Machine Learning Techniques for Text

Module 6: Teaching Machines to Translate

Dr. Nikos Tsourakis

Course outline

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Module 0: Python Crash Course
- Module 1: Intro to Machine Learning
- Module 2: Detecting Spam Emails
- Module 3: Classifying Topics of Newsgroup Posts
- Module 4: Extracting Sentiments from Product Reviews
- Module 5: Recommending Music Titles
- **Module 6: Teaching Machines to Translate**
- Module 7: Summarizing Wikipedia Articles
- Module 8: Detecting Hateful and Offensive Language
- Module 9: Generating Text in Chatbots
- Module 10: Clustering Speech-to-Text Transcriptions

Overview

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Systems for unhindered translation between languages have not been fully realized
- This shortcoming comes as no surprise, given human languages' fluidity, inherent ambiguity, and flexibility
- This module seeks to present the different methods for machine translation and, at the same time, enhance your skillset with many standard techniques for NLP
    - The differences in the methods presented are an excellent opportunity to contrast the design philosophy of top-down and bottom-up approaches
    - We will focus on the evolution of machine translation systems over the years in terms of four main category types
    - There will be a special focus on understanding complex architectures for sequence-to-sequence learning
    - Finally, we will present specific evaluation methods and metrics to assess the performance of relevant systems

Module objectives

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

After completing this module, you should be able to:

- Understanding machine translation
- Introducing rule-based machine translation
- Introducing statistical machine translation
- Introducing sequence-to-sequence learning
- Measuring translation performance

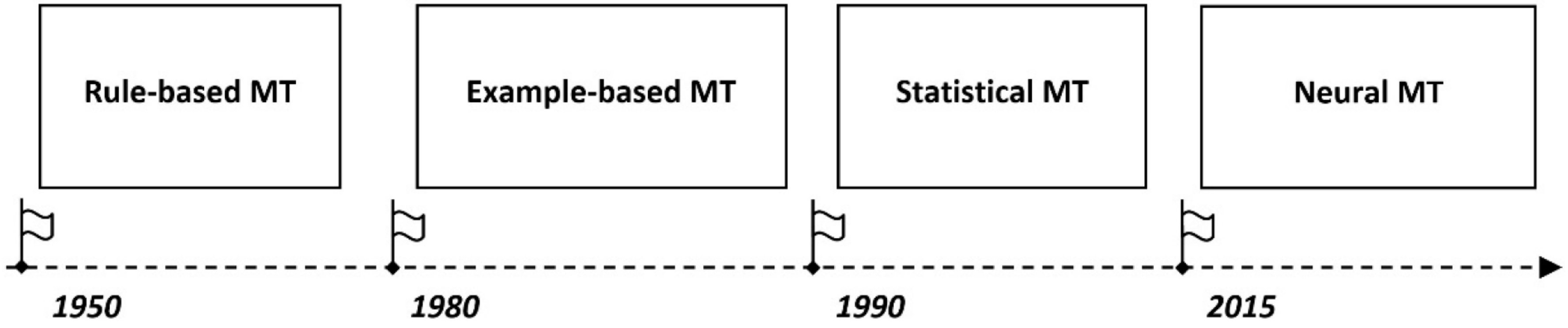
Machine Learning Techniques for Text

Section 1: Understanding machine translation

Introduction

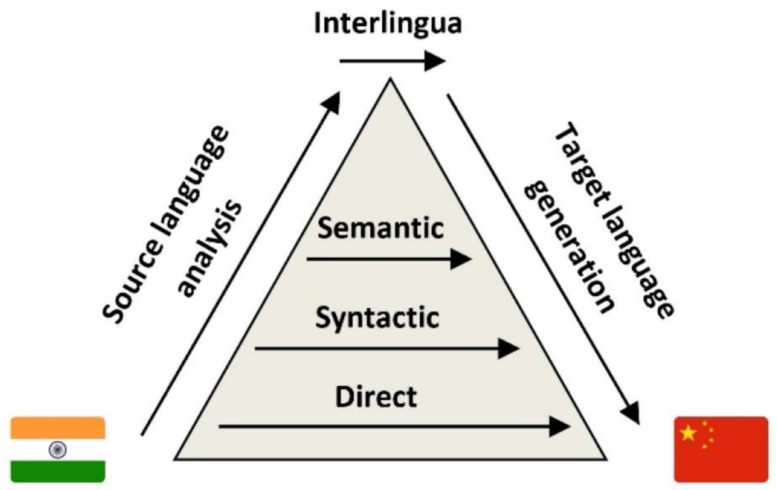
‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

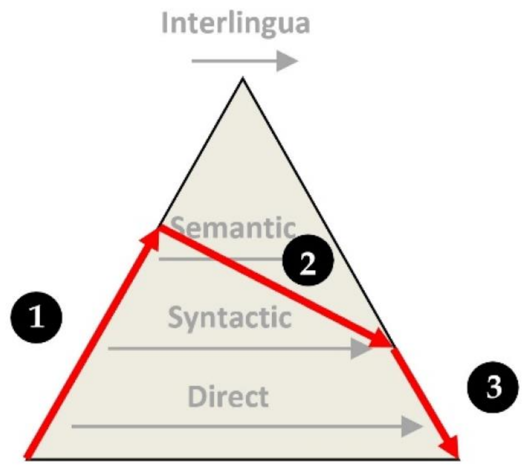
- ***Machine translation*** (MT) is the process automatically converting a piece of text from a source into a target language without human intervention
- MT demands the synergy of various emerging fields to address the peculiarities of human language
- The technology, however, is not new. In the 50s, it was part of the first computing applications



The Vauquois triangle

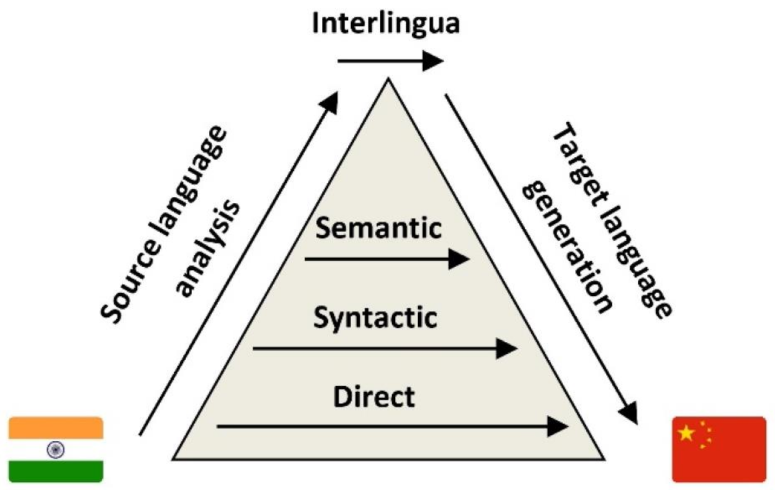
- The ***Vauquois triangle*** is a representation that acts as a roadmap for translating a source sentence into the correct target
- At a high level, the MT algorithm needs to perform three distinct steps, shown schematically in the plots

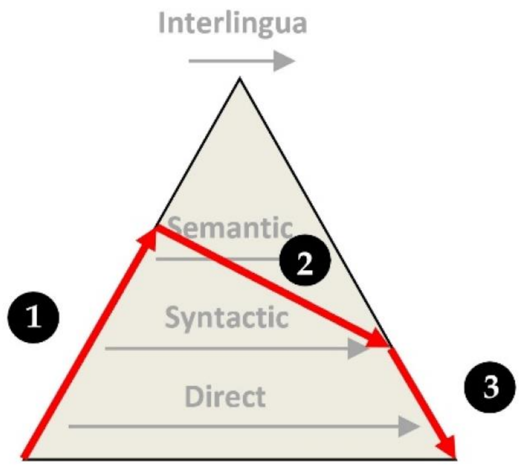




The Vauquois triangle

1. First, ascend the left-hand side of the triangle and create an internal representation of the source sentence (***Source language analysis***)
2. Then, traverse the triangle and transfer the source representation to the target one. The different levels inside the triangle signify different levels of analysis
3. Finally, descend to the right-hand side and generate the target sentence (***Target language generation***)





Machine Learning Techniques for Text

Section 2: Introducing MT techniques

Rule-based machine translation

- ***Rule-based machine translation*** (RBMT), aims to exploit linguistic information about the source and target languages
- RBMT techniques fall under the broad category of ***knowledge-based systems***, which mainly aim to capture the knowledge of human experts to solve complex problems

Source sentence (EN)		Target sentence (FR)
The sky is blue		***Le ciel est bleu***
The	$\rightarrow$	Le
sky	$\rightarrow$	ciel
is	$\rightarrow$	est
blue	$\rightarrow$	bleu

Rule-based machine translation

- ***Rule-based machine translation*** (RBMT), aims to exploit linguistic information about the source and target languages
- RBMT techniques fall under the broad category of ***knowledge-based systems***, which mainly aim to capture the knowledge of human experts to solve complex problems

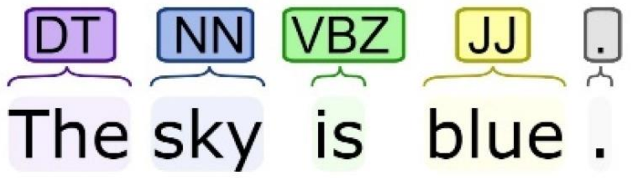
Source sentence (EN)		Target sentence (FR)
The sky is blue		***Le ciel est bleu***
The	→	Le
sky	→	ciel
is	→	est
blue	→	bleu

⚠

A word-for-word translation rarely works in practice

Part-of-speech tagging

- The first task in the Vauquois triangle focuses on analyzing the source sentence
- We parse the source sentences to extract richer internal representations:
  - identify how words are combined to form constituents (linguistic parts of a larger sentence, phrase, or clause)
  - how words relate to other words, and other
- ***Part-of-speech*** (POS) tagging categorizes words in a piece of text with a particular tag, based on their definition and context



Context-free grammar

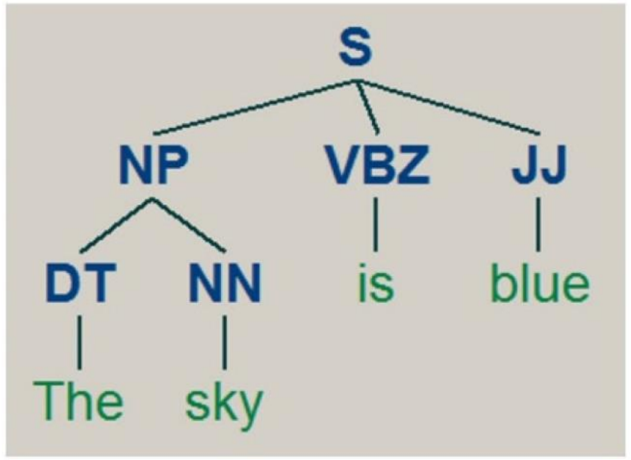
- A more insightful representation is the ***parse tree***, which provides information on how a word joins with the other words of the sentence
- We can quickly build our parse tree in Python using a ***context-free grammar*** (CFG)
  - A CFG is a set of recursive rules used to parse or generate patterns of strings

```
analysis_grammar = nltk.CFG.fromstring("""
    S -> NP VBZ JJ
    NP -> DT NN
    DT -> 'The'
    NN -> 'sky'
    VBZ -> 'is'
    JJ -> 'blue'
    """)
```

Context-free grammar

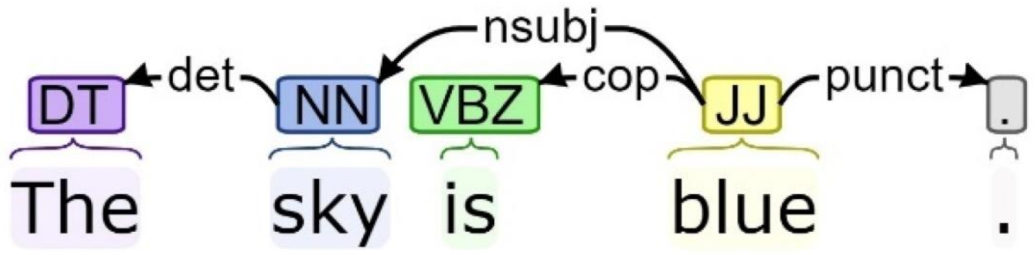
- A more insightful representation is the ***parse tree***, which provides information on how a word joins with the other words of the sentence
- We can quickly build our parse tree in Python using a ***context-free grammar*** (CFG)
  - A CFG is a set of recursive rules used to parse or generate patterns of strings

```
analysis_grammar = nltk.CFG.fromstring("""
    S -> NP VBZ JJ
    NP -> DT NN
    DT -> 'The'
    NN -> 'sky'
    VBZ -> 'is'
    JJ -> 'blue'
    """)
```



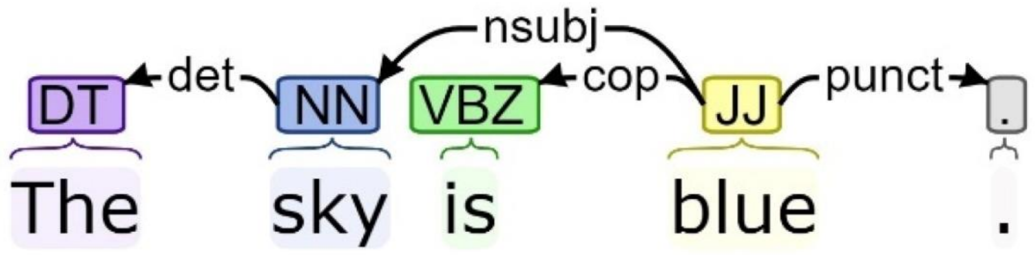
Dependency grammar

- While CFG is concerned with how words and sequences of words combine to form constituents, a ***dependency grammar*** focuses on how words relate to other words
- Using a ***dependency parser***, we can analyze the grammatical structure of a sentence and obtain the relationships between words

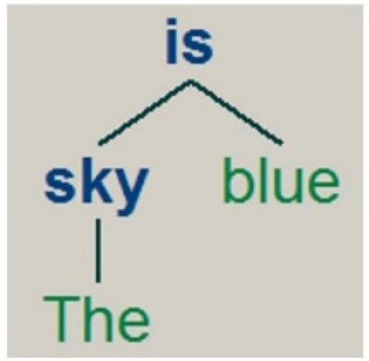


Dependency grammar

- While CFG is concerned with how words and sequences of words combine to form constituents, a ***dependency grammar*** focuses on how words relate to other words
- Using a ***dependency parser***, we can analyze the grammatical structure of a sentence and obtain the relationships between words



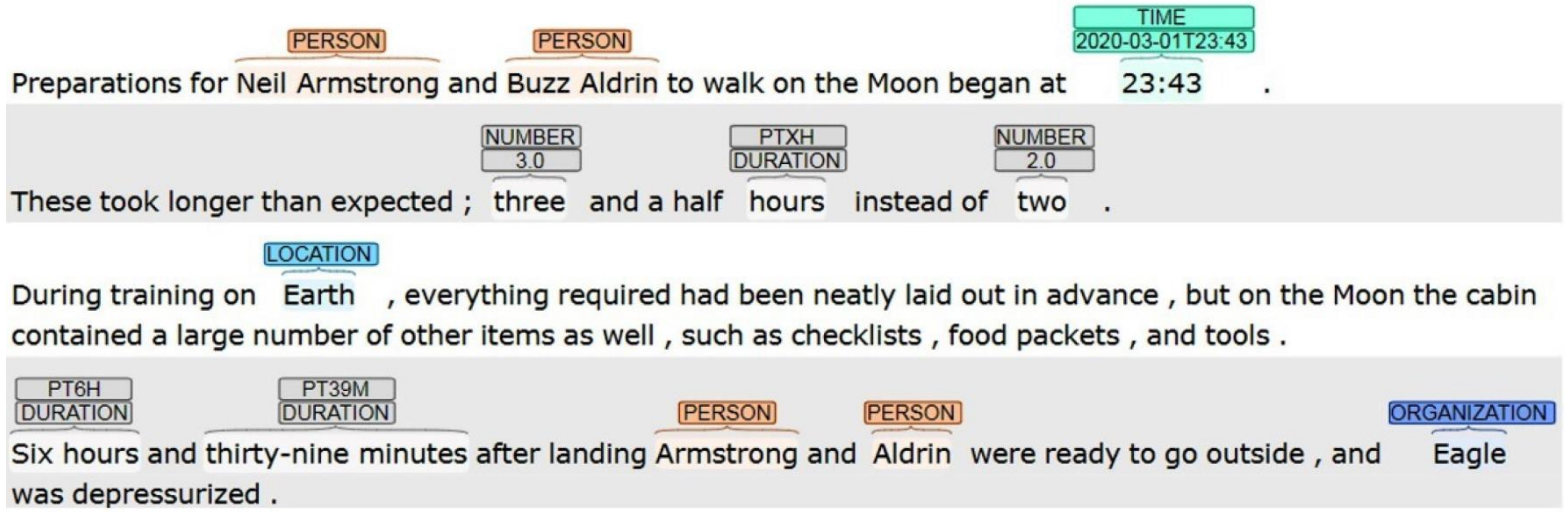
Dependency tree



Named-entity resolution

Machine Learning Techniques for Text

- During any translation task, we can encounter words in the source language that should not be translated into the target
  - E.g.: names of people, organizations, expressions of times, locations, and so
- ***Named-entity resolution*** (NER) seeks to locate and classify named entities in text into predefined categories



Preparations for Neil Armstrong and Buzz Aldrin to walk on the Moon began at 23:43 .

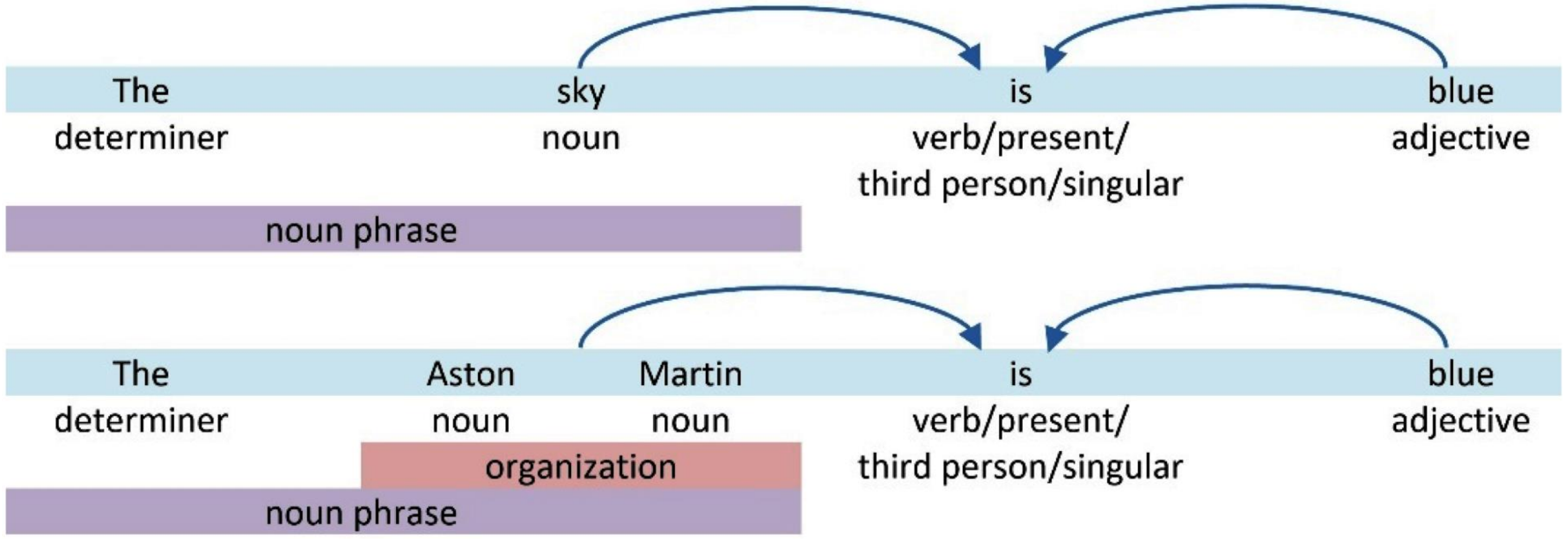
These took longer than expected ; three and a half hours instead of two .

During training on Earth , everything required had been neatly laid out in advance , but on the Moon the cabin contained a large number of other items as well , such as checklists , food packets , and tools .

Six hours and thirty-nine minutes after landing Armstrong and Aldrin were ready to go outside , and Eagle was depressurized .

Final source representation

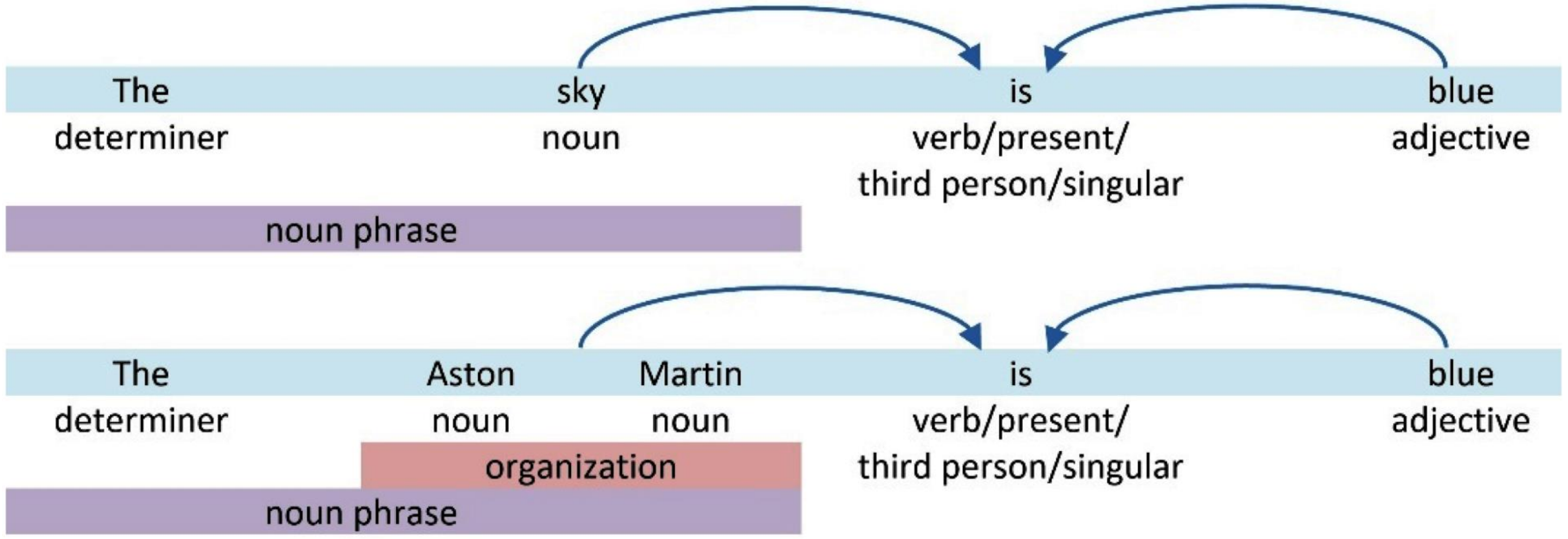
- All the different steps for analyzing a source sentence let us create the representations for two examples phrases

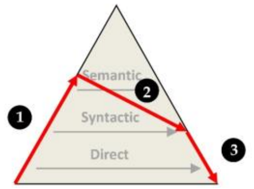


Final source representation

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- All the different steps for analyzing a source sentence let us create the representations for two examples phrases

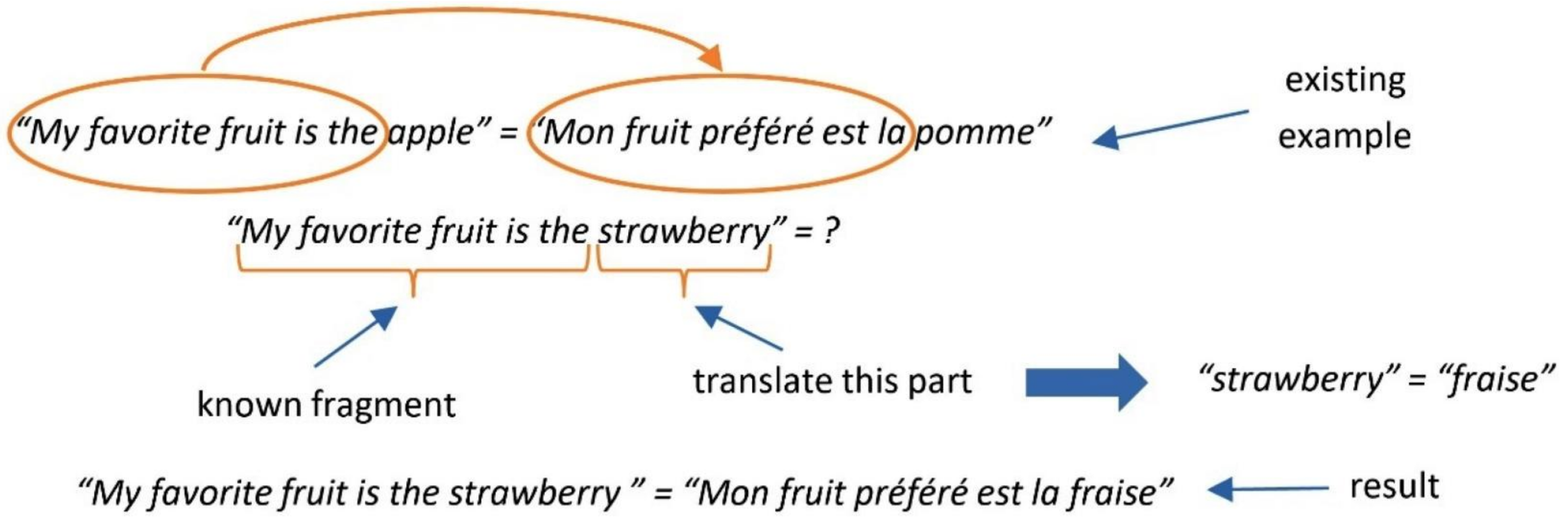




Need to create rules that can ***transfer*** the source representation to the target one and rules to ***generate*** the translation emitted

Example-based machine translation

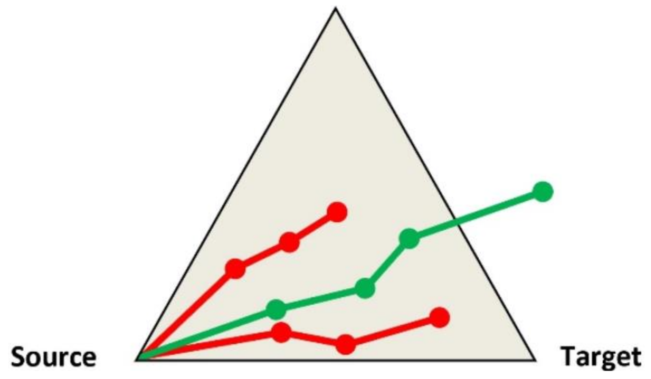
- In ***example-based machine translation*** (EBMT) we use a corpus of already-translated examples could serve as a model to base the translation task on
- Split the source into smaller fragments, and translate the pieces by analogy into previous examples
- Finally, recombine those translations into the target sentence


"My favorite fruit is the apple" = "Mon fruit préféré est la pomme"
existing example
"My favorite fruit is the strawberry" = ?
known fragment
translate this part
"strawberry" = "fraise"
"My favorite fruit is the strawberry" = "Mon fruit préféré est la fraise"
result

Statistical machine translation

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- RBMT techniques follow a top-down approach, and domain experts are required to create models that can replicate the data
- Conversely, data-driven approaches are bottom-up, and the data derives the model
- ***Statistical machine translation*** (SMT) involves exploiting models whose parameters are learned from bilingual text corpora
- SMT systems work on the assumption that every sentence in one language can be translated into any sentence in the target one
- The overarching goal is to find the most probable translation in each case
- Starting from the source sentence, different alternative paths are constructed
- Low-probability paths are pruned until we reach the most probable outcome



‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- MT techniques

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-06/machine-translation.ipynb

Machine Learning Techniques for Text

Section 3: Introducing sequence-to-sequence learning

Introduction

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

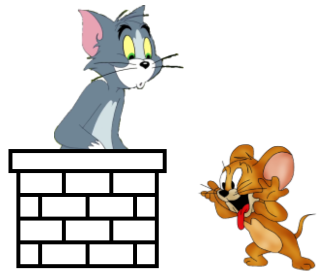
- Many kinds of problems in machine learning involve transforming an input sequence into an output one
- ***Sequence-to-sequence*** (seq2seq) learning has proven useful in applications that demand this transformation
- Seq2seq, pronounced as seek-to-seek, learning falls under the category of neural MT, and unlike solutions based on RBMT and SMT, no domain knowledge of the languages involved is necessary
- You can treat the translation problem as the association between input and output tokens of words or characters
- Moreover, the translation is end-to-end, which means that one model is required instead of many

Context persistence

packt
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- A common problem in seq2seq learning is ***context persistence***, especially when the sequences become too lengthy
- Why does this matter?
- Consider, for instance, the following:

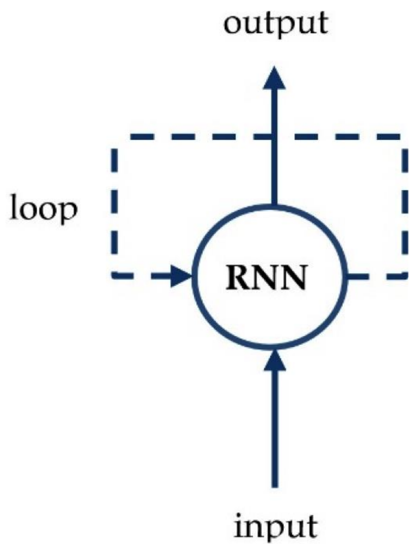
Tom and Jerry are playing outside. They like it a lot!



- In this case, we must read the first sentence to make sense of the word ***They*** in the second
- While reading this slide, you constantly do a similar task, keeping track of information far beyond adjacent sentences

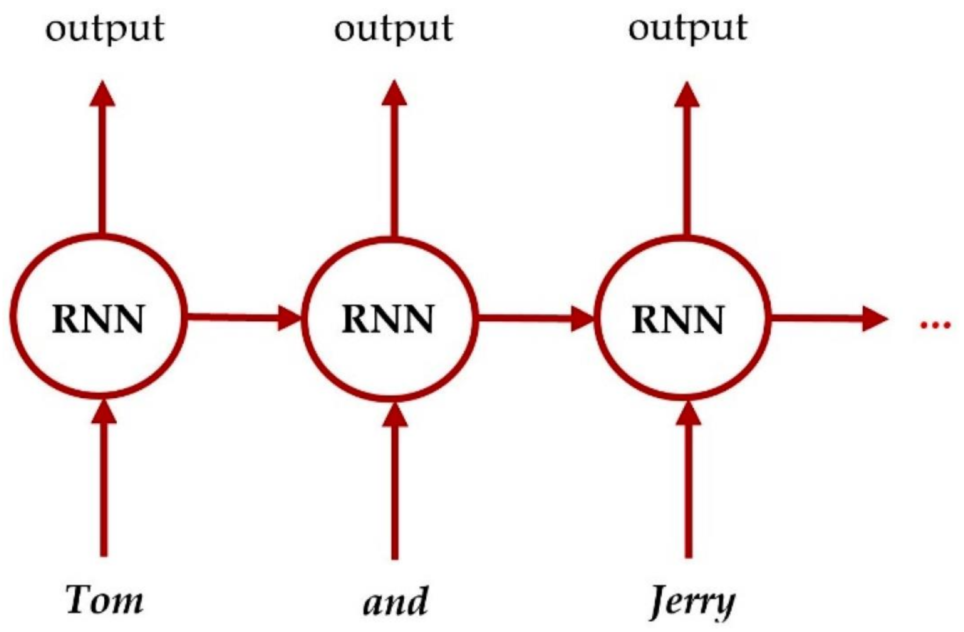
Recurrent neural networks

- Context is crucial for choosing the correct translation in an MT system
  - E.g.: the word ***back*** can be either a noun, verb, adjective, or adverb
- ***Recurrent neural networks*** (RNNs) alleviate this problem as they are suitable for sequential data and have loops for persisting information
- The main difference with a feed-forward neural network is the presence of a memory loop, thus the name recurrent
- RNNs have a memory to register all information that has been extracted so far
- The loop feeds the context back to the model at every timestep



Recurrent neural networks

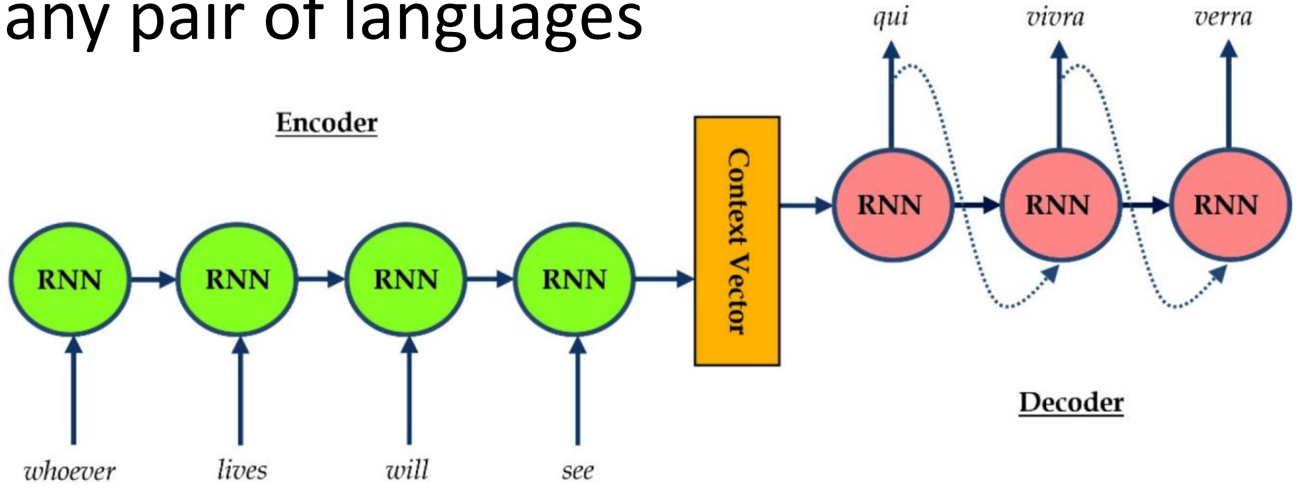
- During the first timestep, the RNN receives the word ***Tom*** as input without any context
- Then, the RNN produces some kind of output, depending on the task, and a context vector that is passed in the second timestep, along with the word ***and***
- This process repeats until the last word in the input sentence is consumed
- By the end of this process, the propagated message should contain enough information to associate ***Tom*** and ***Jerry*** with the word ***They***



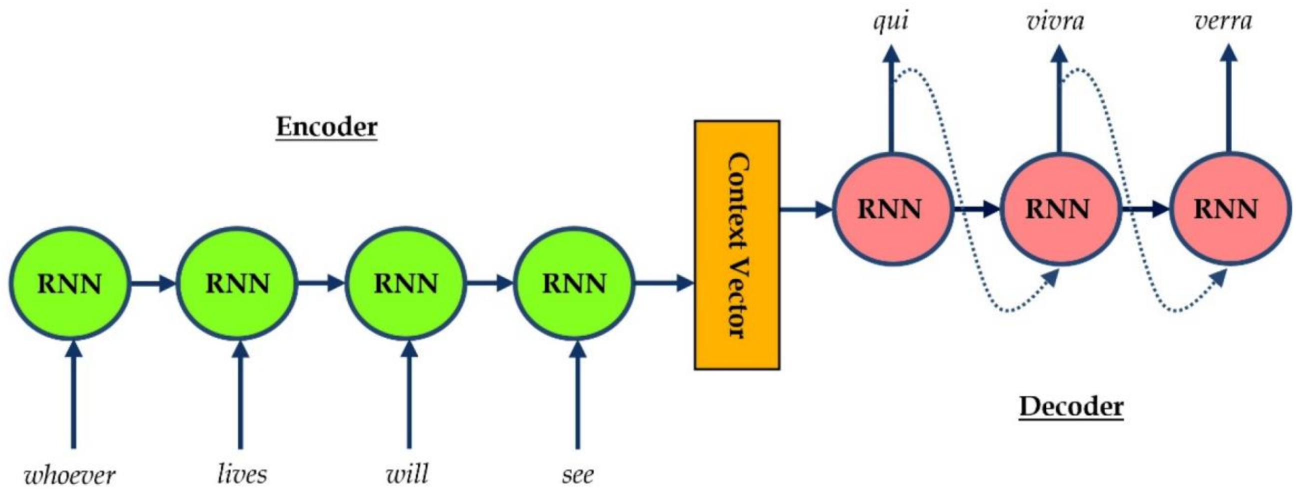
The encoder/decoder architecture

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- We model the source and target sequences with a separate recurrent network that can extract the necessary dependencies
- The first network encodes the source sentence, hence the name ***encoder***, while the second network decodes the input into the target language, hence the name ***decoder***
- Gluing these networks together creates an end-to-end translation pipeline for any pair of languages



The encoder/decoder architecture



- The ***encoder*** (left part) consumes words sequentially from the input sentence. The first RNN cell processes the first source word and emits a hidden state that is fed to the subsequent cell
- The ***context vector*** (middle part) is essentially the final hidden state of the encoder. It aims to encapsulate the information from all input elements
- The ***decoder*** (right part) predicts a word as output (translation) at each timestep. The first RNN cell utilizes the context vector to produce an output and a hidden state. Both are fed to the subsequent cell, which repeats similar processing

Long short-term memory units

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Although, in theory, RNNs can handle long-term dependencies, they can effectively process sequences with a length of less than 10 tokens
- The solution is to use another recurrent layer type called ***long short-term memory*** (LSTM) units
- The basic principle behind these units is analogous to RNN: at each timestep, use a token from the data sequence, along with input from the previous timestep
- The distinctive feature of LSTM, however, is its ability to remember information for long periods
- For this reason, the structure of an LSTM is much more complex compared to the structure of an RNN

Long short-term memory units

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

An analogy for LSTM

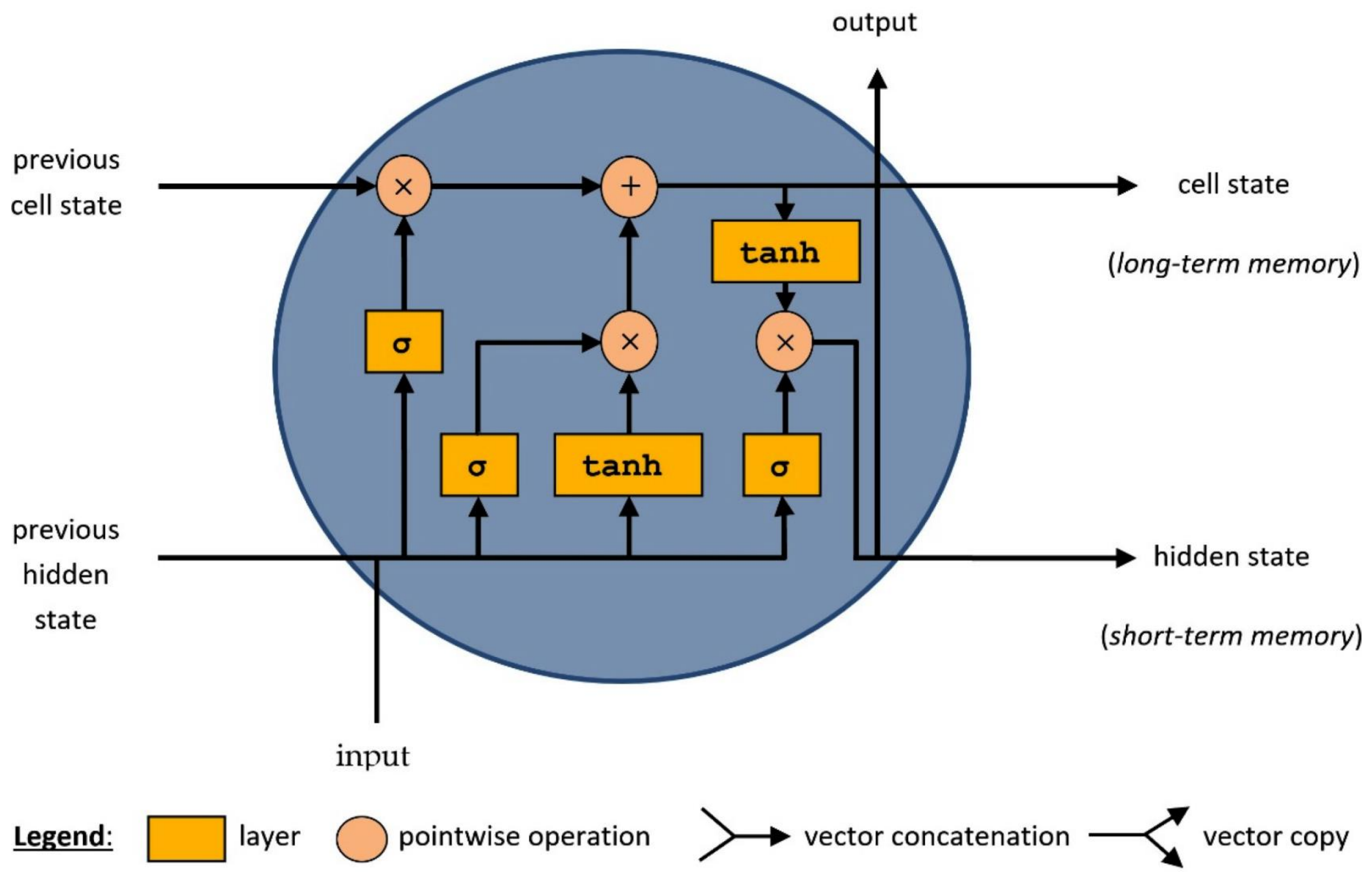
- Consider reading a book one word at a time, and for each word, you update your understanding based on the current word and what you've read so far
- The reader maintains an internal memory state that evolves as new words (or tokens) are processed
- As each word is read, this person considers the current input, updates its memory state, and decides what information to remember or forget
- This enables to capture dependencies and relationships in sequential data, like understanding the meaning of a word in the context of preceding words in a sentence

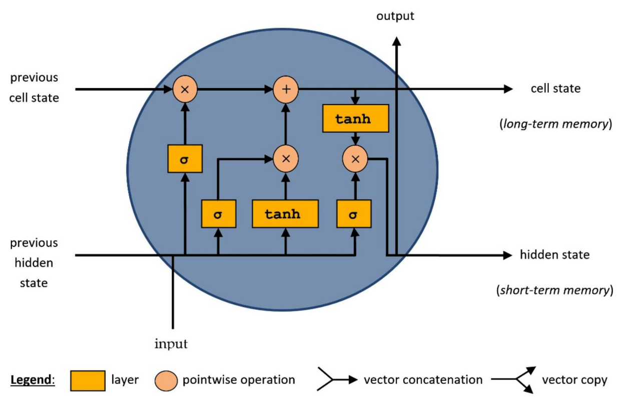
Long short-term memory units

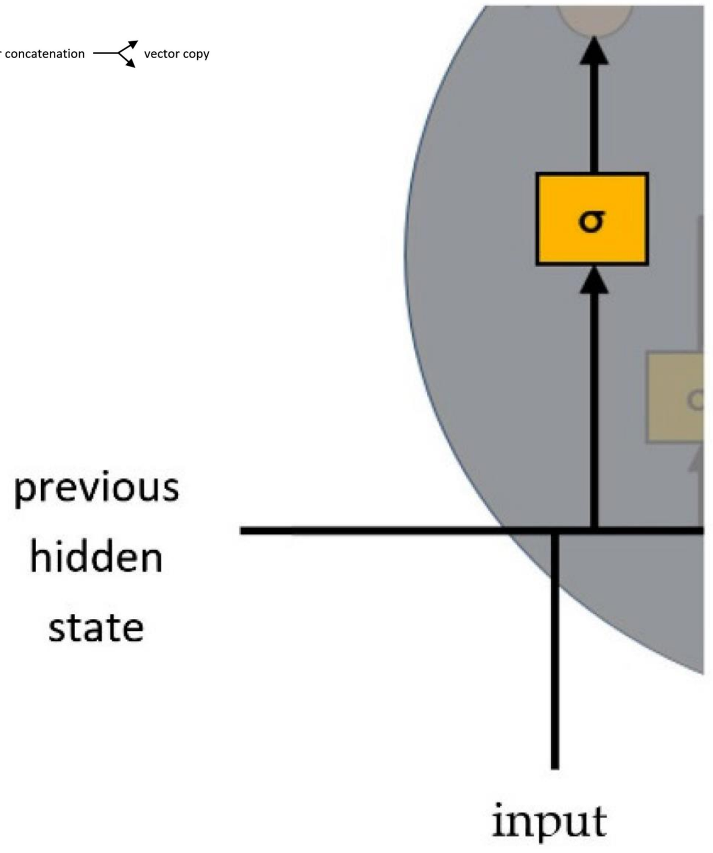
‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation





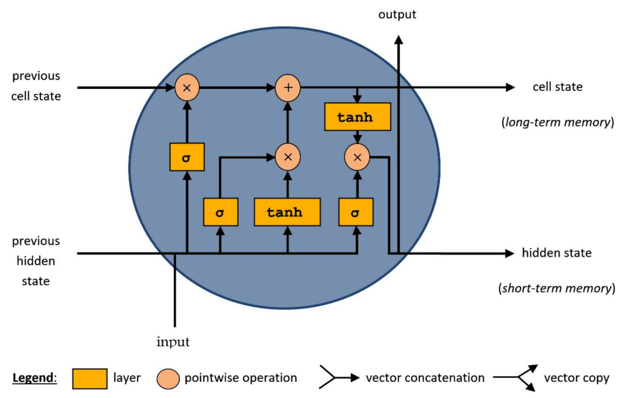


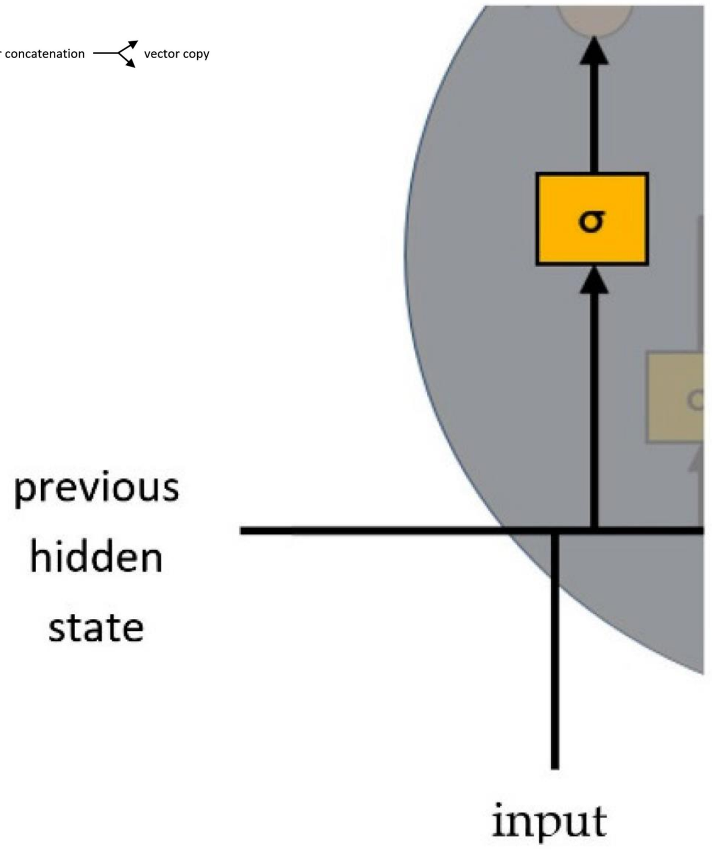
This step decides whether we should keep the information from the previous timestamp or forget it

The sigmoid layer (***σ***) outputs numbers between ***0*** and ***1*** and determines the amount of information to pass in the next step

An output of ***0*** from the sigmoid layer means that any input should be discarded entirely

Conversely, an output of ***1*** lets everything pass through





For example, if the network encounters a feminine name such as Alice, and after a few sentences, a male name such as Bob appears, it makes sense to forget the first person

Memorizing information in this way allows the proper gender pronoun to be applied later in the process

For this reason, the structure shown in this figure is called a ***forget gate***

Long short-term memory units

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- LSTM aims to keep track of both long and short-term dependencies in the input sequence
- As the learning process evolves, the network needs to memorize important information and throw away information that is not relevant anymore
- Essentially, it is a ***gate*** that allows or blocks information from passing through the next layer

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- MT techniques
- Sequence-to-sequence MT

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-06/seq2seq-LSTM.ipynb

Machine Learning Techniques for Text

Section 4: Measuring translation performance

Using standard metrics

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

. . . and in the rain your letters flow in the rivers

Reference: *“et sous la pluie tes lettres coulent dans les rivières”*

$$\textbf{Prediction}\text{: “}\underbrace{\text{sous la}}_{\text{2-gram}}\text{ pluie }\cancel{\text{les}}\text{ }\underbrace{\text{lettres coulent dans}}_{\text{3-gram}}\text{ }\cancel{\text{la rivière}}\text{”}$$

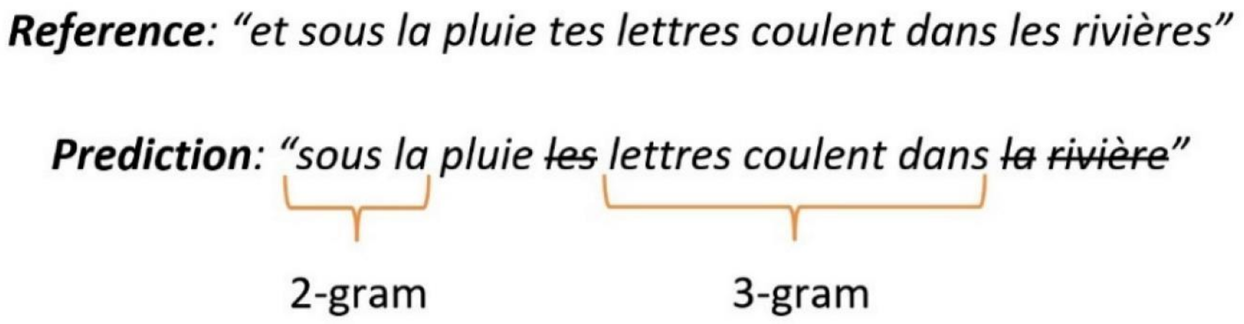
Using standard metrics

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

… and in the rain your letters flow in the rivers


Reference: *“et sous la pluie tes lettres coulent dans les rivières”*

Prediction: *“sous la pluie ~~les~~ lettres coulent dans ~~la rivière~~”*

- Here, we can make the following calculations:

$$Precision = \frac{correct\_words}{\#words\_prediction} = \frac{6}{9} = 67\%$$

$$Recall = \frac{correct\_words}{\#words\_reference} = \frac{6}{10} = 60\%$$

$$F\text{—}score = 2 * \frac{precision * recall}{precision + recall} = 2 * \frac{0.67 * 0.6}{0.67 + 0.6} = 63\%$$

Using standard metrics

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

. . . and in the rain your letters flow in the rivers

Reference: *"et sous la pluie tes lettres coulent dans les rivières"*

Prediction: *"$\underbrace{\text{sous la}}_{\text{2-gram}}$ pluie ~~les~~ lettres $\underbrace{\text{coulent dans}}_{\text{3-gram}}$ ~~la rivière~~"*

- Here, we can make the following calculations:

$$Precision = \frac{correct\_words}{\#words\_prediction} = \frac{6}{9} = 67\%$$

$$Recall = \frac{correct\_words}{\#words\_reference} = \frac{6}{10} = 60\%$$

$$F\text{–}score = 2 * \frac{precision * recall}{precision + recall} = 2 * \frac{0.67 * 0.6}{0.67 + 0.6} = 63\%$$

➢ But, how to interpret these values under the prism of MT?

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

The BLEU score

- A more appropriate metric for evaluating the quality of MT systems is the ***BiLingual Evaluation Understudy*** (BLEU) score
- Using BLEU, we compare the generated prediction to a reference sentence by counting matching n-grams in both cases
  - For each n-gram whose size is between 1 and 4 in the prediction, we count the number of times they appear in the reference
  - Notice that the order of the n-grams does not play any role
  - A brevity penalty is also added to the score to prevent very short candidates from receiving too high BLEU values
  - This way, predictions closer to the length of the reference translation get a higher score

The BLEU score

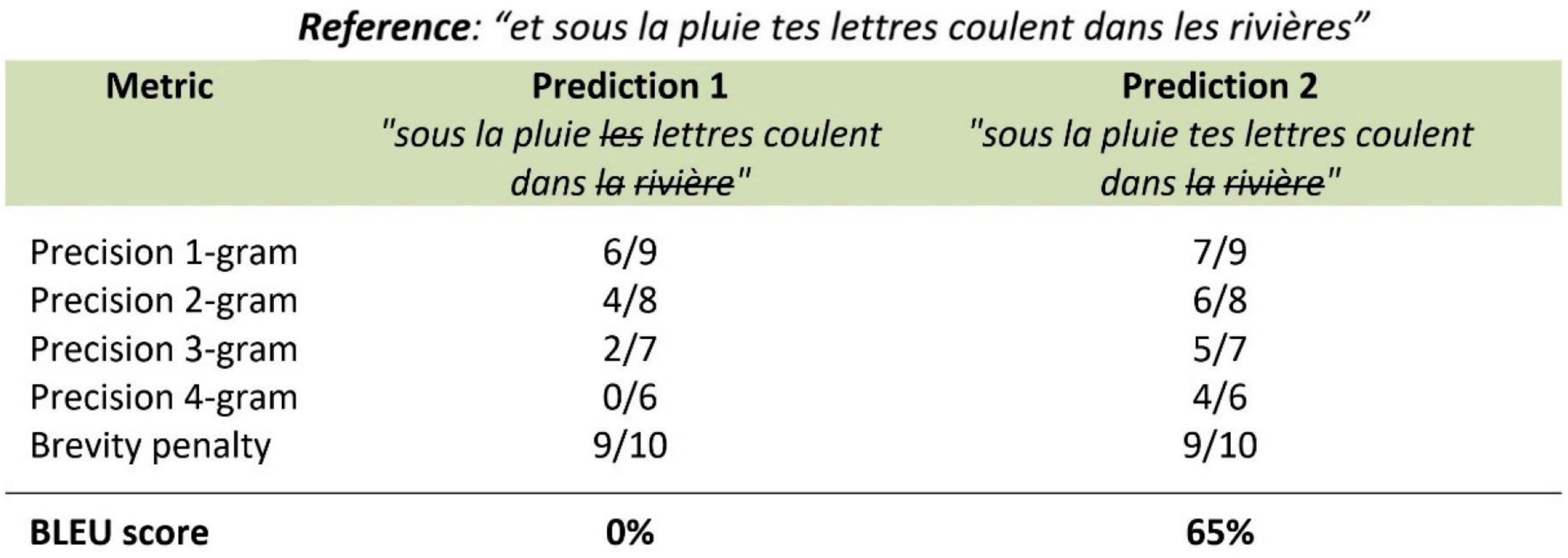
- The formula of BLEU is as follows:

$$BLEU = \min(1, \frac{\#words\_prediction}{\#words\_reference})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

- A score of ***1*** indicates a perfect match, while a score of ***0*** indicates a perfect mismatch
- The benefits of BLEU are that it is easy to apply and understand and correlates well with human evaluation
- The more common patterns that are found in both the prediction and reference, the more confident we are about the translation
- To compare two different MT models, we calculate BLEU using a large corpus of annotated examples and pick the one with the highest score

The BLEU score

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



Reference: *“et sous la pluie tes lettres coulent dans les rivières”*

Metric	**Prediction 1** *"sous la pluie ~~les~~ lettres coulent dans ~~la rivière~~"*	**Prediction 2** *"sous la pluie tes lettres coulent dans ~~la rivière~~"*
Precision 1-gram	6/9	7/9
Precision 2-gram	4/8	6/8
Precision 3-gram	2/7	5/7
Precision 4-gram	0/6	4/6
Brevity penalty	9/10	9/10
BLEU score	**0%**	**65%**

- ***Prediction 2*** receives a higher score and is preferable compared to ***Prediction 1***
- Notice that the absence of one common 4-gram in the first case is sufficient to yield a score of 0%

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- MT techniques
- Sequence-to-sequence MT
- Performance

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-06/seq2seq-LSTM.ipynb

Key takeaways

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

1ST EDITION

Text preprocessing

- Part-of-speech tagging
- Parse trees
- Name Entity Resolution

ML algorithms & models

- Rule-based MT
- Example-based MT
- Statistical MT
- Recurrent Neural Networks

Performance metrics

- BLEU score

Machine Learning Techniques for Text

Questions?