Named entity recognition for Fintech — NER in financial news

Yixi Zhou[1]*, Jingyi Zeng, Qianru Li

[1]Shanghaitech University
393 Middle Huaxia Road
Pudong, Shanghai, 201210
zhouyx2022@shanghaitech.edu.cn

Abstract

This paper is for the Shanghaitech CS181 Artificial intelligence in 24Spring final project. We apply the Hidden Markov Model(HMM) we learned in our course to train out NER model for Fintech, which can extract the named entity from the financial news. However, HMM assumes that each word in the observed sentence is independent of each other, which is flawed in solving the problem of named entity recognition, and then we try the Conditional random Field(CRF) to utilize the dependency between the words in financial news.

The application of artificial intelligence in the financial field has become more and more common. And in a specific field, we know that financial news has a great influence in the financial market, for example, it can affect market sentiment. We can further facilitate our work by extracting named entities from the news, for example, by building a knowledge graph, establishing a connection between the named entities in the news and stock price changes, or finding associations between the named entities. So the prerequisite for these tasks is to perform named entity recognition on the news articles.

Although named entity recognition techniques have already been applied in many domains, in the financial field, due to the rapid updates of news, new proper nouns often emerge. Therefore, we need timely data and a reasonable named entity recognition model.

In the research process, we found that the mainstream methods for named entity recognition are based on deep learning algorithms such as Bi-LSTM and Bi-LSTM+CRF. However, these algorithms have not been covered in our course, so we will instead use the HMM (Hidden Markov Model) that we have learned in class for training. However, we found that the HMM model has two assumptions: 1) the observed values are strictly independent of each other, and 2) the state transition process is only related to the previous state. In the scenario of named entity recognition, HMM considers each character in the observed sentence to be independent, and the current moment's annotation is only related to the previous moment's annotation.

But in reality, named entity recognition often requires more features, such as part-of-speech, context of words, and the current moment's annotation should be related to the previous and next moment's annotation as well. Due to these two assumptions, the HMM model is obviously deficient in solving the problem of named entity recognition.

On the other hand, Conditional Random Fields (CRFs) do not have this problem. By introducing customizing feature functions, CRFs can not only express the dependence between observations, but also represent the complex dependence between the current observation and multiple previous and next states, effectively overcoming the problems faced by the HMM model.

Finally, in our preliminary research, we also found that some researchers have explored using better large model prompts for named entity annotation, and we will also use this method to simplify the initial data annotation.

The following article will be divided into five parts. The first part will discuss the latest news data collection and the prompt + human-annotated labeling method for large models. The second part will introduce the HMM model we have built. The third part will briefly introduce the use of the CRF model for named entity recognition. The fourth part is an evaluation of the above methods. In the last part, we will summarize and look ahead to the above methods.

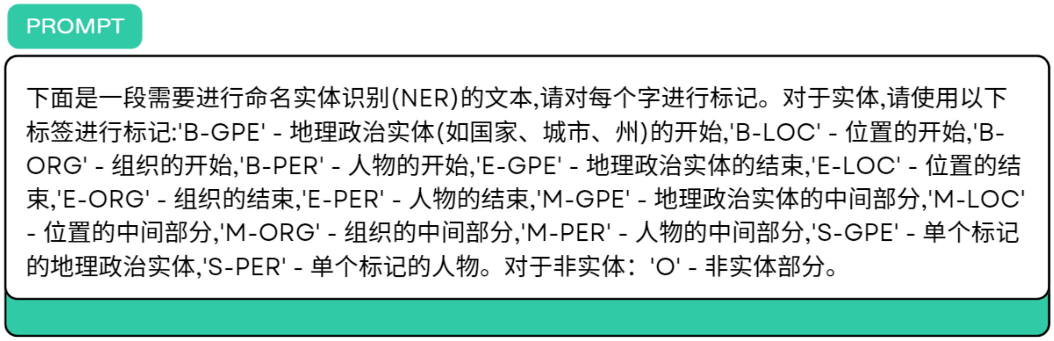
Data Collection and Annotation

To obtain the news dataset, we collected thousands of financial news articles from April 2024 to June 2024 from public financial news sources such as the People's Bank of China, Yahoo Finance, Sina Finance, and investment banks, as well as the data collected from Sina Finance in the paper ACL 2018 Chinese NER using Lattice LSTM. We labeled each entity in the news articles as shown in the figure, with each line containing a character and its corresponding label. The label set uses the BMOES format (B for beginning, E for end, M for inside, O for outside of an entity) in tables1, with a blank line separating sentences.

In the preliminary research, we read papers related to using large language models for named entity recognition. Thanks to Professor Ren's suggestion, we will try to use large language models to label the corresponding data in order to reduce the manual labeling effort. Our prompts are as follows in Figure1.

However, we found that although the large language model has been trained on more financial news data, for

[*] These authors contributed equally.



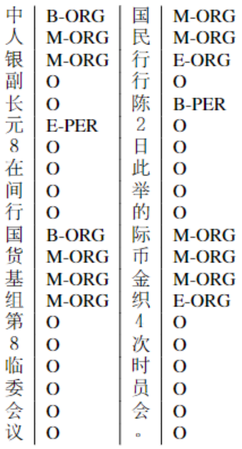
PROMPT

下面是一段需要进行命名实体识别(NER)的文本,请对每个字进行标记。对于实体,请使用以下标签进行标记:'B-GPE' - 地理政治实体(如国家、城市、州)的开始,'B-LOC' - 位置的开始,'B-ORG' - 组织的开始,'B-PER' - 人物的开始,'E-GPE' - 地理政治实体的结束,'E-LOC' - 位置的结束,'E-ORG' - 组织的结束,'E-PER' - 人物的结束,'M-GPE' - 地理政治实体的中间部分,'M-LOC' - 位置的中间部分,'M-ORG' - 组织的中间部分,'M-PER' - 人物的中间部分,'S-GPE' - 单个标记的地理政治实体,'S-PER' - 单个标记的人物。对于非实体：'O' - 非实体部分。

Figure 1: We design the prompt for LLM.

'B-GPE'	Beginning of a Geopolitical Entity
'B-LOC'	Beginning of a Location
'B-ORG'	Beginning of an Organization
'B-PER'	Beginning of a Person
'E-GPE'	End of a Geopolitical Entity
'E-LOC'	End of a Location
'E-ORG'	End of an Organization
'E-PER'	End of a Person
'M-GPE'	Middle of a Geopolitical Entity
'M-LOC'	Middle of a Location
'M-ORG'	Middle of an Organization
'M-PER'	Middle of a Person
'S-GPE'	Single-token Geopolitical Entity
'S-PER'	Single-token Person
'O'	Outside of an entity (non-entity)

Table 1: Named Entity Recognition Tags



中	B-ORG	国	M-ORG
人	M-ORG	民	M-ORG
银	M-ORG	行	E-ORG
副	O	行	O
长	O	陈	B-PER
元	E-PER	2	O
8	O	日	O
在	O	此	O
间	O	举	O
行	O	的	O
国	B-ORG	际	M-ORG
货	M-ORG	币	M-ORG
基	M-ORG	金	M-ORG
组	M-ORG	织	E-ORG
第	O	4	O
8	O	次	O
临	O	时	O
委	O	员	O
会	O	会	O
议	O	。	O

Figure 2: We design the prompt for LLM.

some entity nouns that have newly appeared in financial news, the large language model is unable to correctly perform named entity tagging. We manually corrected the data that the large language model tagged incorrectly, and cleaned the labeled dataset, dividing it into training and test sets. We ensure that the data in the training and test sets do not overlap, and we have open-sourced all the datasets on GitHub.

After obtaining the training dataset, we can assign a numerical ID to each label (tag) and each unique word that appears in the dataset. This process is referred to as:

1. tag2id: This is a dictionary or mapping that associates each unique label or tag with a numerical ID. For example, 'B-ORG' might be assigned the ID 0, 'M-ORG' might be 1, 'B-PER' might be 2, and so on.

2. word2id: This is a dictionary or mapping that associates each unique word in the dataset with a numerical ID. This allows you to represent the words in the dataset using these numerical IDs instead of the actual text.

HMM Named Entity Recognition

Hidden Markov models describe a process where an unobservable random sequence of states is generated by a hidden Markov chain, and then each state generates an observation, producing an observed random sequence. Hidden Markov models are determined by the initial state distribution, the state transition probability matrix, and the observation probability matrix.

HMM model in our project

In the process of named entity recognition (NER) for a news article, the observable objects are the individual characters in the news article, and the hidden states are the label tags behind each character(The relation is shown in Figure 3. The transition probabilities from one tag to another form the transition matrix, and the probabilities from the tags to the individual characters form the observation probability matrix.

During the test phase, we are given a news article that has not appeared in the training set. Based on these observable quantities, we can use the Viterbi algorithm to calculate the most probable sequence of tags, which allows us to complete the task of named entity recognition, the algorithm is shown in algorithm 1.

In other words, the Viterbi algorithm is used to infer the optimal sequence of hidden states (the named entity tags) given the observed sequence of characters in the news article. This is a classic application of hidden Markov models

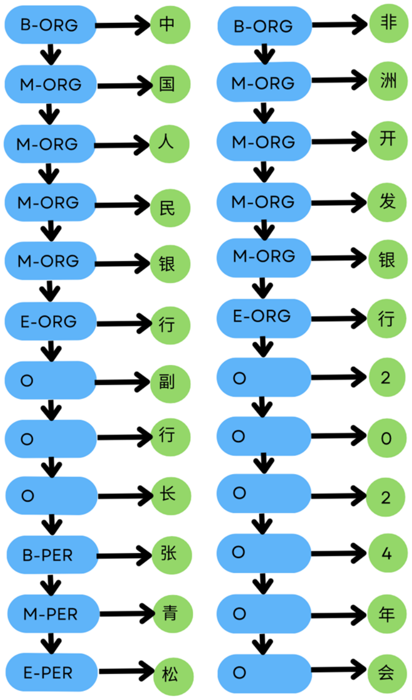


Figure 3: HMM network

(HMMs) in natural language processing tasks like NER.

Training the HMM model

The **HMM** class encapsulates the essential elements of an HMM, including the state transition probabilities, emission probabilities, and initial state probabilities.

The initial part of method initializes these core HMM components, setting up the necessary data structures to store the model parameters. The number of states (N) corresponds to the distinct tags or labels in the problem domain, while the number of observations (M) represents the unique words or tokens.

The crucial step is the **train** method, which takes in the labeled training data and estimates the HMM model parameters. This is achieved through the following key steps:

- Estimating the state transition probabilities (A) by counting the transitions between tags in the training sequences and normalizing the values.
- Estimating the emission probabilities (B) by counting the occurrences of words given their corresponding tags in the training data, again normalizing the values.
- Estimating the initial state probabilities (Pi) by counting the frequencies of the first tag in the training sequences and normalizing the values.
- **To address the issue of zero-valued elements in the probability distribution, we apply a small adjustment to the matrix. Specifically, any elements in the matrix that are equal to 0 are replaced with a very small value, 1e-10. This step ensures that the matrix does not contain any zero-valued probabilities, as such values would cause issues in subsequent calculations and modeling.**

Algorithm 1: Viterbi Algorithm

```
Input: Observations of length $T$, state-graph of length $N$
Output: Best path, path probability
Create a path probability matrix $\mathbf{viterbi}[N, T]$
for each state $s$ from 1 to $N$ do
    $\mathbf{viterbi}[s, 1] \leftarrow \pi_s * b_s(o_1)$
    $\mathbf{backpointer}[s, 1] \leftarrow 0$
end for
for each time step $t$ from 2 to $T$ do
    for each state $s$ from 1 to $N$ do
        $\mathbf{viterbi}[s, t] \leftarrow \max_{s'=1}^{N} \mathbf{viterbi}[s', t-1] * a_{s',s} * b_s(o_t)$
        $\mathbf{backpointer}[s, t] \leftarrow \arg\max_{s'=1}^{N} \mathbf{viterbi}[s', t-1] * a_{s',s} * b_s(o_t)$
    end for
end for
$\mathbf{bestpathprob} \leftarrow \max_{s=1}^{N} \mathbf{viterbi}[s, T]$
$\mathbf{bestpathpointer} \leftarrow \arg\max_{s=1}^{N} \mathbf{viterbi}[s, T]$
$\mathbf{bestpath} \leftarrow$ the path starting at state $\mathbf{bestpathpointer}$, that follows $\mathbf{backpointer}$ to states back in time
return $\mathbf{bestpath}, \mathbf{bestpathprob}$
```

Question anout the transition matrix

However, what we need to declare is: for some cases where the transition probability between certain entity tags is zero, it may not necessarily be because the training data sample is too small, but rather because such a transition relationship does not actually exist in real-world scenarios. For example, in Chinese, there is a connecting word between an organization and the name of one of its employees, so there must be a relationship between the organization and the person entity. Therefore, the transition between organization and person cannot always be the non-entity (O) tag.

In other words, the zero transition probabilities between certain entity tags may not just be due to insufficient training data, but could reflect the inherent relationships (or lack thereof) between different entity types in the real world. This is an important consideration when designing and training the HMM model, as it suggests that the model architecture and parameter constraints should be informed by domain knowledge, not just the statistics of the training data. Insufficient or imbalanced training data:

- If certain entity types appear very rarely in the training data, the model will have difficulty learning their features and is likely to misclassify them as the ’O’ class.
- If the training data contains far more samples of certain entity types than others, the model will perform very well on the majority types but less well on the minority types.

Algorithm 2: Counting Transitions in Tag Sequences

```
Input: $tag\_lists$ - a collection of tag sequences
Parameter: $tag2id$ - a function that maps tags to integer IDs
Output: $A$ transition matrix where $A[i][j]$ represents the number of transitions from tag $i$ to tag $j$
Initialize a 2D matrix $A$ of size $N \times N$, where $N$ is the total number of unique tags, with all elements set to 0.
for each $tag\_list$ in $tag\_lists$ do
    $seq\_len \leftarrow |tag\_list|$
    for $i = 0$ to $seq\_len - 1$ do
        $current\_tagid \leftarrow tag2id(tag\_list[i])$
        $next\_tagid \leftarrow tag2id(tag\_list[i+1])$
        $A[current\_tagid][next\_tagid] \leftarrow A[current\_tagid][next\_tagid] + 1$
    end for
end for
return $A$
```

- Solution: Increase the weight of the minority named entities

First, we define a set of possible entity tags. Then, it iterates through different combinations of entity tag sets and weight values, trains and tests the HMM model, and calculates the average precision as an evaluation metric. During this process, the code dynamically updates the best entity tag combination, weight value, and the corresponding highest F1 score, and finally outputs these optimal results. The overall process aims to find the optimal configuration of entity tags and weights that can maximize the performance of the HMM model on the given test data by adjusting the entity tags and weights. Our algorithm is in algorithm 5.

After the Optimal HMM Hyperparameter Search, the precision rises from 86.48% to the 86.78%.

Testing for the HMM model

The following test part implements the Viterbi algorithm, a dynamic programming technique used for decoding in HMM . The primary purpose of this decoding method is to infer the most likely sequence of hidden states given an observed sequence of words.

$$
\begin{aligned}
\boldsymbol{m}_{1:t+1} &= \mathrm{VITERBI}\left(\boldsymbol{m}_{1:t}, e_{t+1}\right) \\
&= P\left(e_{t+1} \mid X_{t+1}\right) \max_{x_t} P\left(X_{t+1} \mid x_t\right) \boldsymbol{m}_{1:t}\left[x_t\right]
\end{aligned}
$$

Here's a detailed explanation of the Viterbi algorithm:

- The method takes in three parameters: word_list (the sequence of words to be labeled), word2id (a dictionary mapping words to their unique integer IDs), and tag2id (a dictionary mapping tags to their unique integer IDs).
- The code first converts the HMM model parameters (state transition probabilities A, emission probabilities B, and initial state probabilities Pi) to their log-space representations. **This is done to address the potential underflow issues that can occur when multiplying many small probabilities.**

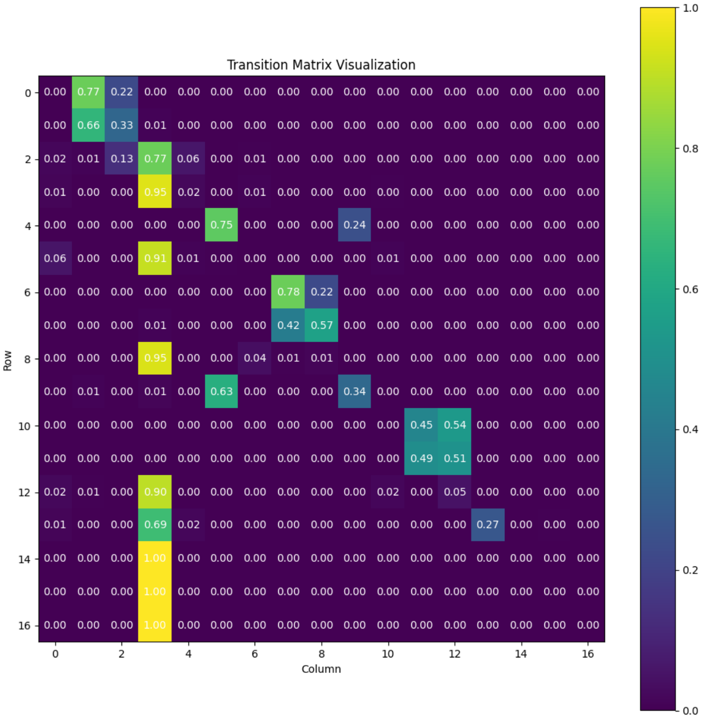


Figure 4: transition matrix

- The Viterbi matrix **viterbi** and the backpointer matrix **backpointer** are initialized. The **viterbi** matrix stores the maximum probability of reaching a particular state at a given time step, while the **backpointer** matrix stores the previous state that led to the maximum probability.
- The first step of the Viterbi algorithm is handled separately. The initial probabilities are set based on the first word in the sequence. **If the first word is not found in the word2id dictionary, a uniform probability distribution is assumed.**
- The main Viterbi recursion is then performed for the remaining time steps. For each time step and each possible tag, the maximum probability of reaching that tag is computed based on the probabilities from the previous time step, the state transition probabilities, and the emission probabilities.
- After the Viterbi recursion, the method finds the maximum probability path by tracing back the **backpointer** matrix. The best path probability and the corresponding state sequence (tag list) are returned.
- The final step is to convert the sequence of state IDs (in the best_path list) back to the corresponding tags using the id2tag dictionary.

The Viterbi algorithm is a crucial component in HMM-based sequence labeling tasks, as it allows for efficient and optimal inference of the most likely hidden state sequence given the observed data.

```
Algorithm 3: Counting Word-Tag Associations
Input: $tag\_lists, word\_lists$ - two parallel collections of tag sequences and word sequences
Parameter: $tag2id, word2id$ - functions that map tags and words to integer IDs
Output: $B$ - a matrix where $B[i][j]$ represents the probability of word $j$ given tag $i$
Initialize a 2D matrix $B$ of size $M \times N$, where $M$ is the total number of unique tags and $N$ is the total number of unique words, with all elements set to 0.
for $tag\_list, word\_list$ in $zip(tag\_lists, word\_lists)$ do
    $assert\ len(tag\_list) == len(word\_list)$
    for $tag, word$ in $zip(tag\_list, word\_list)$ do
        $tag\_id \leftarrow tag2id(tag)$
        $word\_id \leftarrow word2id(word)$
        $B[tag\_id][word\_id] \leftarrow B[tag\_id][word\_id] + 1$
    end for
end for
for $i = 0$ to $M - 1$ do
    for $j = 0$ to $N - 1$ do
        if $B[i][j] == 0$ then
            $B[i][j] \leftarrow 1e - 10$
        end if
    end for
    $B[i] \leftarrow B[i]/sum(B[i])$
end for
return $B$
```

Try on CRF for NER

In the scenario of named entity recognition, HMM considers each character in the observed sentence to be independent, and the current moment's annotation is only related to the previous moment's annotation.

But in reality, named entity recognition often requires more features, such as part-of-speech, context of words, and the current moment's annotation should be related to the previous and next moment's annotation as well. Due to these two assumptions, the HMM model is obviously deficient in solving the problem of named entity recognition.

On the other hand, Conditional Random Fields (CRFs) do not have this problem. By introducing customizable feature functions, CRFs can not only express the dependence between observations, but also represent the complex dependence between the current observation and multiple previous and next states, effectively overcoming the problems faced by the HMM model.

However, owing to the coverage of our class CS181, we lack a lot of the knowledge of Conditional Random Fields. We failed to complete the CRF model completely by ourselves, so in the code part we can only use third-party libraries to implement the methods. For some parameters and computation methods in the third-party libraries, we do not have sufficient knowledge to resolve them. Therefore, the purpose of this part is only to compare the accuracy of named entity recognition with the previous HMM part, and it is not part of the workload for this project. We strongly believe in the future when we get enough knowledge accumulation we can solve this problem by ourselves.

```
Algorithm 4: Estimating Initial Tag Probabilities
Input: $tag\_lists$ - a collection of tag sequences
Parameter: $tag2id$ - a function that maps tags to integer IDs
Output: $\Pi$ - a vector where $\Pi[i]$ represents the probability of the initial tag being $i$
Initialize a vector $\Pi$ of size $N$, where $N$ is the total number of unique tags, with all elements set to 0.
for $tag\_list$ in $tag\_lists$ do
    $init\_tagid \leftarrow tag2id(tag\_list[0])$
    $\Pi[init\_tagid] \leftarrow \Pi[init\_tagid] + 1$
end for
for $i = 0$ to $N - 1$ do
    if $\Pi[i] == 0$ then
        $\Pi[i] \leftarrow 1e - 10$
    end if
end for
$\Pi \leftarrow \Pi/sum(\Pi)$
return $\Pi$
```

Method Evaluation

In this study, we evaluated the performance of hidden Markov models (HMMs) and conditional random fields (CRFs) on the task of named entity recognition. We used a standard named entity recognition dataset and calculated the precision, recall, and F1-score for both methods.

Precision

- Precision refers to the ratio of correctly identified named entities to the total number of named entities identified by the system.
- The formula is: Precision = Number of true named entities / Total number of named entities identified by the system
- A high precision indicates that the system is accurately identifying named entities.

Recall

- Recall refers to the ratio of named entities correctly identified by the system to the total number of actual named entities in the data.
- The formula is: Recall = Number of true named entities identified by the system / Total number of actual named entities
- A high recall indicates that the system is able to identify a large proportion of the named entities present in the data.

F1-score

- The F1-score is the harmonic mean of precision and recall, providing a balanced metric for evaluating the overall performance of the system.
- The formula is: F1-score = 2 * (Precision * Recall) / (Precision + Recall)

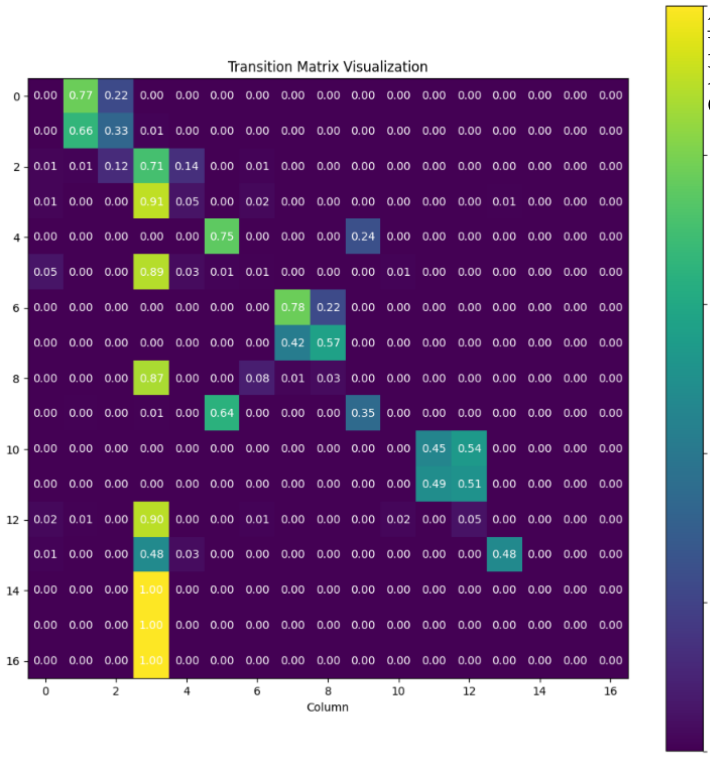


Figure 5: weighted transition matrix

Algorithm 5: Optimal HMM Hyperparameter Search

Input: Training data, test data, vocabulary, list of entity tags
Parameter: Weight range, weight step
Output: Optimal entity tags, optimal weight, best F1 score

```
Define the possible entity tags: $entity\_tags = ['GPE', 'LOC', 'ORG', 'PER']$
Define the weight range and step: $weight\_range = [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5]$, $weight\_step = 0.5$
Initialize the best evaluation metric and the corresponding entity tags and weight: $best\_f1 = 0$, $best\_entity\_tags = []$, $best\_weight = 0$
for $tags$ in $\texttt{powerset}(entity\_tags)$ do
    for $weight$ in $weight\_range$ do
        Initialize the HMM model: $hmm = \texttt{HMM}(len(set(tags)), len(vocabulary))$
        Train the HMM model: $hmm.\texttt{train}()$
        Evaluate the model on the test data: $pred\_tags = hmm.\texttt{test}(test\_data)$
        Calculate the F1 score: $f1 = \texttt{calculate\_f1}(pred\_tags, test\_labels)$
        if $f1 > best\_f1$ then
            $best\_f1 = f1$
            $best\_entity\_tags = tags$
            $best\_weight = weight$
        end if
    end for
end for
return $best\_entity\_tags$, $best\_weight$, $best\_f1$
```

- A high F1-score indicates that the system has a good balance between precision and recall.

Confusion Matrix

- The confusion matrix is a fundamental tool in evaluating the performance of classification models. It provides a comprehensive view of a model's predictions by presenting the counts of true positives, true negatives, false positives, and false negatives.

For the HMM model, we observed a precision of 86.48%, a recall of 87.24%, and an F1-score of 86.84%. This indicates that the HMM model was able to reasonably identify named entities, but exhibited some errors in its recognition. And the result is shown in Figure 9. The confusion matrix is shown in Figure 10. In contrast, the CRF-based model demonstrated superior performance. The CRF model achieved a precision of 91.27%, a recall of 93.13%, and an overall F1-score of 91.24%. This suggests that the CRF model was able to more accurately identify named entities and had a broader coverage. And the result is shown in Figure 11. The confusion matrix is shown in Figure 12.

Analysis

- Feature Richness:
  - CRF can utilize richer features, including features of the observation sequence, features of the label sequence, and features between observations and labels. These complex features can better describe the complexity of the problem.
  - While HMM mainly relies on state transition probability and observation probability, its feature expression ability is relatively weak.
- Global Optimization:
  - CRF adopts a globally conditional probability-based training method, which can perform joint prediction on the entire sequence to obtain the globally optimal label sequence.
  - HMM, on the other hand, adopts a locally probability-based training method, which only focuses on the state transition at the current moment and cannot capture global dependencies.
- Label Bias Problem:
  - HMM is influenced by the frequency of labels during training, which can easily lead to the label bias problem. That is, the model tends to predict the labels with higher frequency.
  - CRF can balance the importance of different label categories by introducing weight factors, which can better handle the label bias problem.

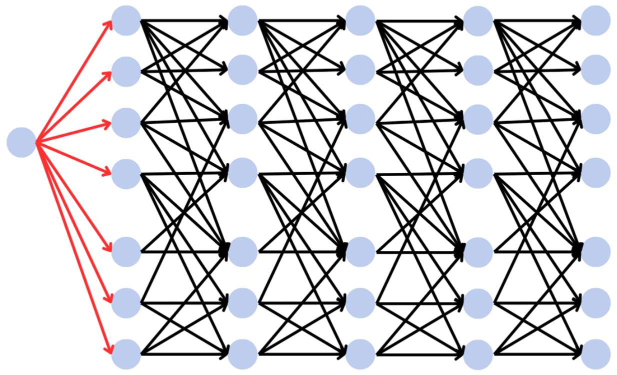


Figure 6: HMM model used for viterbi algorithm

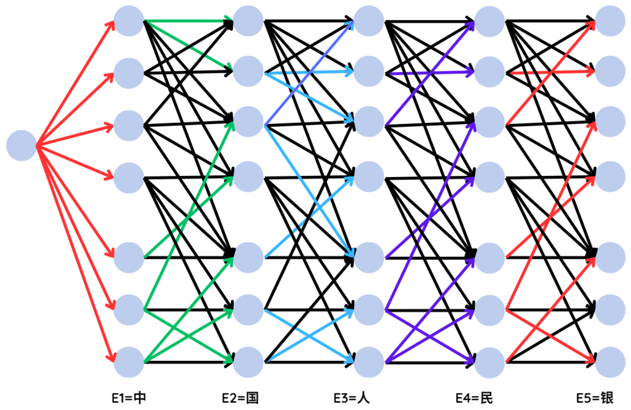


Figure 7: After the Viterbi recursion

Reference

Reference for the theory

In order to make our project complete and sound, we refer to some theoretical materials.

We use this thesis to have a better prompt for our LLM-generating named-entity:
Dhananjay Ashok, Z. C. L. 2023. PromptNER : Prompting For Named Entity Recognition. On arxiv.

We use this material to build up our viterbi algorithm in HMM:
Martin, D. J. . J. H. 2024. Hidden Markov Models. *Speech and Language Processing*, 1(1): 8–9.

We use this material to start a frame of HMM:
QAQ. 2024. Explain step-by-step how to implement Named Entity Recognition using Hidden Markov Models. https://blog.csdn.net/qq_41496421/article/details/127623738. Accessed: 2024-06-06.

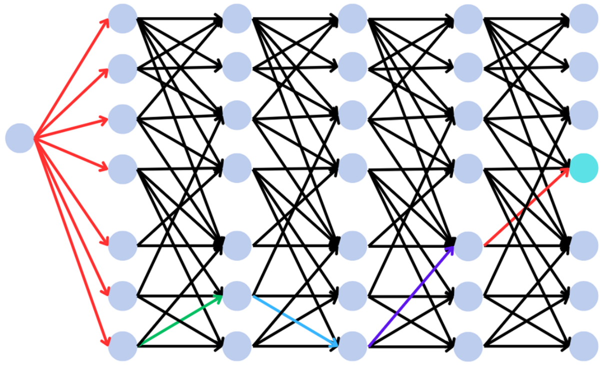


Figure 8: Find the maximum probability path

Algorithm 6: CRF Sequence Labeling Model

```
Input: List of sentences, List of label sequences
Parameter: Algorithm ('lbfgs'), Regularization parameters (c1=0.1, c2=0.1), Max iterations (100), All possible transitions (False)
Output: List of predicted label sequences
1: Define the word2features function to extract features for a single word
2: Define the sent2features function to extract features for an entire sentence
3: Create a CRF model object with the specified parameters
4: Convert the list of sentences into a list of feature sequences
5: Train the CRF model using the feature sequences and label sequences
6: Convert the input list of sentences into a list of feature sequences
7: Use the trained CRF model to predict the labels for each sentence
8: return the list of predicted label sequences
```

Reference for the data

To generate the data satisfy our requirement, we refer to some existing dataset and their ways of lable:

We use some important datasets here:
Smoothnlp. 2019. FinancialDatasets for LatticeLSTM. https://github.com/jiesutd/LatticeLSTM/tree/master. Accessed: 2024-06-06.

We use some guidance here how to label our data:
Jiesutd. 2019. FinancialDatasets. https://github.com/smoothnlp/FinancialDatasets. Accessed: 2024-06-06.

Reference for the library

For better coding and visualization, we import some third-party libraries into our project

- from **sklearn_crfsuite** import CRF, we use this because

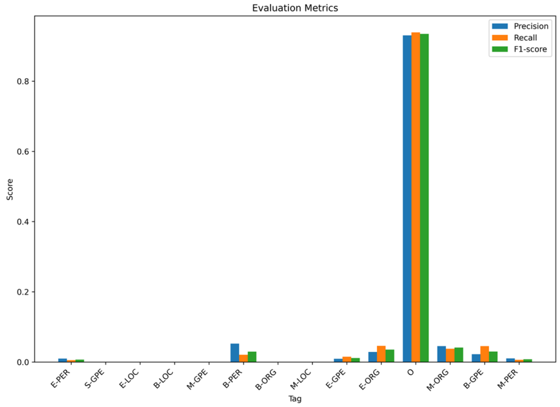
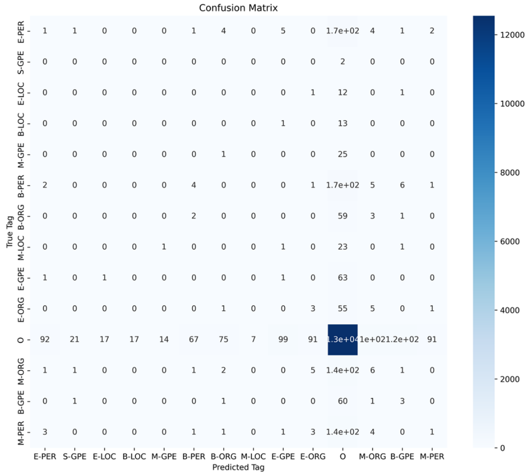


Figure 9: Score for HMM



	E-PER	S-GPE	E-LOC	B-LOC	M-GPE	B-PER	B-ORG	M-LOC	E-GPE	E-ORG	O	M-ORG	B-GPE	M-PER
E-PER	1	1	0	0	0	1	4	0	5	0	1.7e+02	4	1	2
S-GPE	0	0	0	0	0	0	0	0	0	0	2	0	0	0
E-LOC	0	0	0	0	0	0	0	0	0	1	12	0	1	0
B-LOC	0	0	0	0	0	0	0	0	1	0	13	0	0	0
M-GPE	0	0	0	0	0	0	1	0	0	0	25	0	0	0
B-PER	2	0	0	0	0	4	0	0	0	1	1.7e+02	5	6	1
B-ORG	0	0	0	0	0	2	0	0	0	0	59	3	1	0
M-LOC	0	0	0	0	1	0	0	0	1	0	23	0	1	0
E-GPE	1	0	1	0	0	0	0	0	1	0	63	0	0	0
E-ORG	0	0	0	0	0	0	1	0	0	3	55	5	0	1
O	92	21	17	17	14	67	75	7	99	91	1.3e+04	1e+02	1.2e+02	91
M-ORG	1	1	0	0	0	1	2	0	0	5	1.4e+02	6	1	0
B-GPE	0	1	0	0	0	0	1	0	0	0	60	1	3	0
M-PER	3	0	0	0	0	1	1	0	1	3	1.4e+02	4	0	1

Figure 10: Confusion Matrix HMM

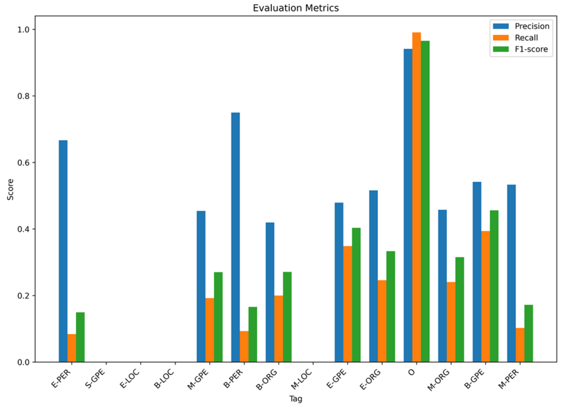
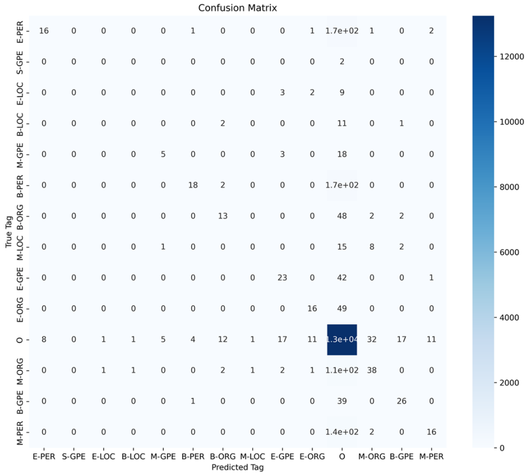


Figure 11: Score for CRF



	E-PER	S-GPE	E-LOC	B-LOC	M-GPE	B-PER	B-ORG	M-LOC	E-GPE	E-ORG	O	M-ORG	B-GPE	M-PER
E-PER	16	0	0	0	0	1	0	0	0	1	1.7e+02	1	0	2
S-GPE	0	0	0	0	0	0	0	0	0	0	2	0	0	0
E-LOC	0	0	0	0	0	0	0	0	3	2	9	0	0	0
B-LOC	0	0	0	0	0	0	2	0	0	0	11	0	1	0
M-GPE	0	0	0	0	5	0	0	0	3	0	18	0	0	0
B-PER	0	0	0	0	0	18	2	0	0	0	1.7e+02	0	0	0
B-ORG	0	0	0	0	0	0	13	0	0	0	48	2	2	0
M-LOC	0	0	0	0	1	0	0	0	0	0	15	8	2	0
E-GPE	0	0	0	0	0	0	0	0	23	0	42	0	0	1
E-ORG	0	0	0	0	0	0	0	0	0	16	49	0	0	0
O	8	0	1	1	5	4	12	1	17	11	1.3e+04	32	17	11
M-ORG	0	0	1	1	0	0	2	1	2	1	1.1e+02	38	0	0
B-GPE	0	0	0	0	0	1	0	0	0	0	39	0	26	0
M-PER	0	0	0	0	0	0	0	0	0	0	1.4e+02	2	0	16

Figure 12: Confusion Matrix CRF

we don't have the basic knowledge of CRF, we just import it for the comparison with HMM model.

- import **torch**, PyTorch provides powerful tensor manipulation capabilities, which makes it convenient to represent and compute the probability matrices (transition probability matrix, observation probability matrix) and the initial state probability vectorthe core parameters of the HMM model. PyTorch allows for easy matrix and vector operations.
- import **matplotlib.pyplot** as plt, import **seaborn** as sns for visualization

Announcement of the LLMs

To reduce some of the repetitive mechanical work, we have used Large Language Model to assist with a portion of the content in this project. However, **the core HMM code section will not involve LLM assistance**. We will declare the areas where we have used LLMs:

- In the data annotation part of named entity recognition, we used LLM for preliminary labeling, and completed the annotation of financial news using human verification.
- In the evaluation function part, we used LLM to assist with the visualization work and guide us which index should be evaluated.
- During the paper writing process, we used LLM for partial paragraph translation and grammar checking.
- In the principle analysis process, we used LLM to write pseudocode.
- For the tables in the LaTeX part, we used LLM for typesetting.

Acknowledgments

Thank you for reading our paper carefully. We look forward to your suggestions about our project!