01.112 Machine Learning - 2017

Design Project - Sentiment Analysis

By: Shruthi Shangar - 1001630
Vanessa Tan - 1001827

Objective

Analyse natural language texts typed, shared and read by the general population across social platforms like twitter, facebook etc. These sites are popular for understanding the general reaction towards a current event. We will be designing a sequence labelling model for informal texts using Hidden Markov Model.

Long term Goal

Complex, Intelligent and Reliable system for analysing social media text.

Part 2:

Objective

Estimate the emission parameters and use the values to compute the most probable tag sequence for a given observation sequence.

Implementation

To compute the emission parameters $b_i(o)$, we use the Maximum Likelihood Estimate (MLE):

$$b_i(o) = \frac{count(i \to o)}{count(i)}$$

where $count(i \to o)$ is the number of times the observation o was emitted from the state i, and $count(i)$ is the number of times the number of times tag i appears in the training data.

Therefore, we first count the number of times each emission appears in the training set, and the number of times each tag was labelled. To account for variations in the training and test data, the number of times each observation appears in the training set was also recorded, and a special token `##UNK##` is used to replace words that appear less than a specified k number of times.

* The value of k is arbitrary.

Learnings

1) Using only the emission parameters to estimate the tag leads to low accuracy. This is due to ignoring of transition parameters which play an important role in tag generation.

Part 2 - Results

Test Results for EN:

Entity in gold data: 226 **Entity in prediction:** 1201	**Correct Entity :** 165 **Entity precision:** 0.1374 **Entity recall:** 0.7301 **Entity F:** 0.2313	**Correct Sentiment :** 71 **Sentiment precision:** 0.0591 **Sentiment recall:** 0.3142 **Sentiment F:** 0.0995

Test Results for FR:

Entity in gold data: 223 **Entity in prediction:** 1149	**Correct Entity :** 182 **Entity precision:** 0.1584 **Entity recall:** 0.8161 **Entity F:** 0.2653	**Correct Sentiment :** 68 **Sentiment precision:** 0.0592 **Sentiment recall:** 0.3049 **Sentiment F:** 0.0991

Test Results for CN:

Entity in gold data: 362 **Entity in prediction:** 3318	**Correct Entity :** 183 **Entity precision:** 0.0552 **Entity recall:** 0.5055 **Entity F:** 0.0995	**Correct Sentiment :** 57 **Sentiment precision:** 0.0172 **Sentiment recall:** 0.1575 **Sentiment F:** 0.0310

Test Results for SG:

Entity in gold data: 1382 **Entity in prediction:** 6599	**Correct Entity :** 794 **Entity precision:** 0.1203 **Entity recall:** 0.5745 **Entity F:** 0.1990	**Correct Sentiment :** 315 **Sentiment precision:** 0.0477 **Sentiment recall:** 0.2279 **Sentiment F:** 0.0789

Part 3:

Objective

From part 2, we can see the results are not very accurate. To improve accuracy, we will be calculating the transition parameters and use it along with the emission parameters to decode the most probable tag sequence in the Viterbi algorithm.

Implementation

Similar to the emission parameters, the transition parameters used in the decoding process are taken to be the MLE of the transition probabilities of a bigram of tag sequence, expressed as follows:

$$a_{i,j} = \frac{count(i,j)}{count(i)}$$

where $count(i,j)$ is the number of times two successive tags (i,j) appears and $count(i)$ is the number of times tag i appears in the training data.

After the transition parameters are estimated, we can find the most optimal tag sequence Y^* as follows:

$$Y^* = argmax_{y_1 \dots y_n} \prod_{i=1}^{n+1} a_{y_{i-1}, y_i} \prod_{i=1}^{n} b_{y_i}(x_i)$$

As the HMM has a simple dependence structure, a dynamic programming algorithm such as Viterbi can be used. In addition, the optimal previous tag was stored along with the maximum score in the 2-dimensional array π to reduce the complexity of the backtracking algorithm.

Viterbi Algorithm

- Base Case:

  - $\pi(0, u) = \{1 \; if \; u = \; start \}$
    $\{0 \; otherwise\}$

- Recursive Case:

  - For any $k \in \{1, ..., n\}$
    $\pi(k, v) = max_{u \in T} \{ \pi(k-1, u) \cdot a_{u,v} \cdot b_v(x_k) \}$
    $bp(k, v) = argmax_{u \in T} \{ \pi(k-1, u) \cdot a_{u,v} \cdot b_v(x_k) \}$

- Finally,

  - $max_{y1,...,yn} p(x_1, ..., x_n, y_0 = START, y_1, ..., y_{n+1} = STOP) = max_{v \in T} \{\pi(n, v) \cdot a_{v,STOP}\}$

Backtracking

- For $k = (n-1)...1$,

$$y_k^* = bp(k+1, y_{k+1}^*)$$

The time complexity of the algorithm implemented (as stated above) is $O(nT^2)$ and the space complexity is $O(nT)$.

Learnings

1) As compared to the case where only the emission parameters were used (Part 2), it is very clear that the accuracy with Viterbi has increased significantly as expected.

Part 3 - Results

Test Results for EN:

Entity in gold data: 226 **Entity in prediction:** 162	**Correct Entity:** 104 **Entity precision:** 0.6420 **Entity Recall:** 0.4602 **Entity F:** 0.5361	**Correct Sentiment:** 64 **Sentiment Precision:** 0.3951 **Sentiment Recall:** 0.2832 **Sentiment F:** 0.3299

Test Results for FR:

Entity in gold data: 223 **Entity in prediction:** 166	**Correct Entity:** 112 **Entity Precision:** 0.6747 **Entity Recall:** 0.5022 **Entity F:** 0.5758	**Correct Sentiment :** 72 **Sentiment Precision:** 0.4337 **Sentiment Recall:** 0.3229 **Sentiment F:** 0.3702

Test Results for CN:

Entity in gold data: 362 **Entity in prediction:** 158	**Correct Entity:** 64 **Entity Precision:** 0.4051 **Entity Recall:** 0.1768 **Entity F:** 0.2462	**Correct Sentiment:** 47 **Sentiment Precision:** 0.2975 **Sentiment Recall:** 0.1298 **Sentiment F:** 0.1808

Test Results for SG:

Entity in gold data: 1382 **Entity in prediction:** 723	**Correct Entity:** 386 **Entity Precision:** 0.5339 **Entity Recall:** 0.2793 **Entity F:** 0.3667	**Correct Sentiment:** 244 **Sentiment Precision:** 0.3375 **Sentiment Recall:** 0.1766 **Sentiment F:** 0.2318

Part 4

Objective

Implementing max-marginal decoding algorithm

Implementation

We first calculated the emission and transition parameters (using the functions described in Part 2 and 3) and used a dynamic programming approach to calculate the values of α and β as shown below:

1. Calculating α

- Forward Probabilities:
  $\alpha_u(j) = p(x_1, ..., x_{j-1}, y_j = u; \theta)$

- Base Case:
  $\alpha_u(1) = a_{START,u} \ \forall u \in 1, ..., N-1$

- Recursive Case:

  $\alpha_u(j+1) = \sum_v \alpha_v(j) a_{v,u} b_v(x_j) \ \ \forall u \in 1, ..., N-1, j = 1, ..n-1$

2. Calculating β

- Backward Probabilities:
  $\beta_u(j) = p(x_j, ..., x_n | y_j = u; \theta$

- Base Case:
  $\beta_u(n) = a_{u,STOP} b_u(x_n) \ \ \forall u \in 1, ..., N-1$

- Recursive Case:

  $\beta_u(j) = \sum_v a_{u,v} b_u(x_j) \beta_u(j+1) \ \forall u \in 1, ..., N-1, j = n-1, ..., 1$

The values of α and β are then used to derive the optimal tag sequence with the following equation:

$$y_i^* = argmax_u \alpha_u(i)\beta_u(i)$$

Learnings

1. The max-marginal decoding algorithm predicts one tag at a time, regardless of other tags, so empirically the max-marginal and Viterbi could return very different results which is what occurred in our case.

Part 4 - Results

Test Results for EN:

Entity in gold data: 226 **Entity in prediction:** 175	**Correct Entity:** 108 **Entity Precision:** 0.6171 **Entity Recall:** 0.4779 **Entity F:** 0.5387	**Correct Sentiment:** 69 **Sentiment Precision:** 0.3943 **Sentiment Recall:** 0.3053 **Sentiment F:** 0.3441

Test Results for FR:

Entity in gold data: 223 **Entity in prediction:** 173	**Correct Entity:** 113 **Entity Precision:** 0.6532 **Entity Recall:** 0.5067 **Entity F:** 0.5707	**Correct Sentiment:** 73 **Sentiment Precision:** 0.4220 **Sentiment Recall:** 0.3274 **Sentiment F:** 0.3687

Part 5:

Objective

To achieve a better accuracy of the predicted tag sequence, we have decided to implement the Structured Perceptron Algorithm.

Implementation

Before running the algorithm, we conducted a pre-processing step to convert all the observations in the input data set into lowercase (e.g. 'tree' is the same as 'Tree' or 'TREE') during the training and testing phase.

We then implemented a version of the Structured Perceptron algorithm to train the model as follows:

- ❖ Initialization:
- ❖ Set the parameter vector α to 0..
- ❖ For $t = 1...T$, $i = 1...n$,
  - ➢ Use the Viterbi algorithm to find the output of the model on the i^{th} training sentence with the current parameter settings, i.e.,

    $$[1 : n_i] = argmax_{u_{[1:n_i]} \in T^{n_i}} \sum_s \alpha_s \Phi_s(w^i_{[1:n_i]}, u_{[1:n_i]})$$

    Where T is the number of iterations over the training set, n is the number of training sentences in the training set, T^{n_i} is the set of all tag sequences of length n_i .
  - ➢ If $z[1 : n_i] \neq t^i_{[1:n_i]}$ then update the parameters

    $$\alpha_s = \alpha_s + \Phi_s(w^i_{[1:n_i]}, t^i_{[1:n_i]}) - \Phi_s(w^i_{[1:n_i]}, z_{[1:n_i]})$$

The optimal parameters of α calculated in the training phase is then used to decode the observation sequence in the validation step.

Learnings

1. While an attempt was made to use the averaged parameters to decode the optimal tag sequence in the validation phase, the validation accuracy was found to be worse than in HMM. Instead, our implementation uses the parameter vector $\overline{\alpha_s} = \alpha_s^{T,n}/nT$, where $\alpha_s^{T,n}$ is the parameter for the s^{th} feature function after the last iteration of the Structured Perceptron algorithm. This was found to yield the best accuracy when run over the validation set.

Part 5 - Results

Test Results for EN:

Entity in gold data: 226 **Entity in prediction:** 323	**Correct Entity:** 160 **Entity Precision**: 0.4954 **Entity Recall:** 0.7090 **Entity F:** 0.5829	**Correct Sentiment:** 86 **Sentiment Precision:** 0.2663 **Sentiment Recall:** 0.3805 **Sentiment F:** 0.3133

Results obtained with: k=1 and number of iterations=12

Test Results for FR:

Entity in gold data: 223 **Entity in prediction:** 280	**Correct Entity:** 117 **Entity Precision**: 0.6321 **Entity Recall:** 0.7937 **Entity F:** 0.7038	**Correct Sentiment:** 103 **Sentiment Precision:** 0.3679 **Sentiment Recall:** 0.4619 **Sentiment F:** 0.4095

Results obtained with: k=1 and number of iterations=4