Statistical structured prediction
Question set (Part 1-A)

Victoria Beltrán Domínguez

December 2021

Contents

1 Theoretical questions

1.1 Question 1

Briefly explain why languages generated by a probabilistic grammar are not necessarily probabilistic, and why not all probabilistic languages can be generated by a probabilistic grammar.

Probabilistics grammars can generate languages that are not consistent, and therefore, not probabilistic. Take for example the following probabilistic grammar taken from page 50 of *Formal Grammars in Linguistics and Psycholinguistics(Willem J.M. Levelt)*:

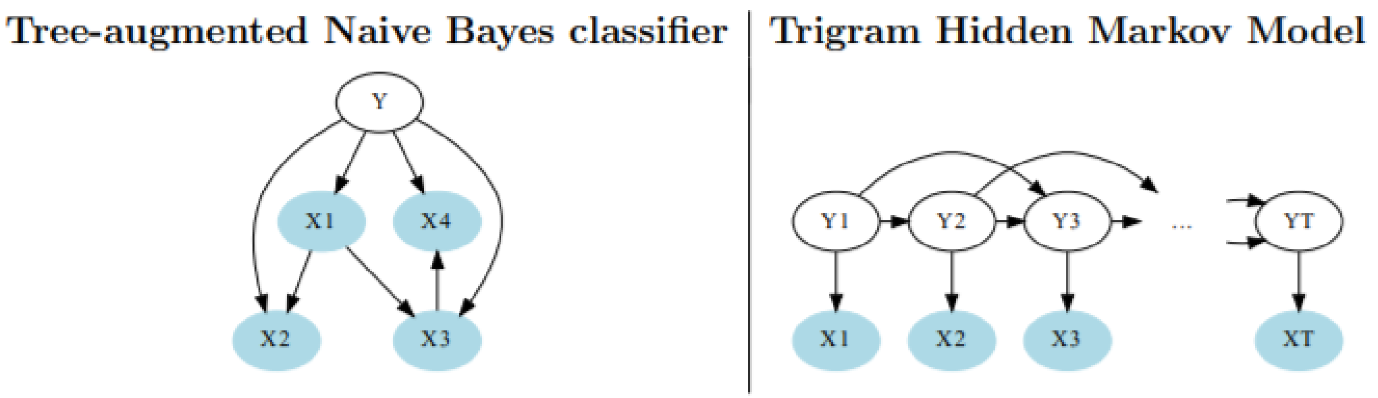
1. $S \rightarrow SS \qquad P(SS|S) = (2/3)$
2. $S \rightarrow a \qquad P(a|S) = (1/3)$

Probabilities for strings "a" and "aa" to be generated by this grammar are 1/3 and 2/27 respectively. It appears that if we somehow summed all the probabilities of the words generated by this grammar in the infinite, $\sum_{n=1}^{\infty} p(a^n) = 0.5$ instead of 1. Therefore, a probabilistic grammar has generated an inconsistent or non probabilistic language.

Moreover, not all probabilistic languages can be generated by a probabilistic grammar.

Let $L = \{a^n b^n | n \geq 0\}$ be a probabilistic language with computable probabilistic function $\phi(a^n b^n) = \frac{1}{en!}$. The proof that this language is probabilistic can be found in the slides of this subject. Nevertheless, there does not exist any probabilistic grammar such that we can accomplish that the probabilities for every word in the language equals the ones obtained with the computable probabilistic function.

1.2 Question 2



Tree-augmented Naive Bayes classifier

Trigram Hidden Markov Model

Given the Bayesian Networks that define the tree-augmented Naive Bayes classifier and the trigram Hidden Markov models, provide the resulting factorization for the joint probabilities $p(x_1^T, y)$ and $p(x_1^T, y_1^T)$.

- **Tree-augmented Naive Bayes classifier**

  $p(x_1^T, y) = p(y)p(x_1|y)p(x_4|y, x_3)p(x_2|y, x_1)p(x_3|x_1, y)$

- **Trigram Hidden Markov models**: Assuming that the access to non existent states will resolve into #, we can use the following factorization:

  $p(x_1^T, y_1^T) = \prod_{t=1}^T q(y_t|y_{t-1}, y_{t-2})e(x_t|y_t)$

1.3 Question 3

Given the Backward algorithm, shown on page 16 from Part-I.3, for bigram transitions, provide a new version of the Backward algorithm for a skip HMM.

- **Definition**: Score for $x_{t+1}...x_T$ starting at $y_t = s \in \mathcal{Y}$:

  $\beta_t(s) = \sum_{y_{t+1}^T; y_t = s} \prod_{i=t+1}^T \Psi_i(y_{i-2}, y_i, x_i)$

- **Inicialization**: $\forall s \in \mathcal{Y}$

  $\beta_T(s) = 1$

  $\beta_{T-1}(s) = 1$

- **Recursion**: $\forall t = T-2...1$; and $\forall s \in \mathcal{Y}$

  $\beta_t(s) = \sum_{s' \in \mathcal{Y}} \Psi_{t+2}(y_t = s, y_{t+2} = s', x_{t+2})\beta_{t+1}(s')$

- **Final Result**: The optimal score for x is:

  $\sum_{s,g} \Psi_1(y_{-1} = \#, y_1 = s, x_1)\beta_1(s)\Psi_2(y_0 = \#, y_2 = g, x_2)\beta_2(g)$

1.4 Question 4

Give a new version of the CKY-based Inside algorithm to calculate the probability of an input sequence, $x_1, ..., x_T$, for (right-branching) grammars. In addition, you must determine the temporary cost of the algorithm proposed.

- **Definition**: Given $x = x_1...x_T \in \sum^*$ and $A \in N$

  $e(A, i, i + l) = P_\theta(A \Rightarrow^* x_{i+1}...x_{i+l})$
- **Inicialization**:$\forall A \in N; \forall i : 0...T - 1;$

  $e(A, i, i + 1) = p(A \to b)\delta(b, x_{i+1})$
- **Recursion**:$\forall A \in N; \forall l : 2...T; \forall i : 0...T - l;$

  $e(A, i, i + l) = \sum_{B,C\in N}\{p(A \to BC)e(B, i, i + 1)e(C, i + 1, i + l)\}$
- **Final Result**: The sentence probability is:

  $P_\theta(x) = e(S, 0, T)$
- **Temporal cost**: The cost of the full algorithm is O(T^2)

1.5 Question 5

Taking as reference the Forward algorithm, shown on page 15 from Part-I.3, give a new version of this Forward algorithm for a trigram CRF.

- **Definition**: Score for $x_1...x_T$ ending at $y_t = s \in \mathcal{Y}$:

  $\alpha_t(s) = \sum_{y_1^t;y_t=s} \prod_{i=1}^t \Psi_i(y_{i-2}, y_{i-1}, y_i, x_i)$
- **Inicialization**: $\forall s \in \mathcal{Y}$

  $\alpha_1(s) = \Psi_1(y_{-1} = \#, y_0 = \#, y_1 = s, x_1)$

  $\alpha_2(s) = \sum_{s'\in\mathcal{Y}} \alpha_1(s')\Psi_2(y_0 = \#, y_1 = s', y_2 = s, x_2)$
- **Recursion**: $\forall t = 3...T$; and $\forall s \in \mathcal{Y}$

  $\alpha_t(s) = \sum_{s'\in\mathcal{Y}} \sum_{w'\in\mathcal{Y}} \alpha_{t-2}(w')\Psi_t(y_{t-2} = w', y_{t-1} = s', y_t = s, x_t)\alpha_{t-1}(s')$
- **Final Result**: The optimal score for x is:

  $\sum_s \alpha_T(s)$

2 Practical assignments

The aim is to evaluate the performance of different models g{1, 2, 3}{eq, is, sc} from a statistical and structural point of view:

2.1 Statistical evaluation

We are going to perform a statistical evaluation over the different provided models. For that, we are going to use *Test Set Perplexity* as metric:

$$logP_M(D) = 2^{(-\frac{1}{N}\sum_x log_2 P_M(x))}$$

In order to accomplish that target, we have developed:

1. A shell script that computes the probabilities over all the models with its respective test sets using the toolkit *SCFG*, and saves them to a .txt file.
2. A python script that takes as input the computed probabilities (.txt file) and computes the Test Set Perplexity using the previously defined formula.

Results obtained for *Test Set Perplexity* are the following:

Model	**Test Set**	**Test Set Perplexity**
G1-EQ	TS-EQ	99095.2
G1-IS	TS-IS	26581.8
G1-SC	TS-SC	33618.3
G2-EQ	TS-EQ	635136.4
G2-IS	TS-IS	52217.6
G2-SC	TS-SC	43543.3
G3-EQ	TS-EQ	986.2
G3-IS	TS-IS	1254.9
G3-SC	TS-SC	1218

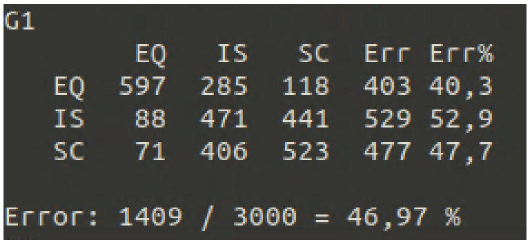
We do not know how models G1, G2 and G3 have been generated. However, we can see that somehow, G2 is the most perplex model and G3 is the least perplex model. In other words, G2 has less information and therefore, we expect it to have a greater error rate. Meanwhile, G3 is the most informed model, and it seems that for this particular problem can obtain good results (or at least, we expect it to get better results than the other ones).

2.2 Classification

This part aims to study the classification results depending on the model used. To do this, it is suggested to calculate the confusion matrix between classes and analyze the results.

In order to accomplish that target, we have developed a shell script that joins all test sets in an orderly manner, and for each model compute over all samples the highest probability for each sample of being classified between EQ, IS, SC.

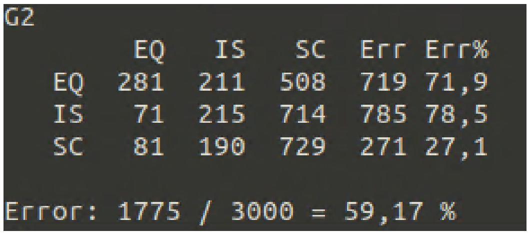
Then, the script also writes down in a .txt file the real label and the predicted label for each sample. Finally, we pass that file (one for each model G1, G2 and G3) to the provided confuss script in order to obtain the following confusion matrixes:



```
G1
       EQ   IS   SC   Err Err%
   EQ  597  285  118  403 40,3
   IS   88  471  441  529 52,9
   SC   71  406  523  477 47,7

Error: 1409 / 3000 = 46,97 %
```

For this model, we see that the error is mainly due to the fact that there is not a good differentiation between the isosceles triangle and the scalene. Out of the 1409 misclassified samples, around 847 correspond to this case. In other words, 60% of the samples that have been misclassified correspond to an isosceles triangle confused with a scalene or vice versa.

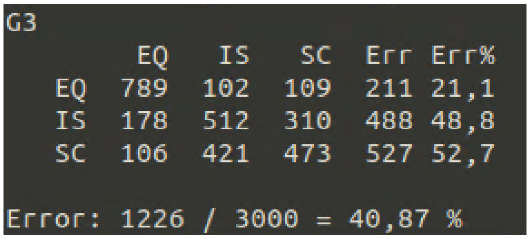


```
G2
       EQ   IS   SC   Err Err%
   EQ  281  211  508  719 71,9
   IS   71  215  714  785 78,5
   SC   81  190  729  271 27,1

Error: 1775 / 3000 = 59,17 %
```

Meanwhile, for model G2 we can see that its main issue seems to be having a big probability associated with the scalene triangle, missclasifying then other types of triangles. That could be due to the lack of information given for equilateral and isosceles triangle, as seen in the previous Test Set Perplexity table.

Furthermore, as we expected in the previous exercise, G2 is the model with higher general perplexity and higher rate error out of the three provided.



```
G3
       EQ   IS   SC   Err Err%
   EQ  789  102  109  211 21,1
   IS  178  512  310  488 48,8
   SC  106  421  473  527 52,7

Error: 1226 / 3000 = 40,87 %
```

Finally, for model G3, even though the errors seem to be a bit homogeneous, we can also notice that it is happening the same as with model G1: there is not a good differentiation between the isosceles triangle and the scalene. Maybe as both triangles have similar test set perplexities, the model has the same information for both but not enough to distinguish them.

Again, as expected, G3 is the model with lowest general perplexity and lowest rate of error (even though it is still pretty high).