Semantic search and similarity ranking

Ane Berasategi

18. July 2019

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Plan

Part 1: Semantic search

- Introduction
- Ontology matching
- Similarity ranking

Part 2: Similarity ranking

- Word vectors
- Contextualized word embeddings: ELMo, BERT, Flair
- Implementation

Introduction

- **Lexical** search: literal matches of the query words.
  - Anthony Hopkins <u>age</u> → good
  - <u>How old</u> is Anthony Hopkins? → bad
- **Semantic** search: search with meaning, understand the query and the intention of the user
  - Why is my bus always late?
  - Why is my laptop overheating?
  - Why do bees follow me?

Ontology matching

The search engine has a huge knowledge graph / ontology with past searches, a representation of semantic relations between documents.

Pipeline:

1. New query arrives
2. Query broken into root terms: POS tagging removal, NER, error correction, conversion to embeddings, etc
3. Return the closest/more relevant/semantically most similar documents from the ontology (similarity ranking)

Plan

Part 1: Semantic search

- Introduction
- Ontology matching
- Similarity ranking

Part 2: Similarity ranking

- Word vectors
- Contextualized word embeddings
- ELMo
- BERT
- Flair
- Implementation

Word vectors: history

"A word is characterized by the company it keeps" – Firth, 1958

How to quantify and categorize semantic similarities between linguistic items based on their distributional properties?

- 1980s, LSA: reduce the number of dimensions using singular value decomposition
- 2000, Bengio et al.: reduce the high dimensionality of words representations in contexts by learning a distributed representation for words
- 2013, Mikolov et al., word2vec: word embedding toolkit to train word vectors in NNs, faster than n-gram models
- Pre-trained word embeddings became the norm (word2vec, GloVe, FastText) as input to NNs

Contextualized word embeddings

- Word embeddings: each word gets an embedding vector
- Irrelevant of the context, part of speech

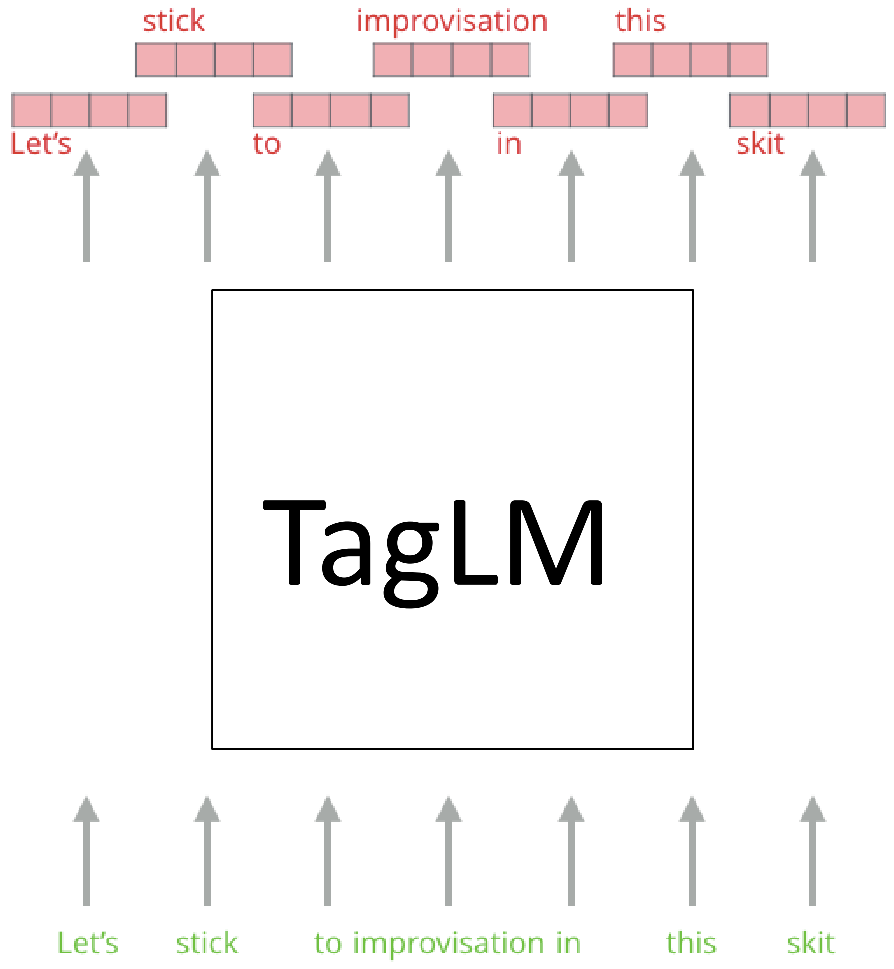
- 2017, Peters et al.: **TagLM**: give the words an embedding vector based on its context, in order to:
  - capture word meaning in that context
  - capture other contextual information

Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

Contextualized word embeddings

- Word embeddings: each word gets an embedding vector
- Irrelevant of the context, part of speech

- 2017, Peters et al.: **TagLM**: give the words an embedding vector based on its context, in order to:
  - capture word meaning in that context
  - capture other contextual information



Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

TagLM: paper #1, Peters et al., 2017

Formally described as: **semi-supervised** approach to add contextual embeddings to word embeddings from **bidirectional language models**

- **Language model** (LM): computes the probability of a token given a token sequence.
- Uses LSTM architecture, produces LM embeddings
  - Forward LM: Given the previous token sequence in a sentence, predict next token
  - Backward LM: Given the future token sequence, predict previous token
  - Bidirectional LM: forwLM and backLM trained separatedly and then concatenated to form the biLM embeddings
- **Semi-supervised** approach:
  - the biLM is trained on a large unlabelled corpus
  - the biLM embeddings are added as additional input to the NLP task

Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

TagLM: paper remarks

- Applied to sequence labelling tasks: assigning a categorical label to each member of a sequence of words.
- Using both forwLM and backLM embeddings boosts performance over forwLM embeddings
- Transfer learning: the biLM embeddings trained in one domain can be transferred to another
- Inputs to **TagLM**:
  - Character representation model, CNN or RNN
  - Token embeddings, initialized using pre-trained word embeddings
  - Recurrent LM: LSTM model with multiple layers
- Output from **TagLM**: a single context-independent representation for each word, **the output layer of the LSTM**.

Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

ELMo: paper #2: Peters et al., 2018

- Why just the last layer of the LSTM? Use all layers

Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

ELMo: paper #2: Peters et al., 2018

- Why just the last layer of the LSTM? Use all layers
- Deep "bidirectional" contextual embeddings (BERT, paper #4, disagrees) . ForwLM and backLM share some weights, not completely independent (unlike paper #1), afterwards shallow concatenation

Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

ELMo: paper #2: Peters et al., 2018

- Why just the last layer of the LSTM? Use all layers
- Deep “bidirectional” contextual embeddings (BERT, paper #4, disagrees) . ForwLM and backLM share some weights, not completely independent (unlike paper #1), afterwards shallow concatenation
- The embeddings are functions of the entire input sentence, computed on top of a 2 layer biLM with character convolutions, as a linear function of the internal network states.

Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

ELMo: paper #2: Peters et al., 2018

- Why just the last layer of the LSTM? Use all layers
- Deep “bidirectional” contextual embeddings (BERT, paper #4, disagrees) . ForwLM and backLM share some weights, not completely independent (unlike paper #1), afterwards shallow concatenation
- The embeddings are functions of the entire input sentence, computed on top of a 2 layer biLM with character convolutions, as a linear function of the internal network states.
- Higher-level layers of the LSTM capture context-dependent aspects of word meaning
- Lower-level layers of the LSTM capture aspects of syntax and can be used for POS tagging

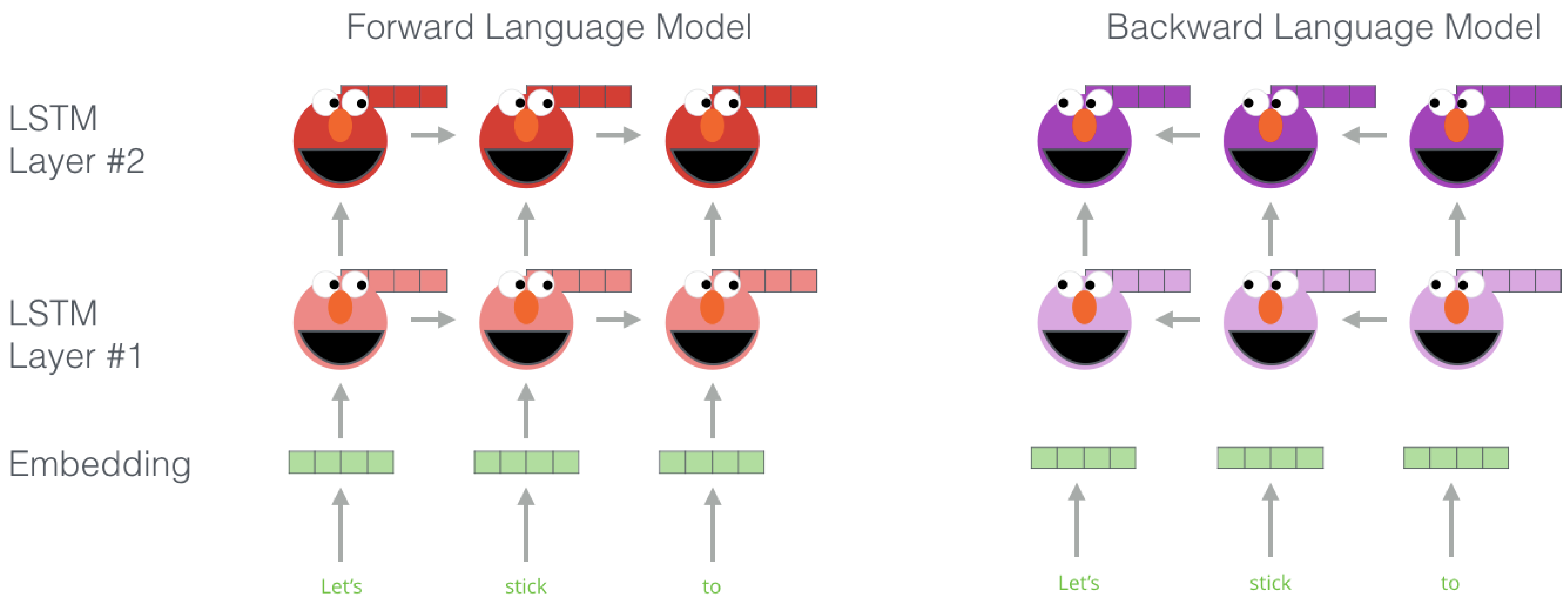
Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

ELMo: paper #2: Peters et al., 2018

- Why just the last layer of the LSTM? Use all layers
- Deep "bidirectional" contextual embeddings (BERT, paper #4, disagrees) . ForwLM and backLM share some weights, not completely independent (unlike paper #1), afterwards shallow concatenation
- The embeddings are functions of the entire input sentence, computed on top of a 2 layer biLM with character convolutions, as a linear function of the internal network states.
- Higher-level layers of the LSTM capture context-dependent aspects of word meaning
- Lower-level layers of the LSTM capture aspects of syntax and can be used for POS tagging
- ELMo is a **feature-based approach** for contextual embeddings: task-specific architectures that include the pre-trained representations as additional features
    - different architectures for different NLP tasks
    - The embeddings are added as additional inputs to the NLP task

Semi-supervised sequence tagging with bidirectional language models, Peters et al., 2017

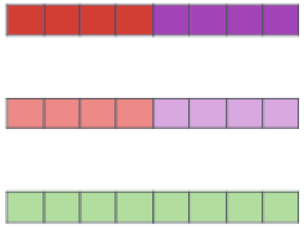
Embedding of ‘stick’ in ‘Let’s stick to’: step #1

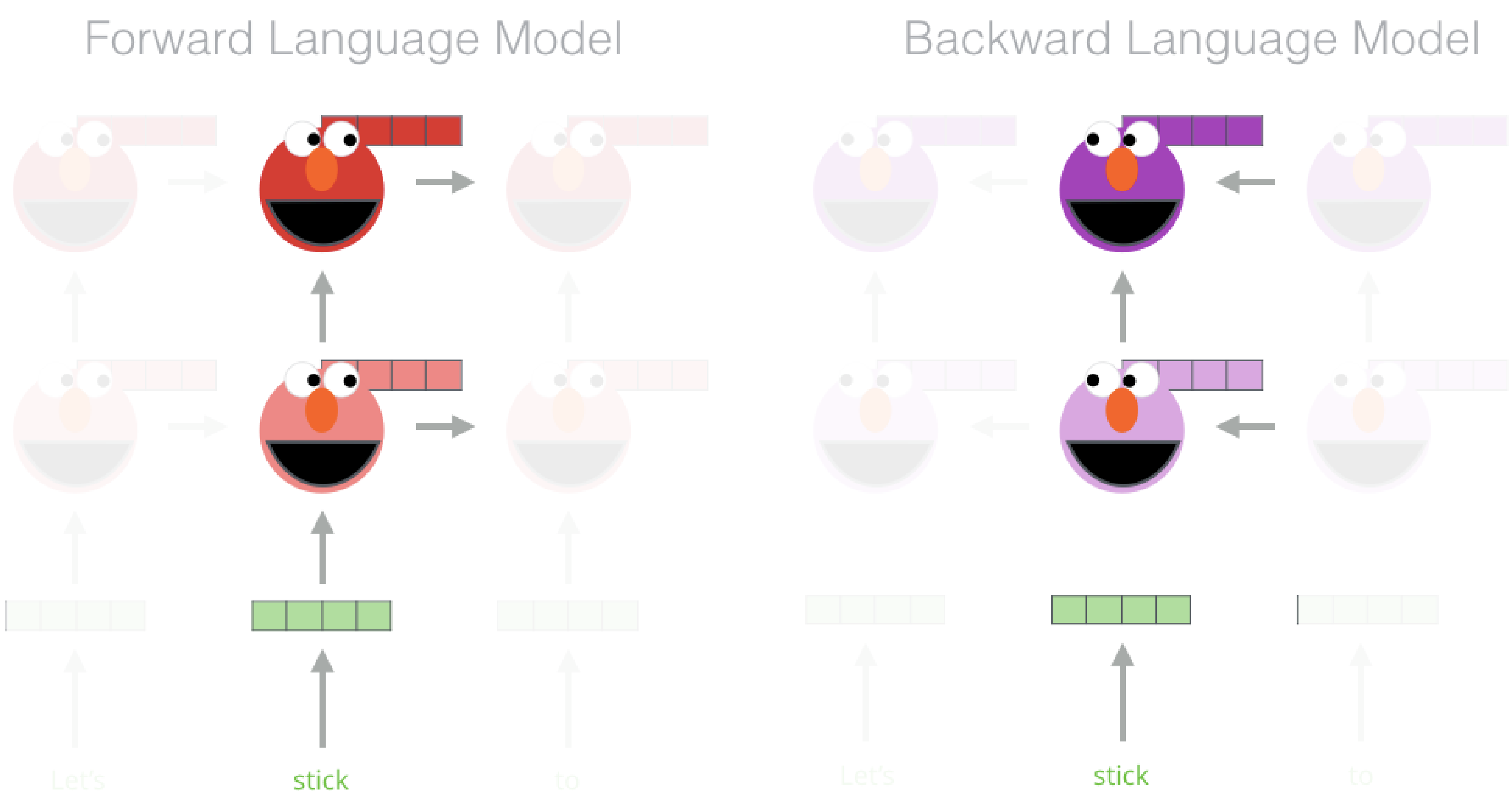


The Illustrated BERT, ELMo, and co., 2018

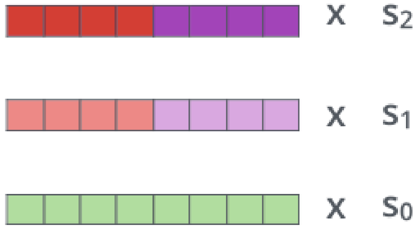
Embedding of ‘stick’ in ‘Let’s stick to’: step #2

1- Concatenate hidden layers





2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors

ELMo embedding of “stick” for this task in this context

Semantic search and similarity ranking

Ane Berasategi

18. July 2019

LMU
LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN