the relationship of different words
2. **RDF Triple:** A tuple of entity-relation-entity
3. **is-a/Hypernym Relation:** An interclass relationship between words. One instance would be "Giraffe is-a ruminant is-a ungulate is-a mammal is-a vertebrate"
4. **Five Main Classes of Algorithms for Relation Extraction:** Hand-written patterns, supervised machine learning, semi-supervised (via bootstrapping and via distant supervision), and unsupervised.

ii. Algorithm

(1) Using Patterns to Extract Relations

Lexical-syntactic patterns is the most widely used algorithm. However, this accurate algorithm heavily relies on hand-built patterns, meaning they are hard to create. The followings are examples of some hand-written rules:

Pattern	**Example(Blued ones are higher hypernyms)**
NP$\{, \text{NP}\}^*\{,\}(and\|or)$ other NP$_H$	temples, treasuries, and other important civic buildings
NP$_H$ such as $\{\text{NP},\}^*\{,\}\{(or\|and)\}$ NP	red algae such as Gelidium
such NP$_H$ as $\{\text{NP},\}^*\{,\}\{(or\|and)\}$ NP	such authors as Herrick, Goldsmith, and Shakespeare
NP$_H\{,\}$including$\{\text{NP},\}^*\{(or\|and)\}$ NP	common-law countries, including Canada and England
NP$_H\{,\}$especially$\{\text{NP},\}^*\{(or\|and)\}$ NP	European countries, especially France, England, and Spain

Table 3: Patterns in a Relation Extraction System

(2) Relation Extraction via Supervised Learning

The procedure is as follows:

1. Choose a hand-annotated training corpus with the relations and entities

2. Use the annotated texts to train classifiers, usually a feature-based classifer like logistic regression or random forests, to annotate an unseen test set

3. A classifier is trained to assign a label to the relations that were found by step 2

```
function FINDRELATIONS(words) returns relations

    relations ← nil
    entities ← FINDENTITIES(words)
    forall entity pairs ⟨e1, e2⟩ in entities do
        if RELATED?(e1, e2)
            relations ← relations+CLASSIFYRELATION(e1, e2)
```

Figure 2: Finding and classifying the relations among entities in a text

(3) Semisupervised Relation Extraction via Bootstrapping

Bootstrap takes in a few high-precision seed patterns or a few seed tuples as classifiers. Then, it takes the entities in the seed pair, and find sentences contain both entities. From all these new sentences, we extract and generalize the context around the entities to learn new patterns. The basic algorithm is as follows:

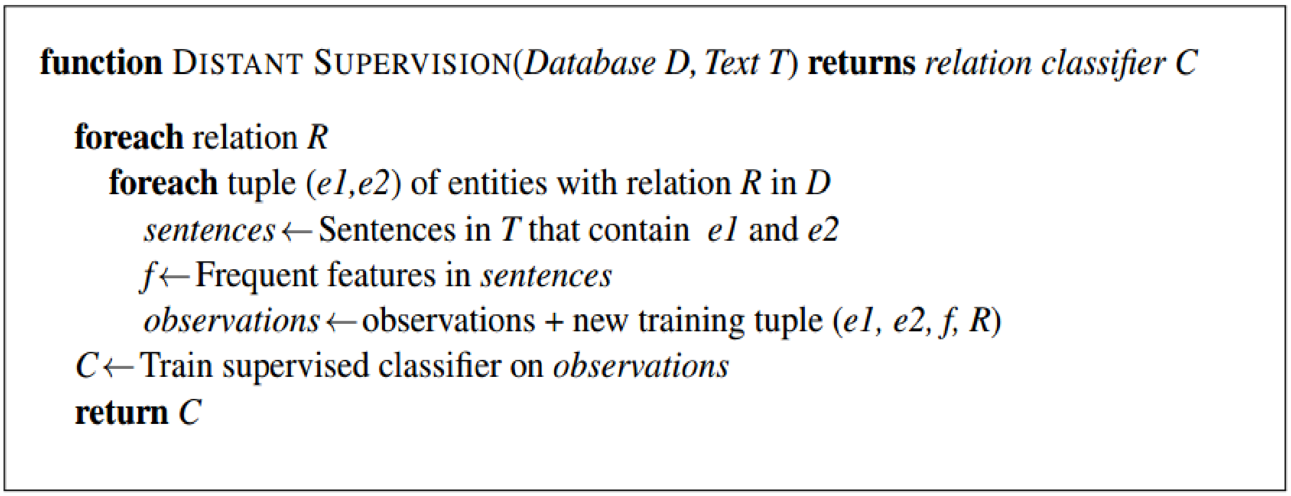
```
function BOOTSTRAP(Relation R) returns new relation tuples

    tuples ← Gather a set of seed tuples that have relation R
    iterate
        sentences ← find sentences that contain entities in tuples
        patterns ← generalize the context between and around entities in sentences
        newpairs ← use patterns to grep for more tuples
        newpairs ← newpairs with high confidence
        tuples ← tuples + newpairs
    return tuples
```

Figure 3: Bootstrapping Algorithm

(4) Distant Supervision for Relation Extraction

Distant supervision applies a large database as source of seed examples in order to create a lot of pattern features from all these examples. Then we put these examples in a supervised classifier.



```
function DISTANT SUPERVISION(Database D, Text T) returns relation classifier C

  foreach relation R
    foreach tuple (e1,e2) of entities with relation R in D
      sentences ← Sentences in T that contain e1 and e2
      f ← Frequent features in sentences
      observations ← observations + new training tuple (e1, e2, f, R)
  C ← Train supervised classifier on observations
  return C
```

Figure 4: Distant Supervision Algorithm

(5) Unsupervised Relation Extraction

Unsupervised learning of relation extraction means to extract relations from the web. This system is also known as an OpenIE. One implementation is **ReVerb** system, using the following steps to complete an unsupervised learning:

1. Run a part-of-speech tagger and entity chunker over s.
2. For each verb in s, find the longest sequence of words w that start with a verb and satisfy syntactic and lexical constraints, merging adjacent matches.
3. For each phrase w, find the nearest noun phrase x to the left which is not a relative pronoun, wh-word or existential there. Find the nearest noun phrase y to the right.
4. Assign confidence c to the relation $r = (x, w, y)$ using a confidence classifier and return it.

iii. Evaluation of Relation Extraction

For supervised learning of relation extraction, we would just be simply looking at the accuracy of the test cases. On the other hand, for both semi-supervised and unsupervised learnings, of which produce new relations from the web or a large text, we would be looking at the accuracy with the following method:

Have a human check the accuracy of a pile of randomly pulled out relations. For instance, we can input a sentence with known relations and then check whether the system could comprehend and give out the relation. Then, we could get an estimated accuracy, stating:

$$P = \frac{\text{the number of correct relations}}{\text{total number of relations}}$$

iv. Implementation

This implementation is based on the notion of an openIE system. Therefore, I am extracting relations from a given online text and tag their relations with my ideal way of extracting relations. This algorithm relies on a package named NLTK, and an NER system, which I have written myself. The algorithm is as following:

1. Unpack the entire paragraph into separate sentences.
2. Get all the entities from the sentences.
3. Get the verbs from the sentences.
4. Find the first entity of the sentence and regard it as the subject, then find the last possible entity and the last possible verb between the two entities.
5. Output the relationship.

(1) input

Takes in a text file. In this example, still the same one as the previous demo.

(2) code

../code/getOnlineText.py

../code/namedEntityRecognition.py

../code/relationExtraction.py

../code/rEDemo.py

(3) Output

A file that contains all the relationships.

(4) Improvement

The accuracy for finding relation needs to be increased. Furthermore, the determination of what is a relation needs to be more abstract.

III. Extracting Times

i. Definition

1. **Absolute Temporal Expression:** Expressions that can be marked directly to calendar dates, times of day, or both.

2. **Relative Temporal Expressions:** Particular times through some other reference point, like a week from last Tuesday.

3. **Durations:** This denotes the span of time at varying level of granularity.

4. **Temporal Anchor:** Most temporal expressions in news articles are incomplete and are only implicitly anchored, often with respect to the dateline of the article.

ii. Concept

(1) Temporal Expression Extraction

Temporal expressions have lexical triggers, examples are as following, as their head. Then, we need to identify the end of this expression. We usually take a rule-based approach to comprehend these expressions. This method usually applies cascades of automata to recognize patterns at increasing levels of complexity.

Category	**Example**
Noun	*morning, noon, night, winter, dusk, dawn*
Proper Noun	*January, Monday, Ides, Easter, Rosh Hashana, Ramadan, Tet*
Adjective	*recent, past, annual, former*
Adverb	*hourly, daily, monthly, yearly*

Table 4: Temporal Expression Headers

A second way is to use a Sequence-labelling system, like IOB to tag all the words in order to obtain temporal expressions. Example features are as follows:

Feature	**Explanation**
Token	The target token to be labeled
Tokens in window	Bag of tokens in the window around a target
Shape	Character shape features
POS	Parts of speech of target and window words
Chunk tags	Base-phrase chunk tag for target and words in a window
Lexical triggers	Presence in a list of temporal terms

Table 5: Temporal Expression Headers

(2) Temporal Normalization

Temporal normalization is the process of mapping a temporal expression to either a specific point in time or to a duration. Normalized times are represented with the VALUE attribute from the ISO 8601 standard for encoding temporal values. Some ISO patterns are shown in the following:

Unit	**Pattern**	**Sample Value**
Fully specified dates	YYYY-MM-DD	1991-09-28
Weeks	YYYY-Wnn	2007-W27
Weekends	PnWE	P1WE
24-hour clock times	HH:MM:SS	11:13:45
Dates and times	YYYY-MM-DDTHH:MM:SS	1991-09-28T11:00:00
Financial quarters	Qn	1999-Q3

Table 6: ISO Time Patterns

Fully qualified date expressions contain a year, month, and day in some conventional form.

IV. Extracting Events and their Times

i. Definition

1. **Event Extraction:** Identify mentions of events in texts. It is generally modeled via supervised learning, detecting events via sequence models with IOB tagging, and assigning event classes and attributes with multi-class classifiers.

2. **Classification of Events:** Actions, States, Reporting Events, Perception Events, etc.

3. **Common Features Used for Event Extraction:** In the table below:

Feature	**Explanation**
Character Affixes	Character-level prefixes and suffixes of target word
Nominalization suffix	Character level suffixes for nominalizations (e.g., $-tion$)
Part of speech	Part of speech of the target word
Light verb	Binary feature indicating that the target is governed by a light verb
Subject syntactic category	Syntactic category of the subject of the sentence
Morphological stem	Stemmed version of the target word sentence
Verb root	Root form of the verb basis for a nominalization
WordNet hypernyms	Hypernym set for the target

Table 7: Common Features for Event Extraction

ii. Concept

(1) Temporal Ordering of Events

Determining ordering of events can be viewed as a binary relation detection and classification task. All temporal relations between events are classified into one of the standard set of Allen relations, which is shown in the following figure:

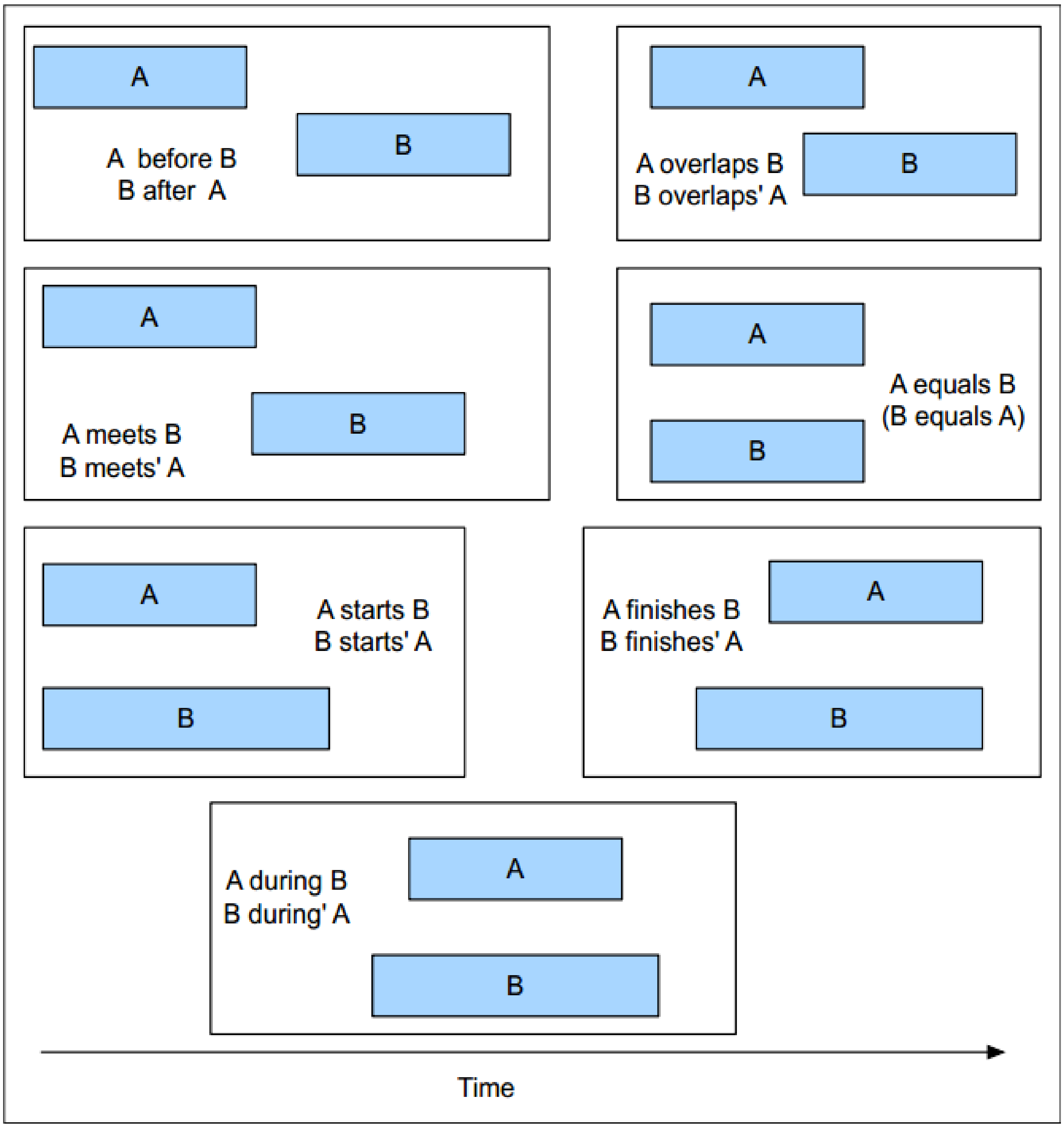


Figure 5: Finding and classifying the relations among entities in a text

V. Template Filling

i. Definition

1. **Scripts:** Consists prototypical sequences of sub-events, participants, and their roles
2. **Template:** Consists of fixed sets of slots that take as values slot-fillers belonging to particular classes
3. **Template Filling:** Find documents that invoke particular scripts and then fill the slots in the associated templates with fillers extracted from the text
4. **Finite State Transducer(FST):** It provides a method for performing mathematical operations on ordered collections of context-sensitive rewrite rules such as those commonly used to implement fundamental natural language processing tasks.

ii. Concept

(1) Machine Learning Approaches to Template Filling

In the standard paradigm for template filling, we are trying to fill fixed known templates with known slots, and also assumes training documents labeled with examples of each template, and the fillers of each slot marked in the text.
There are two system handling this task, the first one is template recognition, and the other one is role-filler extraction. Template recognition can be treated as a text classification task, with features extracted from every sequence of words that was labeled in training documents as filling any slot from the template being detected. Role-filler extraction is a separate classifier trained to detect each role (LEAD- AIRLINE, AMOUNT, and so on). This can be a binary classifier that is run on every noun-phrase in the parsed input sentence, or a sequence model run over sequences of words.

(2) Earlier Finite-State Template-Filling Systems

Early systems for dealing with these complex templates were based on cascades of transducers based on hand-written rules. The first four stages use hand-written regular expression and grammar rules to do basic tokenization, chunking, and parsing. Stage 5 then recognizes entities and events with a FST-based recognizer and inserts the recognized objects into the appropriate slots in templates.

Bibliography

Jurafsky, Daniel, and James H. Martin. *Speech and Language Processing.* 3rd ed., 2018.

Li, Susan, and Susan Li. Named Entity Recognition with NLTK and SpaCy. *Towards Data Science*, Towards Data Science, 17 Aug. 2018, towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da.

Named-Entity Recognition Based on Neural Networks, www.davidsbatista.net/blog/2018/10/22/Neural-NER-Systems/.