WORD EMBEDDING

Basics on Contextualized Word Embedding

Prof. Marco Viviani

marco.viviani@unimib.it

A VERY QUICK RECAP

Let's do some tidying up

- **Text Representation** (TR) forms the basis of NLP and Text Mining (TM).

- It is concerned with the adequate **encoding and formatting of natural language** so that a machine can solve a NLP/TM task.

- In order for a machine to derive meaning from text and solve more complex tasks, the unstructured, discrete symbols have to be transformed into a structured representation.

→ **Numeric vectors**

- The two predominant approaches to do this are **local** and **distributed representations**.

Local representations

- **Local representations** refer to methods of representing linguistic information in a discrete and symbol-based manner.
  - Individual words (or tokens) are assigned unique (and isolated) representations.

- **Main** local representations:
  - One-hot encoding (word level)
  - Count-based representations (document/word level)

One-hot encoding

- **Each word** or token in a vocabulary is represented as a **binary vector**, where only one element is "hot" (set to 1), and all others are "cold" (set to 0).
- The position of the **"hot" element** in the vector corresponds to the word's unique identifier or index in the vocabulary.
- One-hot encoding is simple and interpretable but **does not capture word relationships**.
  - This is due to vector dimensions being orthogonal to each other and each individual token exhibiting the same distance to any other token in the vocabulary.

Count-based representations (Documents)

- A **count vector**, also known as **Bag-of-Words** (BoW), constitutes a local representation that is able to capture similarity information across documents.
- Each document is represented as a vector where each dimension corresponds to a unique word.
- BoW does not take into account the importance of words or their frequency in the entire corpus.
  - It treats each word in isolation.
- The value in each dimension can be represented by the frequency of that word in the document (Term Frequency, TF), or a weighted version of it (TF-IDF) → Next slide.

TF-IDF

- **Term Frequency – Inverse Document Frequency** (TF-IDF) represent words based on their frequency of occurrence in a document AND in a corpus.

- Term Frequency represents how often a term (word or phrase) appears in a document.

- Inverse Document Frequency quantifies how unique or important a term is across a collection of documents.
  - Terms that appear in many documents have a lower IDF, while terms that are specific to certain documents have a higher IDF.

Count-based representation (Words)

- A **co-occurrence vector** for a word constitutes a local representation that can capture similarity information across words.
- Each word is represented as a vector where each dimension corresponds to a unique word.
- The value in each dimension represents the frequency of the co-occurrence of the target word with a contextual word in the corpus.
- A context window (e.g., ±5 words) is often used to define co-occurrence, and in practice, the co-occurrence matrix can be very large and sparse.

Distributed representations

- **Distributed representations** for texts, also known as **word embeddings** or distributed word representations, represents words (or tokens) as <u>continuous</u> vectors in a multi-dimensional vector space.

- Each dimension of the vector corresponds to a **feature** or aspect of the word's (or token’s) meaning.

- Words with similar meanings or that often appear in similar contexts tend to have similar vector representations.

- Distributed representations allow for capturing semantic relationships between words.

Main approaches

- **Count-based models** (among others)
  - SVD
  - GloVe

- **Predictive models**
  - word2vec
  - FastText

- Basically, had one representation of words:

king [-0.5, -0.9, 1.4, …] queen [-0.6, -0.8, -0.2, …]

TOWARDS CONTEXTUALIZED WORD EMBEDDING

Issues with Glove, word2vec, FastText

- Word embedding is nowadays the basis for **pre-trained NLP**.
  - The use of machine learning models that have been trained on large amounts of text data before being fine-tuned for specific NLP tasks.

king [-0.5, -0.9, 1.4, …]   queen [-0.6, -0.8, -0.2, …]

- **Problems**:
  - Always the same representation for a word type regardless of the context in which a word token occurs.
    - We might want very fine-grained word sense disambiguation.
    - Words have different aspects, including semantics, syntactic behavior, and register/connotations.

Word senses

- One word may have many meanings.

Noun

- S: (n) **mouse#1** (any of numerous small rodents typically resembling diminutive rats having pointed snouts and small ears on elongated bodies with slender usually hairless tails)
- S: (n) shiner#1, black eye#1, **mouse#2** (a swollen bruise caused by a blow to the eye)
- S: (n) **mouse#3** (person who is quiet or timid)
- S: (n) **mouse#4**, computer mouse#1 (a hand-operated electronic device that controls the coordinates of a cursor on your computer screen as you move it around on a pad; on the bottom of the device is a ball that rolls on the surface of the pad)

Verb

- S: (v) sneak#1, **mouse#1**, creep#2, pussyfoot#1 (to go stealthily or furtively)
- S: (v) **mouse#2** (manipulate the mouse of a computer)

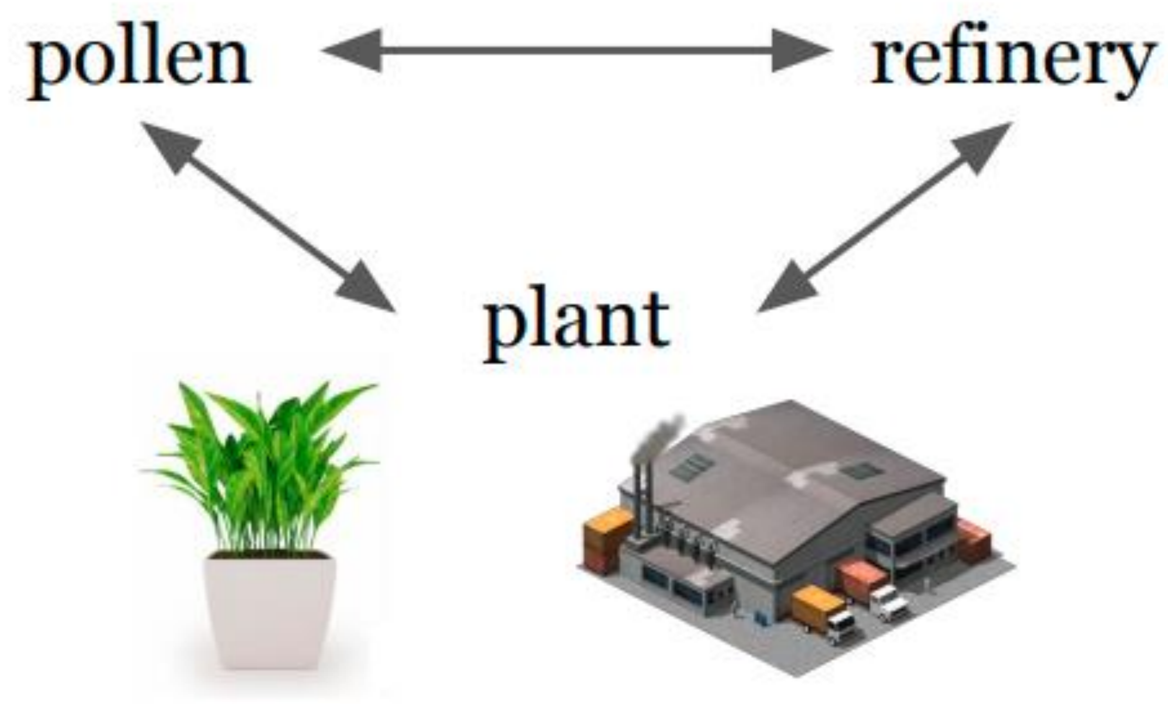
PRINCETON UNIVERSITY

WordNet

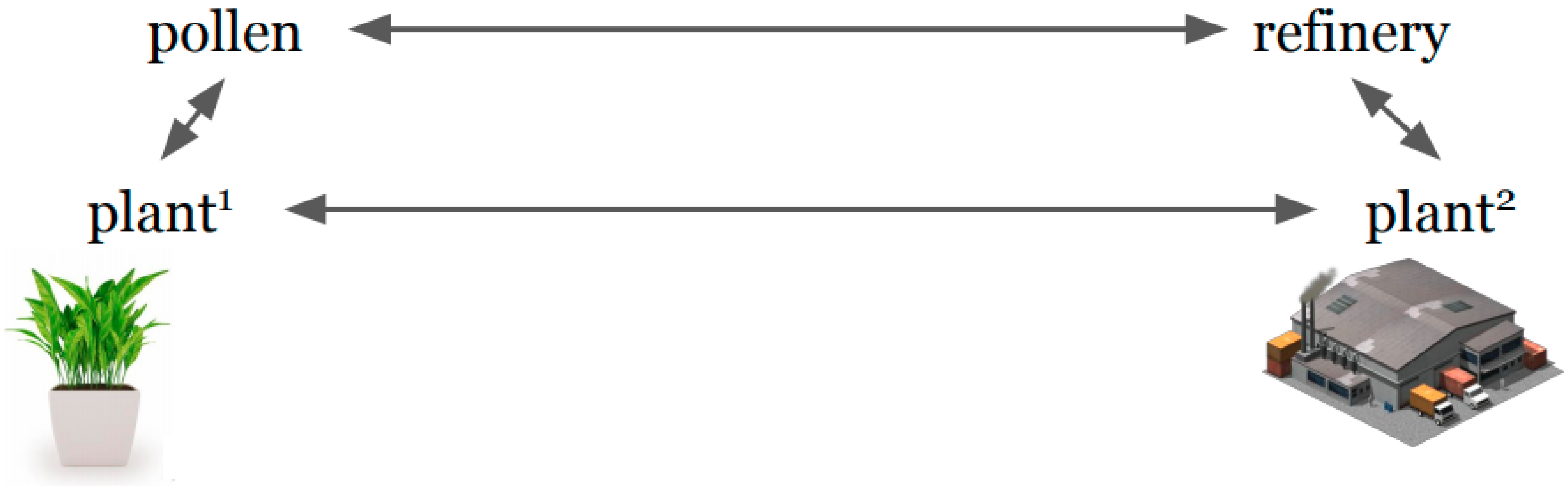
A Lexical Database for English

Conflation deficiency

- “Static” word embedding representations cannot capture polysemy.



Conflation deficiency



How many words?

- How many words are in this sentence below? (Ignoring punctuations)

ask not what your country can do for you,
ask what you can do for your country

Nine words

ask, can, country, do, for not, what, your, you

Seventeen words

ask, not, what, your, country, can, do, for, you, ask, what, you, can, do, for, your, country

How many words?

- How many words are in this sentence below? (Ignoring punctuations)

ask not what your coun

ask what

When we say "words", which interpretation do we mean?

Seventeen words

n, country, do, for not, what, your, you

ask, not, what, your, country, can, do, for, you, ask, what, you, can, do, for, your, country

Towards word interpretation

- Which of these **interpretations** did we use when we looked at word embeddings?

- **Word types**
  - Also known as lexical types, refer to the distinct, unique words or lexemes in a text or a corpus.

- **Word tokens**
  - Refer to the individual occurrences or instances of words in a text.

Word types

- Word types, also known as lexical types, refer to the **distinct, unique words or lexemes** in a text or a corpus.
- They represent the different vocabulary or dictionary words in a language.
- Word types are often lemmatized or stemmed to group together words with the same root form. For example, the word types "run," "running," and "ran" are all related to the lemma "run."
- Word types are typically used to measure vocabulary size and diversity within a text or a collection of texts.

Word types

- Word types are abstract and unique words or lexemes.

ask not what your country can do for you,
ask what you can do for your country

Nine words

ask, can, country, do, for not,
what, your, you

Seventeen words

ask, not, what, your, country,
can, do, for, you, ask, what,
you, can, do, for, your, country

Word tokens

- Word tokens refer to the **individual occurrences or instances of words** in a text.
- Each word token represents a specific occurrence of a word, regardless of whether it is a different form of an existing word type or a new word altogether.
- Word tokens are counted for the purpose of determining the total number of words in a text.
- They are used to perform various text analysis tasks, such as calculating word frequencies, analyzing text structure, or training language models.

Word tokens

- Word tokens refer to the individual occurrences or instances of words in a text.

ask not what your country can do for you,
ask what you can do for your country

Nine words

ask, can, country, do, for not, what, your, you

Seventeen words

ask, not, what, your, country, can, do, for, you, ask, what, you, can, do, for, your, country

Word embedding revisited

- All the word embedding methods we saw so far trained embeddings for **word types**.
  - Used word occurrences, but the final embeddings are “type” embeddings
  - Type embeddings = lookup tables → Static embeddings

- Can we embed word tokens instead?

- What makes a word token different from a word type?
  - We have the context of the word.
  - The context may inform the embeddings.

Word embedding revisited

Word embeddings, able to consider for word tokens instead word types, should…

- **Unify** superficially different words.
  - bunny and rabbit are similar.
- **Capture** information about how words can be used.
  - go and went are similar, but slightly different from each other.
- **Separate** accidentally similar looking words.
  - Words are polysemous.
    - The bank was robbed again.
    - We walked along the river bank.
  - Sense embeddings.

Word embedding revisited

Word embeddings, able to consider for word tokens instead word types, should…

- **Unify** superficially different word…
  - bunny and rabbit…
- **Capture** … words can be used.
  - go and w… similar, but slightly different from each other.
- **Separate** accidentally similar looking words.
  - Words are polysemous.
    - The bank was robbed…
    - We wal…
  - Se… …ge.

“Traditional” word embedding can do this

Need for “revisited” word embedding

Word type embedding VS token embedding

- **Word type embeddings** can be thought of as a lookup table.
  - Map words to vectors independent of any context.
  - A big matrix.

- **Word token embeddings** should be functions.
  - Construct embeddings for a word on the fly.
  - There is no fixed “bank” embedding, the usage decides what the word vector is.

Towards «contextualized» word embeddings

- Given a sentence, we want word embeddings that depends on the sentence itself.

```
[0.9, -0.2, 1.6, …]            [-1.9, -0.4, 0.1, …]
open a bank account             on the river bank
```

- Formally, build a **contextualized word embedding**:

$$\mathbf{c}_k = f(w_k | w_1, w_2, \dots, w_n) \in \mathbb{R}^d$$

Towards «contextualized» word embeddings

- $\mathbf{c}_k = f(w_k | w_1, w_2, \dots, w_n) \in \mathbb{R}^d$

- This means that:

$$\mathbf{c}_{play_1} = f(play \mid Elmo\ and\ Cookie\ Monster\ play\ a\ game)$$
$$\neq$$
$$\mathbf{c}_{play_2} = f(play \mid The\ Broadway\ play\ premiered\ yesterday)$$

$$\mathbf{c}_{play_1} = [\dots\ 0.17\ \ 0.87\ \ 0.15\ \ \dots]$$
$$\mathbf{c}_{play_2} = [\dots\ 0.44\ \ -0.12\ \ 0.77\ \ \dots]$$

Towards «contextualized» word embeddings

- The big new thing in **2017-18**.
  - "A Structured Self-attentive Sentence Embedding"
    - by Zhouhan Lin et al. (2017)
  - "ELMo: Deep contextualized word representations"
    - by Matthew Peters et al. (2018)
  - "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"
    - by Jacob Devlin et al. (2018)

- Nowadays → **Transformers** (but not only).
  - Quick overview on NNs and text representation → Next slide.

NNs and text representation

- **Neural Networks** (NNs) are nowadays widely used for text representation and related TM and NLP tasks.
  - Feedforward Neural Networks (**FFNNs**).
  - Convolutional Neural Networks (**CNNs**).
  - Recurrent Neural Networks (**RNNs**).
  - **Transformers.**

- They have different architectures and approaches for this purpose.

Feedforward Neural Networks

- **FFNNs** are used in simple text representation tasks, such as creating word embeddings or sentence embeddings.
  - In the case of word embeddings, a single-layer FFNN can be used to project words from a high-dimensional space into a lower-dimensional continuous space, creating distributed word representations.
  - For sentence embeddings, FFNNs can take the average or maximum of word embeddings to create fixed-size vectors representing sentences (the same for documents).

Convolutional Neural Networks

- CNNs are often used in text representation for tasks like text classification and sentiment analysis.
  - In text classification, CNNs use convolutional layers to extract local features (n-grams) from the input text.
  - These features are then used for classification.

- Pre-trained word embeddings can be used as input to CNNs, allowing them to capture semantic information.

Recurrent Neural Networks

- **RNNs include**:
  - Long-Short Term Memory (**LSTMs**) networks.
  - Other RNN variants:
    - Vanilla RNNs.
    - Gated Recurrent Units (**GRUs**).

- They are a category of neural networks designed to handle sequential data, making them suitable for text representation tasks where context and sequence matter.

- LSTMs can capture long-range dependencies.
  - They have been used for generating contextualized word embeddings by considering the sequential context of words.

Transformers

- A **Transformer** is a Deep Learning architecture that was introduced in the paper "Attention is all you need" by Vaswani et al. in 2017.

- Transformers are the most versatile and widely used architecture for **text representation nowadays**.
  - Transformers like **BERT**, **GPT**, and their variants provide **contextualized word embeddings**.
  - Transformers have the ability to capture **bidirectional context** (depending on the model) and have brought significant improvements to text understanding and representation.
  - They are used for a broad range of TM/NLP tasks, including text classification, sentiment analysis, machine translation, question answering, and more.

Bidirectional context

- The difference between **monodirectional** and **bidirectional context** in the context of NLP and Deep Learning is related to how a model processes and understands the surrounding words or tokens in a sequence.
  - **Monodirectional context**
    - The model only considers the words or tokens that precede the current word or token in the sequence. This means the model captures information from left to right or right to left but not both simultaneously.
  - **Bidirectional context**
    - The model considers both the words that come before and after the current word or token in the sequence. It captures information from both directions simultaneously.

Key concepts (1)

- The development of Transformer-based contextualized word embeddings involves several **foundational concepts and components**.
  - Transformers
    - Attention and self-attention.
    - Masking.
  - Large Language Models (**LLMs**)
    - Pre-training
      - A model is pre-trained on a massive amount of text data, often involving terabytes of text.
    - Fine-tuning
      - Models can be fine-tuned on specific NLP tasks, allowing them to adapt to the nuances of a particular application while retaining the general language understanding learned during pre-training.
    - Transfer Learning
      - These models serve as feature extractors, providing contextualized embeddings for downstream tasks.

Key concepts (2)

- **Self-attention**
  - For every word (token), the model learns how much attention to give to every other word in the sentence.
  - Example:

The pizza that the chef made was delicious.

Key concepts (2)

- **Self-attention**
  - For every word (token), the model learns how much attention to give to every other word in the sentence.
  - Example:

The pizza that the chef made was delicious.

When the model processes “delicious”, self-attention lets it *look at* surrounding words and decide which ones matter most.

- It pays high attention to “pizza” → what is delicious
- It pays some attention to “chef” → who made the pizza
- It pays very little attention to words like “the” or “that”

Key concepts (3)

- The Transformer architecture is based on **two main components**

- **Encoder** (understand text)
  - Reads the whole input sentence at once (bidirectionally)
  - Self-attention to understand how all the words relate to each other
  - It creates contextualized embeddings
  - Used in models like BERT (understanding text)

- **Decoder** (produces text)
  - The decoder generates new text, one word at a time
  - It looks at the previous words and at the encoder's output
  - It is used for text generation or translation
  - Used in models like GPT (generation) or T5/BART (translation, summarization)

More technical details

- They will be provided in the course:
  - **Natural Language Processing**
  - Elisabetta Fersini and Alessandro Raganato
  - Second semester
  - https://elearning.unimib.it/enrol/index.php?id=61162