ALMA MATER STUDIORUM
UNIVERSITA' DI BOLOGNA

The Deep Comedy

Rispoli Luca
luca.rispoli@studio.unibo.it
Andrea Lavista
andrea.lavista@studio.unibo.it

March 3, 2021

Indice

Introduction
Overview

- The project has been entirely developed with **Keras** library.
- After a first research phase, we experimented with different LSTM architectures, comparing the results to see which one was the most suitable for the task at hand.
- In order to evaluate our models, we used the recommended metrics, which are:
  - plagiarism
  - average hendecasyllables
  - average rhymeness
  - average structuredness

LSTM Architectures

Recurrent Neural Networks

- **Recurrent Neural Networks (RNN)** are widely used in tasks such as: translation, handwriting recognition and text generation.
- They handle well inputs where individual samples are dependent from each other, moreover they can process sequences of any length.
- However, RNNs are prone to problems such as exploding and vanishing gradient, also they fail to remember long-term dependencies.
- In our project we decided to adopt **Long Short-Term memory (LSTM)**, a special kind of RNNs that adresses some of its weaknesses.

- LSTM employs memory cells and several layers, called "gates", in order to deal with long-term dependencies.

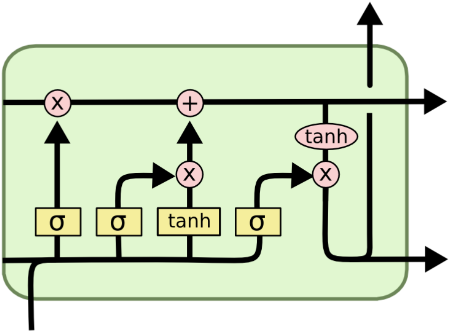


Figure: 1.1 Overview of a standard LSTM cell

LSTM Architectures

Baseline LSTM

- Our first model is based on two stacked LSTM layers, followed by a dense layer used to compute the output.
- The model was tested both with a **character** based approach and a **word** based one.

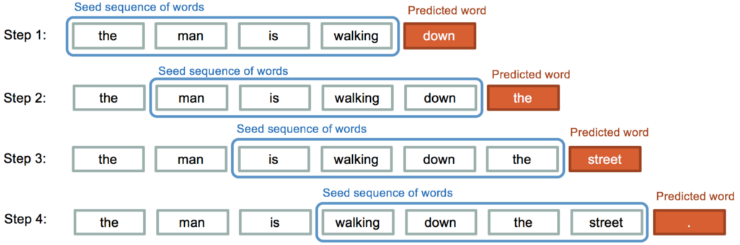


Figure: 1.2 Functioning of the word-based model

LSTM Architectures

Baseline LSTM

Plagiarism	Structure	Hendecasyllable	Rhyme
0.97	0.84	0.79	0.1

Table: scores of the baseline word-based LSTM

Plagiarism	Structure	Hendecasyllable	Rhyme
1.0	0.89	0.77	0.07

Table: scores of the baseline char-based LSTM

- We decided to pursue the word-based approach, trying to improve its rhyme score.

- In order to improve the rhyming score, we used **fast text** to generate the initial weights of the embedding matrix.

Closest Neighbours				
Word	**1° Neigh**	**2° Neigh**	**3° Neigh**	**4° Neigh**
Ardito	Dito (0,77)	Sito (0,76)	Udito (0,72)	Lito (0,67)
Ossa	Rossa (0,94)	Mossa (0,92)	Grossa (0,84)	L'ossa (0,65)
Tolse	Rivolse (0,75)	Volse (0,73)	Scese (0,64)	Rimase (0,62)

Table: Example of some word's closest neighbours

LSTM Architectures

Tweaks

Plagiarism	Structure	Hendecasyllable	Rhyme
0.87	0.98	0.84	0.12

Table: scores of the fasttext word-based LSTM

- Fast text brought a slightly increase to the networks' scores.
- We opted for a double network approach, one specialized in generating the correct rhyme, one generating the rest of the verse according to the hendecasyllable rules.

LSTM Architectures

Rhyming Network

- The rhyming network was trained using a **custom loss** and a dataset created ad-hoc
- The network has to generate only the last word of the verse, but that would ruin the structuredness scores, therefore, we generate verses in reverse.

Plagiarism	Structure	Hendecasyllable	Rhyme
0.99	0.98	0.87	0.90

Table: scores of the Rhyming+Tercet networks

- The results were a great improvement compared to the starting networks, next, we experimented with Sequence to Sequence, another algorithm widely used for text generation.

LSTM Architectures
Output Example

non pur già vero ben ti con l’altre ventiquattro
ma che non dee di retro parte
perché l’ombra che prima quattro

la materia che coi col fu arte
ché di quella vita fu santa globo
mi vidi ne la cenno parte

ma e tra ne erto duce
da me quella che l’ardita novella
parlare e ’ a lui tace e e luce

ma ogne mai mi fa far lui favella
questi va parve suo viver sì spelonche
quel risposto che e io li io lui bella

- Seq2Seq models are widely used in natural language processing tasks, such as translation, question answering and chatbot and, more in general, in text generation.

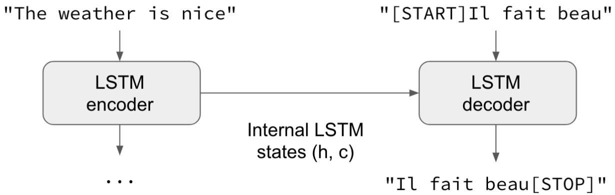


Figure: 2.1 Overview of an example of the Seq2Seq architecture

- Our Seq2Seq model is composed by a Bidirectional LSTM for the encoder and a single layer LSTM for the decoder, combined with the attention mechanism, followed by a dense layer used to compute the output.

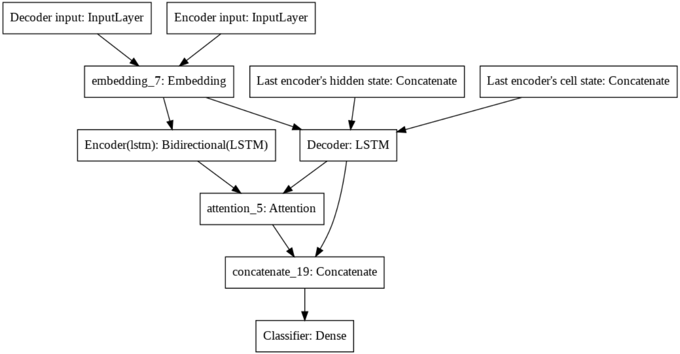
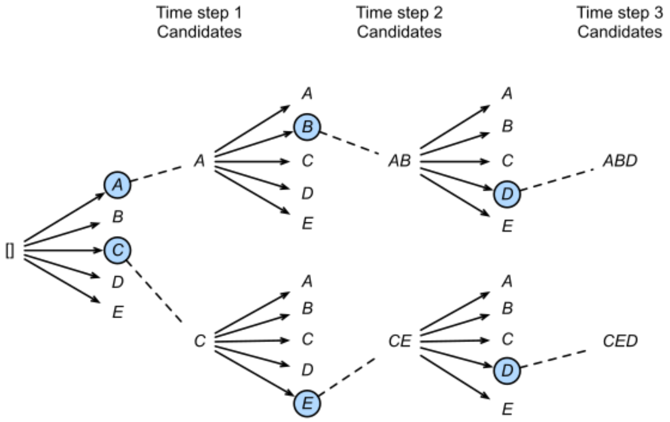


Figure: 2.2 Our architecture

Sequence 2 Sequence
Beam search

- Seq2Seq is often combined with beam search, a search algorithm that instead of taking as output for each time step the token with the highest probability, it explores more paths choosing the best one.



Sequence to Sequence
Scores

- The model was tested both with a **character** based approach and a **word** based one.
- The best results were reached taking for each time step the token with the highest probability, so without **beam search**.

Plagiarism	Structure	Hendecasyllable	Rhyme
0.917	0.986	0.898	0.34

Table: scores of the word-based Seq2Seq

Plagiarism	Structure	Hendecasyllable	Rhyme
0.914	0.996	0.953	0.962

Table: scores of the char-based Seq2Seq

Sequence to Sequence
Output Example

e io a lui si rivolse in su lenti
che se non che l’andare a la sua schiante
che l’aver li piedi e con le sue senti

e quella che parea che l’alte strante
che di là dal mondo in su la sua vita
che si fa che sien di quella con quante

così l’anima che per la sua vita
di quella che con le sue parole strada
che di là dal mondo del mondo strita

che l’altra vista di là dal mal sada
e per lo suo passo del ciel che siene
che non si può con la vista si scada

Rhyme encoder
Training

- We trained a neural network capable of classifying words, where each class contains the words with the same subword from the accent to its end
- Embedding output = <16,30>, Encoder output = <1,10>
- Accuracy = 84,3%, F1-score = 57,3%

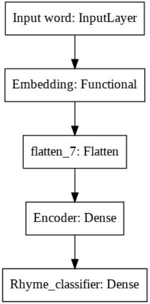


Figure: 3.1 Model for the classification of the rhymes

Rhyme encoder
Exporting

- Then we exported the embedding and the encoding layers, evaluate them and tried them in combination with the previous models.

Evaluation

- Taking 100 pairs of words that did rhyme: the average similarity was 0.739 and 88% of them had a similarity above 0.3
- Taking 1000 pairs of words that didn't rhyme the average similarity was -0.009 and 83,2% of them had a similarity below 0.3

Test

- At the end, we used them with the word-based models (both the baseline LSTM and the Seq2Seq), concatenating the output of the rhyme encoder with the output of the embedding layer

Conclusions

- Both the double LSTM and the Seq2Seq approaches yielded positive results
- Sequence2Sequence model is the one that better reflects Dante's style, but some of the generated words are made up.

Improvements

- Experiment with other types of layers, such as **Transformers**
- Pre-train the model on different texts in vulgar language.

Acknowledgments

" Text Generation Using LSTM " by Harsh Bansal (2020)
" Understanding LSTM Networks " by Colah (2016)
"A ten-minute introduction to sequence-to-sequence learning in Keras" by Francois Chollet (2017)
"Dive into deep learning" by Michael Fullan (2019)