Human Language Technologies

Federico Matteoni

A.A. 2021/22

Index

0.1 Introduction

Prof. Giuseppe Attardi
Prerequisites are: proficiency in Python, basic probability and statistics, calculus and linear algebra and notions of machine learning.

What will we learn Understanding of and ability to use effective modern methods for **Natural Language Processing**. From traditional methods to current advanced ones like RNN, Attentions...
Understanding the difficulties in dealing with NL and the capabilities of current technologies, with experience with **modern tools** and aiming towards the ability to build systems for some major NLP tasks: word similarities, parsing, machine translation, entity recognition, question answering, sentiment analysis, dialogue system...

Books Speech and Language Processing (Jurafsky, Martin), Deep Learning (Goodfellow, Bengio, Courville), Natural Language Processing in Python (Bird, Klein, Loper)

Exam Project (alone or team of 2-3 people) with the aim to experiment with techniques in a realistic setting using data from competitions (Kaggle, CoNLL, SemEval, Evalita...). The topic will be proposed by the team or chosen from a list of suggestions.

Experimental Approach

1. Formulate hypothesis
2. Implement technique
3. Train and test
4. Apply evaluation metric
5. If not improved:

   Perform error analysis

   Revise hypothesis
6. Repeat!

Motivations Language is the most distinctive feature of human intelligence, **it shapes thought**. Emulating language capabilities is a scientific challenge, a **keystone for intelligent systems** (see: Turing test)

Structured vs unstructured data The largest amount of information shared with each other is unstructured, primarily text. Information is mostly communicated by e-mails, reports, articles, conversations, media... and attempts to turn text to structured (HTML) or microformat only scratched the surface.
Problems: requires universal agreed **ontologies** and additional effort. Entity linking attempts to provide a bridge.

0.2 State of the Art

Early History During 1950s, up until AI winter.

Resurgence in the 1990s Thanks to statistical methods, novelty, to study language. Challenges arise: NIST, Netflix, DARPA Grand Challenge...
During 2010s: deep learning, neural machine translation...

Statistical Machine Learning Supervised training with **annotated** documents.
The paradigm is composed of the following:

Training set $\{x_i, y_y\}$

Representation: choose a set of features to represent data $x \mapsto \phi(x) \in R^D$

Model: choose an hypothesis function to compute $f(x) = F_\Theta(\phi(x))$

Evaluation: define the cost function on error with respect to examples $J(\Theta) = \sum_i (f(x_i) - y_i)^2$

Optimization: find parameters Θ that minimize $J(\Theta)$

It's a generic method, applicable to any problem.

Traditional Supervised Learning Approach Freed us from devising algorithms and rules, requiring the creation of annotated training sets and imposing the tyranny of feature engineering.
Standard approach for each new problem:

Gather as much labeled data as one can

Throw a bunch of models at it

Pick the best

Spend hours hand engineering some features or doing feature selection/dimensionality reduction

Rinse and repeat

Technological Breakthroughs Improved ML techniques but also large annotated datasets and more computing power, provided by GPUs and dedicated ML processors (like the TPU by Google).
ML exploits parallelism: stochastic gradient descent can be parallelized (asynchronous stochastic gradient descent). No need to protect shared memory access, and low (half, single) precision is enough.

Deep Learning Approach Was a big breakthrough.

Design a model architecture

Define a loss function

Run the network letting the parameters and the data representations **self-organize** as to minimize the loss

End-to-end learning: no intermediate stages nor representation

Feature representation Use a vector with each entry representing a feature of the domain element
Deep Learning represents data as vectors. Images are vectors (matrices), but words? **Word Embeddings**: transform a word into a vector of hundreds of dimensions capturing many subtle aspects of its meaning. Computed by the means of **language model**.
From a discrete to distributed representation. Words meaning are dense vectors of weights in a high dimensional space, with algebraic properties.
Background: philosophy, linguistics and statistics ML (feature vectors).

Language Model Statistical model which tells the probability that a word comes after a given word in a sentence.

Dealing with Sentences A sentence is a sequence of words: build a representation of a sequence from those of its words (compositional hypothesis). Sequence to sequence models.
Is there more structure in a sentence than a sequence of words? In many cases, tools forgets information when translating sentences into sequences of words, discarding much of the structure.

0.3 Language Modeling

Probabilistic Language Model The goal is to assign a probability to a sentence.

Machine Translation: $P(\text{high winds tonight}) > P(\text{large winds tonight})$

Spell Correction: $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$

Speech Recognition: $P(\text{I saw a van}) > P(\text{eye saw a van})$

Language Identification: s from unknown language (italian or english) and Lita, Leng language models for italian and english $\Rightarrow L\text{ita}(s) > L\text{eng}(s)$

Summarization, question answering...

We want to compute

$P(W) = P(w_1, w_2, \dots, w_n)$ the probability of a sequence

$P(w_4 \mid w_1, w_2, w_3, w_4)$ the probability of a word given some previous words

The model that computes that is called the **language model**

Markov Model and N-Grams Simplify the assumption: the probability of a word given all the previous is the same of the probability of that word given just few (one, two...) previous words. So $P(w_i \,|\, w_{i-}, \dots, w_1) = P(w_i \,|\, w_{i-1})$ (First order Markov chain).
With a N-**gram**: $P(w_n \mid w_1^{n-1}) \simeq P(w_n \mid w_{n-N+1}^{n-1})$
In general it's insufficient: language has **long distance dependencies**, but we can often get away with N-gram models. For example:

"The **man** next to the large oak tree near the grocery store on the corner **is** tall."

"The **men** next to the large oak tree near the grocery store on the corner **are** tall."

Or even semantic dependencies:

"The **bird** next to the large oak tree near the grocery store on the corner **flies** rapidly."

"The **man** next to the large oak tree near the grocery store on the corner **talks** rapidly."

So more complex models are needed to handle such dependencies.

Maximum likelihood estimate

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{\text{count}(w_{n-N+1}^{n-1}, w_n)}{\text{count}(w_{n-N+1}^{n-1})}$$

Maximum because it's the one that maximize $P(\text{Training set} \mid \text{Model})$

Shannon Visualization Method Generate random sentences:

Choose a random bigram $(\langle s \rangle, w)$ according to its probability

Choose a random bigram (w, x) according to its probability

Repeat until we pick $\langle /s \rangle$