Text processing pipelines: the Deep Learning chapter

Radu ION

Research Institute for AI, Romanian Academy

radu@racai.ro

Contents

- Definition of the text processing domain: what is text processing and what is it good for?
- “Classical” text processing algorithms: HMM POS tagging
- Deep Learning text processing algorithms: BiLSTM + CRF POS tagging.
- Open-source tools and resources
- Where to go next

Text processing: what is it?

- **Automatic text analysis at different linguistic levels** in order to provide a structured representation for further analysis or different applications
- Analyses examples: tokenization
  - Easy: “I walk along the line.”
  - Not so easy: “I used a **0.12mg/l** dose.”
- Analyses examples: POS tagging
  - “can” is a NOUN in “I don’t want to eat fish from the **can**.”
  - “can” is a VERB (modal) in “I **can** do this!”

Applications and text processing

- Suppose you want to cluster Twitter comments into two classes: those who like person A and those who don’t.
- Usually, you need to detect verb occurrences such as “hate”, “hates”, “don’t like”, “likes”, “loves”, “love” and construct your cluster algorithm accordingly
- “hate” and “hates” are two word forms of the same lemma “hate”
- In general, **you want to eliminate the lexical variants of a word and use its lemma** to better estimate different statistics related to the comment
- To be able to do lemmatization you need tokenization and POS tagging, **two standard text processing steps**.

Applications and text processing (2)

- Suppose you want to build a question answering system from Wikipedia, dealing with factual questions such as:
  - "**When was Albert Einstein born**?"
  - "What is the value of Pi?"
- Whatever the answering algorithm, you need a way to analyze the input question:
  - What is the subject of the question? (e.g., "Albert Einstein")
  - What is the predicate of the question? (e.g., "to be born")
  - What is to be answered? (the date of his birth)
- In order to carry out such analysis, we need **syntactic analysis** of the question, which requires **POS tagging**, which, in turn, requires **tokenization**.

Standard levels of text processing

- Sentence splitting:
  - "I was here.**/**He was there."→ two sentences
- Tokenization:
  - "**I**", "**was**", "**there**", "**.**" → four tokens, one punctuation token
- POS tagging:
  - "I/**pronoun**", "was/**verb**", "there/**adverb**", "./**punctuation**"
- Lemmatization/stemming:
  - "was → **to be**"
  - "reanalyzed" → **analyze**

Other levels of text processing

- Syntactic analysis (constituents or dependency parsing)
    - “**[**I**]**$_{NP}$ **[**was **[**there**]**$_{AP}$ **[**all night**]**$_{NP}$.**]**$_{VP}$”
- Word sense disambiguation
- Named entity recognition
    - “**Roger Federer/PERSON** is, probably, the best tennis player in history.”
- For speech processing:
    - Syllable splitting
    - Phonetic transcription
    - Detection the accented syllable

Text processing and error propagation

- All levels of text processing require that some other levels have already been run
  - POS tagging depends on tokenization
  - Syntactic analysis depends on POS tagging
- **Any linguistic processing algorithm is error prone**
  - Some are better than others
  - POS tagging: 97-98% https://aclweb.org/aclwiki/POS_Tagging_(State_of_the_art)
  - Dependency parsing: 85% LAS https://universaldependencies.org/conll18/results-las.html
- Errors propagate down the pipeline (e.g., parsing is 85% accurate if POS tagging is 98% accurate)

Literature on text processing

- Simply put, it is **huge**.
- A processing level has its own literature
  - Some well-known processing levels, such as POS tagging, have, probably, thousands of papers published for all languages
  - Some less-interesting processing levels, such as tokenization, are taken for granted (e.g., “it’s easy, just split at space boundaries”) when this should not be the case (e.g., Romanian has clitics, Hungarian is agglutinative, etc.)
- To get started on a text processing level literature, Google it with queries such as:
  - <text processing level> survey
  - <text processing level> tutorial

Text processing: old and new

- Automatic linguistic analysis has (almost always) been done using Machine Learning (ML)
  - “Classic” methods such as Support Vector Machines or Conditional Random Fields were used
  - Feature engineering was done separately
- ML has **spectacularly** grown in popularity with the advent of “complex neural networks”: NNs with huge number of parameters (ranging from a few millions to hundreds of millions), defining many **stacked layers**
  - Until recently, these networks could not be trained unless very expensive hardware was at hand
  - With the progress in GPU design and, especially their low cost, coupled with the existence of virtual machines “in the cloud”, Deep Learning took off at an incredible speed
- Main difference: the DNN takes a vectorial representation of a word and *learns the relevant feature for the processing level on the fly*

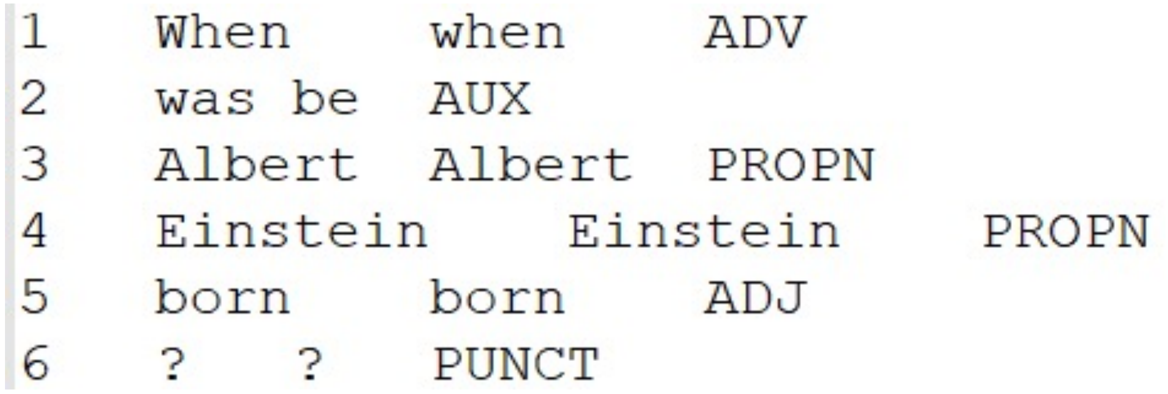
Case study 1: HMM POS tagging

- Hidden Markov Models POS tagging works with the following equation (the TnT POS tagger):

$$\underset{t_1 \ldots t_T}{\operatorname{argmax}} \left[\prod_{i=1}^{T} P(t_i | t_{i-1}, t_{i-2}) P(w_i | t_i) \right]$$

- Words w_i are seen in the training corpus, and tags $t_{i\text{-}2}$, $t_{i\text{-}1}$ and t_i form trigrams whose training corpus frequencies help compute the conditional probabilities
- *For unknown words, $P(w_i\,|\,t_i)$ is computed for inflections, e.g., *s words can be plural nouns or present tense verbs, such as "cars" or "runs".*
- For best results, only valid inflections should be considered.

Training corpus



```
1	When	when	ADV
2	was be	AUX
3	Albert	Albert	PROPN
4	Einstein	Einstein	PROPN
5	born	born	ADJ
6	?	?	PUNCT
```

- f(ADV, AUX, PROPN) = 1
- f(AUX, PROPN, PROPN) = 1
- …
- P(PROPN|AUX, PROPN) =
  f(AUX, PROPN, PROPN) / f(AUX, PROPN)
- f(when, ADV) = 1
- P(when|ADV) = f(when, ADV) / f(ADV) = 1

HMM POS tagging “human intervention”

- The emission probability $P(w_i | t_i)$ can be made more complex so that it can incorporate linguistic evidence from the context of the word at ith position:
  - Word has a special casing, e.g., “Python3”, “HelpNet”, etc.
  - If word can be a finite verb but we already have a finite verb previously assigned, choose some other possible tag
  - If a word can be either an adjective or a past tense verb (e.g., “included”) but it occurs in a likely noun phase, choose the adjective reading
- Use only valid inflections for the language, e.g., for English “-s”, “-ing”, “-ed”, “-ies”, etc.

Case study 2: BiLSTM + CRF POS tagging

- Every input word w_i is represented as a real-valued vector that is obtained from the concatenation of:
  - Its character-level embedding computed with CharWNN (https://arxiv.org/abs/1505.05008)
  - Its word embedding computed with any conventional method (e.g., FastText available at https://fasttext.cc/docs/en/unsupervised-tutorial.html)
- *The DNN can learn to POS tag without any other feature engineering effort!*
- Ma and Hovy (2016) show that the POS tagging accuracy is 97.55%, compared to 96.46% of TnT.
  - Xuezhe Ma and Eduard Hovy. (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. arXiv:1603.01354 [cs.LG]

CharWNN

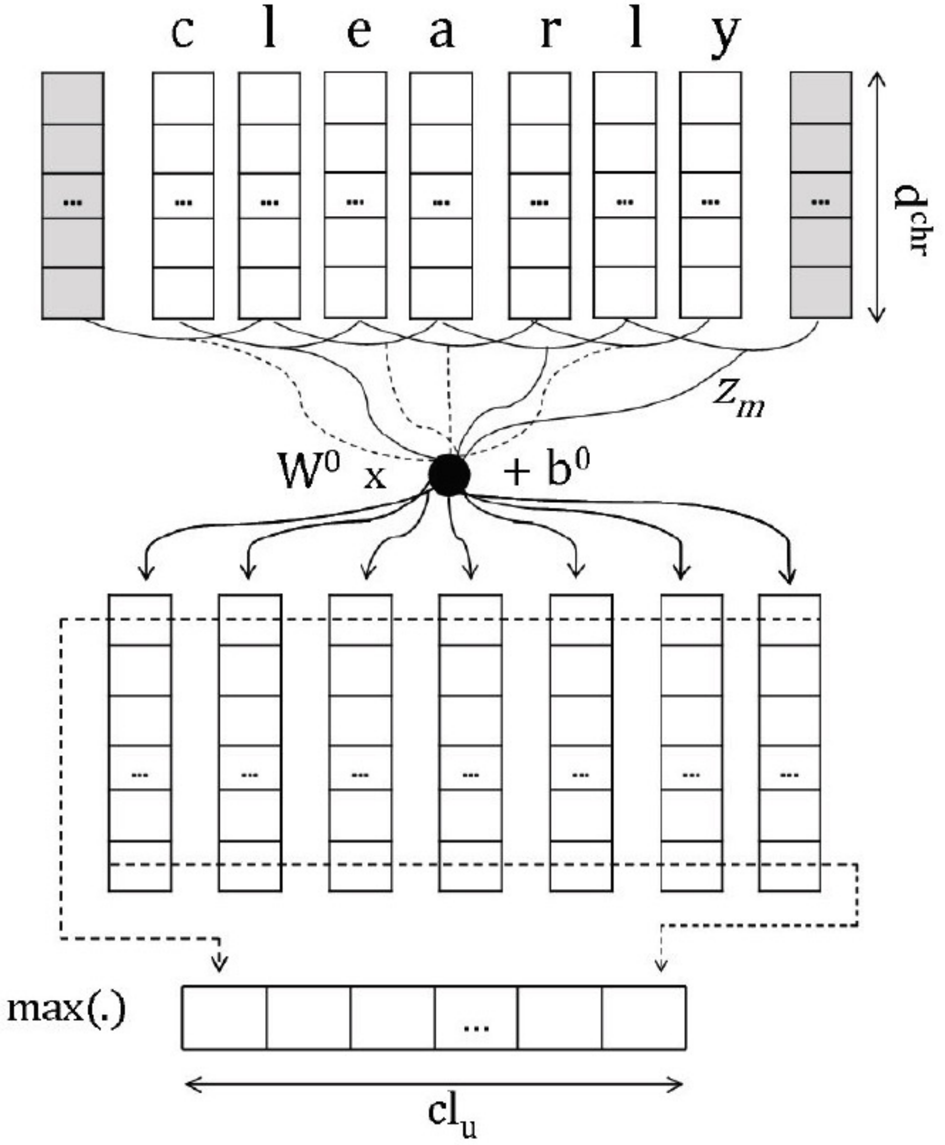
http://proceedings.mlr.press/v32/santos14.pdf

$$r^{chr} = W^{chr} \cdot v^{c}$$

$$x = \left\{ r_1^{chr}, r_2^{chr}, \ldots, r_M^{chr} \right\}$$

$$z_m = \left(r_{m-(k^{chr}-1)/2}^{chr}, \ldots, r_{m+(k^{chr}-1)/2}^{chr} \right)^T$$

$$\left[r^{wch}\right]_j = max \left[W^0 z_m + b_0\right]_j, 1 < m < M$$



BiLSTM + CRF

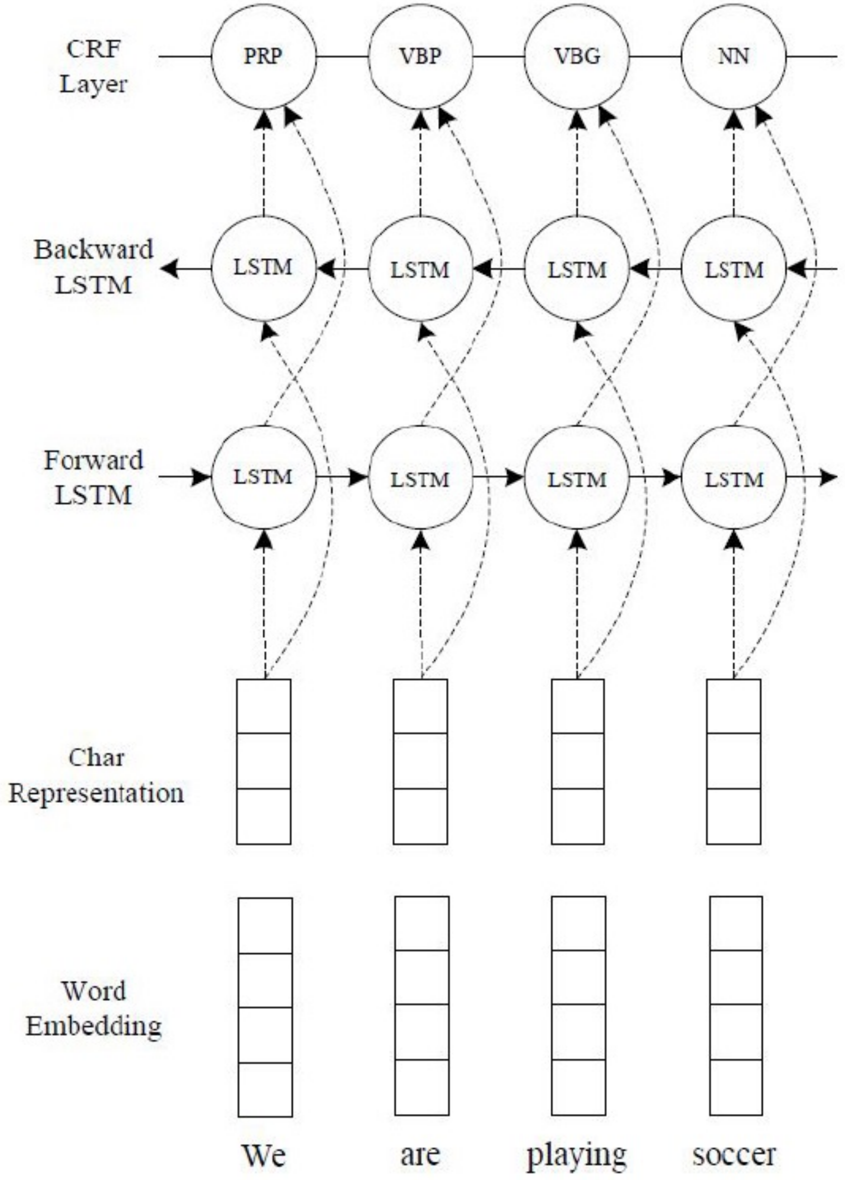
- Ma and Hovy (2016)

$$z = \{z_1, z_2, \ldots, z_n\}$$

$$y = \{y_1, y_2, \ldots, y_n\}$$

$$p(y|z; W, b) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, z_i)}{\sum_{y' \in Y(z)} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, z_i)}$$

$$\psi_i(y_{i-1}, y_i, z_i) = e^{W^T_{y_{i-1}, y_i} \cdot z_i + b_{y_{i-1}, y_i}}$$



Open-source text processing DNNs

- NLP-Cube: https://github.com/adobe/NLP-Cube
  - Has models for all languages in UD (Romanian included)
  - Python3 module and API REST web service server
  - Technology: BiLSTM with a final Attention layer
- UD-Pipe (1, 2 and 3): https://ufal.mff.cuni.cz/udpipe
  - Has models for all languages in UD (Romanian included)
  - Python3 module (support for other programming languages)
- Stanza: https://stanfordnlp.github.io/stanza/
  - Has to be trained for Romanian (3-4 days)
  - Python3 module
  - Best performing parsing system at CoNLL 2018 Dependency Parsing Shared Task
  - Technology: BiLSTM and Deep Biaffine transformations with explicit modeling of dependency relations orientation (https://arxiv.org/abs/1611.01734)

TEPROLIN

- It is a project, in which ICIA and “Universitatea Alexandru Ioan Cuza” were partners, that built a configurable and scalable text processing platform for bimodal corpora in Romanian
- Entry point: https://relate.racai.ro/ for small tests (demo)
- TEPROLIN (https://github.com/racai-ai/TEPROLIN) is also exposed as a Docker container:
  - Just pull it with `docker pull raduion/teprolin:1.1`, run it, and point your browser at http://localhost:5000/ for REST API docs

Learning resources

- Coursera.org (best learning portal, if you ask me)
  - https://www.coursera.org/specializations/deep-learning
  - About 50 US Dollars/month but you will learn everything you need to know to kick start your DL career
- Papers with code:
  - https://paperswithcode.com/area/natural-language-processing
  - It is the most up-to-date proof that NLP is now done with DNNs
  - You can choose your processing level (e.g., POS tagging, dependency parsing) and see the top performers on standard test sets
  - The result is accompanied by the scientific paper and most of them have public GitHub repositories

What to do next

- There is no better way to learn than experimenting:
  - Take any DL course you can find
  - Install your favorite DL toolkit:
    - TensorFlow (https://www.tensorflow.org/) – use Keras for an easier start
    - PyTorch (https://pytorch.org/)
    - MXNet (https://mxnet.apache.org/)
  - Pick a repository from “Papers with code” and reproduce their result
  - See if you can improve it for Romanian
- If you want to take part in the “DL for Romanian” research program that I lead at ICIA, let me know (`radu@racai.ro`)