Named Entity Recognition via Conditional Random Fields (CRF)

Ashish Vijay Tendulkar
ashishvt@gmail.com

Note: Most of the material is copies from Information Extraction Survey by Prof. Sunita Sarawagi. Please do not distribute. Instead distribute the original paper. This is for preparing our presentation.

1 Introduction: NER

Entity recognition is one of the central task in Information Extraction (IE). The entities are typically noun-phrases and comprise of one to a few tokens in the unstructured text. The most popular form of entities is *named entity* like names of persons, location and organizations. The extraction of named entity is referred to as a named entity recognition(NER) task. NER task takes a document as an input and annotates it with the named entities. For example, for input document containing a single sentence: ”Show me the US Air flights between Atlanta and Boston”, the NER system will tag named entities as shown in bold faced letters - ”Show me the **US Air** flights from **Atlanta** to **Boston**.”. Here ”US Air” is an organization, while ”Atlanta” and ”Boston” are locations.

2 NER methods

The NER task can be carried out with two broad classes of methods:

- Rules based methods
- Statistical methods

Statistical methods of entity extraction convert the extraction task to a problem of designing a decomposition of the unstructured text and then labeling various parts of the decomposition, either jointly or independently. The most common form of decomposition is into a sequence of tokens obtained by splitting an unstructured text along a predefined set of delimiters (like spaces, commas, and dots). In the labeling phase, each token is then assigned an entity label or an entity subpart label. Once the tokens are labeled, entities are marked as consecutive tokens with the same entity label. We methods are referred to as token-level methods since they assign label to each token in a sequence of tokens

A second form of decomposition is into word chunks. A common method of creating text chunks is via natural language parsing techniques that identify noun chunks in a sentence. During labeling, instead of assigning labels to tokens, we assign labels to chunks. This method is effective for well-formed natural language sentences. It fails when the unstructured source does not comprise of well formed sentences, for example, addresses and classified ads.

3 Problem Formulation

We denote the given input of unstructured text by $\mathbf{x}$ and its tokens as $x_1, x_2, \ldots, x_n$ where n is the number of tokens in $\mathbf{x}$. At the time of NER task, each x_i will be labeled with one of a set $\mathcal{Y}$ of labels. This gives rise to a tag sequence $\mathbf{y} = y_1, y_2, \ldots, y_n$. The set of labels $\mathcal{Y}$ comprise of the set of entity types E and a special label "other" for tokens that do not belong to any of the entity types.

Since entities typically comprise of multiple tokens, there are two popular encodings for decomposing each entity label:

- **BCEO** where we decompose each entity label into "Entity Begin"(B), "Entity End"(E), "Entity Continue"(C) and "Other."(O)

- **BIO** where we decompose each entity label into "Entity Begin"(B), "Entity Inside"(I) and "Other"(O).

We will use $\mathcal{Y}$ to denote the union of all these distinct labels and m to denote the size of $\mathcal{Y}$. Token labeling can be thought of as a generalization of classification where instead of assigning a label to each token, we assign labels to a sequence of tokens. Features form the basis of this classification process.

4 Features

A typical extraction task depends on a diverse set of clues capturing various properties of the token a and the context in which it lies. Each of these can be thought of as a function $f : (x, y, i) \to R$ that takes as argument the sequence x, the token position i, and the label y that we propose to assign x_i, and returns a real-value capturing properties of the i-th token and tokens in its neighborhood when it is assigned label y. Typical features are fired for the i-th token x_i, for each token in a window of w elements around x_i, and for concatenation of words in the window.

The following are common families of token properties used in typical extraction tasks:

- **Word features**: The surface word itself is often a very strong clue for the label it should be assigned. Two examples of token features at position 4 of our example sequence x is:

$$\begin{aligned} f_1(y, \mathbf{x}, i) &= [[x_i\ equals\ \backslash US"]]\ .\ [[y = Organization]] \\ f_2(y, \mathbf{x}, i) &= [[x_{i+1}\ equals\ \backslash Air"]]\ .\ [[y = Organization]], \end{aligned}$$

where $[[p]] = 1$ if the predicate p is true.

- Orthographic Features Many valuable features for entity extraction are derived from various orthographic properties of the words, viz, its capitalization pattern, the presence of special symbols and alphanumeric generalization of the characters in the token. Two examples of orthographic features are

  $$\begin{aligned} f_3(y, \mathbf{x}, i) &= [[x_i matchesCapsWord]]\ .\ [[y = Location]] \\ f_4(y, \mathbf{x}, i) &= [[x_i x_{i+1} matchesCapsWordCapsWord]]\ .\ [[y = Organization]]. \end{aligned}$$

  Feature f_3 fires when a token x_i is a capitalized word, and it is being labeled Location. Feature f_4 fires when a token x_i and x_{i+1} are both capitalized and i-th token is being labeled as Organization.

- Dictionary Lookup Features: There is often a database of entities available at the time of extraction. Match with words in a dictionary is a powerful clue for entity extraction. This can be expressed in terms of features as follows:

  $$f_5(y, x, i) = [[x_i inLocationdictionary]] \Delta [[y = Location]]$$

5 Models for Labeling Tokens

Variety of different models can be used for assigning labels to the sequence of tokens in a sentence. An easy model is to independently assign the label y_i of each token x_i using features derived from the token x_i and its neighbors in $\mathbf{x}$. One can use classifiers like Naive Bayes, Logisic Regression, SVM, Decision Trees to label the tokens. However, in typical extraction tasks the labels of adjacent tokens are seldom independent of each other. For example, it might be difficult to classify “Air” as being a word from an organization. However, when the word to the left of it is labeled an organization, it makes sense to label “Air” as an organization too. This has led to a number of different models for capturing the dependency between the labels of adjacent words. The simplest of these is the ordered classification method that assigns labels to words in a fixed left to right order where the label of a word is allowed to depend on the label of the word to its left. Other options include: HMM, Max-Entropy and Conditional Markov Models.

CRFs provide a powerful and flexible mechanism for exploiting arbitrary feature sets along with dependency in the labels of neighboring words. Empirically, they have been found to be superior to all the earlier proposed methods for sequence labeling.

A Conditional Random Field (CRF) models a single joint distribution $\Pr(y|x)$ over the predicted labels $\mathbf{y} = y_1, \ldots, y_n$ of the tokens of $\mathbf{x}$. The tractability of the joint distribution is ensured by using a Markov random field to express the conditional independencies that hold between elements y_i of $\mathbf{y}$. In typical extraction tasks, a chain is adequate for capturing label dependencies. This implies that the label y_i of the i-th token is directly influenced only by the labels of tokens that are adjacent to it. In other words, once the label y_{i-1} is fixed, label y_{i-2} has no influence on label y_i. The dependency between the labels of adjacent tokens is captured by a scoring function $\psi(y_{i-1}, y_i, \mathbf{x}, i)$ between nodes y_{i-1} and y_i. This score is defined in terms of weighted functions of features as follows:

$$\begin{aligned} \psi(y_{i-1}, y_i, \mathbf{x}, i) &= \exp(\sum_{k=1}^{K} w_k f_k(y_i, \mathbf{x}, i, y_{i-1}) && (1) \\ &= \exp(\mathbf{w}\ .\ \mathbf{f}(y_i, \mathbf{x}, i, y_{i-1}) && (2) \end{aligned}$$

With these per-edge scores, the conditional distribution of a label sequence $\mathbf{y}$ given a token $\mathbf{x}$ is given as

$$\begin{aligned} \Pr(\mathbf{y}|\mathbf{x}, \mathbf{w}) &= \frac{1}{Z(\mathbf{x})} \prod_{i=1}^{n} \psi(y_{i-1}, y_i, \mathbf{x}, i) && (3) \\ &= \frac{1}{Z(\mathbf{x})} \exp(\sum_{i=1}^{n} \mathbf{w}\ .\ \mathbf{f}(y_i, \mathbf{x}, i, y_{i-1}) && (4) \\ &= \frac{1}{Z(\mathbf{x})} \exp(\mathbf{w}\ .\ \mathbf{f}(\mathbf{x}, \mathbf{y}) && (5) \end{aligned}$$

The term $Z(x)$ is a normalizing constant and is equal to $\sum_{\mathbf{y}} \exp(\mathbf{w}.\mathbf{f}(\mathbf{x}, \mathbf{y})$ where $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbf{f}(y_i, \mathbf{x}, i, y_{i-1})$ the sum of the feature vector over all token positions of the sequence. Some subset of these features can be simplified further to depend only on the current state and are independent of the previous state. We will refer these as **state features** and denote these by $f(y_i, \mathbf{x}, i)$ when we want to make the distinction explicit. The term **t**ransition features refers to the remaining features that are not independent of the previous label.

Given m labels and a sequence $\mathbf{x}$ of length n, there can be $O(mn)$ possible labeling of $\mathbf{x}$. This exponential complexity is cut down to $O(nm^2)$ by the famous dynamic programming-based Viterbi Algorithm.

6 Training

Let $D = (\mathbf{x}_l, \mathbf{y}_l)_{i=1}^{N}$ denote the labeled training set. The goal during training is to choose a weight vector $\mathbf{w}$ such that probability of the correct output as given in the training data is maximized. Let $L(\mathbf{w})$ denote the log of the probability of the training data when the weight vector is $\mathbf{w}$. We can write it as:

$$L(\mathbf{w}) = \sum_{l} \log \Pr(\mathbf{y}_1|\mathbf{x}_l, \mathbf{w}) = \sum_{l} (\mathbf{w}\ .\ \mathbf{f}(\mathbf{x}, \mathbf{y} - \log(\mathbf{Z}_{\mathbf{w}}(\mathbf{x}_l)). \qquad (6)$$

Routinely, a term such as $-\frac{||\mathbf{w}||_2}{C}$ is added to prevent overfitting by penalizing large swings in the parameter values. This term has the effect of performing a soft form of feature selection. Ideally, we would like to maximize accuracy while using the smallest number of features. Since this leads to a difficult discrete optimization problem we instead use $||\mathbf{w}||_2$ weighted by a user-provided constant $1/C$. The training objective can then be expressed as:

$$\mathbf{max}_w \sum_l (\mathbf{w} \ . \ \mathbf{f}(\mathbf{x}, \mathbf{y} - \log(\mathbf{Z}_{\mathbf{w}}(\mathbf{x}_l)) - \frac{||\mathbf{w}||_2}{C} \tag{7}$$

The above equation is concave in w, and can thus be maximized by gradient ascent type of methods that iteratively move toward the optimum $\mathbf{w}$. At each iteration they require the computation of the gradient $\nabla L(\mathbf{w})$ of the objective which can be calculated as:

$$\begin{aligned} \nabla L(\mathbf{w}) &= \sum_l \mathbf{f}(\mathbf{x}_l, \mathbf{y}_l) - \frac{\sum'_y \mathbf{f}(\mathbf{x}_l, \mathbf{y}')\exp(\mathbf{w}.\mathbf{f}(\mathbf{x}_l, \mathbf{y}'))}{Z_{\mathbf{w}}(\mathbf{x}_l)} - 2\mathbf{w}/c & (8) \\ &= \sum_l \mathbf{f}(\mathbf{x}_l, \mathbf{y}_l) - E_{\Pr(\mathbf{y}'|\mathbf{x},\mathbf{w})}\mathbf{f}(\mathbf{x}_l, \mathbf{y}') - 2\mathbf{w}/c. & (9) \end{aligned}$$

The first term is easy to compute. However, the second term that involves a summation over exponentially many outputs $\mathbf{y}'$ require special algorithms that exploit the decomposability of the feature vector.

7 Inference

For a given input sequence $\mathbf{x}$, find out the label sequence y^* such that:

$$y^* = \text{argmax}_y \mathbf{w} \ . \ \mathbf{f}(\mathbf{x}, \mathbf{y}) \tag{10}$$

The key to solving this efficiently, in spite of the exponential number of possible values of $\mathbf{y}$, is that the feature vector $\mathbf{f}(\mathbf{x}, \mathbf{y})$ decomposes over finer substructures in $\mathbf{y}$. This enables us to design a dynamic programming algorithm for this task.

In sequence labeling the feature vector decomposes over labels of adjacent positions, that is, $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{|x|} \mathbf{f}(y_i, \mathbf{x}, i, y_{i-1})$. Let $\mathcal{V}(i|y)$ denote the best score of the sequence from 1 to i with the ith position labeled y. We can express its value using dynamic programming as follows:

$$\mathcal{V}(i|y) = \begin{cases} \max'_y \mathcal{V}(i-1, y') + \mathbf{w} \ . \ \mathbf{f}(y, \mathbf{x}, i, y') & \text{if } x > 0 \\ 0 & \text{if } x = 0. \end{cases} \tag{11}$$

The best label then corresponds to the path traced by $\max_y \mathcal{V}(n|y)$, where n is the length of x. The running time of this algorithm is $O(nm^2)$, where m is the number of labels. This is well-known as the Viterbi algorithm.