ISSN 2959-6157

Support Vector Machine and Hidden Markov Model in Name Entity Recognition of Natural Language Processing

Jiaheng Li[1]

[1]Arcadia High School, Arcadia, United States of America
*Corresponding author: 45155@students.ausd.net

Abstract:

This paper illustrates a comparison between the Hidden Markov Model and the Support Vector Machine, two important methodologies and tools, used in Natural Language Processing. Breaking down the model formulations of each, this paper first describes the mathematical motivations behind their applications in NLP. The mathematical motivations will be discussed through formulas, ideas, and examples. Then, this paper applies two real pre-established algorithms, one for each model, as examples to further rationalize their unique characteristics, similarities, and differences. These aspects will be broken down further into algorithmic efficiency, effectiveness, and other factors. Based on their performances analyzed through each factor, specific toolkits will be proposed, explained, and tested to optimize the test results, as the improving method. Some examples of toolkits include YamCha, TinySVM, etc. Overall, Name Entity Recognition involves different methodologies, and SVM and HMM, which represent two leading areas of NLP research, can best describe future trends and current situations.

Keywords: Name Entity, support Vector Machines, hidden markov model, natural language Processing.

1. Introduction

With the developments of computer technologies and Artificial Intelligence, new methods and models in the field of Natural Language Processing (NLP) have been introduced. Specifically, the majority of these methods have been classified into three categories: Symbolic, Statistical, and Neural. The three types of methods have their unique sets of advantages and disadvantages in terms of factors such as interpretability, scalability, and more.

Amongst the three, statistical methods have been the most controversial over almost 40 years of its development. Compared to Neural and Symbolic approaches, statistical models prevail in suggesting auto-complete and recognizing handwriting with lexical acquisition even if it's in a poorly written text. Furthermore, the accessible nature of data in statistical methods makes them stand out in efficiency and stability as compared to NN algorithms in tasks about syntactic and semantic analysis of NLP. [1] Indeed, statistical methods have been favored and utilized for decades due to these reasons. Nonetheless, statistical models were significantly criticized for their over-reliance on assumptions about data distribution and their requirements for elaborate feature engineering. As a result of their drawbacks, neural methods gradually replace statistical methods and eventually dominate the majority of NLP research. Uses of statistical methods, however, are still prevalent in recent NLP studies for their irreplaceable benefits.

In Natural Language Processing, various fields of research are foundational to the overall process. Part-of-speech-tagging (POT), Speech Recognition, and Text Classifications are examples of the major topics in NLP for computer scientists to study. Noteworthily, the applications of statistical models are especially extensive in Name Entity Recognition (NER) as they train probabilistic models based on textual patterns and relationships to predict named entities in new text data. [2] In this case, statistical methods' benefits of using feature engineering to represent context relevance and determine name entities outweigh their drawbacks, as previously mentioned. [3]

With the incorporation of machine learning algorithms, traditional statistical methods such as Hidden Markov Model (HMM) and non-traditional methods such as Support Vector Machine (SVM) have been widely applicable in different processes of NLP, including the research trends in NER. This paper studies the functionality of the Hidden Markov Model and the Support Vector Machine in treating Name Entity Recognition problems while also comparing them based on their similarities and differences.

This paper first discusses the mathematical motivations

and formulations for each model. Based on the formulations, corresponding algorithms and examples will be provided to explain the test efficiency of each algorithm, with an indication of the different factors involved. Lastly, a thorough comparison between the results will be given, leading to a conclusion regarding the cumulative net benefits of each model.

2. Model Formulation

2.1 . Hidden Markov Model Formulation

The idea of a Markov chain is essential to understanding a Hidden Markov Model. Any Markov chain can be described by three variables: x, P, π. Let X denote a set of random variables: $X_0, X_1, X_2, X_3, ...$ and x denote a set of possible states values for $X: x_0, x_1, x_2, x_3 ...$ When X is a stochastic process and the probability of each event depends solely on the state of the previous event, it is said to be a first-order Markov chain. In other words, the stochastic process X must suffice for the following relationship, also known as the Markov property.

$$P(X_{t+1} = x_{t+1} \mid X_t = x_t) = P(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, ..., X_0 = x_0) \quad (1)$$

Based on the first-order Markov chain, an n-th order Markov chain can be described when the state s_{t+1} is dependent on the previous n states. This is formulated through an extended version of the Markov property.

$$P(X_{t+1} = x_{t+1} \mid X_t = x_t, ..., X_n = x_n) = P(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1}, ..., X_0 = x_0) \quad (2)$$

To simplify the probability calculations of state conversions, a transition matrix P with size $\Omega \cdot \Omega$ can be constructed. In the transition matrix, the rows i represent the current state, and the columns j represent the future states. All the elements in all rows should sum up to 1 while all the elements in all columns generally don't. In addition, the probability of going from state i to state j can be expressed as:

$$P_{ij} = P(X_{t+1} = j \mid X_t = i) \quad (3)$$

To find the probability of going from state i to state j in n steps, the expression can be generalized into:

$$P(X_t = j \mid X_0 = i) = (P^n)_{ij} \quad (4)$$

By incorporating the initial probability distribution π, the common distribution of a Markov chain after t steps can be computed through:

$$P(X_0 = x_0 ... X_t = x_t) = \pi_{s_0} \cdot p_{s_0 s_1} \cdot p_{s_1 s_2} ... p_{s_{t-1} s_t} \quad (5)$$

Now, a Hidden Markov Model is a first-order Markov chain with the previous states x being unobservable. However, a set of emission symbols, v, is now observable. The probability of dependence of v on state x is $P(v(t) \mid x(t))$. Thus, based on the existing three variables: x, P, π from Markov chains, two additional variables V, b have now been formulated in HMM. V here represents the set of emission symbols $v_1, v_2, v_3 ... v_t$ and b represents the set of emission probabilities of states. Additionally, the emission probabilities represented by b can be expressed in matrix form just like the transition matrices.

The probability that results in a specific observed sequence is equivalent to the summation of all possible paths of hidden states. To optimize the complexity, a dynamic programming approach called the Viterbi algorithm is proposed. The Viterbi algorithm, instead of tracking all probabilities plainly, only tracks the state sequences with the maximized probability. Applications of the Viterbi algorithm are commonly seen in solving HMM-based problems.

2.2 . Support Vector Machine Model Formulation

The Support Vector Machine is a supervised machine-learning algorithm used for classification purposes. The principles behind an SVM classification model involve the definition of a support vector. Support vectors are data points lying closest to the divided line, also known as the hyperplane in a higher dimension. The hyperplane is described by the equation $\vec{x} \cdot \vec{w} + b = 0$, where $\vec{w}$ denotes the normal vector to the decision boundary and $\vec{x}$ denotes the set of data points.

Practically. to classify a specific data point in the space using SVM mathematically, the idea of vectors and dot product operations is required. We take the projection of $\vec{x}$ on $\vec{w}$ by calculating their dot products. The resulting value should fall into one of the following three cases, where d represents the distance between the origin and the decision boundary:

$\vec{x} \cdot \vec{w}$ = d (This indicates that the data point lies exactly on the decision boundary)

$\vec{x} \cdot \vec{w}$ > d (This indicates that the data point belongs to the positive samples)

$\vec{x} \cdot \vec{w}$ < d (This indicates that the data point belongs to the

negative samples)

This relationship can also be expressed as the decision function $\vec{x}\cdot\vec{w}+b=0$, as previously mentioned, where d is equivalent to $-b$.

In general, the support vectors are the hardest to classify. To solve this issue, the SVM model needs to maximize the margin between the two sets of data points while ensuring the plane is equidistant to both data sets. There are two types of SVM while doing this: Hard SVM and Soft SVM. The hard SVM refers to a linearly separable setting, where the optimization of the margin follows:

Objective Function:

$$w,b = \arg\ min_{w,b} \frac{1}{2}||w||^2$$

Constraints:

$$\forall i: t^{(i)}(w^T x^{(i)} + b) \geq 1$$

On the other hand, when dealing with linearly inseparable data points, we introduce the well-known soft SVM. Instead of solely maximizing the marginal distance like hard SVM, soft SVM makes tradeoffs between margin maximization and the amount of misclassification errors ξ. Now, the objective function and the constraints become:

Objective Function:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \quad (6)$$

Constraints:

$$\forall i: t^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i \quad (7)$$

$$\forall i: \xi_i \geq 0 \quad (8)$$

Noteworthily, when the dataset is nonlinear, the idea of kernel functions is utilized. Kernel functions help to convert the current dataset into either higher or lower dimensional space where a decision boundary is detectable. Some examples of kernel functions include Polynomial Kernel, Sigmoid Kernel, RBF Kernel, and Bessel Function Kernel.

3. Analysis

Starting with the Hidden Markov Model, the Hidden Markov Model was used primarily in Natural Language Processing for Name Entity Recognition specifically. A proposed HMM-based algorithm contains three stages: data preparation and parameter estimation. [4] In the data preparation stage, the program converts the raw data for it to be trainable. To do so, the algorithm will tokenize the words separated in the sentence. It might as well tag the words if required. Moving on to the parameter estimation for the HMM model to be defined. Using the algorithm, we can find the states, transition probability, and emission probability. In NER, the states generally represent the name entity tags, a category of similar words that can be classified together. On the other hand, calculating the start probability can involve the formula of dividing the number of sentences with a specific tag by the total number of sentences given. Likewise, the transition probability is formulated by dividing the total number of sequences from two given name tags T_i and T_j by the total number of T_i. Lastly, finding the Emission Probability requires dividing the total number of word occurrences as tags by the total occurrences of that specific one.

This algorithm meets the standards through different aspects. For example, the program has approximately 90% accuracy during the best testing. For another example, the methodology can be applied to multiple languages such as English, Hindu, and Urdu. Besides experimental excellence, this HMM-based algorithm is also useful in solving other NLP problems involving part-of-speech tagging, classifications, etc. It can also be generalized for testing stories and sentences which can be commonly applicable.

In terms of the Support Vector Machine, the NER process is also a two-step process: training and classifications. [5] The training can be carried out by available SVM toolkits and tools such as YamCha and TinySVM Classifier. Within the toolkits, kernel functions can be used for the algorithm. First, the SVM algorithm will extract some patterns and match them against an unannotated corpus. Based on the result and corresponding classifications, the extracted entity will either be positive, negative, or error in the sample. These three categories will be further used to tag the training corpus. This entire process can be repeatedly conducted until reaches its limit. Using these patterns, we can apply an SVM model by combining the elements of the set containing the best features for NER. The features should include things like context word features, word suffixes, prefixes, etc.

4. Conclusion

In conclusion, both the Hidden Markov Model and Support Vector Machine represent methodologies developed in Natural Language processing and have been extensively applied in Name Entity Recognition. Amidst studies, these two models have been utilized differently based on their unique functionalities. While HMM appeals to a 90% accuracy with a variety of applications, the SVM algorithm is widely recognized for its repetitive and recursive nature of logic and principles. Overall, both models exert indispensable benefits in researching Natural Language Processing topics. However, based on the preference for a specific factor, this paper makes it self-evident whether a researcher should select statistical models or neural mod-

els.

References

[1] Sarle, Warren S. Neural Networks and Statistical Models. Proceedings of the Nineteenth Annual SAS Users Group International Conference, April 1994, SAS Institute Inc., Cary, NC, USA.

[2] Agarwal, Oshin, Yinfei Yang, Byron C. Wallace, and Ani Nenkova. Interpretability Analysis for Named Entity Recognition to Understand System Predictions and How They Can Improve

[3] Padmanabhan, Pyari. Named Entity Recognition using Statistical Model Approach. KMCT College of Engineering, University of Calicut, Kerala, India.

[4] Morwal, Sudha and Jahan, Nusrat and Chopra, Deepti, Named Entity Recognition using Hidden Markov Model (HMM). International Journal on Natural Language Computing (IJNLC) 2012(1), 4.

[5] .Ekbal, Asif, and Sivaji Bandyopadhyay. Named Entity Recognition using Support Vector Machine: A Language Independent Approach.