FST for Spoken Language Understanding

Festo Owiny
festo.owiny@studenti.unitn.it
LUS Midterm Project

I. Introduction

The main purpose of the project is to compare performances between generative models for concept tagging w.r.t. the n-gram features. In this work we considered 9 n-gram features; unigram, bigram, trigram, 4-gram ,..., 9-gram respectively. We also subject the model to five different smoothing algorithms and compared the results. Each model was trained and tested with respect to the specified training features. In other words, for every smoothing algorithm, I vary the model features to test five different models (specified by the n-grams) and subject each model to 5 smoothing algorithms; witten_bell, absolute, katz, kneser_ney, presmoothed and unsmoothed.

The goal of this task requires locating and classifying chunks of words that identify entity names of the movie domain.

II. Dataset

The data is NL-SPARQL Data Set containing preprocessed and annotated training and test set from the movie domain, presented in the IOB data format. There are 22 entity names identifiable in our dataset, a total of $7,117$ tokens with $1,091$ phrases. The data is distributed with training and test sets having $3,338$ and $1,084$ sentences respectively; Approximately a ratio of $3:1$.

The data represents a bunch of queries in a movie domain. Queries search answers for common questions like name of film, year of production, name of actor, etc; Every word is associated with its IOB-tag.

III. Fundamentals

A. FSA, FST and WFST

Finite State Acceptor (FST) is a finite-state machine (FSM) with one tape which accepts/recognizes strings. Once all input has been received, if the current state is an accepting state, the input is accepted; otherwise it is rejected. A common application of FSA is word checking for regular expression patterns.

A finite-state transducer (FST) is FSM with two memory tapes which translates input to output string. It consists of a finite number of states which are linked by transitions labeled with an input/output pair. The transition table defines FST movements from the start state through different intermediate states to the final state. Formally, a finite transducer is a 7-tuple $(Q, \Sigma, \Gamma, \delta, \sigma, I, F)$ such that:
Q is a finite set of states, Σ is a finite set (input alphabet), Γ is finite set (output alphabet), δ is the transition function, σ is the output function, $I \in Q$ the set of initial states, and $F \subseteq Q$ is the set of final states.

While FSA are mostly used for recognizing, FST can also be used for parsing or translating. Usually FST are not deterministic, but can be. Those type of FST are called sequential FST.

FST can be weighted, where each transition is labeled with a weight in addition to the input and output labels. Weights represents costs for the transitions and can be used to determine special paths in case of non-deterministic FST.

FSA is closed under $Concatenation$, $Union$, $Intersection$, $Kleene^*$, $Difference$ and $Reversal$ but NOT $Complement$. However, FST and WFST is closed under all the above.

B. Generative Language Model

Generative approaches attempt to model the probability distribution of the data, $P(X, Y)$

Assume a finite set of words V, and a finite set of tags K. Define S to be the set of all sequence/tag-sequence pairs $\langle x_1...x_n, y_1...y_n \rangle$ such that $n \geq 0, x_i \in V$ for $i = 1...n$, and $y_i \in K$ for $i = 1...n$. A generative tagging model is then a function P such that [3]:

- For any $\langle x_1...x_n, y_1...y_n \rangle \in S$,

$$P(x_1...x_n, y_1...y_n) \geq 0 \tag{1}$$

- In addition,

$$\sum_{\langle x_1...x_n, y_1...y_n \rangle \in S} P(x_1...x_n, y_1...y_n) = 1 \tag{2}$$

Hence $P(x_1...x_n, y_1...y_n)$ is a probability distribution over pairs of sequences (i.e., a probability distribution over the set S). Given a generative tagging model, the function from sentences $x_1...x_n$ to tag sequences $y_1...y_n$ is defined as [3].

$$f(x_1...x_n) = \arg\max_{y_1...y_n} P(x_1...x_n, y_1...y_n) \tag{3}$$

$$f(x_1...x_n) = \arg\max_{y_1...y_n} p(x_1...x_n/y_1...y_n) p(y_1...y_n) \tag{4}$$

Using Bayes theorem and markov model iid assumptions, generative language models can be classified under unigram, bigram, trigram,...,n-gram. For example the bigram hidden markov model can be represented as [3];

$$f(x_1...x_n) = \arg\max_{y_1...y_n} \prod_{i=1}^{n} p(x_i/y_i) p(y_i/y_{i-1}) \tag{5}$$

1) Estimating the Parameters of a Trigram model: Define $c(u, v, s)$ to be the number of times the sequence of three states (u, v, s) is seen in training data. Similarly, define $c(u, v)$ to be the number of times the tag bigram (u, v) is seen. Define $c(s)$ to be the number of times that the state s is seen in the corpus. Finally, define $c(s \to x)$ to be the number of times state s is seen paired with observation x in the corpus. Given these definitions, the maximum-likelihood estimates are;

$$q(s|u, v) = \frac{c(u, v, s)}{c(u, v)} \tag{6}$$

$$e(x|s) = \frac{c(s \to x)}{c(s)} \tag{7}$$

2) Smoothing algorithms: A language model usually has very large number of parameters. The maximum-likelihood parameter estimates above will run into serious issues with sparse data. Even with a large set of training sentences, many of the counts will be low, or will be equal to zero. Smoothed estimation methods alleviate many of the problems found with sparse data. Some of the smoothing methods are; Linear Interpolation, Discounting Methods, witten_bell, absolute, katz, kneser_ney, presmoothed, etc.

IV. Implementation

Using equation 7, we obtain counts and calculate consequent probabilities.

We calculate the parameters using training data to build the concept tagger; input data is converted to fst. The test data usually contains new words which weren't observed in the training data therefore a separate fst is built for processing the unknown words $< unk >$. The final transducer has the possibility to jump from known to unknown words and vice versa in any time with no costs through epsilon-values. WFST weight is considered as the probability of the word occurring given the label for that word.

We used weighted Finite state transducers for processing a language model using OpenFST library and running the necessary commands to obtain the results. The probabilities are modeled as the weights, the words as inputs and the tags as output labels.

The second term, equation 6, strongly depends on which n-gram model we are going to build. In any case we have to convert the training data, which currently is in the 'token-per-line' IOB format, into concept 'sentence-per-line' format.

The lexicon for the training WFST is automatically extracted from the training dataset containing all phrases, words and concepts ordered and numbered. In addition to them other two special word symbols are used; $< esp >$ that will represent the empty transaction for the transducer and $< unk >$ that will be our wild card token for the unknown words.

The resulting transducers obtained are compiled using command, "fstcompile"; then both transducers are unioned.

The trained model is tested with the test set that. The resulting label for each word and the correct label for the word are put together in a file separated by the delimiter in the format "true label", "predicted label". This file is then evaluated using the script file conlleval.pl. The script evaluates the result based on the accuracy measures; Global accuracy, precision, recall and F-measure.

V. Results

The table shows accuracy values interms of Global accuracy, precission, Recall and F-measure. Global accuracy refers to the number of positively classified tags in relation to the total tags. The global accuracy is high across all the smoothing terms since it also includes the *O-tag* which is the most common tag across any SLU task. This *O-tag* is excluded while calculating precission, recall and f-measure. The table shows that accuracy values gradually increase with respect to n-grams across all the smoothing algorithms. However, this increment starts to reduce with higher n-gram values, such that a higher n-gram terms doesn't seem to influence further accuracy growth despite the fact that long range correlations drop exponentially with distance for any Markov model. It is also observable that among the smoothing techniques, the absolute method performs best whereas Katz performed worst. However, all the techniques employed performed averagely same range with little variances. The average values attained are lower than those obtained by state-of-the-art learning algorithms, e.g. CRF [1]. The unigram method has the lowest performance across all the smoothing techniques since It doesn't consider dependencies between terms as the other n-grams. On the other hand the best n-gram feature realized for our case is trigram.

VI. Conclusion

From the the observed result we can conclude that n-gram features are very essential in NLP tasks as It covers the linguistic dependency between words in sentences though performing low in long range dependencies. It is clear that if more features are added to this task such as POS tags, accuracy will be boosted. Some other features that can be used to improve the accuracy are; prefixes/suffixes, token frequency, word features [1]. The generative approach employed yields good result but It is commendable to implement the same with discriminative models and compare the results.

References

[1] Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. *Conditional Random Fields: Discriminative Training over Statistical features for Named Entity Recognition.* , 2013.

[2] Yashar Mehdad, Vitalie Scurtu, Evgeny Stepanov. *Italian Named Entity Recognizer Participation in NER task Evalita 09.* , 2016.

[3] Michael Collins, Columbia University. *Language Modeling.* , 2015.

witten_bell	Global acc	precision	recall	$F_{\beta=1}$
Unigram	88.84%	55.51%	60.04%	57.68
Bigram	92.68%	78.51%	74.34%	76.37
Trigram	92.62%	76.58%	74.61%	75.58
4-gram	92.86%	77.11%	75.34%	76.22
5-gram	92.90%	77.03%	75.62%	76.32
6-gram	92.01%	74.35%	73.05%	73.69
7-gram	91.51%	70.59%	71.95%	71.27
8-gram	91.79%	72.04%	72.50%	72.27
9-gram	91.67%	72.05%	72.32%	72.19
Average	91.88%	72.64%	72.20%	72.40

Absolute	Global acc	precision	recall	$F_{\beta=1}$
Unigram	88.84%	55.51%	60.04%	57.68
Bigram	92.69%	78.51%	74.34%	76.37
Trigram	92.65%	76.67%	74.70%	75.67
4-gram	92.85%	77.16%	75.25%	76.19
5-gram	92.88%	76.97%	75.34%	76.15
6-gram	92.89%	77.04%	75.34%	76.18
7-gram	92.26%	74.24%	74.24%	74.24
8-gram	92.16%	73.94%	73.33%	73.63
9-gram	92.24%	74.12%	73.51%	73.82
Average	92.16%	73.80%	72.90%	73.33

Katz	Global acc	precision	recall	$F_{\beta=1}$
Unigram	88.84%	55.51%	60.04%	57.68
Bigram	92.62%	78.03%	73.88%	75.89
Trigram	92.09%	75.60%	72.69%	74.11
4-gram	91.88%	72.57%	73.97%	73.26
5-gram	88.16%	56.95%	71.31%	63.33
6-gram	88.06%	56.95%	72.14%	63.65
7-gram	87.75%	56.39%	71.22%	62.94
8-gram	87.80%	56.33%	70.94%	62.80
9-gram	87.92%	56.98%	71.49%	63.41
Average	89.46%	62.81%	70.85%	66.34

kneser_ney	Global acc	precision	recall	$F_{\beta=1}$
Unigram	88.84%	55.51%	60.04%	57.68
Bigram	92.68%	78.41%	74.24%	76.27
Trigram	92.64%	76.67%	74.70%	75.67
4-gram	92.78%	76.87%	75.53%	76.19
5-gram	92.90%	76.79%	75.53%	76.16
6-gram	92.81%	76.73%	75.25%	75.98
7-gram	92.64%	75.63%	74.79%	75.21
8-gram	92.51%	75.40%	74.15%	74.77
9-gram	92.67%	75.65%	74.34%	74.99
Average	92.27%	74.18%	73.17%	73.66

Presmoothed	Global acc	precision	recall	$F_{\beta=1}$
Unigram	88.84%	55.51%	60.04%	57.68
Bigram	92.65%	78.41%	74.24%	76.27
Trigram	90.74%	64.81%	71.40%	67.95
4-gram	89.97%	62.28%	72.50%	67.01
5-gram	90.00%	62.36%	72.14%	66.89
6-gram	90.12%	62.91%	72.59%	67.40
7-gram	90.04%	62.53%	72.04%	66.95
8-gram	90.07%	62.77%	71.86%	67.01
9-gram	90.30%	64.39%	71.59%	67.80
Average	90.30%	64.00%	70.93%	67.22

Unsmoothed	Global acc	precision	recall	$F_{\beta=1}$
Unigram	88.84%	55.51%	60.04%	57.68
Bigram	92.60%	78.39%	74.15%	76.21
Trigram	92.55%	76.55%	74.52%	75.52
4-gram	92.67%	77.15%	74.89%	76.00
5-gram	92.83%	77.35%	74.79%	76.05
6-gram	92.69%	76.72%	74.61%	75.65
7-gram	92.61%	76.24%	74.43%	75.32
8-gram	92.44%	75.90%	73.60%	74.73
9-gram	92.67%	76.68%	74.15%	75.40
Average	92.21%	74.50%	72.80%	73.62

Table I
ACCURACY RESULTS ON THE MOVIE TEST SET.