IRT SAINT EXUPÉRY

NLP to LLMs

Master ValDom

NLP to LLMs

fanny.jourdan@irt-saintexupery.com

mouhcine.mendil@irt-saintexupery.com

joseba.dalmau@irt-saintexupery.com

Fanny JOURDAN

CM1 : NLP tasks and preprocessing
CM2 : Text vectorization

Mouhcine MENDIL

CM3 : Classical ML techniques

TP 1 + TP2

Joseba DALMAU

CM4 : Classical DL techniques
CM5 : Transformers
CM6 : Feature space

TP3 + TP4 + TP5

NLP to LLMs

Written Exam: March 3, 14h00-15h45
MCQs + course questions + exercises

- The students will be allowed a single A4 two-sided handwritten sheet of paper with the notes they deem useful for the exam.
- The instructors will provide the students with sample exercises so that they can train themselves for the exam.

Ressources

GitHub
https://github.com/jdalch/Valdom-NLP2LLM

FRENCH INSTITUTES OF TECHNOLOGY

NLP Task and Preprocessing

Master ValDom – 07/01/2025

Summary

NLP Tasks

Text Classification

Text Classification

Goal: Assign a single label to an entire text.

Process:

- The input text is transformed into numerical data (e.g., embeddings or tokenized vectors).
- The model processes the input as a whole to predict the most probable label (e.g., "positive," "negative," or "neutral").
- The final prediction corresponds to the label with the highest probability.

Key Insight:
The model considers the entire input text at once to make its decision.

Sentiment Analysis

Goal : Determine whether a text expresses a positive, negative, or neutral opinion.

Example :

- *Text*: “I love this product!”
- *Output*: “Positive feeling”

Language Detection

Goal : Identify the language of a text.

Example :

- *Text*: “Hello, how are you?”
- *Output*: “Language: French”

Topic Classification

Goal : Identify a text's topic or theme.

Example :

- *Text*: “PSG won the match last night.”
- *Output*: “Theme: Sport”

Grammaticality Classification

Goal : Identify whether a sentence is grammatically correct.

Example :

- *Text*: “The cat eats mouse one.”
- *Output*: “Incorrect sentence”

Text Classification (general)

Goal : Assign a general category to a text (binary, multi-class or multi-label).

Example :

- *Binary classification*: Spam or non-spam in e-mails.
- *Multi-class classification*: Article categories (sport, politics, science).
- *Multi-tag classification*: Assign multiple tags (e.g. “politics” and “environment”).

NLP Tasks

Information Extraction

Information Extraction

Goal: Assign a label to each token in a sequence.

Process:

- The input text is split into smaller units (tokens).
- For each token, the model predicts a label (e.g., "B-PER" for the beginning of a person's name, or "O" for outside any entity).
- Labels are combined to extract structured information (e.g., "John Doe" → person name).

Key Insight:
The model processes each token individually, but uses context from the entire sentence for predictions.

Named Entity Recognition - NER

Goal : Identify specific entities in a text and classify them (People, Places, Organizations, etc.).

Example :

- *Text*: “Marie left for Paris yesterday.”
- *Output*: {“Marie” : Person, “Paris” : Place, “yesterday” : Date}

Part-of-Speech (POS) Tagging

Goal : Assign a grammatical class to each word (verb, noun, adjective, etc.).

Example :

- *Text*: “The cat eats a mouse.”
- *Output*: {“The”: Determiner, “cat”: Noun, “eats”: Verb, “mouse”: Noun}

Relation Extraction

Goal : Identify and classify relationships between entities in a text.

Example :

- *Text*: “Marie works at Google.”
- *Output*: {“Marie”, “employee”, “Google”}

NLP Tasks

Text Generation

Text Generation

Goal: Generate coherent text one token at a time.

Process:

- The model predicts the next token based on the input and previously generated tokens.
- At each step, it selects the most probable token from the model's vocabulary.
- This process is repeated recursively until the text is complete (or a stop condition is reached).

Key Insight:
Text generation is a sequential process where each token depends on what has already been generated.

Machine Translation

Goal : Translate a text from a source language to a target language.

Example :

- *Text*: "Hello, how are you?"
- *Output*: "Bonjour, comment ça va ?"

Text Summarization

Goal : Generate a concise, informative summary from a longer text.

Example :

- *Text*: “Artificial intelligence is revolutionizing many sectors, especially medicine, where it is enabling better diagnostics.”
- *Output*: “AI improves medical diagnostics.”

Question Answering - QA

Goal : Generate a text response based on a question and context.

Example :

- *Context*: “The Eiffel Tower was built in 1889.”
- *Question*: “When was the Eiffel Tower built?”
- *Output*: “In 1889.”

Free Text Generation

Goal : Generate fluid text from minimal input or a prompt.

Example :

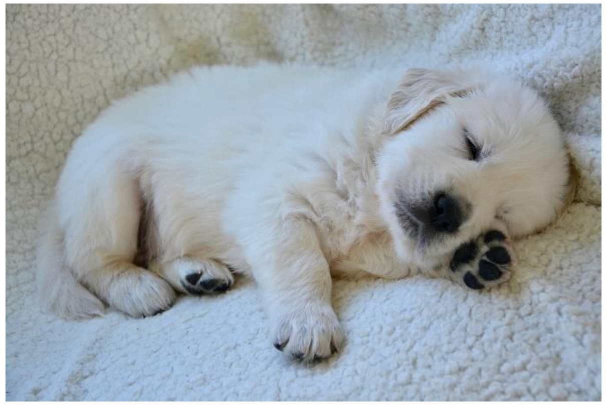
- *Text*: “Tell a story about a magical cat.”
- *Output*: “Once upon a time, there was a cat who lived in an enchanted forest and could talk to animals.”

Image-to-Text

Goal : Generate a text description from an image.

Example :

- *Input*:



- *Output*: “A baby dog sleeping on a beige blanket.”

Data-to-Text Generation

Goal : Convert structured data into natural text.

Example :

- *Input*:

Name	Age	Occupation
John	30	Doctor

- *Output*: “John is 30 and a doctor.”

Data-to-Text Generation

Goal : Convert structured data into natural text.

Example :

- *Input*:

- *Output*: “Hello everyone!”

Preprocessing for Traditional Methods

Introduction to Traditional Approaches

Simpler and rule-based methods:

Focused on breaking text into smaller components (e.g., words or characters).

Relied heavily on word frequency and patterns in the text.

Key Characteristics:

Count-based techniques to measure how often words appear in text.

Statistical methods to identify relationships between words and their contexts.

Advantages:

- **Simplicity**: Easy to implement and interpret. + Works well for small datasets or straightforward tasks.
- **Computational Efficiency**: Requires less processing power compared to neural models.
- **Domain-Specific Customization**: Rules can be tailored to specific applications (e.g., legal or medical texts).
- **Explainability**: Outputs are often easier to understand and debug.

Text Cleaning

Objective: Enhance text data quality before vectorization.

Steps:

- **Removing unnecessary characters:** punctuation, emojis, special characters.
- **Normalization:**
  - Lowercasing text.
  - Removing accents.
- **Stopword removal:** Removing non-informative words (e.g., "the," "and").

Lemmatization, and Stemming

Stemming: Reduces a word to its root (e.g., "running" → "run").

Lemmatization: Converts a word to its base form, considering context (e.g., "was" → "be").

Comparison:

- **Stemming:** faster but less precise.
- **Lemmatization:** more accurate.

Tokenization

Whitespace Tokenization:
Splits text at spaces.

Example: "I can't play outside because it's raining."→ ["I", "can't", "play", "outside", "because", "it's", "raining."]

Limitations: Treats punctuation as part of tokens, can't handle contractions like "don't".

Tokenization

Word-based Tokenization:
Divides text into words while applying additional rules to handle punctuation, contractions, etc.

Example: "I can't play outside because it's raining." → ["I", "ca", "n't", "play", "outside", "because", "it", "'s", "raining", "."]

Limitations: Slower due to additional rules.

Tokenization

Character-based Tokenization:
Treats each character as a token.

Example: "can’t" → ["c", "a", "n", " ‘ ", "t"]

Use cases: Rarely used alone; mostly for simple or low-resource setups.

Text Vectorization

What Is Text Vectorization?
The process of converting **text into numerical representations** so it can be used by machine learning models.

Key Idea: Words or documents are transformed into **vectors** (arrays of numbers) that encode their features.

Example:
Sentence Tokenized: [“I”, “love”, “NLP”, “and”, “I”, “love”, “AI”]

Bag of Words:
- Vocabulary: [“I”, “love”, “like”, “NLP”, “and”, “AI”, “ML”]
- Vector: [1,1,0,1,1,1,0]

Frequency-Based Representations:
- Vocabulary: [“I”, “love”, “like”, “NLP”, “and”, “AI”, “ML”]
- Vector: [2,2,0,1,1,1,0]

Text Vectorization - Challenges

No Context:
Cannot capture the relationship between words in a sentence (e.g., "good" vs. "not good").

High Dimensionality:
Large vocabularies lead to very high-dimensional vectors.

Sparse Representations:
Most vector entries are zero, leading to inefficiency in computation

Preprocessing for Modern Models

Introduction to Modern Approaches

Embedding-based techniques: Words or phrases are represented as numerical vectors, capturing their meaning in context.

Deep Learning: Models learn patterns and relationships in text using neural networks. + Incorporates attention mechanisms to focus on important parts of the input text.

Key Characteristics:
- Words are no longer isolated; meaning is determined by the context in which they appear.
- Use large datasets and powerful algorithms to learn patterns without manual intervention.

Advantages
- **Contextual Understanding**: Captures the meaning of words based on their usage in sentences. Example: "bank" in "I sat by the bank" vs. "I visited the bank."
- **Scalability**: Handles large datasets and multilingual text efficiently.
- **End-to-End Learning**: Reduces manual feature engineering, as models learn features automatically.

Cleaning

Differences from traditional approaches:

- Lighter cleaning to preserve context.
- No stopword removal or stemming.

Why minimal cleaning is sufficient:

Models like BERT handle tokenization and normalization internally.

Recommended steps:

- Remove unnecessary characters (punctuation, emojis) depending on the task.
- Simple normalization (lowercasing, accent removal).

Tokenization

Subword Tokenization:
Words are divided into smaller units called subwords.

Example:
- Rare word: "unbelievable" → ["un", "believ", "able"]
- Frequent word: "cat" → ["cat"] (remains intact).

A fixed vocabulary (e.g., 30k-50k tokens) is created before model training.
Includes frequent subwords, characters, and special symbols (e.g., [CLS], [SEP]).

Note: The token [CLS] is placed at the beginning of each sentence. It is used to perform classification tasks.

Subword Tokenization Techniques

1) Byte Pair Encoding (BPE)
Used in models like GPT and GPT-2.

How it works:

1. Start with each character as a token.
   Example: "hello" → ["h", "e", "l", "l", "o"].

2. Merge the most frequent pairs of characters to form new tokens.
   Example:
   ["h", "e", "l", "l", "o"] → “h” + “e” : 10 occ ; “e”+ “l”: 4 occ ; **“l” + “l” : 18 occ** ; “l” + “o” ; 6 occ.
   So, we create: ["h", "e", "ll", "o"] → **“h” + “e” : 10 occ** ; “e”+ “ll”: 2 occ ; “ll” + “o” ; 3 occ.
   So we create: ["he", "ll", "o"]. Etc…

3. Continue until reaching a fixed vocabulary size (e.g., 50k tokens).

Advantages:
Efficiently handles rare words.
Reduces vocabulary size.

Subword Tokenization Techniques

2) WordPiece
Used in models like BERT.

How it works:

1. Start with each character as a token.
   Example: "hello" → [“h", "e", "l", "l", "o"].

2. Breaks words into subwords based on statistical likelihoods (rather than frequency like BPE).
   Example:
   P("h","e","llo") = 0.15
   P("he","llo") = **0.35**
   P("hel","lo") = 0.21 etc…

   “hello" → [“he", "##llo"]. (## indicates that the token continues a previous word.)

Advantages:
Avoids excessive fragmentation.
Well-suited for complex languages (e.g., Korean, Turkish).

Subword Tokenization Techniques

3) SentencePiece (with Unigram)
Used in models like T5 and GPT-3.

How it works:

1. Processes raw text without requiring initial segmentation (no splitting by spaces).
   Example: “I can’t play.” → “I_can’t_play”
1. Tokenize a sentence by minimizing the number of tokens while maximizing the overall probability
   Example: “I_can’t_play” → ["▁I", "▁can", "'t", "▁play"] (▁ denotes the start of a new word.)

P("▁I")=0.3
P("▁can’t")=0.04
P("▁ca")=0.002
P("▁can")=0.5
P(“’t")=0.1
P(“_play")=0.12
P(“_pl")=0.003
P(“ay")=0.01

Advantages:
Language-agnostic.
Flexible for different scripts or languages.

Subword Tokenization Techniques

Aspect	BPE	WordPiece	SentencePiece
Principe	Merges the most frequent subword pairs	Merges the most likely subword pairs	Probabilistic model based on a global vocabulary (Unigram)
Spaces	Prior segmentation required	Prior segmentation required	Treats spaces as characters
Indicators	No	Uses indicators (e.g., ##)	Prefix ▁ for new words
Advantages	Simple and fast	Better at capturing context	Flexible, suitable for any language
Disadvantages	Ignores global context	Less suitable for unsegmented corpora	More complex to train
Current use	GPT-2, RoBERTa	BERT, DistilBERT	T5, GPT-3

Padding, and Truncation

Input sequences must have the same length to form a consistent batch.
Models cannot handle sequences of varying lengths directly.

1. Padding

Adds special tokens ([PAD]) to shorter sequences to match the length of the longest sequence in a batch.
Attention masking to ignore padding tokens.

Example:
Sequence 1: ["I", "love", "studying", "transformers", "."]
Sequence 2: ["AI", "is", "fun"]
After Padding:
Sequence 1: ["I", "love", "studying", "transformers ", "."]
Sequence 2: ["AI", "is", "fun“, “[PAD]”, “[PAD]”, “[PAD]”]

Padding, and Truncation

Input sequences must have the same length to form a consistent batch.
Models cannot handle sequences of varying lengths directly.

2. Truncation

Cuts sequences that exceed a predefined maximum length.

Example:
Sequence 1: ["I", "love", "studying", "transformers", "."]
Sequence 2: ["AI", "is", "fun"]
Max Length: 4
After Truncation:
Sequence 1: ["I", "love", "studying", "transformers"]
Sequence 2: ["AI", "is", "fun"]

Padding, and Truncation

In practice, we use a mix of both techniques:

Example:
Sequence 1: ["I", "love", "studying", "transformers", "."]
Sequence 2: ["AI", "is", "fun"]
Max Length: 4
After Truncation & Padding:
Sequence 1: ["I", "love", "studying", "transformers"]
Sequence 2: ["AI", "is", "fun“, “[PAD]”, “[PAD]”]

Embeddings

Mapping to IDs:
Tokens are converted into numeric IDs based on the vocabulary.

Example: ["un", "believ", "able"] → [1234, 5678, 910].

Feeding to the Model:
These IDs are transformed into dense vectors via an embedding layer before being processed by the LLM.

Example: [1234, 5678, 910] → [[0.2,0.45,1.23,1.01], [2.54,0.03,0.42,2.3], [0.11,0.4,0.1,2.54]]

Questions