Multimedia Tools and Applications (2023) 82:803–817
https://doi.org/10.1007/s11042-022-13303-2

A conditional random field framework for language process in product review mining

Yue Ming[1] · Xiyuan Liu[2] · Gang Shen[3] · Di Gao[4] · Yu Wang[5]

Received: 1 August 2021 / Revised: 25 January 2022 / Accepted: 30 May 2022 /
Published online: 10 June 2022

Abstract
The Part-Of-Speech tagging is widely used in the natural language process. There are many statistical approaches in this area. The most popular one is Hidden Markov Model. In this paper, an alternative approach, linear-chain Conditional Random Fields, is introduced. The Conditional Random Fields is a factor graph approach that can naturally incorporate arbitrary, non-independent features of the input without conditional independence among the features or distributional assumptions of inputs. This paper applied the Conditional Random Fields for the car review word Part-Of-Speech tagging and then the feature extraction, which can be used as an input to an opinion mining system. To reduce the computational time, we also proposed applying the Limited-memory BFGS algorithm to train the Conditional Random Fields. Furthermore, this paper evaluated the Conditional Random Fields and the classical graph approach using the car review dataset to demonstrate that the Conditional Random Fields have a more robust result with a smaller training dataset.

Keywords Conditional Random Fields (CRFs) · Natural language processing · POS tagging · Text mining · HMM · L-BFGS algorithm

1 Introduction

With the rapid growth of e-commerce, people are more likely to share their opinions and hands-on experiences on products or services they have purchased. This information is important for both business organizations and potential customers. Companies can make

✉ Xiyuan Liu
liuxyuan@latech.edu

Yue Ming
ming.stat@hotmail.com

1 Syngenta Seeds, LLC, Basel, Switzerland
2 Department of Mathematics and Statistics, Louisiana Tech University, Ruston, LA, USA
3 Department of Statistics, North Dakota State University, Fargo, ND, USA
4 Department of Mathematics and Statistics, Sam Houston State University, Huntsville, TX, USA
5 Department of Statistics, Texas A&M University, College Station, TX, USA

decisions on their strategies for marketing and products improvement, which Customers can make a better decision when purchasing the products or services. Unfortunately, the number of reviews has reached to more than hundreds of thousands in recent days, especially for popular products, which hence poses a challenge for a potential customer to go over all of them. Therefore, it is essential to provide coherent and concise summaries for the reviews.

Researchers have explored different angles on opinion mining to tackle this problem, aiming to extract the essential information from reviews and present it to the users. Previous works have mainly adopted rule-based techniques [3] and statistic methods [10]. Later, a machine learning approach based on the Hidden Markov model (HMMs) was proposed and proved more effective than previous works. However, these HMMs-based methods are limited because it is difficult to model arbitrary, dependent features of the input word sequence.

A Conditional Random Field (CRFs) was introduced to fix this limitation [Lafferty 2001]. Later on, the CRF framework was summarized [Sutton 2012]. The CRFs is a discriminant, factor graph model with the potential to model overlapping and dependent features. Prior works on natural language processing (NLP) have demonstrated that CRFs outperform the classical HMMs [Peng 2006].

Motivated by the findings, we propose a linear-chain CRF-based framework to mine and extract opinions from product reviews on the web. The performance of the CRFs is impressive as the training data for the CRFs is minimal, and the CRFs still perform a relatively similar result comparing with the classical POS tagging method.

The rest of this paper is organized as follows: we will describe the proposed framework and the CRFs for the framework in Section 2. Section 3 demonstrates the experiment result. Section 4 demonstrates a further application using CRFs: feature extraction, i.e., extracting keywords from a sentence. Section 5 summarizes our work, and Section 6 present its future directions.

2 Methodology

Before applying the CRFs to the POS tagging, some problems need to be solved. First is the pre-data process. Then, the feature design for the CRFs. At last, parameter estimation for the CRFs.

2.1 Proposed framework

The architectural overview of the framework can be divided into the following steps: First, pre-processing that includes crawling raw review data and cleaning. Step 2, POS tagging on review data. In this step, we manually labeled the data using the Penn Treebank POS tagging. Step 3, training the linear-chain CRFs model using the pre-defined POS tagging. Step 4, applying the model to the test set and extract opinions. For comparison, the Python Natural Language Toolkit (NLTK 3.3) is applied [2]. Step 5, we used the POS tagging result generated by the CRFs model to further extract opinions by extracting only Nouns and Adjectives words from the review sentences.

2.2 Conditional random fields

Conditional random fields (CRFs) are conditional probability distributions on an undirected graph model [Lafferty 2001]. To reduce the complexity, we employed linear-chain CRFs as an approximation to restrict the relationship among tags. A 1^{st} order CRF (X, Y) is specified by a vector F of local features and a corresponding weight vector λ. Each local feature is either a transition feature A_{y_{t-1},y_t} or an emission feature O_{y_t,x_t}, where y is the label sequence, x is the input sequence, and t is the position of a token in the sequence. We define the 1^{st} order features:

- The assigment of current tag y_t is supposed to depend on the current word x_t only. The feature function is represented as an emission feature O_{y_t,x_t} in the form $F_k(y_t|x_t) = \mathbb{1}_{\{y_t=y\}}\mathbb{1}_{\{x_t=x\}}$.
- The assignment of current y_t is supposed to depend on the previous tag y_{t-1} only. The feature function is represented as a transition feature A_{y_{t-1},y_t} in the form $F_k(y_t|y_{t-1}) = \mathbb{1}_{\{y_{t-1}=y'\}}\mathbb{1}_{\{y_t=y\}}$.

With the definition of

$$F_k(y_{t-1}, y_t, x_t) = F_k(y_t|x_t)F_k(y_t|y_{t-1}),$$

the conditional probability can be written as:

$$P(y|x) = \frac{1}{Z(x)}\prod_{t=1}^{T}\exp\left\{\sum_{k=1}^{K}\lambda_k \cdot F_k(y_{t-1}, y_t, x_t)\right\} \tag{1}$$

where

$$Z(x) = \sum_{y}\left(\prod_{t=1}^{T}\exp\left\{\sum_{k=1}^{K}\lambda_k \cdot F_k(y_{t-1}, y_t, x_t)\right\}\right), \tag{2}$$

is called the partition function (or a normalization factor), which is the summation over all possible combinations of sequences (transitions and emissions). Hence, the most probable label sequence for input sequence x:

$$\hat{y} = \arg\max_{y} P(y|x) \tag{3}$$

can be found with Viterbi algorithm.

Therefore, the task of review mining can be transformed to an automatic labeling task, and the problem can then be formalized as: given a sequence of words $x = x_1x_2, ..., x_T$ and it's corresponding POS $y = y_1y_2, ..., y_T$, the objective is to find an appropriate sequence of tags which can maximize the conditional likelihood according to (3).

2.2.1 Parameter estimation

To estimate the parameters of a linear-chain CRF $\theta = \{\lambda_k\}$, given identically independent distributed (iid) training data $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$, where $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_{T_i}^{(i)}\}$ is the observation sequence and each $y^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \ldots, y_{T_i}^{(i)}\}$ is a sequence of the desired predictions (i.e. labels), the conditional log likelihood can be obtained as:

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^{N}\log P(y^{(i)}|x^{(i)}) \\ &= \sum_{i=1}^{N}\left\{\sum_{t=1}^{T_i}\sum_{k=1}^{K}\left[\lambda_k F_k(y_{t-1}^{(i)}, y_t^{(i)}, x_t^{(i)}) - \frac{\lambda_k^2}{2\sigma^2}\right]\right. \\ &\quad \left. - \log[Z(x^{(i)})]\right\}\end{aligned} \tag{4}$$

where $\sum_{k=1}^{K}\frac{\lambda_k^2}{2\sigma^2}$ is the $L2$ regularization term added to the likelihood function in order to reduce overfitting. σ is assigned a Gaussian prior and the value of σ^2 is often taken up to 10 (we take $\sigma^2 = 10$ in our experiment). Since in general the function $\ell(\theta)$ cannot be maximized in closed form, so dynamic programming and L-BFGS algorithm can be used to optimize objective function. The partial derivative, or the gradient of the objective function is computed as:

$$\begin{aligned}\frac{\partial \ell}{\partial \lambda_k} =& \sum_{i=1}^{N}\sum_{t=1}^{T_i} F_k(y_{t-1}^{(i)}, y_t^{(i)}, x_t^{(i)}) \\ & - \sum_{i=1}^{N}\sum_{y} F_k(y_{t-1}^{(i)}, y_t^{(i)}, x_t^{(i)}) P(y_{t-1}, y_t|x^{(i)}) - \frac{\lambda_k}{\sigma^2}\end{aligned} \tag{5}$$

where the first term is the empirical count of feature k in the training data, the second term is the expected count of this feature under the current trained model. Hence, the derivative measures the difference between the empirical count and the expected count of a feature under the current model.

In order to obtain the gradient (5), we need to calculate the conditional probability $P(y_{t-1}, y_t|x^{(i)})$ that requires the sum over the whole label sequence y, which is intractable in a naive fashion. Hence we need to employ some dynamic programming techniques for the calculation.

2.2.2 Dynamic programming for CRF probability as matrix computations

For a linear-chain CRF where each label sequence is augmented by $start$ and end states for y_0 and y_{t+1} respectively, the conditional probability of label sequence y given an observation sequence x can be efficiently computed using matrices.

Let $\mathcal{Y}$ be the collection of all possible labels, define a set of $n + 1$ matrices $\{M_t(x)|t = 1, \ldots, t + 1\}$, where each $M_t(x)$ is a $|\mathcal{Y}_{t-1} \times \mathcal{Y}_t|$ matrix with elements of the form:

$$M_t(y', y|x) = \exp\left[\sum_k \lambda_k F_k(y_{t-1}, y_t, x, t)\right] \tag{6}$$

Hence, the conditional probability can be written as the product of the appropriate elements of the $n + 1$ matrices for that pair of y and x sequences as

$$P(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T+1} M_t(y_{t-1}, y_t|x) \tag{7}$$

The partition function $Z(x)$ is given by the $(start, end)$ entry of the product of all $n + 1$ $M_t(x)$ matrices:

$$Z(x) = \left[\prod_{t=1}^{T+1} M_t(x)\right]_{start,end} \tag{8}$$

Therefore, the conditional probability can be calculated by a dynamic programming method that is similar to the forward-backward algorithm for HMMs. Define the forward and backward vectors α_t and β_t starting with the base cases:

$$\begin{aligned}\alpha_0 &= \begin{cases} 1 & \text{if } y = start \\ 0 & \text{otherwise} \end{cases} \\ \beta_{t+1} &= \begin{cases} 1 & \text{if } y = stop \\ 0 & \text{otherwise,} \end{cases}\end{aligned} \tag{9}$$

and for recurrence:

$$\begin{aligned}\alpha_t(x)^T &= \alpha_{t-1}(x)^T M_t(x)\\ \beta_t(x) &= M_{t+1}(x)\beta_{t+1}(x)\end{aligned} \tag{10}$$

Finally, the conditional probability can be written as:

$$\begin{aligned}&P(Y_{t-1} = y', Y_t = y|x^{(i)}, \lambda)\\ &= \frac{\alpha_{t-1}(y'|x)M_t(y',y|x)\beta_t(y|x)}{Z(x)}\end{aligned} \tag{11}$$

which can thus be plugged into (5) to calculate the gradient.

2.3 Training with limited-memory quasi-newton method

The traditional Newton methods for nonlinear optimization require calculating the inverse of the Hessian matrix (curvature information) of the log-likelihood to find the search direction, which in our case, is impractical. Limited-memory BFGS (L-BFGS) estimates the curvature information based on previous m gradients and weight updates. There is no theoretical guidance on how much information from previous steps should be kept to obtain sufficiently accurate curvature estimates. In our experiment, we used the previous $m = 10$ gradient and weight pairs, which worked well.

Assume all vectors are column vectors, given λ_k as the updates at the k^{th} iteration, and the gradient $g_k \equiv \nabla f(\lambda_k)$ where f is the objective function being minimized (negative log likelihood). The last m updates of the form $s_k = \lambda_{k+1} - \lambda_k$ and $y_k = g_{k+1} - g_k$ are stored. Define $\rho_k = \frac{1}{y_k^T s_k}$, and $H_k^0 = \frac{y_{k-1}s_{k-1}^T}{y_{k-1}^T y_{k-1}}$ as the initial approximate of the inverse Hessian at k^{th} iteration. The search direction $d_k = -H_k g_k$ can be approached through two-loop recursion [6]

– 1^{st} *Loop:* Define a sequence of vectors

$$q_k[q_{k-m}, ..., q_k] = g_k$$

and its element

$$q_i := \left(I - \rho_i y_i s_i^T\right) q_{i+1}.$$

Define $a_i = \rho_i s_i^T q_{i+1}$, and hence the first recursion calculates $q_i = q_{i+1} - a_i y_i$.

– 2^{nd} *Loop:* Define another sequence of vectors where each element $z_i[z_{k-m}, \cdots, z_k] = H_i q_i$. The second recursion calculates $z_{k-m} = H_k^0 q_{k-m}$, thus obtains $b_i = \rho_i y_i^T z_i$ and $z_{i+1} = z_i + (a_i - b_i)s_i$. Hence, the value z_k is the approximation for the search direction. (Note: when performing minimization, the search direction is the negative of z.)

After obtaining the search direction at each step, a backtracking line search method is implemented to find and tune the learning rate (step size) such that it satisfies the sufficient decrease condition given by:

$$f(\lambda_k + \gamma_k d_k) \le f(\lambda_k) + \sigma \cdot \gamma_k^\eta \cdot g_k^T d_k \tag{12}$$

where γ_k is the step size, $\sigma \in (0, 1)$ is a control parameter and η is the scaling parameter that fits (12) iteratively until the condition is met. In our experiment, the initial step size is $\gamma_0 = 0.5$, $\sigma = 0.4$ and $\eta = \{1, 2, \cdots, 20\}$. This step determines the optimal η value, and then the γ_k^η becomes the new step size (learning rate) for the next iteration.

2.3.1 Path prediction with viterbi algorithm

After training the model, the aim is to find the most probable label sequence for a given sequence with observed words and corresponding POS tags. The Viterbi algorithm was employed to score all candidate tags with the trained model and then search for the best path with the maximal score.

Given an observed sequence $X = \{x_1, x_2, \cdots, x_T\}$ (T being the number of tokens in this sequence) with the trained feature (transition and emission) weights being obtained, the most likely state sequence $Y = \{y_1, y_2, \cdots, y_T\}$, where each $y_t \in L = \{l_1, l_2, \cdots, l_V\}$ (L being the label space obtained through training) can be calculated by the recurrence relations (forward step):

$$V_1 = O_{y_1,x_1} \tag{13}$$
$$V_t = \max_{Y \in L}(O_{y_t,x_t} + A_{y_{t-1},y_t}) \tag{14}$$

where V_t is the score of the most probable state sequence responsible for the first t observations. The Viterbi path can then be retrieved by saving back pointers that remember which state y was used in (14). Let $Ptr(y_t, t)$ be the function that returns the value of y_t used to compute V_t, then we have:

$$y_T = \max_{Y \in L}(V_T) \tag{15}$$
$$y_{t-1} = Ptr(y_t, t) \tag{16}$$

3 Numeric experiment

In order to demonstrate the performance of the CRFs in the POS tagging, the CRF model was applied to the Car review datasets.

3.1 Data description

We crawled the car review dataset on Toyota and Honda cars from Cars.com using Python Scrapy. A total of 1,126 reviews were collected. After the initial cleaning and duplicates removal, 1,094 reviews were kept. Inspired by [4], additional transformations using regular expressions (also known as rational expression or regex) were used on the training and testing dataset. As a result, a total number of 18,440 words are used.

We tokenized the review sentence into word-level (18,440 words), and then POS tagged each word manually with Penn Treebank POS Tags, and 45 POS tags are used (see Appendix Table 10). Notice that a Verb Past Participles (*VBN*) can be used as adjectives (*JJ*) to describe nouns.

3.2 Train the conditional random field part-of-speech tagger

The performance of the CRF model is measured using 10-fold Cross-validation using the transformed dataset. That means, for each validation, the transformed dataset was then divided into training with 998 reviews and testing with 96 reviews. For such a small dataset, 10% as test samples can provide an intuition about the model. After the pre-processing that included tokenizing the corpus, there are 549 transition features and 2,475 emission features, which means there were a total of 3,024 parameters to be estimated. We ran the algorithm for 100 iterations, and the negative Log-Likelihood converged quite well.

Figure 1 shows the distribution for trained weights, as most of the feature weights have values around 0. There are a few features having values that are towards the tails, meaning that certain words are likely/unlikely to emit certain POS tags, or certain transitions, e.g. [*Adjective (JJ)* → *Noun (NN)*] vs [*Adjective (JJ)* → *Verb (VB)*], are likely/unlikely to happen.

3.3 Performance evaluation

The performance is evaluated based on precision, recall, and F-score. Precision, also referred to as positive predictive value, talks about how precise/accurate the model is out of those $PredictedPositive$, how many of them are $ActualPositive$; Recall is defined as the true positive rate or sensitivity, calculates how many of the $ActualPositives$ the model captures through labeling it as $Positive$ (True Positive):

$$\begin{aligned} Precision &= \frac{TruePositive}{TruePositive+FalsePositive} \\ &= \frac{TruePositive}{TotalPredictedPositive} \end{aligned} \tag{17}$$

$$\begin{aligned} Recall &= \frac{TruePositive}{TruePositive+FalseNegative} \\ &= \frac{TruePositive}{TotalActualPositive} \end{aligned} \tag{18}$$

and F_1 score is the harmonic mean of the precision and recall, which helps seek a balance between precision and recall:

$$F_1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{19}$$

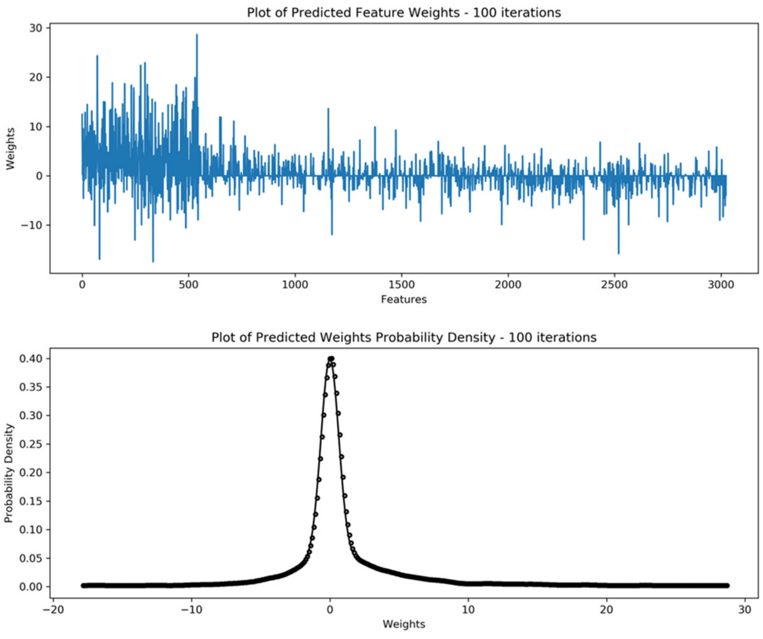

Fig. 1 Distribution of Predicted Feature Weights

We computed both the macro and micro values for precision and recall. A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally). In contrast, a micro-average will aggregate the contributions of all classes to compute the average metric. In a multi-class classification setup, micro-average is preferable if one suspects there is a class imbalance.

3.3.1 Validation

To validate our CRF model, we incorporated 10-fold cross-validation where the training set was randomly partitioned into 898 for training and the rest 100 for validation. Further, after each cycle, we would reshuffle the training set and go through the 10-fold CV process again. The process was repeated 20 times to ensure the generality of our proposed CRF model. Hence, we obtained 200 validation results and calculated the three metrics accordingly, with corresponding means and standard deviations listed in Table 1. In summary, the overall performance is good, as the lower bounds of the 95% confidence intervals rest above our threshold of 90% for both precision (0.9393) and recall (0.9195), indicating no further model tuning is required at the moment.

3.3.2 Testing

For the testing set, Fig. 2 displays the confusion matrix, where the overall accuracy is 0.9252 (however, overall accuracy is not a metric to use when evaluating a model). Table 2 shows the average precision, recall and F_1 metrics.

We also computed these metrics for each label (overall 31 labels in our experiment), displayed in Table 3. Our tagger managed to capture each POS feature fairly well, given such a small data set.

The error matrix displayed in Fig. 3 shows the details of mispredicted classes, and we see that most misclassified tokens were between *VBZ* and *NNS*.

Based on the above metrics, CRF performed well in sequential labeling for Toyota and Honda cars reviews. Taking the first sentence in our testing data as an example, comparing the true path and predicted path is shown in Table 4 where the only misclassification was on the word [inside].

3.3.3 Comparison

We compared the performance of the CRF tagger to the baseline tagger in Python NLTK 3.3, which is based on HMM. The side-by-side comparison is displayed in Table 5. The performances of the two competing taggers were very close, which is impressive as CRF was performed on a small training data set. However, as we observed from the tagging results, the performance of the baseline tagger has been inconsistent. The baseline tagger tends to classify any word with the first letter capitalized to NNP, e.g. [Gas] and [Nice]

Table 1 Validation Performance - Mean, Standard Deviation and 95% C.I. of Precision, Recall and F_1

	Precision	Recall	F_1
Mean	0.9423	0.9224	0.9202
Standard Deviation	0.0218	0.0212	0.0212
95% C.I.	[0.9393, 0.9453]	[0.9195, 0.9253]	[0.9173, 0.9231]

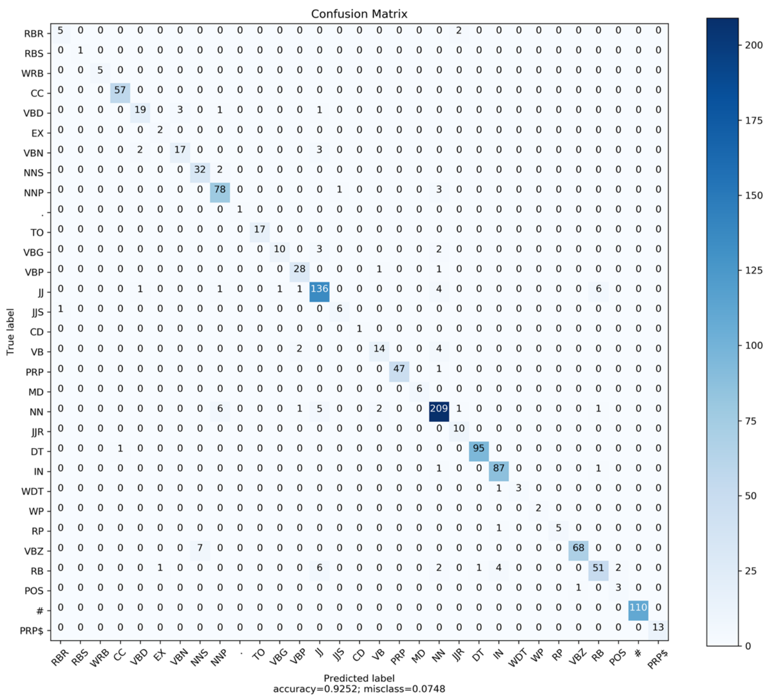

Fig. 2 Confusion Matrix

would be classified as NNP instead of the ground truth of NN and JJ. Hence, in our data, the CRF tagger performance is more robust.

4 Feature extraction

After successful training of the CRF tagger, we then extracted features based on the tagging result. As the first step, we extracted only Nouns and Adjectives from the review sentences as these words contain the most information one would need to generalize the ideas. Furthermore, Since these keywords contain useful information in the review mining, we can use these keywords as an input for an opinion mining system (e.g., using a new CRF model to classify the opinions in 5 levels).

An example shown in Table 6 gives the idea about how it works. When one is interested in finding out how people think about a specific feature (e.g., transmission), our framework takes in the keywords [transmission, transmissions] and output any summarized reviews that contain these keywords. From the generalized report on feature transmission as shown in Table 7 people will get abundant information on how transmission performs.

Table 2 Overall Performance - Precision, Recall and F_1

	Precision	Recall	F_1
Macro	0.9322	0.9290	0.9264
Micro	0.9352	0.9352	0.9352

Table 3 Performance on Individual Tags - Precision, Recall and F_1

POS Tag	Precision	Recall	F_1
RBR	0.8333	0.7143	0.7692
RBS	1.0000	1.0000	1.0000
WRB	1.0000	1.0000	1.0000
CC	0.9828	1.0000	0.9913
VBD	0.8636	0.7917	0.8261
EX	0.6667	1.0000	0.8000
VBN	0.8500	0.7727	0.8095
NNS	0.8205	0.9412	0.8767
NNP	0.8864	0.9512	0.9176
.	1.0000	1.0000	1.0000
TO	1.0000	1.0000	1.0000
VBG	0.9091	0.6667	0.7692
VBP	0.8750	0.9333	0.9032
JJ	0.8831	0.9067	0.8947
JJS	0.8571	0.8571	0.8571
CD	1.0000	1.0000	1.0000
VB	0.8235	0.7000	0.7568
PRP	1.0000	0.9792	0.9895
MD	1.0000	1.0000	1.0000
NN	0.9207	0.9289	0.9248
JJR	0.7692	1.0000	0.8696
DT	0.9896	0.9896	0.9896
IN	0.9355	0.9775	0.9560
WDT	1.0000	0.7500	0.8571
WP	1.0000	1.0000	1.0000
RP	1.0000	0.8333	0.9091
VBZ	0.9855	0.9067	0.9444
RB	0.8644	0.7612	0.8095
POS	0.6000	0.7500	0.6667
#	1.0000	0.8333	0.9091
PRP$	1.0000	1.0000	1.0000

5 Conclusion

We proposed and built a CRF based framework and integrated it with L-BFGS. The advantage of CRF is that it makes fewer assumptions than the generative models and hence allows a better level of flexibility on feature engineering. Compared with the existing method, which has been trained over a large training set, the CRF model has a very similar accuracy and shows a more robust result even though it is trained over a minimal training set. Hence, the CRF model can be used as part of an exploratory data analysis.

Furthermore, similar to deep learning, the CRF-based framework can be used to classify car reviews in the future study by defining more precise feature functions.

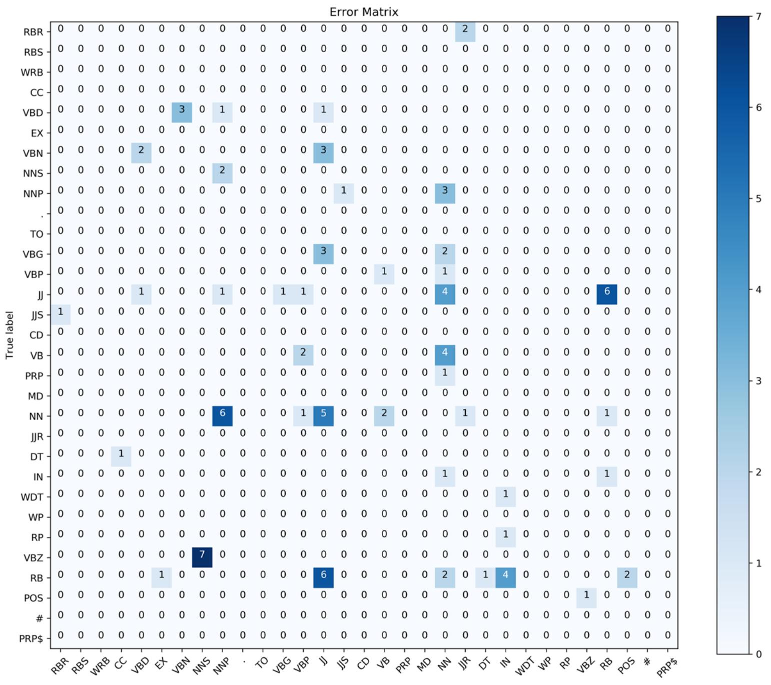

Fig. 3 Error Matrix

Table 4 Example: Tagging Output &. Comparison

Original sentence	The car is roomy inside, comfortable, handles and performs great and is fun to drive.
Processed sentence	the car is roomy inside comfortable handles and performs great and is fun to drive
True Path	*DT NN VBZ JJ IN JJ VBZ CC VBZ JJ CC VBZ JJ TO VB*
Predicted Path	*DT NN VBZ JJ RB JJ VBZ CC VBZ JJ CC VBZ JJ TO VB*

Table 5 Performance Comparison: CRF vs NLTK Baseline Tagger

	Precision	Recall	F_1
CRF	0.9322	0.9290	0.9264
NLTK Baseline Tagger	0.9248	0.9210	0.9201

Table 6 Example: Word Extraction from Review Sentence

Original sentence	The car is roomy inside, comfortable, handles and performs great and is fun to drive.
Processed sentence	the car is roomy inside comfortable handles and performs great and is fun to drive
Predicted Path	*DT NN VBZ JJ RB JJ VBZ CC VBZ JJ CC VBZ JJ TO VB*
Extracted words	car roomy comfortable great fun

Table 7 Summarized Report on Feature: Transmission

’transmission’, ’jerky’, ’gas’, ’mileage’, ’terrible’
’transmission’, ’not’
’lack’, ’power’, ’transmission’, ’problem’, ’car’, ’down’, ’shifts’
’transmission’, ’smooth’
’problem’, ’transmission’, ’computer’, ’chips’, ’difference’
’transmission’, ’cruise’, ’control’, ’joke’
’miles’, ’auto’, ’shop’, ’times’, ’last’, ’call’, ’dealer’, ’transmission’
’transmission’, ’not’, ’smoothest’
’hp’, ’speed’, ’auto’, ’transmission’, ’responsive’, ’smooth’
’new’, ’transmission’, ’not’, ’smooth’, ’accelerating’, ’stop’
’speed’, ’transmission’, ’shifts’, ’manual’
’transmission’, ’computer’, ’major’, ’issue’
’transmission’, ’jerky’, ’gas’, ’mileage’, ’terrible’
’major’, ’transmission’, ’issues’, ’twice’
’transmission’, ’driving’, ’crazy’
’transmission’, ’absolute’, ’worst’, ’dangerous’, ’cause’, ’accident’
’manual’, ’transmission’, ’lack’, ’power’, ’great’, ’fuel’, ’economy’
’transmission’, ’big’, ’issue’, ’rattles’, ’more’, ’miles’

6 Future research

The current CRF model can be further expanded in the future. For example, since we only extracted information that is carried by the Nouns and Adjectives at the current stage, some information that is carried by verbs or verb phrases such as “recommend,” “outperform,” “disappoint,” etc. are not inherited. Hence, we can improve the CRF model by introducing a set of self-defined entities and corresponding feature functions listed in Table 8.

For word that is not an entity, it will be represented as background word by (*B*). Furthermore, an entity can be a single word or a phrase. For phrase entity, a position feature is assigned to each word in the phrase, and there are three possible positions denoted at beginning of the phrase (*Entity-B*), middle of the phrase (*Entity-M*) and end of the phrase (*Entity-E*). As for opinion entity, polarity can be represented by positive (*P*) and negative (*N*), and use (*Exp*) and (*Imp*) to respectively indicate explicit opinion (opinion expressed explicitly) and implicit opinion (opinion needs to be induced from the review).

Table 9 shows the solution by using the hybrid tag. In the example, [car] is the component of a car, [inside], [handles], [performs] and [to drive] are features of a car. [Roomy] is a positive, explicit opinion expressed on the feature [inside], so it is tagged as the hybrid tag (*Opinion-B-P-Exp*). Therefore, after obtaining all the hybrid tags, we can identify the

Table 8 Different Types of Entities [7]

Components	Pysical objects of a product, e.g. engine, transmission, brake, seat ...
Functions	Capabilities provided by a product, e.g. horsepower, acceleration, adjustable seat ...
Features	Properties of components or functions, e.g. mileage, confort, size, color, design ...
Opinions	Thoughts expressed by users on components, functions or features

Table 9 Label with New Tags

Original sentence	The car is roomy inside, comfortable, handles and performs great and is fun to drive.
Processed sentence	the car is roomy inside comfortable handles and performs great and is fun to drive
POS tags	*DT NN VBZ JJ RB JJ VBZ CC VBZ JJ CC VBZ JJ TO VB*
Hybrid tags	*B Component-B B Opinion-B-P-Exp Feature-B Opinion-B-P-Exp Feature-B B Feature-M Opinion-B-P-Exp B B Opinion-B-P-Exp Feature-M Feature-E*

opinion orientation if a word is an opinion entity. Thus, second-order feature functions can be expanded on top of the first-order feature function defined in Section 2.

Appendix

Table 10 Penn Treebank Part-of-Speech Tags

POS Tag	Description
CC	conjunction, coordinating
CD	numeral, cardinal
DT	determiner
EX	existential there
FW	foreign word
IN	preposition or conjunction, subordinating
JJ	adjective or numeral, ordinal
JJR	adjective, comparative
JJS	adjective, superlative
LS	list item marker
MD	modal auxiliary
NN	noun, common, singular or mass
NNP	noun, proper, singular
NNPS	noun, proper, plural
NNS	noun, common, plural
PDT	pre-determiner
POS	genitive marker
PRP	pronoun, personal
PRP$	pronoun, possessive
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	particle
SYM	Symbol
TO	“to” as preposition or infinitive marker
UH	interjection
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle or gerund

Table 10 (continued)

POS Tag	Description
VBN	verb, past participle
VBP	verb, present tense, not 3rd person singular
VBZ	verb, present tense, 3rd person singular
WDT	WH-determiner
WP	WH-pronoun
WP$	WH-pronoun, possessive
WRB	Wh-adverb
$	dollar
“	Opening quotation marks
”	Closing quotation mark
(	Opening brackets
)	Closing brackets
,	Comma
–	dash
.	sentence terminator
:	Punctuation

Declarations

Conflict of Interests On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Azzalini A (2005) The skew-normal distribution and related multivariate families. Scand J Stat 32:159–188
2. Bird S, Loper E, Klein E (2009) Natural language processing with Python. OReilly Media Inc
3. Hu MQ, Liu B (2005) Mining and summarizing customer reviews. In: 10th ACM SIGKDD international conference on knowledge discovery and data mining, pp 168–177
4. Jin W, Ho HH, Srihari RK (2009) Opinion miner: a novel machine learning system for web opinion mining and extraction. In: Proceedings of international conference on machine learning, pp 465–472
5. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proc 18th International conf on machine learning. Morgan Kaufmann, pp 282–289
6. Nocedal J (1980) Updating Quasi-Newton matrices with limited storage. Math Comput 35:773–782
7. Peng FC, McCallum A (2006) Accurate information extraction from research papers using conditional random fields. In: Human language technology conference and North American chapter of the association for computational linguistics, pp 963–979

8. Qu L, Toprak C, Jakob N, Gurevych I (2008) Sentence level subjectivity and sentiment analysis experiments in NTCIR-7 MOAT challenge. In: Proceedings of the 7th NTCIR workshop meeting on evaluation of information access technologies: information retrieval, question answering, and cross- lingual information access. Tokyo, pp 210–217
9. Sutton C, McCallum A (2012) An introduction to conditional random fields. Found Trends Mach Learn 4:267–373
10. Turney PD (2002) Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: 15th ACM SIGKDD International conference on knowledge discovery and data mining, pp 417–424