HW4: Word Segmentation

Virgile Audi
vaudi@g.harvard.edu
Nicolas Drizard
nicolasdrizard@g.harvard.edu

April 1, 2016

1 Introduction

The goal of this assignement is to implement reccurent neural networks for a word segmentation task. The idea is to identify the spaces in sentence based on the previous characters only. This could be particularly helpful for processing languages written without spaces such as Korean or Chinese.

2 Problem Description

The problem that needs to be solve in this homework is the following: given a sequence of characters, predict where to insert spaces to make a valid sentence. For instance, consider the following sequence of character:

I A M A STUDENT IN C S 2 8 7

the implemented algorithm should be capable of segmenting this sequence into valid words to give:

I am a student in CS 287

To solve this problem, we will train different language models including count-based models, basic neural networks, and recurrent neural networks, combined with two search algorithms to predict the right position for spaces, i.e. a greedy search algorithm and the Viturbi algorithm.

3 Model and Algorithms

3.1 Count-based Model

3.2 Neural Language Model

As a second baseline, we implemented a neural language model to predict whether the next character is a space or not. The model is similar to the Bengio model coded in HW3 but is adapted to characters. Similarly to what we did for word prediction, we imbed a window of characters

in a higher dimension using a look-up table. We first apply a first linear model to the higher dimensional representation of the window of characters, followed by a hyperbolic tangent layer to extract non-linear features. A second linear layer is then applied followed by a softmax to get a probability distribution over the two possible outputs, i.e. a character or a space.
We can summarize the model in the following formula:

$$nnlm_1(x) = \tanh(\mathbf{x}\mathbf{W} + \mathbf{b})\mathbf{W}' + \mathbf{b}'$$

where we recall:

- $\boldsymbol{x} \in \Re^{d_{in} \cdot d_{win}}$ is the concatenated character embeddings
- $\boldsymbol{W} \in \Re^{(d_{in} \cdot d_{win}) \times d_{hid}}$, and $\boldsymbol{b} \in \Re^{d_{hid}}$
- $\boldsymbol{W'} \in \Re^{d_{hid} \times 2}$, and $\boldsymbol{b'} \in \Re^2$.

3.3 Algorithm to generate spaces sequences

As mentioned in the problem description, in order to predict the position of a space, we will use two search algorithm. Both of these algorithm use the language models mentioned above to predict the next character or space given the prior context.

3.3.1 Greedy

The greedy algorithm implemented is an algorithm that chooses the locally optimum choice at every step in the sequence. This algorithm does not generally lead to a global maxium but has the advantage of being easilly implementable and efficient both in memory and complexity. The pseudo-code of the algorithm is presented below:

procedure GREEDYSEARCH
    s=0
    $c \in \mathcal{C}^{n+1}$
    $c_0 = \langle s \rangle$
    for i = 1 to n **do**
        Predict the distribution $\hat{\mathbf{y}}$ over the two classes given the previous context
        Pick the next class that maximises the distribution $c_i \leftarrow \arg\max_{c_i'} \hat{\mathbf{y}}(c_{i-1})_{c_i}$
        Update the score of the chain: $s + \log \hat{\mathbf{y}}(\mathbf{c_{i-1}})_{c_i}$
        Update the chain/context by adding a space or the following character
    return the chain and the score

3.3.2 Viterbi

The second search algorithm that we implemented is the dynamic programming algorithm named after Andrew Viterbi. The main difference between t

3.4 Recurrent Neural Networks

Generic RNN Transducer

LSTM

GRU

4 Experiments

4.1 Count-based Model

4.2 Neural Language Model

4.3 Recurrent Neural Networks

4.4 Ensemble Method

5 Conclusion

End the write-up with a very short recap of the main experiments and the main results. Describe any challenges you may have faced, and what could have been improved in the model.