Internship Log Book

Marius Ortega

April 7, 2023

Abstract

This logbook contains the full extent of my research and discovery during my internship. My final objective is the evaluate the Bert Score and Bart Score NLP metrics.

1 Introduction

In the context of my Master 1 internship, I will have to benchmark Bert Score and Bart Score metrics. Having only a limited experience in NLP Study, my first goal is to better understand NLP as a global subject.

To do so, I will study LSTM's internal structure, read and synthesize scientific articles, and then study and implement Bert and Bart Metrics.

2 LSTM Internal Structure

This section aims at understanding the logical structure of an LSTM Cell. To do so, we will use Figure.1 as a visual support.

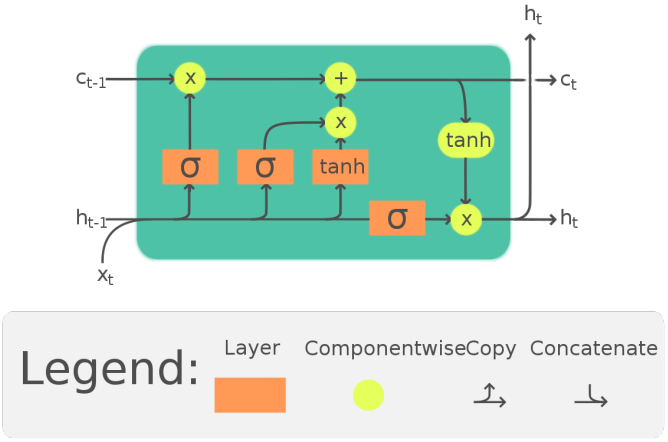

Figure 1: LSTM Cell's Logical Structure Diagram

- **Variables and Symbols :**
  - h_{t-1} and h_t : Respectively the previous and current hidden states
  - c_{t-1} and c_t : Respectively the previous and current cell states
  - x_t : Input value
  - σ : Layer with sigmoid activation function
  - tanh : Layer with hyperbolic tangent activation function

- **Different components of a LSTM cell** :
  - Long Term Memory (Cell State) :

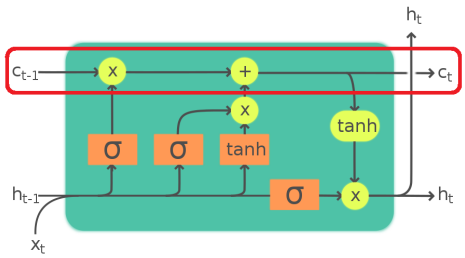

Figure 2: LSTM Cell's Long Term Memory (Cell State)

    * Cell state isn't modified by any bias or weight
    * Prevents the gradient from vanishing or exploding
    * Only gets modified by a multiplicative and an additive component
  - Short Term Memory (Hidden State) :

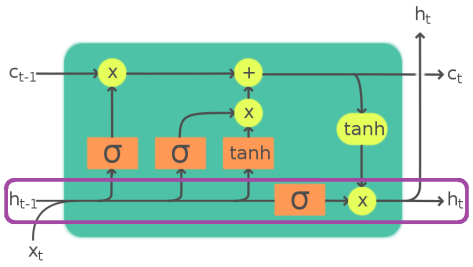

Figure 3: LSTM Cell's Short Term Memory (Hidden State)

    * Gets summed with the input value to update the long term memory and create the new short term memory

- **Gates of an LSTM Cell** :
  - Forget Gate :

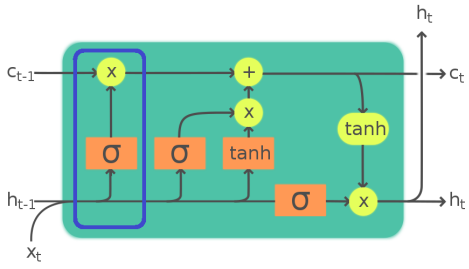

Figure 4: LSTM Cell's Forget Gate

    * Determines what percentage of the long term memory to keep
    * Sums Hidden state and Input
    * Runs the result trough a Sigmoid layer which gives a factor $f \in [0, 1]$
    * Multiplies f with the previous long term memory c_{t-1}

– Input Gate :

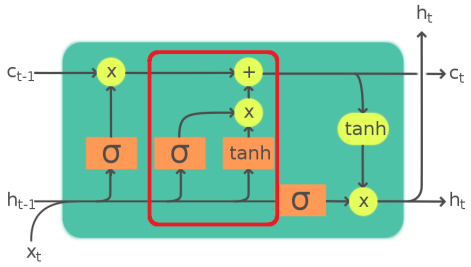

Figure 5: LSTM Cell's Input Gate

* Updates the long term memory
* Sums Hidden state and Input
* Runs the result though a hyperbolic tangent layer to create a potential long term memory
* In parallel, runs the previous result through a sigmoid function
* Multiples both of these outputs to determine which percentage of the potential long term memory to keep
* Sums the previous long term memory and the potential one to finalize the update

– Output Gate :

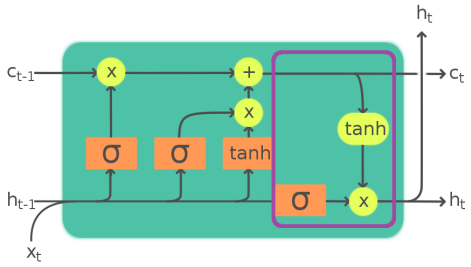

Figure 6: LSTM Cell's Output Gate

* Updates the short term memory
* Runs the updated long term memory through a hyperbolic tangent layer to determine a potential short term memory
* Sums Hidden state and Input
* Runs it through a sigmoid layer
* Multiples the potential short term memory and the sigmoid layer output to determine what percentage of the potential short term memory to keep

- **Output of the LSTM cell** :
  – The updated long term memory becomes the cell state of the next LSTM cell
  – The update short term memory becomes the hidden state of the next LSTM cell

3 Articles Syntheses

3.1 ROUGE : A Package for Automatic Evaluation of Summaries

3.1.1 Rouge : Global Definition

- **Rouge** : Recall-Oriented Understudy for Gisting Evaluation.
  - Gisting : Use of machine Translation (MT) to translate foreign texts.
- Rouge **aims** and **main specs** :
  - Automatic evaluation of summaries.
  - Show correlation of it's results with human judgement.

3.1.2 Rouge-N : N-gram Co-Occurrence Statistics

N-gram : Contiguous sequence of n items from a given sample of text or speech

$$Rouge_N = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

  - With :
    - n : length of the n-gram
    - $gram_n$: n-gram
    - $Count_{match}(gram_n)$: maximum number of n-grams co-occurring in a candidate summary and a set reference summaries
    - $Count(gram_n)$: number of n-grams occurring at the reference summary side
- **Comments** :
  - Recall-related measure because the denominator is equivalent to $TP + FN$
  - While numerator is equivalent to TP
- **Strengths** :
  - A candidate summary that contains words shared by more references is favored. We prefer when there is some kind of consensus among reference summaries.
  - Intuitive and reasonably simple to explain.

Multi-Reference Rouge-N :

$$Rouge_{N_{multi}} = argmax_i(Rouge_N(r_i, s))$$

- With :
  - r_i : i^{th} reference summary
  - s : candidate summary

3.1.3 Rouge-L : Longest Common Subsequence

- **Vocabulary** :
  - LCS : Longest Common Subsequence
  - LCSR : LCS Ratio : $LCSR = \frac{len(LCS(x,y))}{argmax(len(words_x), len(words_y))}$

- **Definition of a Subsequence** :
  - Let :
    - X a sequence, $X = [x_1, x_2, .., x_n]$
    - Z another sequence, $Z = [z_1, z_2, .., z_n]$
    - I strictly increasing sequence of X indices, $I = [i_1, i_2, .., i_n]$
  - Then :

$$\exists I, \forall j \in [1, k], x_{ij} = z_j \Rightarrow \text{Z is a subsequence of X}$$

- **Sentence-Level Rouge-L**:

$$\begin{cases} R_{LCS} = \frac{LCS(X,Y)}{m} & , m = length(X) \\ P_{LCS} = \frac{LCS(X,Y)}{n} & , n = length(Y) \\ \beta = \frac{P_{LCS}}{R_{LCS}} & \\ Rouge_{L_{sentence}} = F_{LCS} = \frac{(1+\beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} & \end{cases} \tag{1}$$

  - With :
    - X : reference summary
    - Y : candidate summary
    - LCS(X, Y) : length of the LCS of X and Y
  - Strengths of the sentence-level Rouge-N :
    - In contrast to Rouge-N, **Rouge-L takes into account the order of words**.
  - LCS cons :
    - Only takes the main in-sequence words while ignoring other alternatives (Rouge-N does not suffer this problem).

- **Summary-Level Rouge-L** :
  - Let :
    - C : Candidate sequence : $C = [c_1, c_2, \ldots, c_n]$
    - r_i : Reference summary sentence
    - u : Number of sentences in the reference summary
  - Then :

$$\begin{cases} R_{LCS} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{m} & , m = length(X) \\ P_{LCS} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{n} & , n = length(Y) \\ \beta = \frac{P_{LCS}}{R_{LCS}} & \\ Rouge_{L_{summary}} = F_{LCS} = \frac{(1+\beta^2) R_{LCS} P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} & \end{cases} \tag{2}$$

  - Strength :
    - $Rouge_{L_{summary}}$ doesn't suffer from the same "tunnel vision" effect as $Rouge_{L_{sentence}}$.
    - It takes into account every matching LCS and subsequence even if they are not adjacent given that is only considers there union.

- **Normalized Pair-Wise Rouge-L** :
  - LCS_{MEAD} : Normalized Pair-Wise LCS.

$$\begin{cases} R_{LCS_{MEAD}} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{m} & , m = length(X) \\ P_{LCS_{MEAD}} = \frac{\sum_{i=1}^{u} LCS_{\cup}(r_i, C)}{n} & , n = length(Y) \\ \beta = \frac{P_{LCS_{MEAD}}}{R_{LCS_{MEAD}}} & \\ LCS_{MEAD}(L_1, L_2) = \frac{(1+\beta^2) R_{LCS_{MEAD}} P_{LCS_{MEAD}}}{R_{LCS_{MEAD}} + \beta^2 P_{LCS_{MEAD}}} & \end{cases} \tag{3}$$

  - Difference between LCS_{MEAD} and Rouge-L :
    - Normalized pairwise LCS takes the best LCS score while ROUGE-L takes the union LCS scores.

3.1.4 Rouge-W : Weighted Longest Common Subsequence

- **Vocabulary** :
  - WLCS : LCS but we keep in a dynamic 2D array the longest consecutive matches.
- **WLCS Algorithm** :

```
(1) For (i = 0; i <=m; i++)
      c(i,j) = 0  // initialize c-table
      w(i,j) = 0 // initialize w-table
(2) For (i = 1; i <= m; i++)
      For (j = 1;j <= n;j++)
        If x_i = y_j Then
          // the length of consecutive matches at
          // position i-1 and j-1
          k = w(i-1,j-1)
          c(i,j) = c(i-1,j-1) + f(k+1) – f(k)
          // remember the length of consecutive
          // matches at position i, j
          w(i,j) = k+1
        Otherwise
          If c(i-1,j) > c(i,j-1) Then
            c(i,j) = c(i-1,j)
            w(i,j) = 0          // no match at i, j
          Else c(i,j) = c(i,j-1)
            w(i,j) = 0          // no match at i,j
(3) WLCS(X,Y) = c(m,n)
```

Figure 7: WLCS Algorithm pseudo-code

  - With :
    - c : dynamic table with c_{ij} storing the WLCS score ending at word x_i of X and at word y_j of Y.
    - f : function of consecutive matches at the table position c(i, j).
- **Rouge-W Formula** :

$$\begin{cases} R_{WLCS} = f^{-1}(\frac{WLCS(X,Y)}{m}) & , m = length(X) \\ P_{WLCS} = f^{-1}(\frac{WLCS(X,Y)}{n}) & , n = length(Y) \\ \beta = \frac{P_{WLCS}}{R_{WLCS}} & \\ Rouge_W = F_{WLCS} = \frac{(1+\beta^2)R_{WLCS}P_{WLCS}}{R_{WLCS}+\beta^2 P_{WLCS}} & \end{cases} \tag{4}$$

- **f weighting function** :

  $f(a+b) > f(a) + f(b), a, b \in N$

  - Examples :
    - $f(k) = \alpha k - \beta, \qquad k \geq 0, \alpha > 0, \beta > 0$
    - $f(k) = k^\alpha, \qquad \alpha > 1$
      - We prefer polynomials that can be inverse in closed form like k^2.
- **Strength of Rouge-W** :
  - It rewards consecutive matches in contrast to Rouge-L.

3.1.5 Rouge-S : Skip-Bigram Co-Occurrence Statistics

- **Vocabulary** :
  - Skip-Bigram : All combinations of possible bi-grams in a sentence.
- **Rouge-S Formula** :

$$\begin{cases} R_{SKIP2} = \frac{SKIP2(X,Y)}{C_2^m} & , m = length(X) \\ P_{SKIP2} = \frac{SKIP2(X,Y)}{C_2^n} & , n = length(Y) \\ \beta = \frac{P_{SKIP2}}{R_{SKIP2}} & \\ Rouge_W = F_{SKIP2} = \frac{(1+\beta^2)R_{SKIP2}P_{SKIP2}}{R_{SKIP2}+\beta^2 P_{SKIP2}} & \end{cases} \tag{5}$$

  - We can reduce the **skip distance** to only try to match word that are close in the sentence.
  - **Weakness of Rouge-S** :
    - Doesn't give any credit to a candidate sentence when there are no word co-occurring even if they have uni-grams in common.
- **Rouge-SU** :
  - We add uni-gram as counting unit to the basic Rouge-S to eliminate the main weakness of Rouge-S.

3.1.6 Evaluation of Rouge

Before considering the results of the conducted experiment, let's evaluate the quality of the protocol. To do so, we will focus 3 main aspects :

- **Datasets quality** :
  - Is it gold standard or often utilized for benchmarking ?
  - Is its size consequent enough ?
  - Is is biased or noisy (due to preprocessing choices) ?
- **Algorithm's Complexity** :
  - Spacial Complexity
  - Time Complexity
- **Parameters affecting the algorithm's robustness**.

We summarize these KPIs in the following table :

Table 1: Rouge's Evaluation conditions Study

(a) Dataset Study Table

Datasets KPIs	Comments
Benchmark-oriented or Gold Standard	Yes, they are gold standard and human annotated datasets widely used in scientific experiments
Size	Relatively small datasets : Only 1127 annotated summaries in DUC 2001,2002 and 2003 put together (considering all summaries size)
Biased or Noisy	No information were given relatively to the preprocessing but we know they used stop words to improve their algorithms performances

(b) Complexity Study Table

Complexities	Comments
Spatial	No information on the implementation
Time	No information on the benchmarking hardware and the algorithm run time

(c) Robustness to variation Study Table

Parameters	Comments
Stop words	Rouge is subject to noise disturbance : words like "I" or "the"
Summary Size	Some metrics like Rouge-S4 reach peak performance for very short summaries (10 words) but become inconsistent if the size comes to change
Method of Measurement	Only Pearson's correlation results are shown in the paper. However, the measure doesn't take into account non-linear relationships and Independence between variables.

3.1.7 Conclusion of Rouge's performances

Now that we took the context of the evaluation into account, we can study the experiment's results. Let's sum up which measures are the best depending of the required task :

- **Single document summarizing :**
  - Rouge-2, Rouge-L, Rouge-W, Rouge-S
- **Short/Headline Single document summarizing :**
  - Rouge-1, Rouge-L, Rouge-SU4, Rouge-SU9
- **Multi document summarizing :**
  - Rouge-1, Rouge-2, Rouge-S4, Rouge-S9, Rouge-SU4, Rouge-SU9 (couldn't reach 90% correlation)

3.2 BERTScore : Evaluating Text Generation with BERT

3.2.1 Introduction to BERTScore

BERTScore is an automatic evaluation metric for text generation. It addresses two common downsides of n-gram-based metrics :

- Lack of robustness to match paraphrases.
  - BERTScore computes similarity using contextualized embedding.
  - Very efficient technique for paraphrase matching.
- Fail to capture distant dependencies and penalize semantically-critical ordering changes.
  - Contextualized embedding is designed to solve such problems.

3.2.2 Definition of BERTScore

In this section, we will explain how to calculate BERTScore based on Figure.8 :

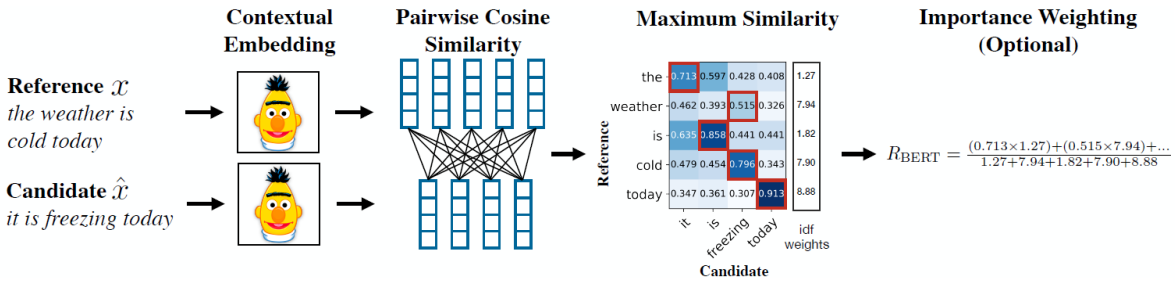

Figure 8: BERTScore Computing Process

- **Contextual Embedding** :
  - Generates different vector representations for the same word in different sentences depending on the surrounding words.
  - Gives context to the embedding.
  - Unlike classical embedding that provides a single vector representation per word.
- **Pairwise Cosine Similarity Measurement** :
  - Let :
    - x_i : Reference token
    - $\hat{x}_j$: Candidate token
    - S : Cosine similarity between x_i and $\hat{x}_j$
  - Then :
$$S_{ij} = \frac{x_i^T \hat{x}_j}{||x_i^T||\ ||\hat{x}_j||}$$
  - Comment :
    - BERTScore implementation pre-normalizes vectors to prevent from calculating the denominator of S_{ij}.

- **BERTScore - Maximum Similarity** :
  Greedy matching is used to maximize the similarity scores. Each token is matched to the most similar token of the other sentence.
  - Let :
    - x_i : Reference token
    - $\hat{x}_j$: Candidate token
  - Then :

$$\begin{cases} Recall & R_{BERT} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} (x_i^T \hat{x}_j) \\ Precision & P_{BERT} = \frac{1}{|\hat{x}|}\sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} (x_i^T \hat{x}_j) \\ F1Score & F_{BERT} = 2\frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \end{cases} \tag{6}$$

- **Importance Weighting (Optional)** :
  - Observation :
    - Rare words can be more indicative for sentence similarity than common words.
    - Thus, we can use IDF (Inverse Document Frequency) to incorporate importance weighting to BERTScore.
    - In the same fashion as search engines.
  - Inverse Document Frequency :
    - Let :
      - M : Number of reference sentences
      - $\{x^{(i)}\}_{i=1}^{M}$: Reference Sentence
      - w : Word-piece token
    - Then :

$$idf(w) = -\log\left(\frac{1}{M}\sum_{i=1}^{M}(\mathbb{1}[w \in x^{(i)}])\right)$$

  - Recall with IDF :

$$R_{BERT} = \frac{\sum_{x_i \in x} idf(x_i) \max_{\hat{x}_j \in \hat{x}} (x_i^T \hat{x}_j)}{\sum_{x_i \in x} idf(x_i)}$$

    - idf score remains the same for a specific test set because it is computed based on reference sentences.
    - Plus-one smoothing is applied to handle unknown word pieces.

- **Baseline Re-scaling** :
  - In theory, BERTScore lies between -1 and 1 (because we use normalized vectors to compute similarity).
  - In practice, BERTScore lies in a much smaller range due to the learned geometry of the contextual embedding.
  - It doesn't affect its performance but makes it hardly readable.
  - Thus we re-scale it between 0 and 1 so it is more evenly distributed in this range. Let's take an example with the recall :

$$\hat{R}_{BERT} = \frac{R_{BERT} - b}{1 - b}$$

    - b : The empirical lower bound computed with Common Crawl Monolingual Datasets

3.2.3 BERTScore Evaluation

:

References

[Lin04] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out (ACL 2004 Workshop)*, 2004.

[Sta22] Josh Starmer StatQuest. Long short-term memory (lstm), clearly explained. Youtube, 2022.

[Wik23] Wikipedia. Long short-term memory, 2023.

[ZKW+20] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yaov Artzi. Bertscore : Evaluating text generation with bert. *ICLR*, 2020.

[Lin04] [Wik23] [Sta22] [ZKW+20]