GetInTheMood 1.0 : Conditional random field based lyrics parser

Ishan Khurjekar
Department of Electrical and Computer Engineering
Texas A&M University
College Station
ishank10@tamu.edu

Radhika Saraf
Department of Electrical and Computer Engineering
Texas A&M University
College Station
saraf.radhika@tamu.edu

Abstract

The field of natural language processing has received much attention due to its ubiquity in the recent years. We explore the foundations of a common task in natural language processing i.e part of speech tagging in sequential data. We use this technique to parse the lyrics of a song and classify the song as either happy or sad. An efficient strategy for this task is using conditional random fields. This topic is of particular interest to us in the context of the course given that conditional random fields are a special case of undirected Markov models. CRFs are capable of modeling the dependencies between states. The paper proposes a sorting and labelling algorithm that employs the machinery of conditional random fields.

1 Introduction

In recent times, the need to analyze complex data has become imminent. Of critical importance is the analysis of sequential data across multiple domains like natural language processing, time-series data in financial models, and gene sequences in bioinformatics. Typically, there have been two approaches : probabilistic modeling methods in the form of hidden Markov models and statistical /data oriented parsing of input sequence. HMMs and statistical parsers are generative models i.e given the data points they estimate the transition probabilities between different states. Although such approaches are robust, a glaring limitation is their inability to model complex dependencies between input nodes. The generative models work by overlaying a joint distribution over input-label pairings.In most cases, listing down all possible input sequences and calculating the joint input feature-label distribution becomes a hard problem. A solution to this problem is conditional random fields (modeling the conditional probability distribution directly). CRF's differ from HMMs in that they model the distribution of label sequences given the input sequences. Due to the conditioning, the strict assumption of independence between states can be relaxed in conditional random fields. We look at a typical natural language processing problem with an emphasis on using conditional random fields. Subsequent sections outline the importance of the problem being studied, description of model and techniques being used. Finally we outline the results and future work.

1.1 Problem Statement

We are investigating the quintessential problem in natural language processing : tagging language data at a word / document level. Common approaches to this problem include statistical distribution of input data. This method includes having knowledge of the class conditional densities which in turn depends on the comprehensiveness of the training dataset and statistical techniques used. Hidden Markov model is also a common solution but it assumes that each state is independent of its ancestors given the preceding state and that every observation is dependent only on the corresponding state. The sequential nature of the data merits the use of conditional random fields in this case. We extend the problem to document classification with input sequence being a sequence of labels assigned to individual words making up the document. The final output is a binary label assignment to the document.

1.2 Why is this important ?

With the rise of big-data era, it has become important to devise inference / learning models that can analyze and / or classify the data. Dealing with sequential data is an important component of this process due to the ubiquitous nature of such data[1]. Conditional random fields have shown good results when the input data is sequential. The structure of CRF's represents a black box. This makes it a preferred solution when we are concerned only with the output given the input.

1.3 Salient Features

The crux of this report lies in the concept of conditional random fields applied to language processing [2]. Due to the nature of data, some level of pre processing is required. This is followed by manually creating a labeled dataset for training the CRF. The CRF does not concern itself with separately modeling the highly complex relationships between input features [3]. Instead it learns the conditional probability of a state given the observation. A distinguishing factor of this exercise is the relatively simple input feature being used.

1.4 Implication of results

Results show that working with relatively simple input features creates a good baseline for the performance of conditional random fields for language processing tasks.The conditional probability model lends flexibility to us by allowing arbitrary input features. This flexibility helps in capturing the complex relationships between individual entities in language inputs.

2 Modeling the system

We are interested in designing an application that will classify songs according to different moods such as happy or sad. This classification is done solely on the basis of the lyrics of the song. Natural language processing techniques are employed to tokenize (discretize the text into individual words) and assign labels to the words [4]. We create a knowledge based dictionary for song labels. The training data is labelled on the basis of this dictionary and fed to a conditional random field tagger. The CRF learns the conditional distribution parameter from this training data and then we use the tagger to accordingly label the test data. The output is a word level tagging for all words in a song. We have further defined a statistical metric to classify the entire song as either sad or happy based on the CRF tagged output.

The system model is as shown in Figure (1).

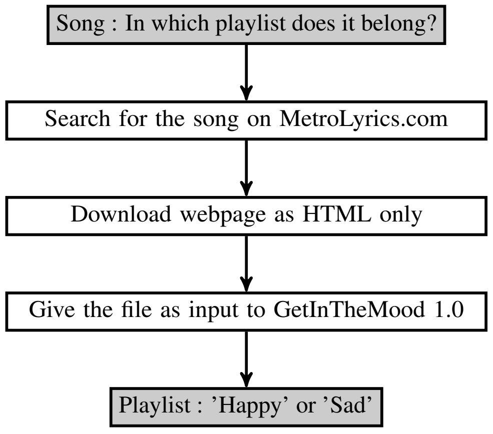
Song : In which playlist does it belong?
Search for the song on MetroLyrics.com
Download webpage as HTML only
Give the file as input to GetInTheMood 1.0
Playlist : 'Happy' or 'Sad'

Figure 1: Flowchart for GetInTheMood

3 Connections to coursework

We studied the problem of document classification in the context of sequential data. The motivation to study this problem came from a class homework wherein we had to build a naive Bayes classifier for text classification. The independence assumption for individual features in naive Bayes classifier led us to explore further graphical modeling techniques that could overcome this limitation. To this end, we made use of the concept of conditional random fields. This topic follows directly from the limitations of hidden Markov models. CRF is an undirected probabilistic graphical model. The advantage of CRF's is their ability to take textual context into account when dealing with sequential data. CRF's enable us to create an efficient and easy to interpret solution to the document classification problem. Generalizing to all language processing tasks, CRF generates part of speech tags (labels) for each entity in the input sequence. The CRF learns the conditional probability parameters which results in suitable representation of input feature dependencies.

4 Techniques discussed/analyzed

As shown in the Figure 2 the inputs are .htm files downloaded from the website `metrolyrics.com` [7]. The initial processing of the input files involves :

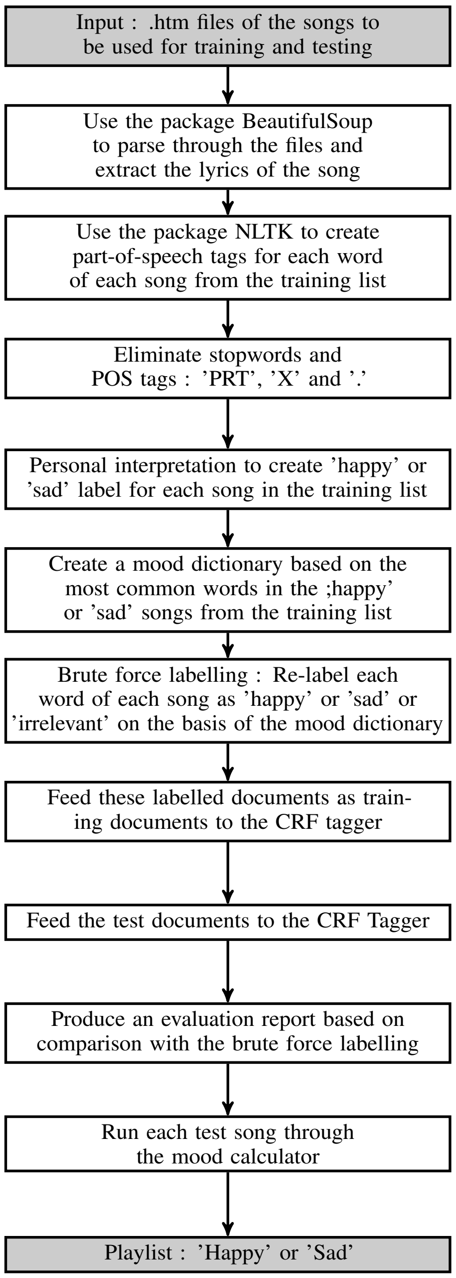
Input : .htm files of the songs to be used for training and testing
Use the package BeautifulSoup to parse through the files and extract the lyrics of the song
Use the package NLTK to create part-of-speech tags for each word of each song from the training list
Eliminate stopwords and POS tags : 'PRT', 'X' and '.'
Personal interpretation to create 'happy' or 'sad' label for each song in the training list
Create a mood dictionary based on the most common words in the ;happy' or 'sad' songs from the training list
Brute force labelling : Re-label each word of each song as 'happy' or 'sad' or 'irrelevant' on the basis of the mood dictionary
Feed these labelled documents as training documents to the CRF tagger
Feed the test documents to the CRF Tagger
Produce an evaluation report based on comparison with the brute force labelling
Run each test song through the mood calculator
Playlist : 'Happy' or 'Sad'

Figure 2: Flowchart for GetInTheMood

1. Using the HTML parser package BeautifulSoup [8].
2. Check the layout of the webpage that displayed the lyrics of the song and isolate the HTML tag and class that contains the text of the lyrics.
3. Split the text into individual words i.e. tokenize the text using the NLTK package [9].
4. Store the words of one song into one document.
5. Create a list of documents containing several songs.

Once we have tokenized the lyrics of the song, we can continue with the natural language processing task of assigning part-of-speech tags to each word. Part-of-speech (POS) tags are labels that are given to the word according to the rules of grammar of the English language such as 'NOUN', 'VERB' or 'ADJECTIVE'. Part of speech tagging requires the knowledge of the sequence of the words in a sentence. For example, in commonly used grammar, a sentence cannot end with a preposition ('PRT'). The NTLK package is used to perform the POS tagging in the program. We explain how the POS tagger works exactly like a CRF.

Let us assume that X is a set of sentences that needs to be labelled, and Y is a vector that contains the corresponding label sequences. Now, $Y_1, \ldots, Y_n$ is limited to a finite number of labels : 'NOUN','ADJ','ADV','CONJ' and so on, whereas X ranges of the many possible natural language combinations of words forming sentences. The CRF framework needs to know the conditional model $P(Y|X)$ and does not require the knowledge of $p(X)$. The marginal density of X can be hard to compute or model due to vast possibilities that exist in natural languages. We use a fixed graph structure G such that Y is indexed by the vertices of the graph. We can say that (X, Y) forms a conditional random field, which when conditioned on X then Y follows the Markov property.

The conditional probability $p_\theta(Y|X)$ depends on the features f which are assumed to given and fixed. The features are such that f_i is true if X_i is upper case and Y_i is 'proper noun'. The feature vectors can be constructed based on the intricacies of the language. There is one feature corresponding to each state-observation pair (x, y) and one feature for each state pair (y, y'). The mappings in the feature vector will depend on various conditions such as is the first character lower case, is it a digit, is it a punctuation mark and where does in occur in the sentence. The sequence is very important in such an application - a period followed by an upper case letter most probably implies that a new sentence has begun, or if an upper case letter occurs without any punctuation in its neighbourhood it might be a proper noun. The entire set of mappings is stored as M of the size of $|\gamma| \times |\gamma|$, where γ is the size of the possible set of POS tags Y.

The next step is to perform parameter estimation for the CRF that maximizes the log likelihood $p_\theta(Y|X)$ of the training data. The parameters θ can be learned based on the given data $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$ with the empirical distribution $\tilde{p}(x, y)$

by:

$$\max_{\theta} \tilde{p}(x, y) \log p_{\theta}(Y|X) \tag{1}$$

A forward-backward algorithm can be used where α is the forward vector and β is the backward vector and they can be computed recursively as :

$$\alpha_i(X) = \alpha_{i-1}(x)M_i(x) \tag{2}$$

$$\beta_i^T(x) = M_{i+1}(x)\beta_{i+1}(x) \tag{3}$$

The partition function $Z_{\theta}(x) = [M_1(x), \ldots, M_{n+1}(X)]$, with the help of which we can calculate the marginal probability that a label $Y_i = y$ given the observation sequence x:

$$P(Y_i = y|x) = \frac{\alpha_i(y|x)\beta_i(y|x)}{Z_{\theta}(x)} \tag{4}$$

Once the CRF trained, it can be used to predict

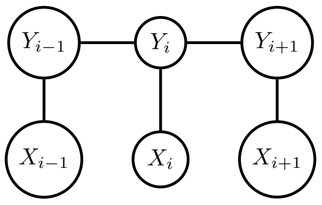

Figure 3: Conditional Random Field

the labels of a test sentence. The accuracy of the CRF depends on the algorithm used to perform parameter estimation, the regularization used and the quality of the training data among other factors. The NLTK POS tagger is a well established tool and performs this task satisfctorily.

The next step in our GetInTheMood 1.0 application is to eliminate the stopwords and certain POS tags. This is done because certain elements of the sentence are not relevant to the sentiment attached to the song.

We curated particular songs to form the data set and classified them as happy or sad. By studying the most commonly occurring words in these songs, we created a mood dictionary comprising of two separate dictionaries : one of 'happy' words and the other of 'sad' words. To overcome the problem of labelling bias, we made sure that the length of two dictionaries is equal. We relabelled the words in the songs as 'happy' or 'sad' or 'irrelevant (I)' based on whether or not they appear in these dictionaries. This relabelling is termed as brute force labelling in the program.

Then, these documents labelled 'happy','sad' and 'I', we fed as training data to another CRF tagger. Using the same principles are discussed above the CRF was trained and used to predict labels for a test document. Note here that the mood dictionary served as the mapping M defined in the POS tagger problem. The rest of the principles follow.

In addition to this, we developed our own metric 'mood ratio' to classify the songs. The test document was fed to the CRF tagger and the words were labelled as 'happy','sad' or 'I'. Depending on the count of 'happy' or 'sad' labels, the mood calculator gives us a mood ratio for the song. The mood ratio goes on to tell us if that song should be put in the 'happy' or 'sad' playlist.

5 Results

The CRF tagger was evaluated using 'sklearn metrics'. The training documents were labelled and classified again to confirm that the program was running successfully. It gave us sufficiently accurate results when the CRF output was compared to the brute-force-labelled song.

The tagger also predicted a mood for a completely new song satisfactorily. The precision was slightly greater for the 'happy' and 'I' labelling than the 'sad'. This could be a consequence of certain overlap in the two dictionaries. The brute-force-labelling checks for a word in the 'happy' dictionary before checking for it in the 'sad' dictionary, and this could be the reason behind this discrepancy.

GetInTheMood 1.0 was used to perform the task of natural language processing of songs based on their lyrics and could be used further, to perform sentiment analysis.

6 Future work

In order to avoid overlap in the two dictionaries, the conditional distributions for the training data could be better defined than the current brute-force method. We could employ better parameter estimation techniques instead of using the most commonly occurring words, possibly by appending synonyms to the mood dictionary.

The future for this application could be to incorporate this feature into a smartphone where it could create a mood dictionary based on the user's existing playlists. This could be used to classify new songs added by the user.

Acknowledgments

The authors would like to thank Prof Krishna Narayanan for his continued guidance and support throughout the project. He lent a patient ear to us and provided us with invaluable insights throughout the course duration.

References

[1] Dietterich (2012) Machine Learning for Sequential Data : A Review. *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 15-30

[2] Lafferty. J, McCallum. A & Pereira F (2001) Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. *ICML* $'01$ *Proc. of the* 18^{th} *Intl. Conf on Machine. Learning*, pp. 282 =–289.

[3] Charles S, & McCallum A , (2012) An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning*

[4] Bird, Steven (2006) NLTK: The Natural Language Toolkit

[5] Manning, C. D, & Schutze H, (1999) *Foundations of statistical natural language processing*. MIT Press, Cambridge.

[6] Yeung, Albert A, (2017) Performing Sequence Labelling using CRF in Python. *Notes on machine learning and A.I.*, http://www.albertauyeung.com/post/python-sequence-labelling-with-crf/

[7] Webpages metrolyrics.com

[8] BeautifulSoup Documentation 4.4.0 https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[9] NLTK package http://www.nltk.org/howto/