Neural approaches to text normalization

Deep Learning for Speech and Language, January 2018

Pau, Miguel and Esteve

About us

- Undergraduate CFIS students @ UPC
- Three different double degrees:
  - Miguel: Mathematics + CS
  - Esteve: Industrial Eng. + CS
  - Pau: Mathematics + Physical Eng.

Index

1. Introduction to text normalization: problem and dataset
2. Neural approaches to the problem
   2.1. Sequence2Sequence in Keras
   2.2. Using Open NMT implementation in Pytorch
   2.3. Attention based models
   2.4. Word embeddings
3. Conclusions

Introduction to text normalization (I): Problem

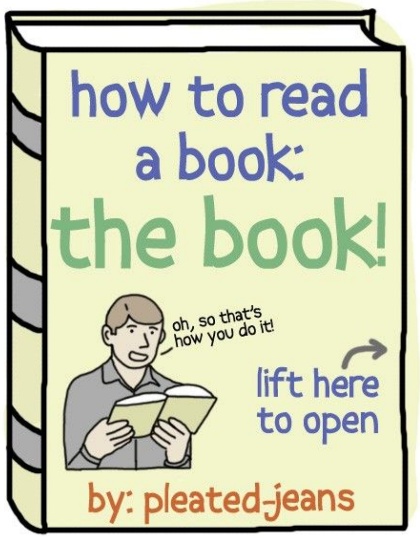
- Objective: Create a system that transforms text into readable expressions (i.e teaching a computer "How to read" unconventional expressions).

For example,

"150lb" → "one hundred fifty pounds"
"200$" → "two hundred dollars".

- The problem was the subject of a kaggle competition ([Link]()), where both deep learning related and non deep learning solutions were offered

- Our work consisted in exploring and comparing between deep learning approaches to the problem



Introduction to text normalization (II): Data

Dataset:

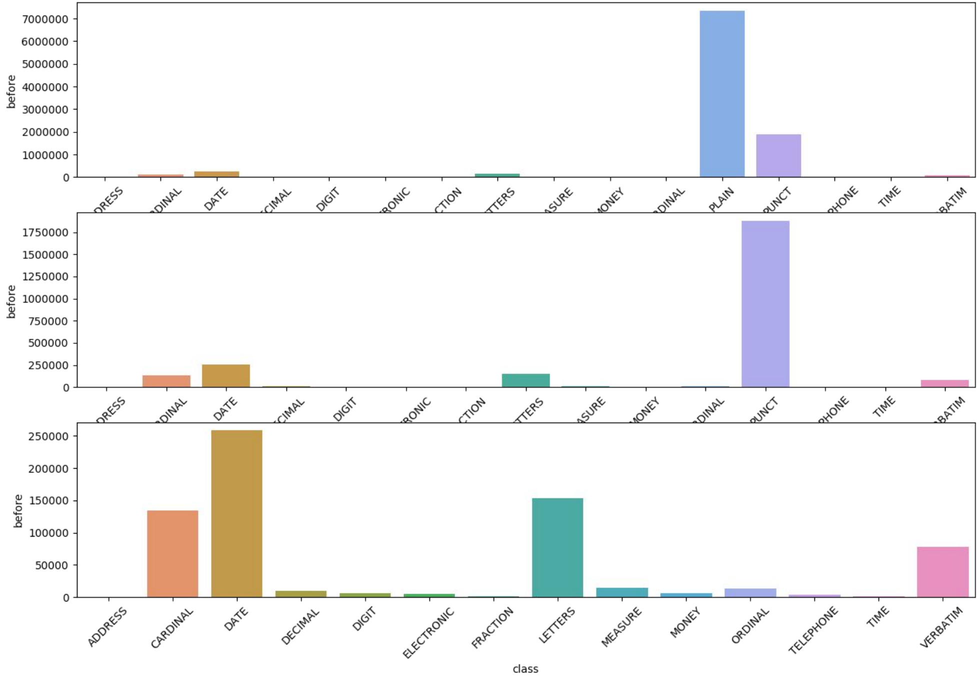
- Train: ~10M words separated in ~750k sentences
  For each word, sentence_id, token_id, class (word type), before (raw word) and after (normalized word) are given.

- Test: ~1M words separated in ~70k sentences
  The word type and the normalized word are not given in this dataset

Example:

	sentence_id	token_id	class	before	after
0	0	0	PLAIN	Brillantaisia	Brillantaisia
1	0	1	PLAIN	is	is
2	0	2	PLAIN	a	a
3	0	3	PLAIN	genus	genus
4	0	4	PLAIN	of	of

Introduction to text normalization (III): Data

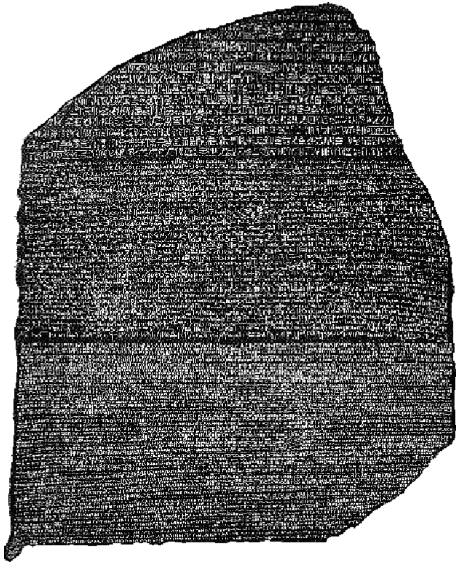


Train dataset:

- ~93.34 % words are equal to the normalized word
- 16 different classes
- The most common class is “PLAIN” (~74.14 %)

Neural approaches to the problem (I): Intro

- Most of the approaches treat the problem as a pure translation one. Therefore, all the classic neural machine translation (NMT) methods can be applied
- We could in theory take advantage of the two “languages” being equal most of the cases with different methods found in literature, but we just explored “vanilla” NMT methods due to lack of time
- We want to see whether the best NMT procedures are also the best for this particular task



First ever NMT dataset?

Neural approaches to the problem (II): Pytorch + OpenNMT framework

- OpenNMT provides a pytorch implementation of different classic NMT methods
- Using these implementations, we tried different possible models to try to see which worked better
- Our final training was performed using the exact same architecture that was used in the WMT‘16 Multimodal Translation task (english to german translation), based mainly in CNNs

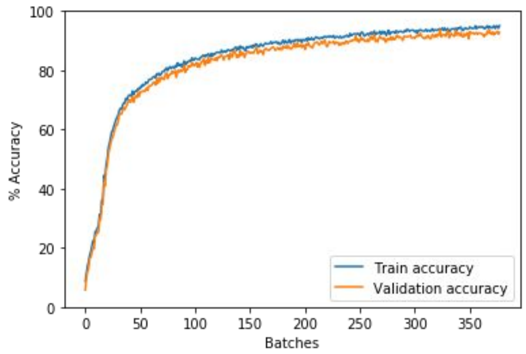
PYTORCH

+

OpenNMT

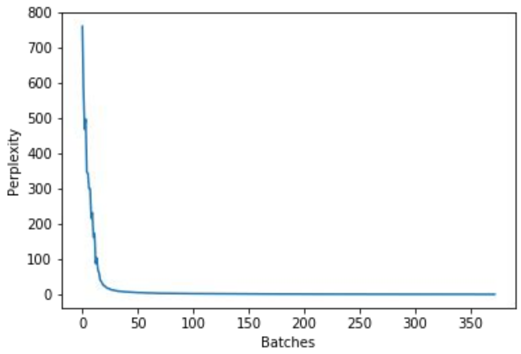
Open-Source Neural Machine Translation in Torch

Neural approaches to the problem (II): Pytorch + OpenNMT framework results



Max Train accuracy: 95.34 %

Max Valid accuracy: 91.36 %



Min perplexity: 1.53

PYTORCH

+

OpenNMT

Open-Source Neural Machine Translation in Torch

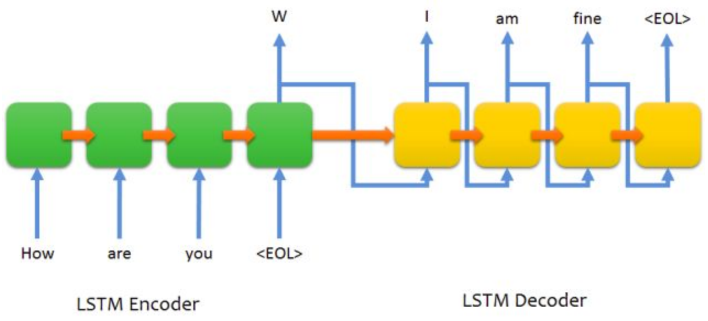
Neural approaches to the problem (III) : Keras sequence2sequence

- Simple LSTM sequence2sequence encoder decoder scheme, coded in Keras

- Character sequence are feed in the encoder in one-hot encoding representation.

- No context used:
    - DL -> Deadline or Deep Learning?

- In some categories context may be useless:
    - 1973 -> nineteen seventy three



Neural approaches to the problem (III) : Keras sequence2sequence

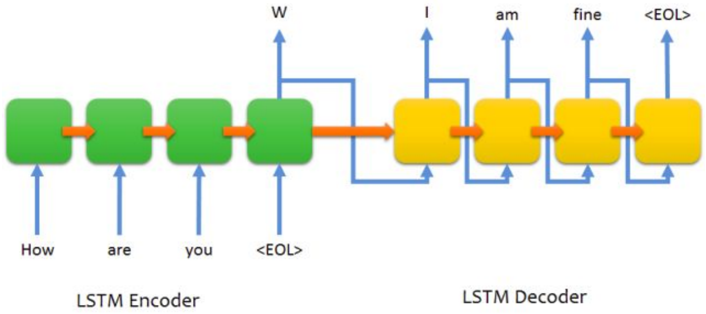
- ~90% Validation accuracy, but overall impressive results in some non-trivial examples considering low training time

  Input sentence: April 10, 2013
  Decoded sentence: april tenth twenty thirteen

  Input sentence: 91
  Decoded sentence: ninety one

  Input sentence: ALCS
  Decoded sentence: a l c s

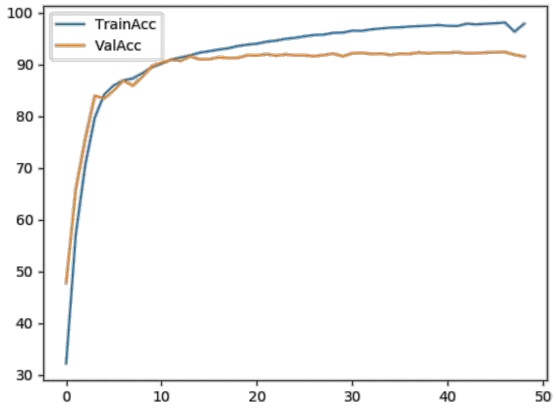
- Invented words + long repeating sequences while undertrained
  2015 → twenty finty twenty finty twenty finty twenty finty twenty finty twenty ...



Neural approaches to the problem (IV): Attention based models

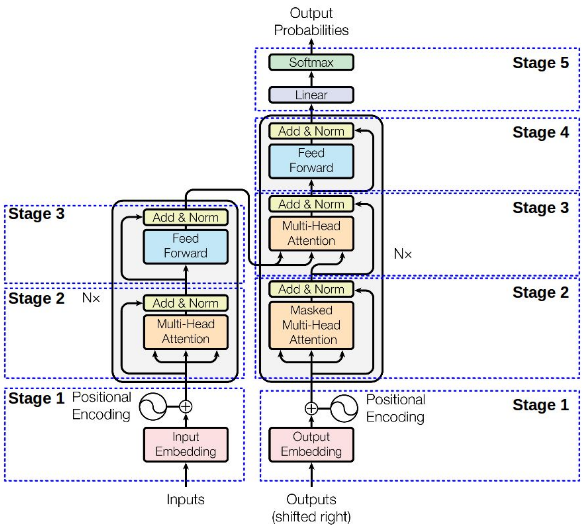
- Thanks to the work performed by former students of the UPC course “Deep Learning for Artificial Intelligence”, attention-based models could also be compared for this particular task

Results for the transformer



Max Train accuracy: 97.836 %

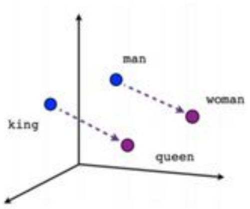
Max Valid accuracy: 92.376 %



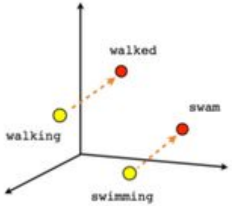
Neural approaches to the problem (Bonus): Can word embeddings help us in the task?

Using the learned vocabulary and two Word2Vec embeddings

- **Accuracy:** 96.6 %

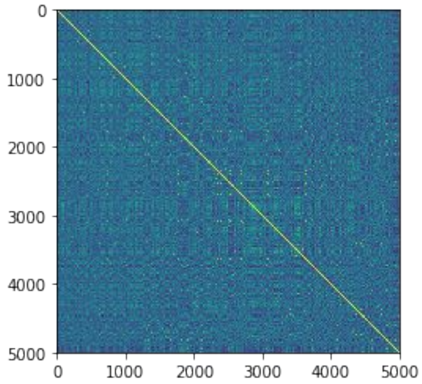


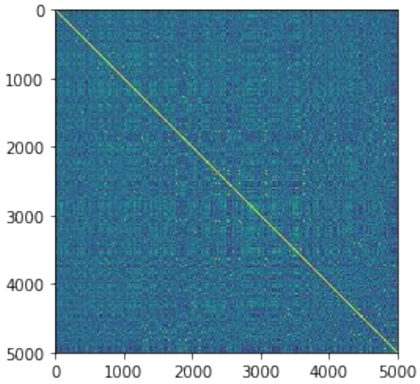
Male-Female



Verb tense

Country-Capital





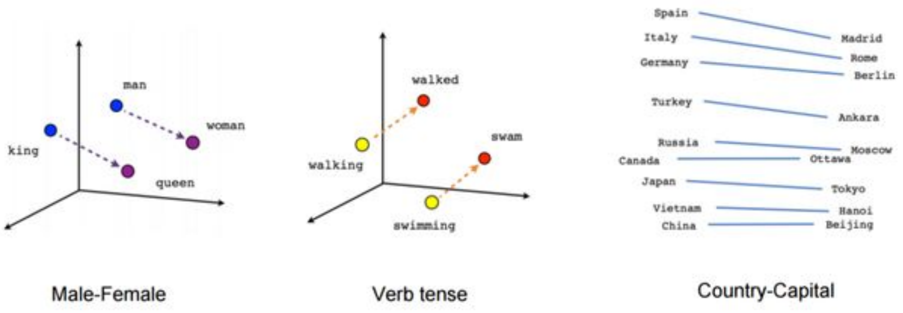
Neural approaches to the problem (Bonus): Can word embeddings help us in the task?

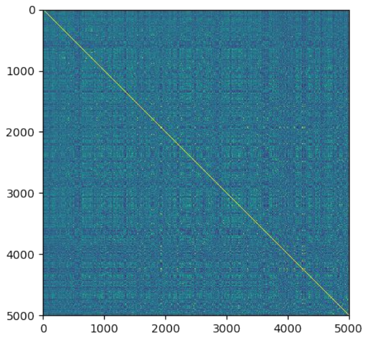
Using word embedding for classifying in 16 classes

- **Accuracy:** 96.87 %

Using word embedding for classifying in two classes

- **Accuracy:** 97.07 %

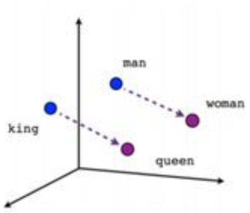




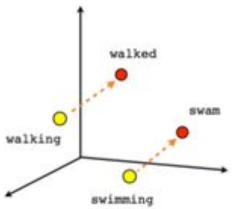
Neural approaches to the problem (Bonus): Can word embeddings help us in the task?

- In fact, it works as a dictionary.

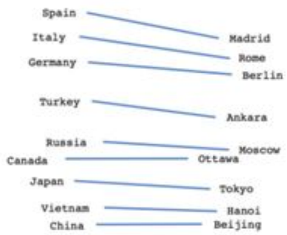
- Word embedding is not a final methodology in this case, because it lacks generalization capacity to unseen examples



Male-Female



Verb tense



Country-Capital

Results & Conclusion

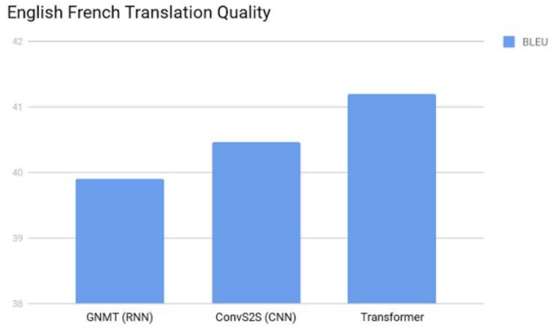
- I was already known that for traditional NMT tasks, Transformer > CNN > RNN

- It was observed that the same order still holds for Text normalization when seen as a translation problem, at least accuracy-wise.

- Usage of Word Embeddings in this particular problem was discussed

- Really interesting and immersive first experience with Neural Machine Translation!

Method	Validation accuracy (%)
RNN	~90
CNN	91.36
Attention	92.376



THANKS!

Any questions?