Sentence-Level embeddings

Dr. Lorenzo Vaiani
Dipartimento di Automatica e Informatica
Politecnico di Torino

Lecture goals

- From word to sentences
- Sentence-level embedding models
  - Doc2Vec
  - Sent2Vec
  - InferSent
- Embedding model evaluation
  - Intrinsic evaluation
  - Extrinsic evaluation
- Bias in word embedding models

Lecture goals

- **From word to sentences**
- Sentence-level embedding models
  - Doc2Vec
  - Sent2Vec
  - InferSent
- Embedding model evaluation
  - Intrinsic evaluation
  - Extrinsic evaluation
- Bias in word embedding models

From words to sentences

- **Word embeddings:** dense vectors that captures the semantics of words. They enable word-to-word similarity computation.

Rome = [0.91, 0.83, 0.17, …, 0.41]

Paris = [0.92, 0.82, 0.17, …, 0.98]

Italy = [0.32, 0.77, 0.67, …, 0.42]

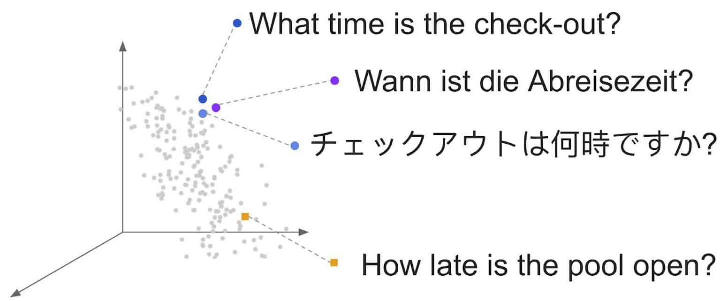
France = [0.33, 0.78, 0.66, …, 0.97]

https://speakerdeck.com/marcobonzanini/word-embeddings-for-natural-language-processing-in-python-at-london-python-meetup?slide=22

- Applications in several NLP tasks (e.g., topic analysis)

From words to sentences

- **Sentence embeddings:** dense vectors the semantic meaning of the entire sentence.



Source: https://megagon.ai/blog/emu-enhancing-multilingual-sentence-embeddings-with-semantic-similarity/

- There are several method to aggregate words into sentences
  - Both **static** and **dynamic** embeddings

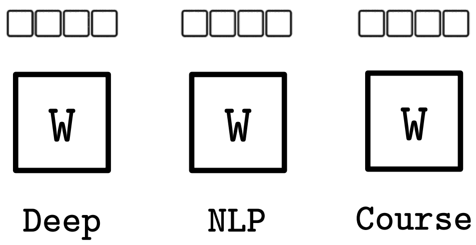
Lecture goals

- From word to sentences
- **Sentence-level embedding models**
  - Doc2Vec
  - Sent2Vec
  - InferSent
- Embedding model evaluation
  - Intrinsic evaluation
  - Extrinsic evaluation
- Bias in word embedding models

Doc2Vec

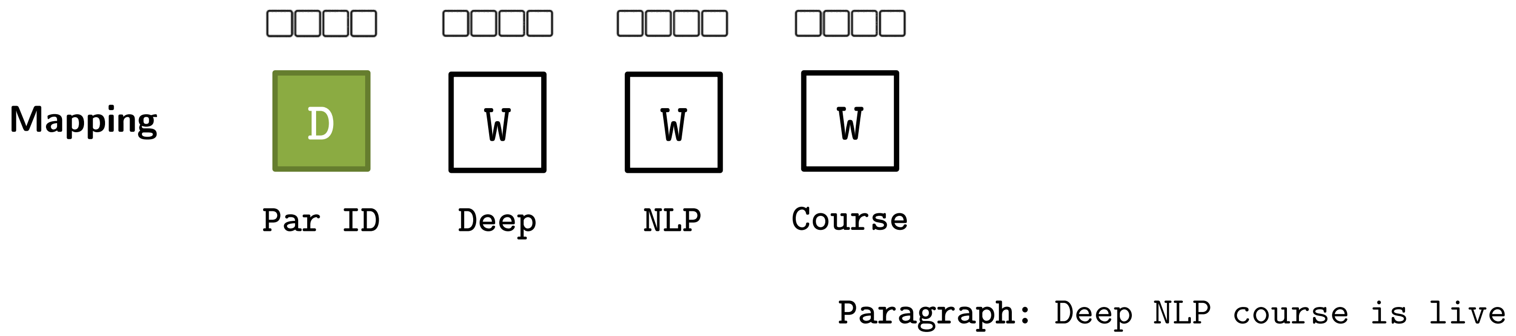
- **Doc2Vec**: extend Word2Vec to longer text sequences.
  - Proposed by Le & Mikolov (2014)
  - Relies on the assumption according to which word's meaning is given by the words that appear nearby
- There are two variations of the algorithm:
  - Distributed Memory model (DM)
  - Distributed Bag of Words (DBOW)

What is possible to obtain with standard Word2Vec model



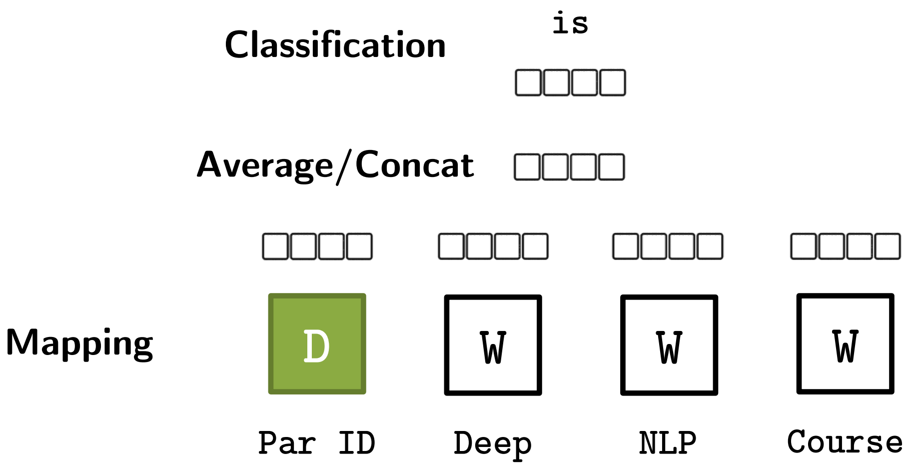
Doc2Vec - DM

- **Distributed Memory model (similar to CBOW)**
  - Every paragraph is mapped to a unique vector (matrix **D**)
  - Every word is mapped to a unique vector (matrix **W**)



Doc2Vec - DM

- **Distributed Memory model (similar to CBOW)**
  1. Every paragraph is mapped to a unique vector (matrix **D**)
  2. Every word is mapped to a unique vector (matrix **W**)
  3. The **paragraph vector** and **word vectors** are averaged or concatenated to predict the next word in a context.

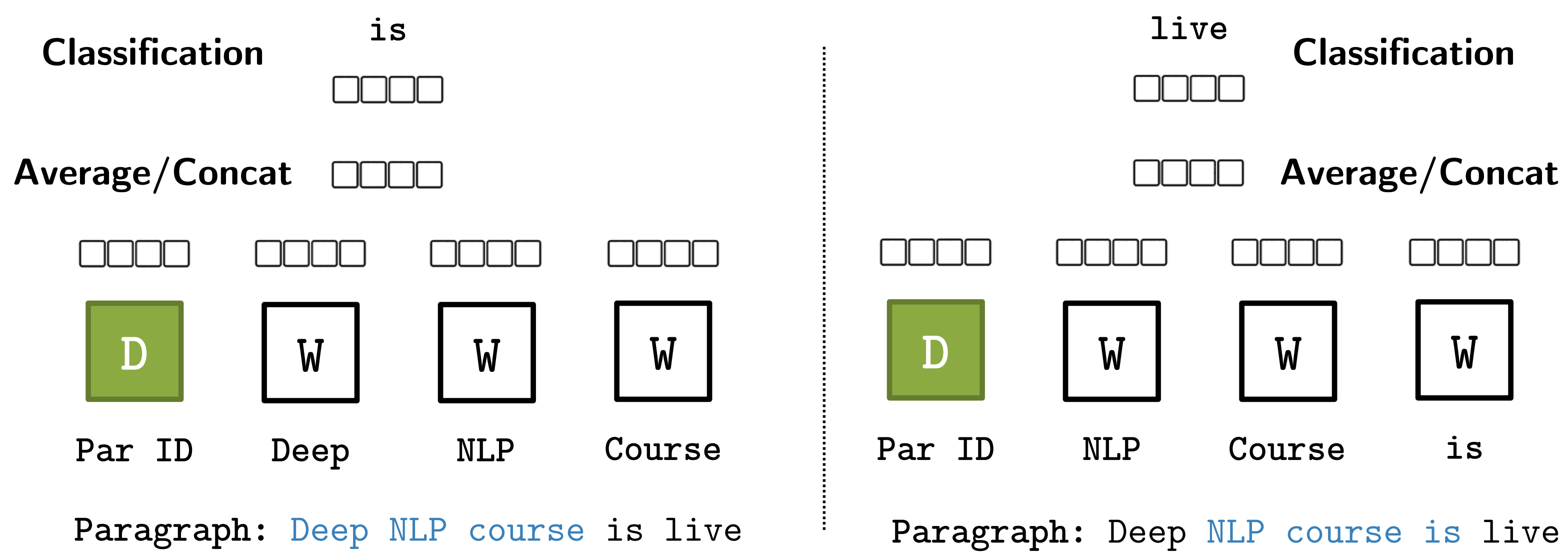


Paragraph: Deep NLP course is live

Doc2Vec - DM

Paragraph Vector: can be thought as an additional word. It acts as a memory that remembers what is missing from the current context ("topic" of the paragraph)

Context: sliding window over the paragraph. The paragraph vector is shared across all contexts generated from the same paragraph (not across paragraphs)



Doc2Vec - DM

Classification objective: paragraph vectors and word vectors are trained using stochastic gradient descent and the gradient is obtained via backpropagation.

During training, at each step of stochastic gradient descent:

1. **Sample** a fixed-length context from a random paragraph
2. **Compute the error** gradient from the network
3. **Use the gradient to update the parameters** in the embedding model.

Doc2Vec - DM

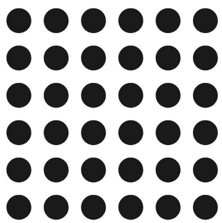
Classification objective: paragraph vectors and word vectors are trained using stochastic gradient descent and the gradient is obtained via backpropagation.

At prediction time the embedding for a new paragraph is obtained by fixing the parameters for word vectors (**W**) and classification

After training, paragraph vectors can be used as features for the paragraph. It is possible to **feed these features** directly **to machine learning techniques** such as logistic regression, support vector machines or K-means.

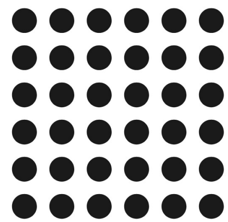
Doc2Vec - DM

Tr: training phase - Te: test/inference phase



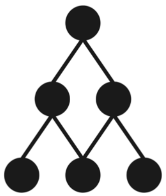
Word matrix (**W**):

- _ in Tr
- _ in Te



Paragraph matrix (**D**):

- _ in Tr
- _ in Te

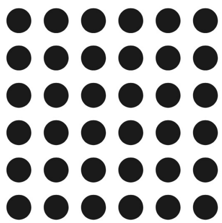


Classification weights:

- _ in Tr
- _ in Te

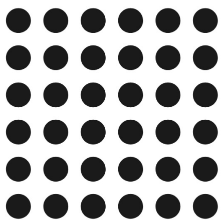
Doc2Vec - DM

Tr: training phase - Te: test/inference phase



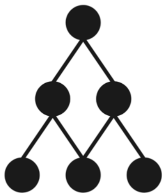
Word matrix (**W**):

- Learned in Tr
- Fixed in Te



Paragraph matrix (**D**):

- Learned in Tr
- **Learned** in Te

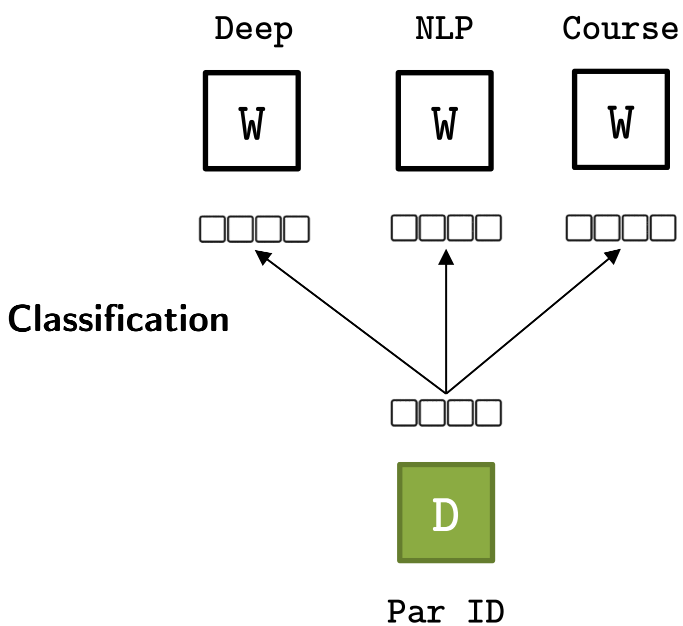


Classification weights:

- Learned in Tr
- Fixed in Te

Doc2Vec - DBOW

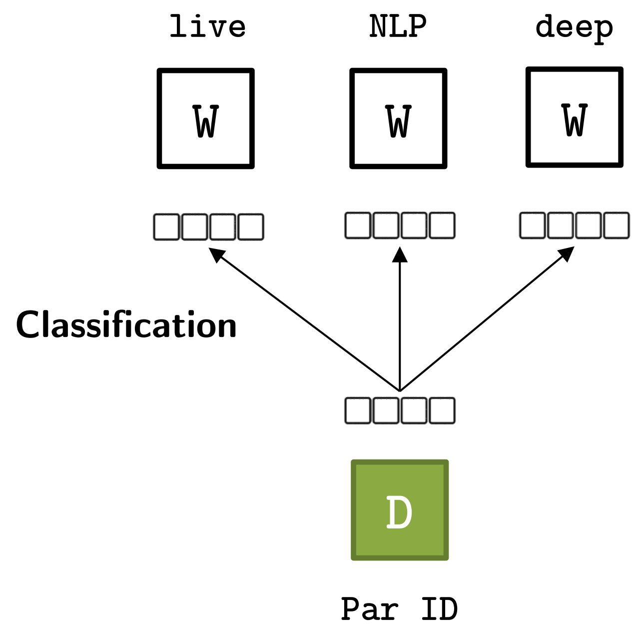
- **DBOW**: it is similar to the Skip-Gram architecture of Word2Vec



Paragraph: Deep NLP course is live

Doc2Vec - DBOW

- **Training Phase:**
  - Represent both paragraphs and words with unique IDs
  - Given paragraph ID:
    - **Input:** one-hot encoded representation of the paragraph ID.
    - **Output:** is one hot encoded representation of a randomly selected word.
  - The neural network transforms one hot encoded representation to paragraph vector. It is passed through a softmax layer. Both set of weights are adjusted during training phase.



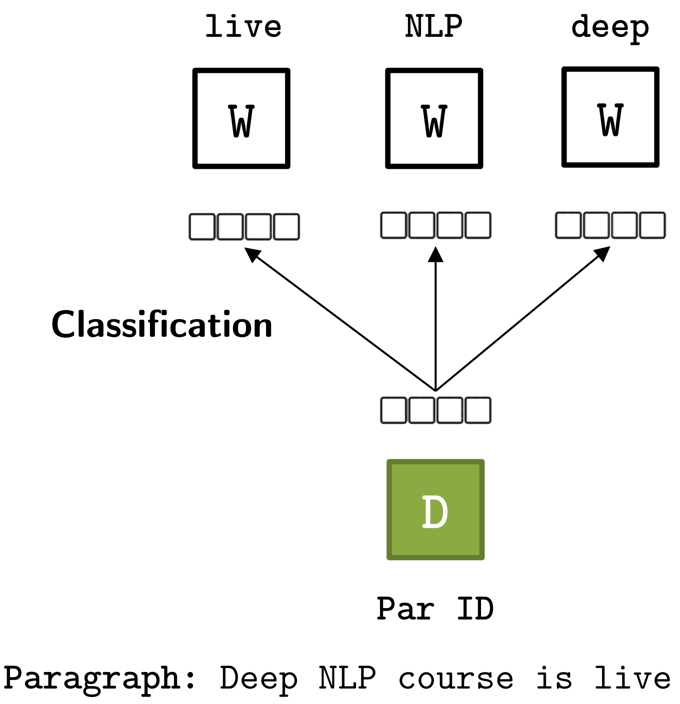
Paragraph: Deep NLP course is live

After training: all paragraph IDs are mapped to a new space such that probabilities for **randomly selected words** in each document are maximized starting from that vector space representation to softmax output.

Doc2Vec - DBOW

Inference:

- What vector space representation is most appropriate for an input document
  - Use the same (trained) set of weights from hidden space to output layer.
  - The weights from hidden layer to softmax output are frozen
- Select a word from new document at random.
- Start with a random representation for the document vector
- Adjust the randomly initialized weights such that softmax probability is maximized for the selected word.



An additional **hyperparameter** is required to set the number of update steps. It represents the number of updates to the paragraph vector before obtaining the final embedding.

Doc2Vec - implementation

```python
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
from sklearn.metrics.pairwise import cosine_similarity

data = ["Deep NLP course is live.",
        "It's a course for PoliTO students.",
        "It's a course for Master students.",
        "PoliTO is located in Turin."]

tagged_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)]) for i, _d in enumerate(data)]


max_epochs = 100
emb_size = 200
alpha = 0.01

model = Doc2Vec(documents=tagged_data,
                vector_size=emb_size,
                alpha=alpha,
                min_alpha=0.0001,
                min_count=1,
                dm =1,
                epochs=max_epochs)

model.save("d2v.model")
print("Model Saved")

vector_1 = model.infer_vector(["nlp", "course"])
vector_2 = model.infer_vector(["polito", "students"])
print (cosine_similarity(vector_1.reshape(1, -1), vector_2.reshape(1, -1)))
```

Represents a document along with a tag, input document format for Doc2Vec

Remove words with the number of occurrences lower than this threshold

Doc2Vec - implementation

GENSIM
topic modelling for humans

```python
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
from sklearn.metrics.pairwise import cosine_similarity

data = ["Deep NLP course is live.",
        "It's a course for PoliTO students.",
        "It's a course for Master students.",
        "PoliTO is located in Turin."]

tagged_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)]) for i, _d in enumerate(data)]


max_epochs = 100
emb_size = 200
alpha = 0.01

model = Doc2Vec(documents=tagged_data,
                vector_size=emb_size,
                alpha=alpha,
                min_alpha=0.0001,
                min_count=1,
                dm =1,
                epochs=max_epochs)

model.save("d2v.model")
print("Model Saved")

vector_1 = model.infer_vector(["nlp", "course"])
vector_2 = model.infer_vector(["polito", "students"])
print (cosine_similarity(vector_1.reshape(1, -1), vector_2.reshape(1, -1)))
```

Embedding size

Flag to use DM algorithm

Doc2Vec - implementation

```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
from sklearn.metrics.pairwise import cosine_similarity

data = ["Deep NLP course is live.",
        "It's a course for PoliTO students.",
        "It's a course for Master students.",
        "PoliTO is located in Turin."]

tagged_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)]) for i, _d in enumerate(data)]


max_epochs = 100
emb_size = 200
alpha = 0.01

model = Doc2Vec(documents=tagged_data,
                vector_size=emb_size,
                alpha=alpha,
                min_alpha=0.0001,
                min_count=1,
                dm =1,
                epochs=max_epochs)

model.save("d2v.model")
print("Model Saved")

vector_1 = model.infer_vector(["nlp", "course"])
vector_2 = model.infer_vector(["polito", "students"])
print (cosine_similarity(vector_1.reshape(1, -1), vector_2.reshape(1, -1)))
```

Initial learning rate

Learning rate will linearly drop to min_alpha as training progresses

Doc2Vec - implementation

GENSIM
topic modelling for humans

```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
from sklearn.metrics.pairwise import cosine_similarity

data = ["Deep NLP course is live.",
        "It's a course for PoliTO students.",
        "It's a course for Master students.",
        "PoliTO is located in Turin."]

tagged_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)]) for i, _d in enumerate(data)]


max_epochs = 100
emb_size = 200
alpha = 0.01

model = Doc2Vec(documents=tagged_data,
                vector_size=emb_size,
                alpha=alpha,
                min_alpha=0.0001,
                min_count=1,
                dm =1,
                epochs=max_epochs)

model.save("d2v.model")
print("Model Saved")

vector_1 = model.infer_vector(["nlp", "course"])
vector_2 = model.infer_vector(["polito", "students"])
print (cosine_similarity(vector_1.reshape(1, -1), vector_2.reshape(1, -1)))
```

Storing the model

Compute cosine similarity between vectors

 |

Sent2Vec

- **Sent2Vec**: compose sentence embeddings using word vectors along with n-gram embeddings
  - Proposed by Pagliardini et al. 2018
  - It simultaneously train composition and the embedding vectors themselves.

It can be seen as an **extension** of the **CBOW** model that allows to train and infer numerical representations of whole sentences instead of single words

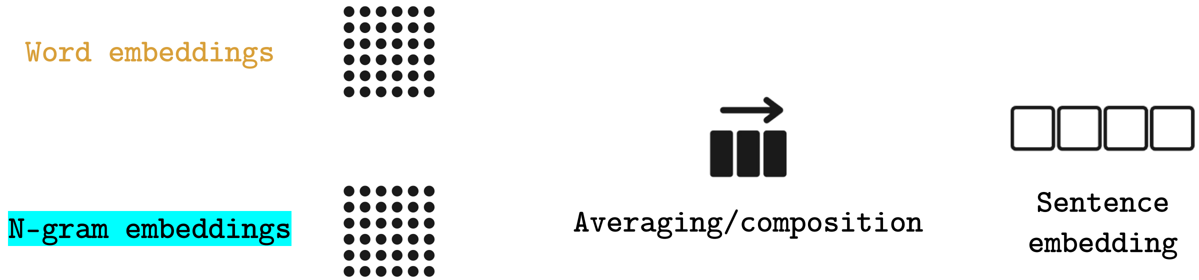
Sent2Vec

The sentence embedding is defined as the average of the source word embeddings of its constituent words

Source embeddings are learned **not only for unigrams** but also for n-grams present in each sentence

Final embeddings are obtained by averaging both n-grams and word embeddings

'Word composition is essential for sentence embeddings'



Sent2Vec – usage pretrained models

```python
import sent2vec
model = sent2vec.Sent2vecModel()

# pretrained model
model.load_model('model.bin')
emb = model.embed_sentence("NLP is powerful")
embs = model.embed_sentences(["Deep NLP is powerful", "CV vs NLP is the AI battle"])
```

Source: https://github.com/epfml/sent2vec

Sent2Vec – usage pretrained models

Source: https://github.com/epfml/sent2vec

```
# Input file should contain one sentence per line
# Input data should prior pre-processed

./fasttext sent2vec -input wiki_sentences.txt -output my_model -minCount 8 -dim 700 -epoch 9 -lr 0.2 -wordNgrams
2 -loss ns -neg 10 -thread 20 -t 0.000005 -dropoutK 4 -minCountLabel 20 -bucket 4000000 -maxVocabSize 750000
-numCheckPoints 10
```

Path to the output model

Loss function (default: negative sampling)

Number of ngrams dropped when training a sent2vec model

InferSent

- **InferSent**: Conneau et al. 2018

Natural Language Inferencing Task: is the task of determining whether a "hypothesis" is true, false, or neutral, given a "premise". It involves high-level reasoning about semantic relationships within sentences.

```
premise: A deep learning model is used to analyze natural language.
hypothesis: NLP tools include deep learning models.
label: Entailment
```


InferSent

- **InferSent**: Conneau et al. 2018

Natural Language Inferencing Task: is the task of determining whether a "hypothesis" is true, false, or neutral, given a "premise". It involves high-level reasoning about semantic relationships within sentences.

```
premise: Syntactic tools are the only way to analyze text
hypothesis: NLP tools include deep learning models.
label: Contradiction
```

InferSent

- **InferSent**: Conneau et al. 2018

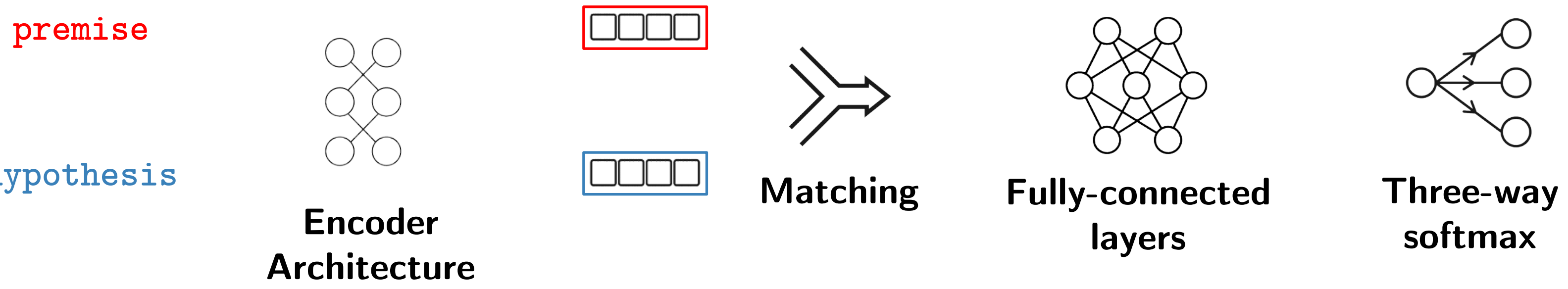
Natural Language Inferencing Task: is the task of determining whether a “hypothesis” is true, false, or neutral, given a “premise”. It involves high-level reasoning about semantic relationships within sentences.

```
premise: In CV you can use deep learning to boost performance
hypothesis: NLP tools include deep learning models.
label: Neutral
```


InferSent

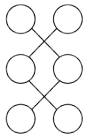
- **Training:**
  - **Input:** premise and hypothesis
  - **Output:** entailment, contradiction, or neutral

- **Embedding matching:**
  - Concatenation
  - Element-wise product
  - Absolute element-wise difference



InferSent

- **Embedding model:**
  1. LSTM
  2. GRU
  3. Bidirectional GRU + concat
  4. Bidirectional LSTM with average pooling
  5. Bidirectional LSTM with max pooling
  6. Self-attentive network
  7. Hierarchical CNN



Embedding model

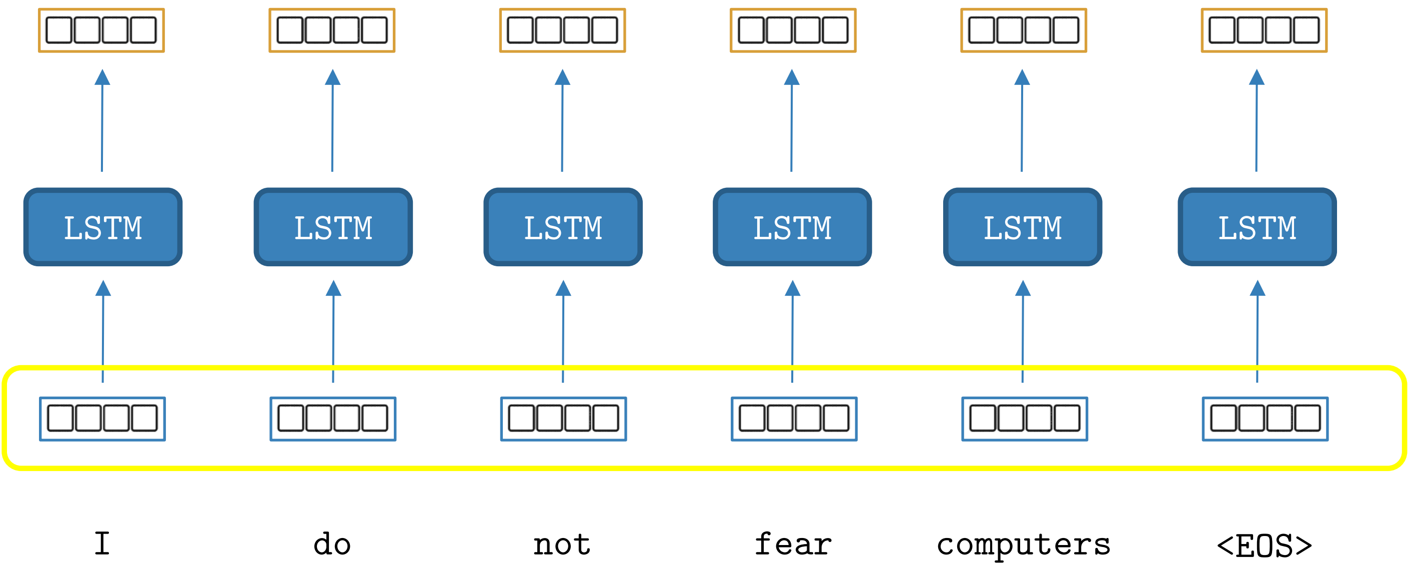
InferSent - preliminaries

```
# preliminaries

git clone https://github.com/facebookresearch/InferSent.git
cd InferSent

mkdir GloVe
curl -Lo GloVe/glove.840B.300d.zip http://nlp.stanford.edu/data/glove.840B.300d.zip
unzip GloVe/glove.840B.300d.zip -d GloVe/
mkdir fastText
curl -Lo fastText/crawl-300d-2M.vec.zip https://dl.fbaipublicfiles.com/fasttext/vectors-english/crawl-
300d-2M.vec.zip
unzip fastText/crawl-300d-2M.vec.zip -d fastText/

# download model
curl -Lo encoder/infersent1.pkl https://dl.fbaipublicfiles.com/infersent/infersent1.pkl
```

https://github.com/facebookresearch/InferSent

Why word embedding model?

InferSent - preliminaries



Word-level input representation

InferSent - usage

```
from models import InferSent

VERSION = 2
MODEL_PATH = 'encoder/infersent%s.pkl' % VERSION
params_model = {'bsize': 64, 'word_emb_dim': 300, 'enc_lstm_dim': 2048,
                'pool_type': 'max', 'dpout_model': 0.0, 'version': VERSION}
infersent = InferSent(params_model)
infersent.load_state_dict(torch.load(MODEL_PATH))

W2V_PATH = 'fastText/crawl-300d-2M.vec'
infersent.set_w2v_path(W2V_PATH)

# sente
infersent.build_vocab(sentences, tokenize=True)
embeddings = infersent.encode(sentences, tokenize=True)
```

https://github.com/facebookresearch/InferSent

Lecture goals

- From word to sentences
- Sentence-level embedding models
  - Doc2Vec
  - Sent2Vec
  - InferSent
- **Embedding model evaluation**
  - Intrinsic evaluation
  - Extrinsic evaluation
- Bias in word embedding models

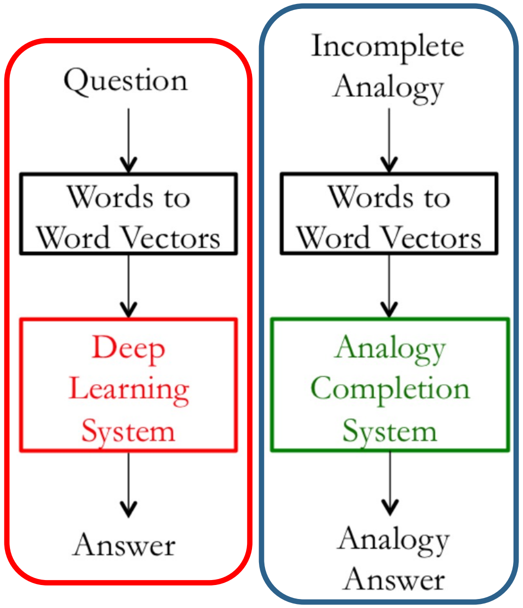
Model evaluation

- Intrinsic evaluation:
  - Generic evaluation of the quality and coherence of the vector space
  - It is independent from model performance on downstream tasks
- Extrinsic Evaluation
  - Assess model perfomance when used as preliminary step in a downstream task (e.g., machine translation)

Intrinsic Evaluation

- Evaluation on a specific, intermediate task
  - Fast to compute performance
- Helps understand vector subsystem
  - Needs positive correlation with real task to determine usefulness

Extrinsic Evaluation
Expensive training of the end-to-end system



Intrinsic Evaluation
a simple intrinsic evaluation technique which can provide a measure of "goodness" of the word to word vector subsystem.

https://cs224d.stanford.edu/lecture_notes/notes2.pdf

Intrinsic Evaluation - *Word Vector Analogies*

In a word vector analogy, we are given an incomplete analogy of the form:

$$\texttt{a : b = c : ?(d)}$$

The goal is to obtain the following relation:

$$\texttt{b - a = d - c}$$

This implies:

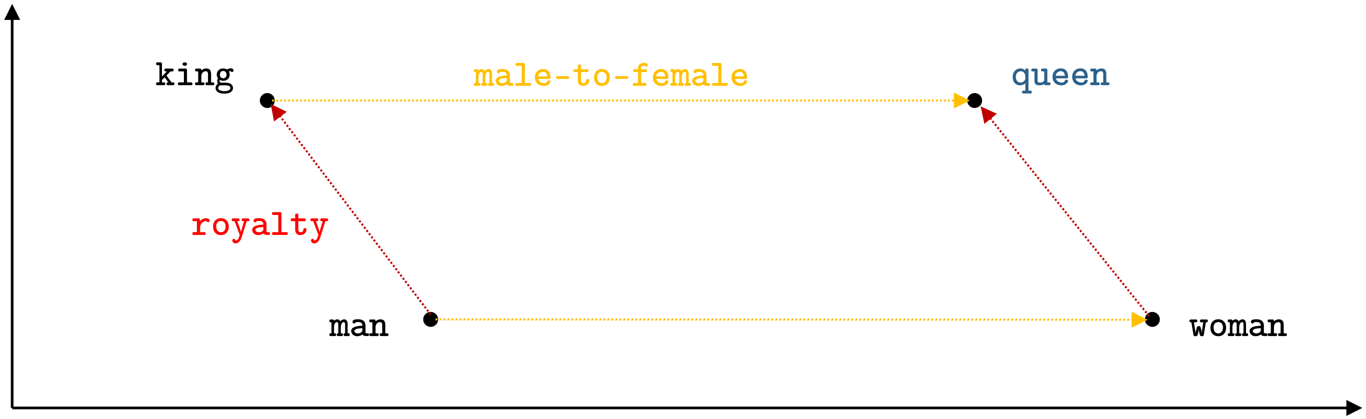
$$\texttt{b - a + c = d}$$

Identify the vector **d** which maximizes the normalized dot-product between the two word vectors (i.e. cosine similarity).

Intrinsic Evaluation - *Word Vector Analogies*

In a word vector analogy, we are given an incomplete analogy of the form:

$$\texttt{a : b = c : ?}$$

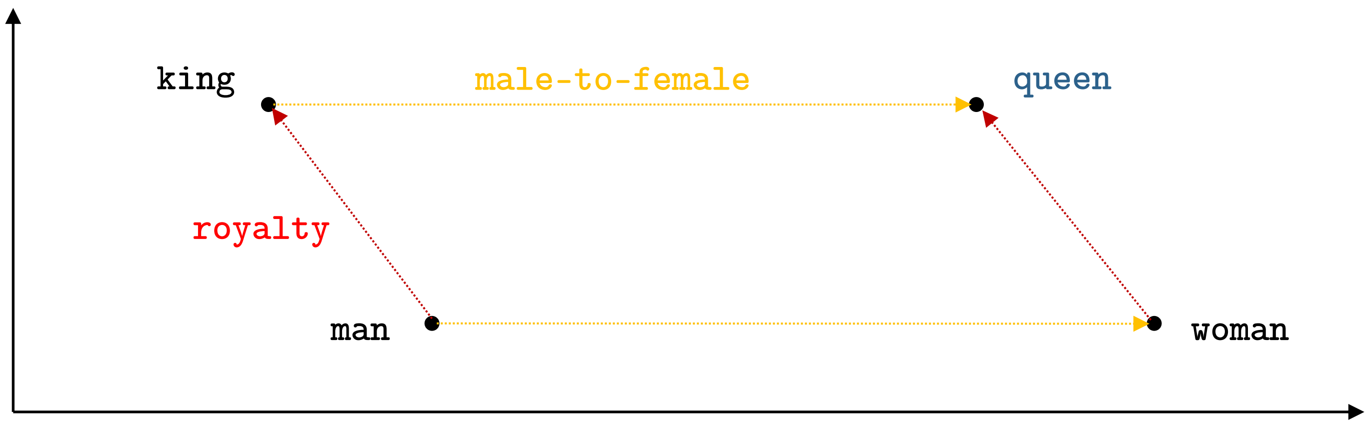


Intrinsic Evaluation - *Word Vector Analogies*

In a word vector analogy, we are given an incomplete analogy of the form:

`man : king = woman : queen`

`king - man + woman = queen`



Intrinsic Evaluation - *Word Vector Analogies*

An example of analogy dataset could be:

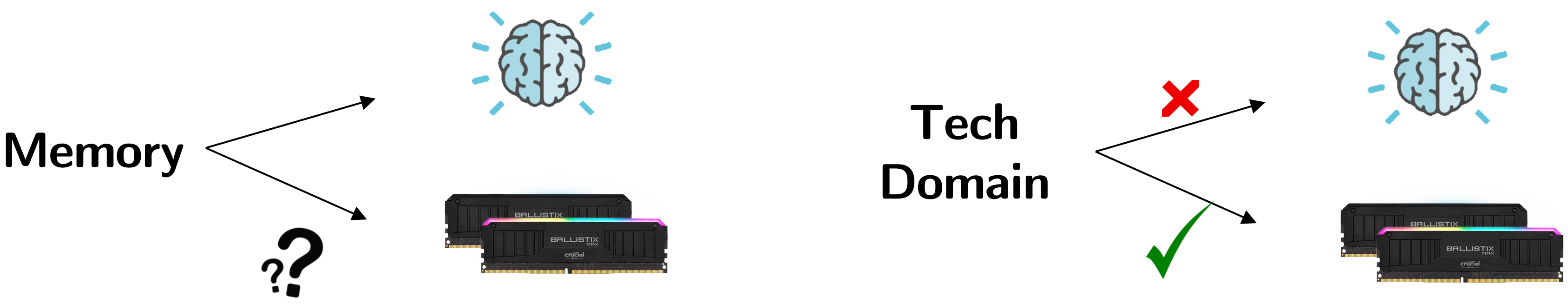
Input	**Result**
`Italy : Rome = France : (?)`	`Paris`
`Italy : Rome = Belgium : (?)`	`Bruxelles`
`Italy : Rome = Germany : (?)`	`Berlin`
`Italy : Rome = Australia : (?)`	`Canberra`
`Italy : Rome = Greece : (?)`	`Athens`
`...`	`...`

What type of similarity?

- **Property-based:** if they share many properties (e.g., bike-motorcycle)
- **Meronymy:** a meronym denote a part and a holonym denoting a whole (e.g., wheel-bike)
- **Antonymy:** the semantic relationship between words that have opposite meanings (e.g., good-bad)
- **Topic relation:** both words refer to the same topic (e.g., zebra-zoo)

Limitations

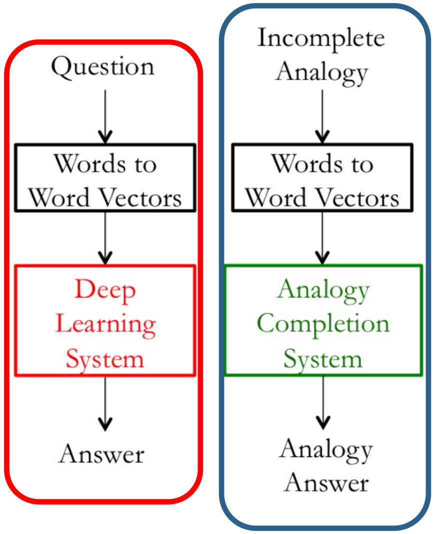
- It only considers the **attributional** similarity between words. Irrelevant for some NLP tasks (e.g., North, south in QA systems)
- **Hubness**: an **hub** in the semantic space is a word that has high semantic similarity with a large number of words.
- **Polysemous nature** of words: a single word can have multiple meanings in different domains



Extrinsic Evaluation

- Evaluation on a real task
- Can be slow to compute performance
- Unclear if subsystem is the problem
- If replacing subsystem improves performance, the change is likely good

Extrinsic evaluation
Evaluation of a set of word vectors generated by an embedding technique on the real task at hand.

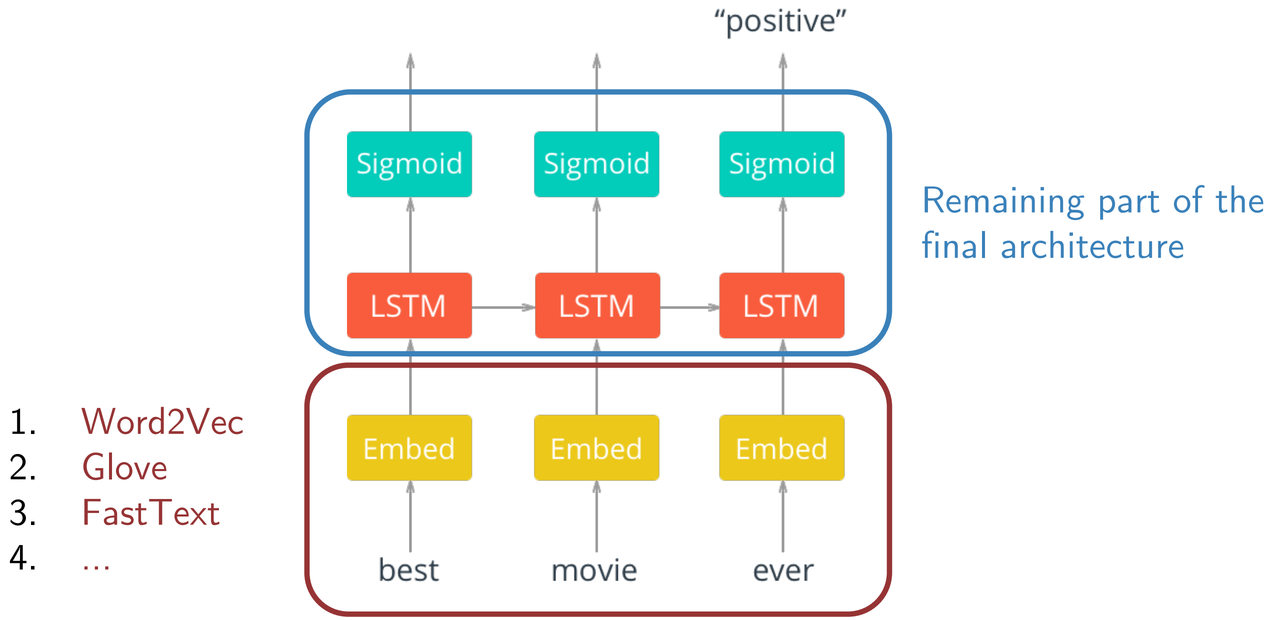


Intrinsic Evaluation
Low correlation with final system performance

https://cs224d.stanford.edu/lecture_notes/notes2.pdf

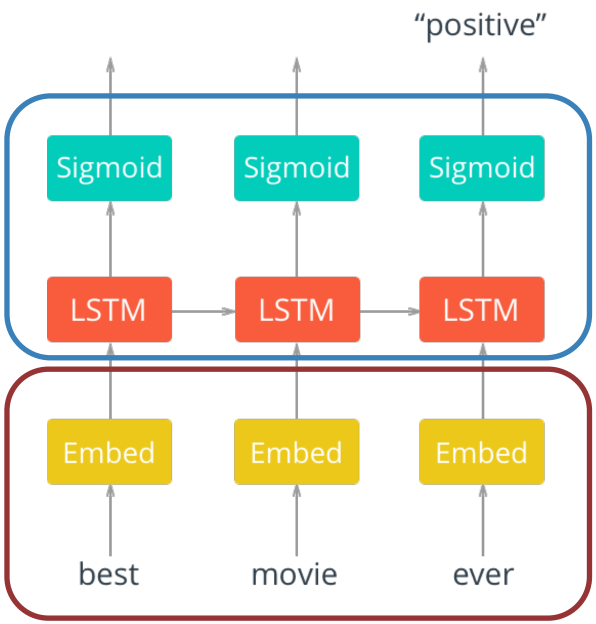
Extrinsic Evaluation

- Most NLP extrinsic tasks can be formulated as classification tasks.
- **Example:** given a sentence, we can classify the sentence to have positive, negative or neutral sentiment.



Extrinsic Evaluation

- It could be **unclear** whether performance improvement is related to word embedding models or other model settings
- Usually, the word vectors used in extrinsic tasks are initialized by optimizing them over a **simpler intrinsic task**.



Remaining part of the final architecture

1. Word2Vec
2. Glove
3. FastText
4. ...

Lecture goals

- From word to sentences
- Sentence-level embedding models
  - Doc2Vec
  - Sent2Vec
  - InferSent
- Embedding model evaluation
  - Intrinsic evaluation
  - Extrinsic evaluation
- **Bias in word embedding models**

Bias in embedding models

- Word embeddings model text meaning, however, text could incorporate implicit biases and stereotypes

`man : king = woman : queen`

`man : programmer = woman : ?`

- **Allocation harm:** when a system allocate resources (jobs or credit) unfairly to different groups.

Bias in embedding models

- In [1] the authors used GloVe embeddings to analyze racial-related bias in the embedding model
- They found that African-American names (Leroy, Shaniqua) had higher cosine similarity with unpleasant words if compared with European-American names (Brad, Greg)
- **Representational harm:** when a system demeans or ignores some social groups.
- **Bias** mitigation or reduction is an open (and discussed) problem in NLP research

[1] Caliskan, A., Bryson, J. J., & Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, *356* (6334), 183-186.

Question and Exercises

What of the following statement holds **False** for word embedding models?

1. FastText encodes subwords.
2. GloVe exploits occurrence-based statistics.
3. GloVe do not support domain adaptation.
4. WordVec relies on a Deep Feed-forward Neural Network.
5. Word2Vec supports domain adaptation.

Question and Exercises

What of the following statement holds **False** for word embedding models?

1. FastText encodes subwords.
2. GloVe exploits occurrence-based statistics.
3. GloVe do not support domain adaptation.
4. WordVec relies on a Deep Feed-forward Neural Network.
5. Word2Vec supports domain adaptation.

It relies on a single-layer NN

Question and Exercises

What of the following statement holds False for word embedding models?

1. FastText encodes subwords. **It is one of the main novelty over Word2Vec**
2. GloVe exploits occurrence-based statistics.
3. GloVe do not support domain adaptation.
4. WordVec relies on a Deep Feed-forward Neural Network.
5. Word2Vec supports domain adaptation.

 |

Question and Exercises

What of the following statement holds False for word embedding models?

1. FastText encodes subwords.
2. GloVe exploits occurrence-based statistics. **It is one of the main novelty over Word2Vec**
3. GloVe do not support domain adaptation.
4. WordVec relies on a Deep Feed-forward Neural Network.
5. Word2Vec supports domain adaptation.

Question and Exercises

What of the following statement holds False for word embedding models?

1. FastText encodes subwords.
2. GloVe exploits occurrence-based statistics.
3. GloVe do not support domain adaptation.
4. WordVec relies on a Deep Feed-forward Neural Network.
5. Word2Vec supports domain adaptation.

It is not possible to adapt to a new corpus without re-training

Question and Exercises

What of the following statement holds False for word embedding models?

1. FastText encodes subwords.
2. GloVe exploits occurrence-based statistics.
3. GloVe do not support domain adaptation.
4. WordVec relies on a Deep Feed-forward Neural Network.
5. Word2Vec supports domain adaptation.

The training can continue over a new corpus to specialize the domain (no global statistics)

Question and Exercises

Considering the Word2Vec model:

1. Describe its inner working and main goals

Question and Exercises

Considering the Word2Vec model:

1. Describe its inner working and main goals

- The word embedding model generates one vector representation per word in the vocabulary.
- Word vectors are not contextualized (unlike, e.g., ELMO)
- It relies on self-supervised training using sliding window.
- Architectures: Skip-gram and CBOW.
- The output of the model is a matrix containing the mapping between each word in the vocabulary and its corresponding vector.

Question and Exercises

Considering the Word2Vec model:

2. Illustrate in details one of the training procedures (at your choosing).

Question and Exercises

Considering the Word2Vec model:

2. Illustrate in details one of the training procedures (at your choosing).

6.2: The CBOW training procedure:

- The context is represented as a bag of the words contained in a fixed-size window aligned with the target word.
- The distributed representations of the context is used to predict the word in the middle of the window.
- The classification model exploits the context to predict the correct target word.
- *Sketch of the architecture.*

Question and Exercises

Considering the Word2Vec model:

3. Discuss (at least) one of the main drawbacks of the model.

Question and Exercises

Considering the Word2Vec model:

3. Discuss (at least) one of the main drawbacks of the model.

Limitation: word representations are not contextual.

The cat eat the $\texttt{mouse}^1$

The $\texttt{mouse}^2$ is not working anymore.

$$\mathrm{W2V}(\texttt{mouse}^1) = \mathrm{W2V}(\texttt{mouse}^2)$$

Question and Exercises

Considering the Word2Vec model:

4. Enumerate the key differences with FastText.

Question and Exercises

Considering the Word2Vec model:

4. Enumerate the key differences with FastText.

FastText creates n-gram embeddings and generates word vectors using compositionality.

E.g. (with n-gram = 4),

$$V(\texttt{processing}) = \mathrm{FT}(\texttt{<pro}) + \mathrm{FT}(\texttt{proc}) + \mathrm{FT}(\texttt{roce}) + \ldots + \mathrm{FT}(\texttt{ing>}) + \mathrm{FT}(\texttt{processing})$$

Question and Exercises

Considering the InferSent model:

1. Describe the main goal and the task on which it is trained

Question and Exercises

Considering the InferSent model:

1. Describe the main goal and the task on which it is trained

- The main goal of InferSent is to create a semantically meaningful sentence representation.
- It is trained on the inferencing task, which involves determining teh truth value (true, false or neutral) of a hypothesis based on a provide premise

Question and Exercises

Considering the InferSent model:

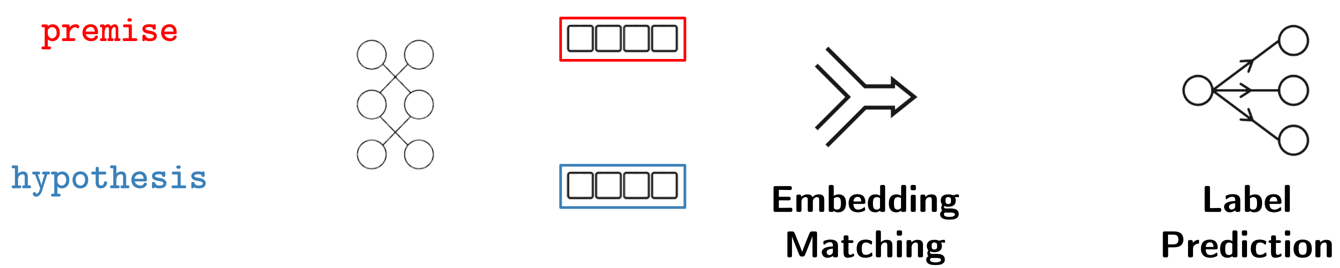
2. Describe the training pipeline (from input data to final model)

Question and Exercises

Considering the InferSent model:

2. Describe the training pipeline (from input data to final model)

- A word embedding model (e.g., Word2Vec, GLOVE) is applied to the words of the sentences
- Word embeddings are fed to an embedding model (e.g., LSTM, GRU)
- An embedding matching strategy is applied to predict the correct label among the 3 possible outcomes (entailment, contraddiction, neutral)
- Sketch of the pipeline:



Question and Exercises

Considering the InferSent model:

3. Indicate at least two embedding matching strategies

Question and Exercises

Considering the InferSent model:

3. Indicates at least two embedding matching strategies reporting also the final size with respect the input vectors

- **Concatenation**: the vector representations of the 2 sentences (premise and hypothesys) are concatenated to obtain a unique vector (with doubled size with respect the original ones)
- **Element-wise product**: corresponding elements of the two vectors are multiplied to obtain a new vector of the same size

Question and Exercises

What of the following statement holds **False** for Doc2Vec-DM?

1. Word and paragraph vector can be combined to predict the next word in a context
2. Paragraph vector always appears in the sliding window
3. All the weights for paragraphs and classification are fixed when you predict a new word vector
4. Every paragraph is mapped into a unique vector
5. None of the above

Question and Exercises

What of the following statement holds **False** for Doc2Vec-DM?

1. Word and paragraph vector can be combined to predict the next word in a context
2. Paragraph vector always appears in the sliding window
3. All the weights for paragraphs and classification are fixed when you predict a new word vector
4. Every paragraph is mapped into a unique vector
5. None of the above

It is the opposite!

Question and Exercises

What of the following statement holds **False** for Doc2Vec-DM?

1. Word and paragraph vector can be combined to predict the next word in a context
2. Paragraph vector always appears in the sliding window
3. All the weights for paragraphs and classification are fixed when you predict a new word vector
4. Every paragraph is mapped into a unique vector
5. None of the above

Yes, they can be averaged or concatenated

Question and Exercises

What of the following statement holds **False** for Doc2Vec-DM?

1. Word and paragraph vector can be combined to predict the next word in a context
2. Paragraph vector always appears in the sliding window **Window slides on word vectors while paragraph vector is fixed**
3. All the weights for paragraphs and classification are fixed when you predict a new word vector
4. Every paragraph is mapped into a unique vector
5. None of the above

Question and Exercises

What of the following statement holds **False** for Doc2Vec-DM?

1. Word and paragraph vector can be combined to predict the next word in a context
2. Paragraph vector always appears in the sliding window
3. All the weights for paragraphs and classification are fixed when you predict a new word vector
4. Every paragraph is mapped into a unique vector
5. None of the above

Unique representation, as for the words

Additional References

1. [Book] Jurafsky, D., & Martin, J. H. (2018). **Speech and language processing.** Available from: https://web.stanford.edu/~jurafsky/slp3.

2. [Book] Pilehvar, M. T., & Camacho-Collados, J. (2020). **Embeddings in natural language processing: Theory and advances in vector representations of meaning**. Synthesis Lectures on Human Language Technologies, 13(4), 1-175.

3. [Blog post] From Word Embeddings to Sentence Embeddings (link)

4. [Paper] Le, Q., & Mikolov, T. (2014, June). **Distributed representations of sentences and documents**. In International conference on machine learning (pp. 1188-1196). PMLR.

5. [Paper] Pagliardini, M., Gupta, P., & Jaggi, M. (2018). **Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features**. In Proceedings of NAACL-HLT (pp. 528-540).

6. [Paper] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017, September). **Supervised Learning of Universal Sentence Representations from Natural Language Inference Data.** In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (pp. 670-680).

Thank you!

