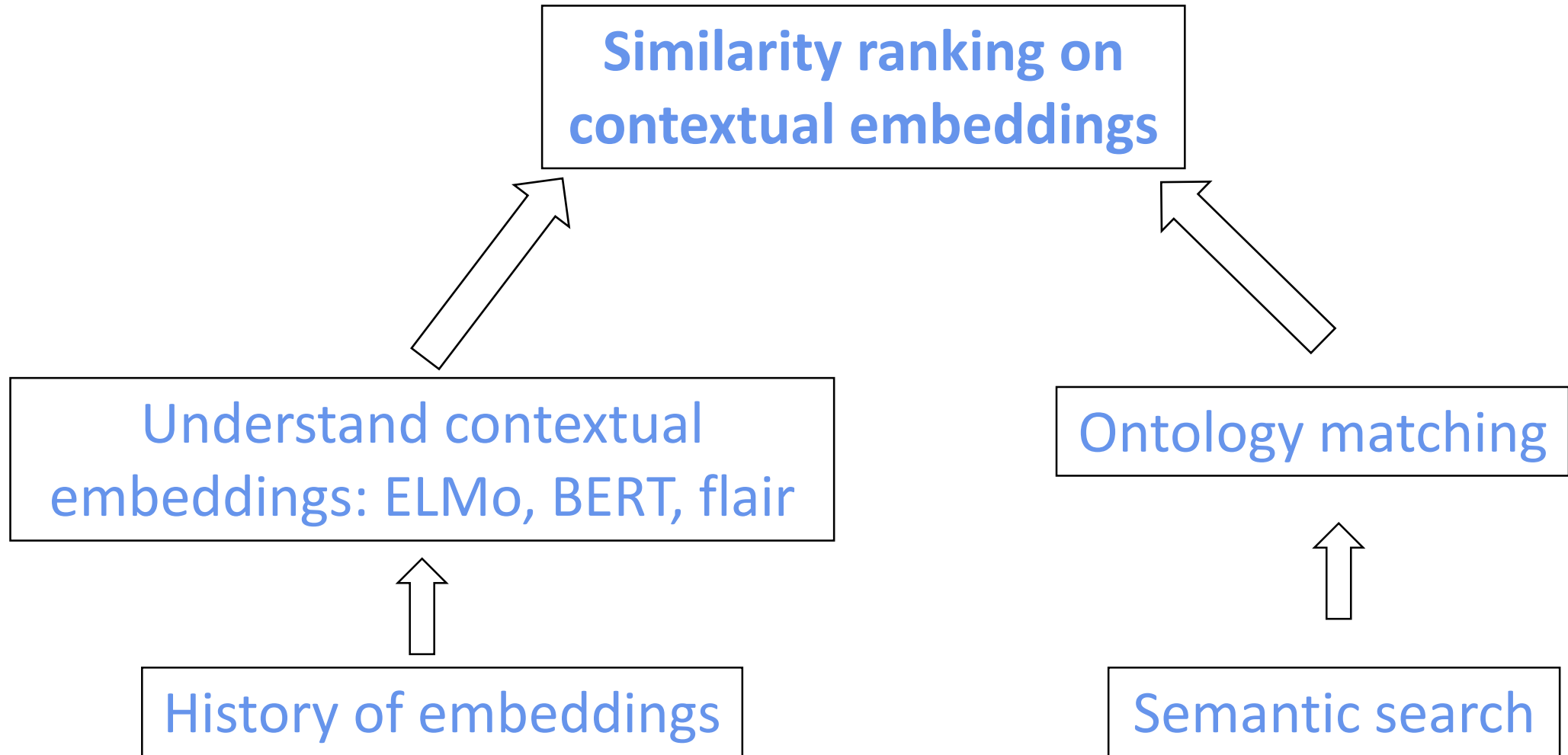


# Semantic search and similarity ranking

Ane Berasategi

18. July 2019





# Why is this interesting?

- Contextual embeddings have achieved unprecedented results in many tasks
  - Natural language understanding
  - Question answering
  - NER
- But what are they, how do they work, how do they represent language?
- Do they follow (my) intuition on sentence similarity?
  - A invited B for lunch vs.
  - A did not invite B for lunch vs.
  - B invited A for lunch

# Plan

## Part 1: Semantic search

- **Ontology matching**

## Part 2: Similarity ranking

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

## Part 3: Experiments

- Word order
- Lexical similarity
- Synonyms
- Out of vocab words

## Part 4: Conclusion

# 1. Semantic search

- **Lexical** search: literal matches of the query words
  - Anthony Hopkins age → relevant results
  - How old is Anthony Hopkins? → not relevant results
- **Semantic** search: search with meaning, understand the intention of the user
  - Why is my laptop overheating?
  - How many continents are there in the world?

# 1.1. Ontology matching

The search engine has a huge **knowledge graph / ontology** with past searches

- Ontology: a representation of semantic relations between documents.

Pipeline:

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

# Plan

## Part 1: Semantic search

- Ontology matching

## Part 2: Similarity ranking

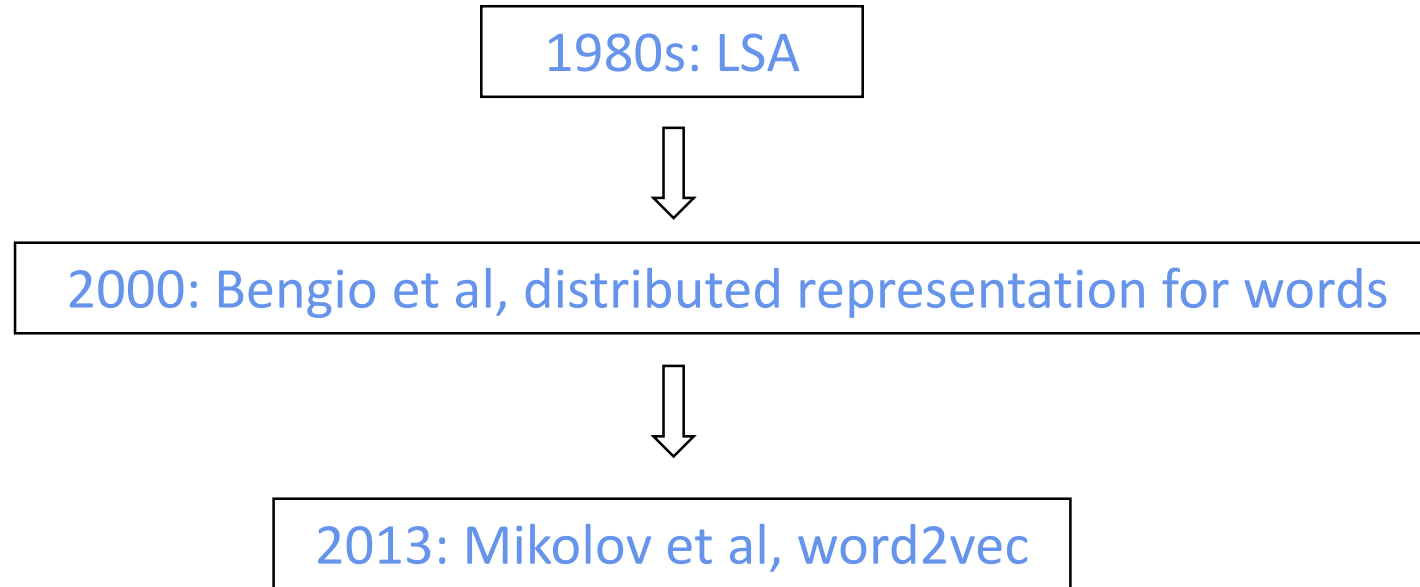
- Word vectors
- Contextualized word embeddings
- ELMo
- BERT
- Flair

## Part 3: Experiments

- Word order
- Lexical similarity
- Synonyms
- Out of vocab words

## Part 4: Conclusion

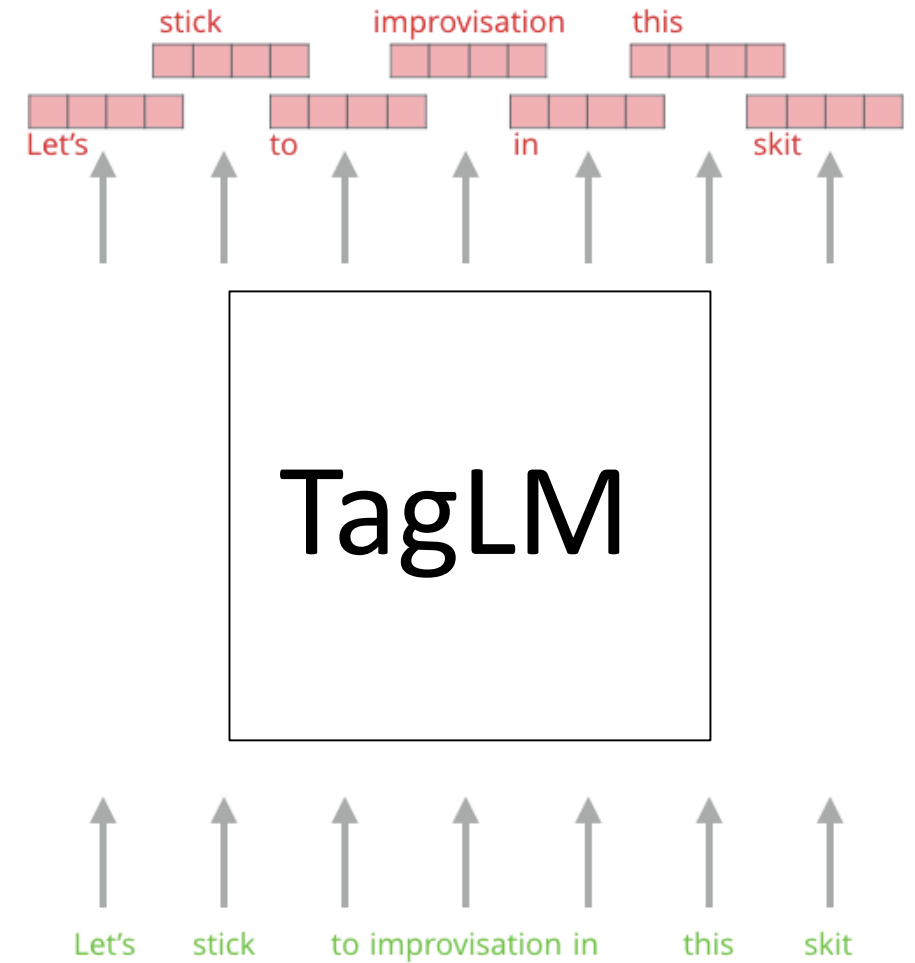
## 2.1. Word vectors: history



- Pre-trained word embeddings became the norm (word2vec, GloVe, FastText) as input to NNs
- Each word gets an embedding vector → Irrelevant of the context, part of speech, polysemy

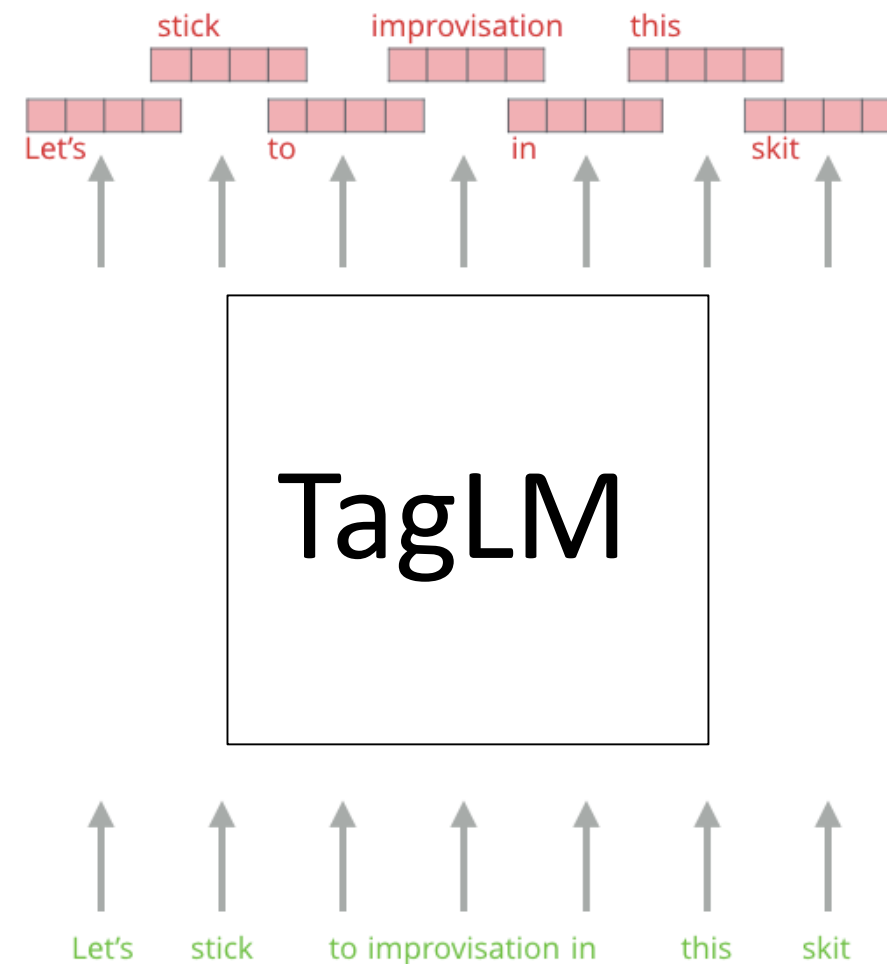


## 2.2. Contextualized word embeddings



## 2.2. Contextualized word embeddings

- TagLM: **semi-supervised** approach to add contextual embeddings to word embeddings from **bidirectional language models**
  - fLM + bLM embeddings has better performance than just fLM embeddings
  - The two LMs are trained independently
- Input for TagLM: sequences of words
- Output from TagLM: a single **context-independent** representation for each word, the output layer of the LSTM.



## 2.3. ELMo: Embeddings from Language Models

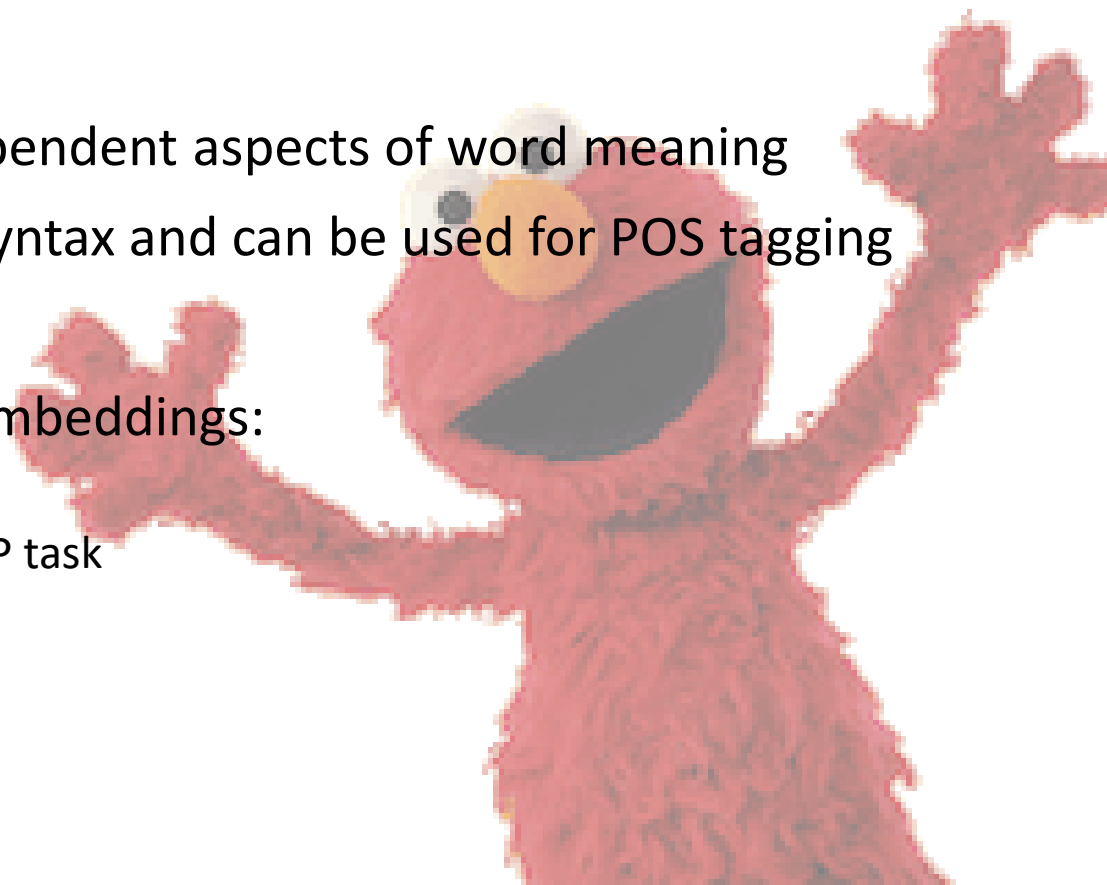
- Why just the last layer of the LSTM? Use all layers + share weights between forwLM and backLM

## 2.3. ELMo: Embeddings from Language Models

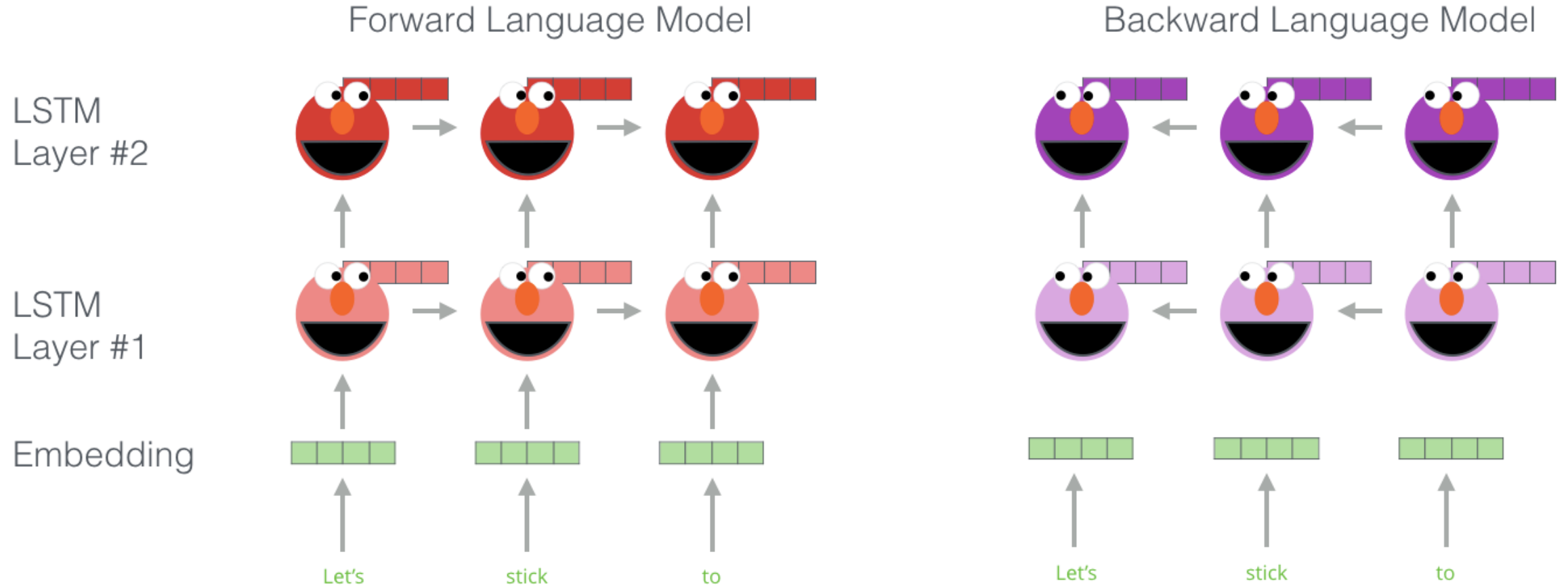
- Why just the last layer of the LSTM? Use all layers + share weights between forwLM and backLM
- Higher-level layers of the LSTM capture context-dependent aspects of word meaning
- Lower-level layers of the LSTM capture aspects of syntax and can be used for POS tagging

## 2.3. ELMo: Embeddings from Language Models

- Why just the last layer of the LSTM? Use all layers + share weights between forwLM and backLM
- 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:
  - different architectures for different NLP tasks
  - The embeddings are added as additional inputs to the NLP task



# Embedding of 'stick' in 'Let's stick to': step #1



# 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

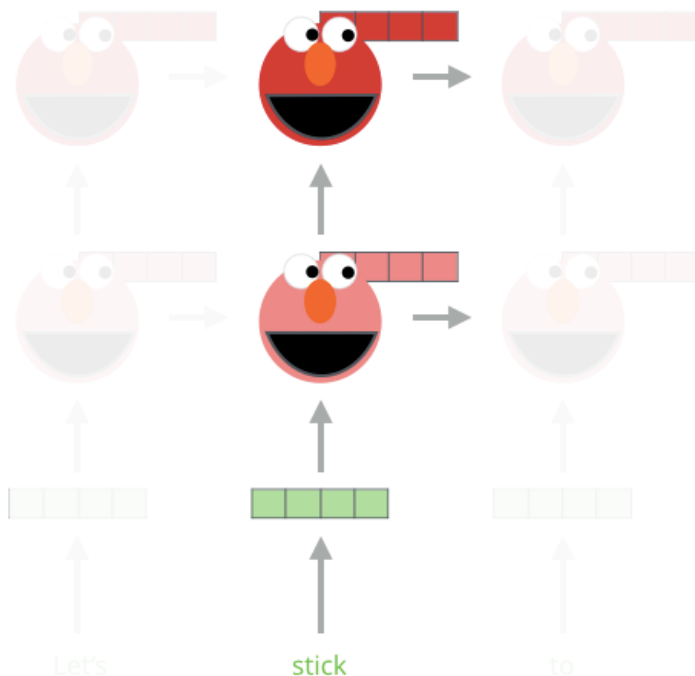


3- Sum the (now weighted) vectors

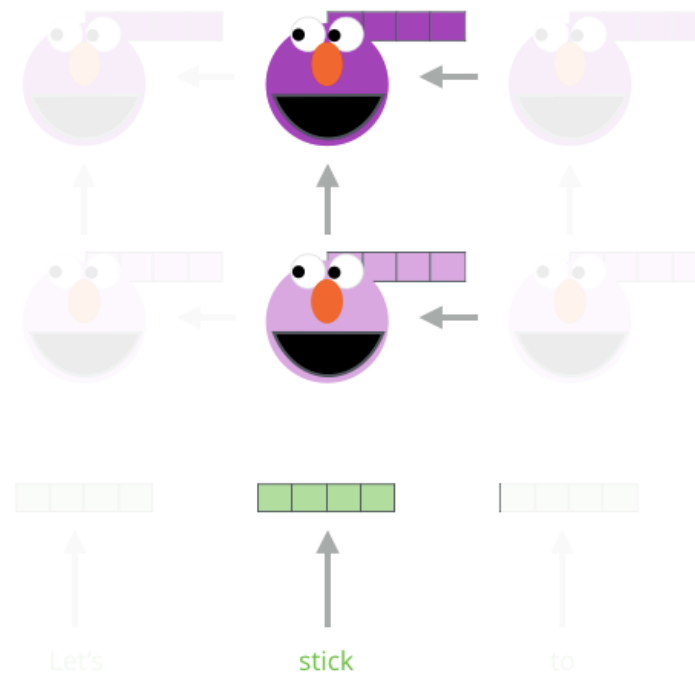


ELMo embedding of "stick" for this task in this context

Forward Language Model



Backward Language Model



## 2.4. The Transformer vs LSTM



## 2.4. The Transformer vs LSTM

- Recurrent, sequential models can't parallelize well → bottleneck at longer sequence lengths
- The Transformer:
  - No sequence, based solely on **self-attention** mechanisms
  - Deals with long-term dependencies better than LSTMs
  - More parallelizable than LSTMs

## 2.4. BERT: Bidirectional Embedding Representations from Transformers

[BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#), Devlin et al., 2018  
[The Illustrated BERT, ELMo, and co.](#), Jay Alammar, 2018

## 2.4. BERT: Bidirectional Embedding Representations from Transformers

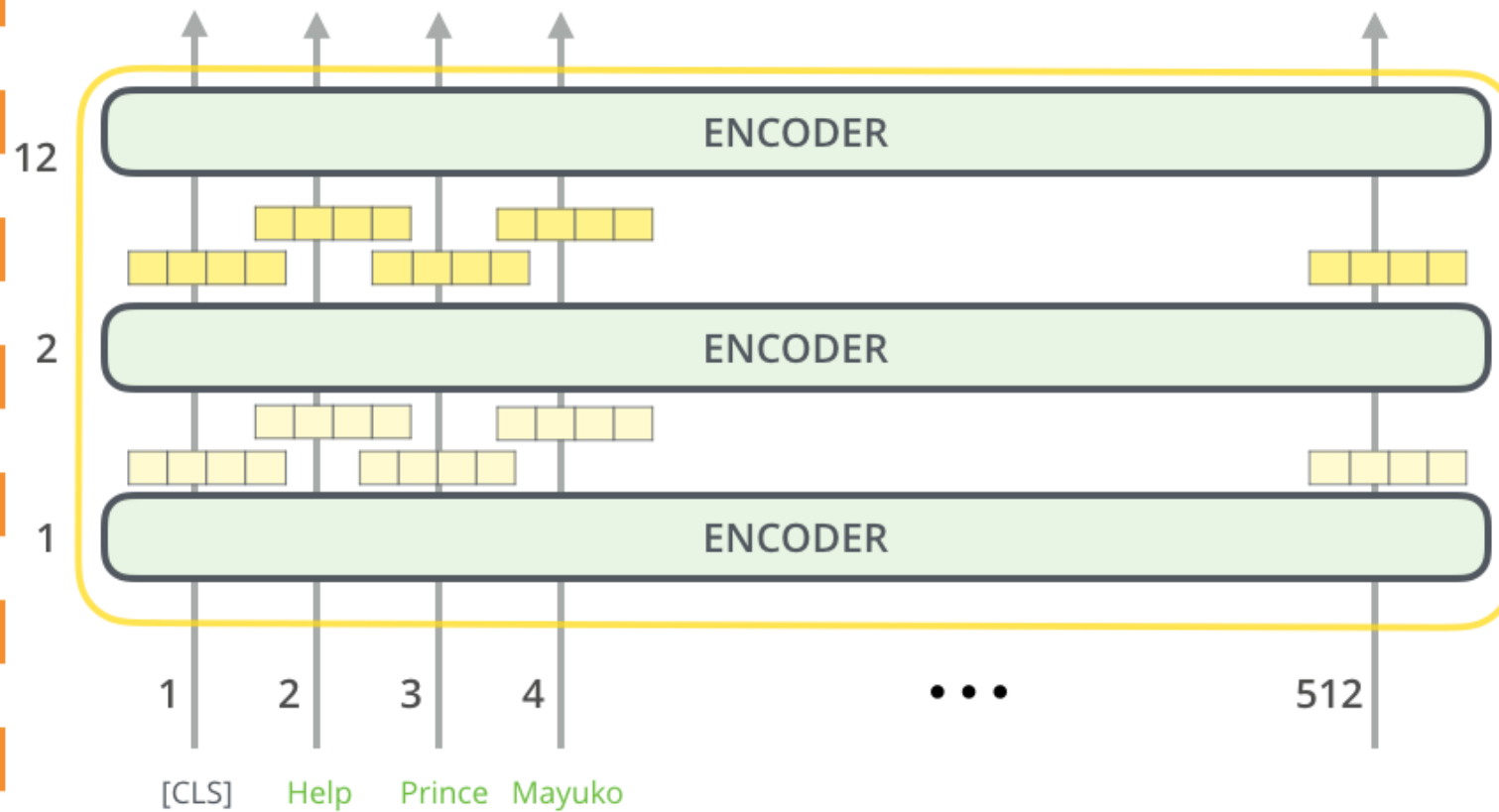
- BERT jointly conditions on both left and right context in all layers (unlike ELMo)
- Input to BERT: sub-words
- BERT is a **fine-tuned** approach
  - to use it with a specific task, all pre-trained parameters are fine-tuned end-to-end

## 2.4. BERT: Bidirectional Embedding Representations from Transformers

- BERT jointly conditions on both left and right context in all layers (unlike ELMo)
- Input to BERT: sub-words
- BERT is a **fine-tuned** approach
  - to use it with a specific task, all pre-trained parameters are fine-tuned end-to-end
- **Pre-trained BERT can be used to create contextualized embeddings**

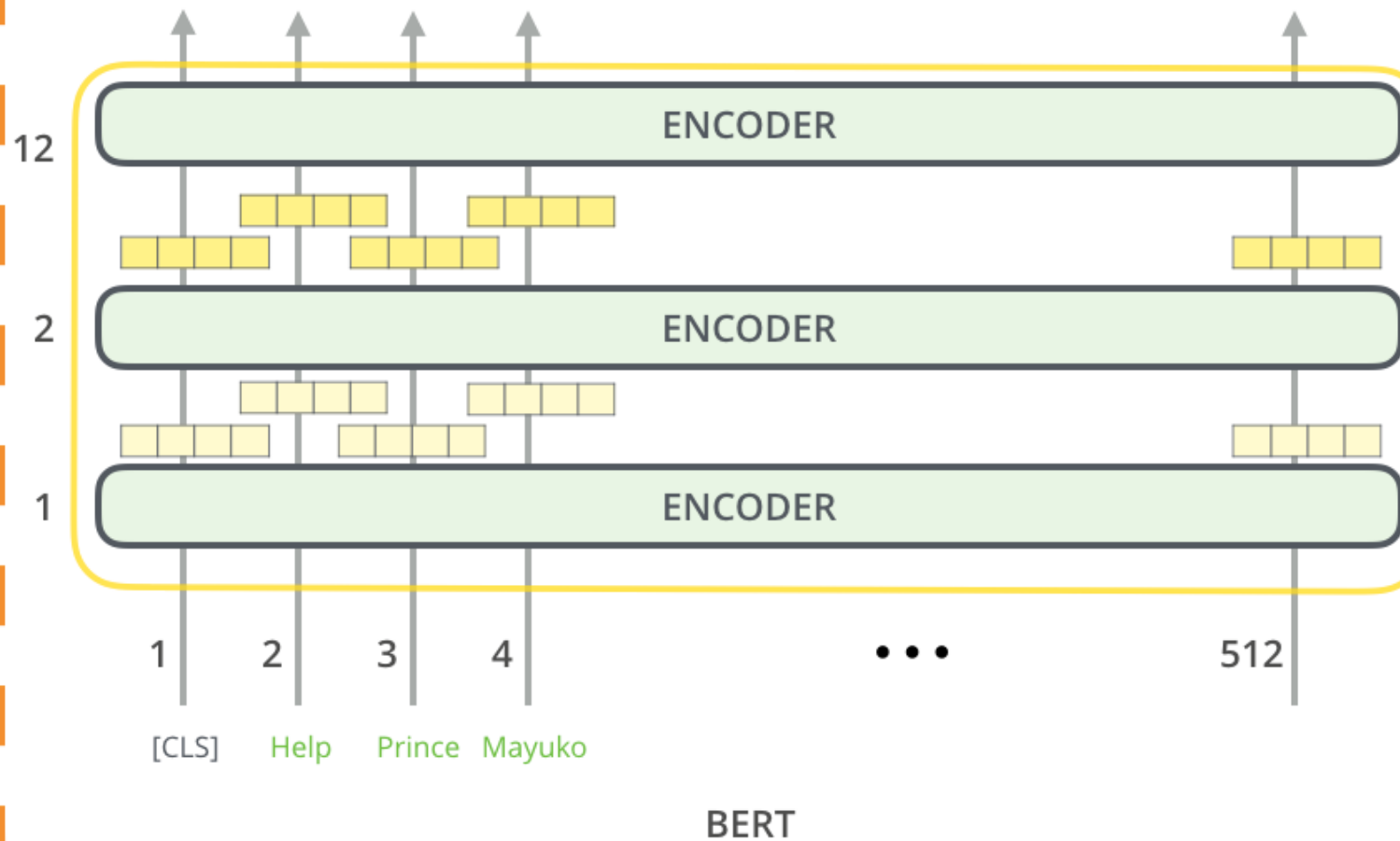


## Generate Contextualized Embeddings

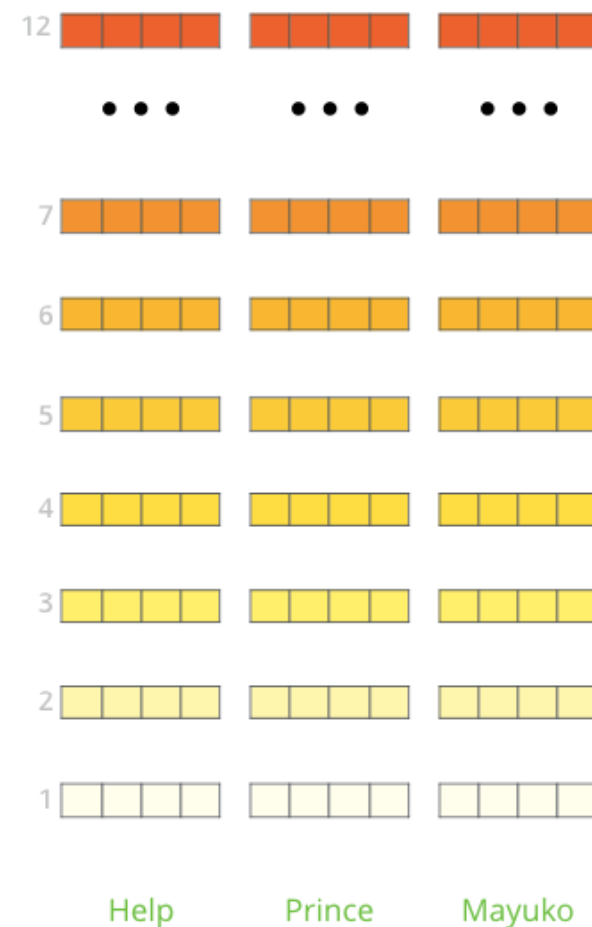


BERT

## Generate Contextualized Embeddings



The output of each encoder layer along each token's path can be used as a feature representing that token.



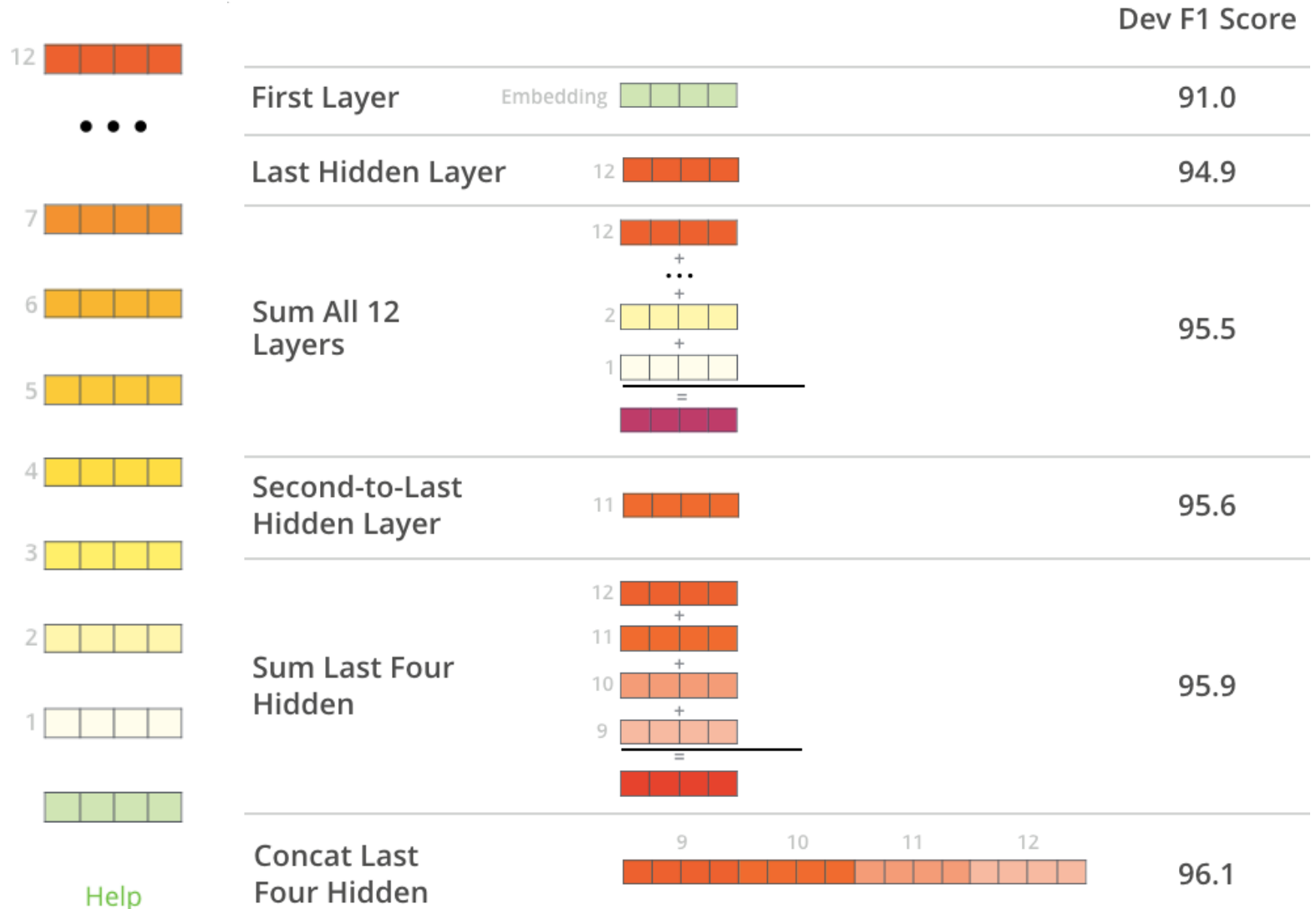
But which one should we use?

# Which contextualized embedding do we take for 'Help' in this context?

- The layer(s) to use depend on the NLP task
- Some tasks look at similarities between sentences, others care more about syntax and POS tagging
- Higher-level layers of the LM capture context-dependent aspects of word meaning
- Lower-level layers of the LM capture aspects of syntax
- For NER CoNLLTask 2003:

# Which contextualized embedding do we take for 'Help' in this context?

- The layer(s) to use depend on the NLP task
- Some tasks look at similarities between sentences, others care more about syntax and POS tagging
- Higher-level layers of the LM capture context-dependent aspects of word meaning
- Lower-level layers of the LM capture aspects of syntax
- For NER CoNLLTask 2003:





## 2.5. flair

- Another type of contextual string embeddings, **trained without an explicit notion of what a word is**
  - Models words as sequences of characters
  - The same word has different embeddings depending on its context
- Inputs to flair: **characters**
- Character-level LM is independent of tokenization and has a fixed vocabulary
- Deal with rare and out of vocabulary words better

# Plan

## Part 1: Semantic search

- Ontology matching

## Part 2: Similarity ranking

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

## Part 3: Experiments

- Word order
- Lexical similarity
- Synonyms
- Out of vocab words

## Part 4: Conclusion

# 3. Experiments

- Experiment with contextual embeddings in English and multilingual (basque)
- Import ELMo, BERT-cased, BERT-uncased, and flair embeddings
  - BERT-uncased: text is lowercased before tokenisation and strips out accent markers
  - BERT paper: “uncased is typically better unless you know that case information is relevant to your task”
- For each experiment, create a reference sentence and several (dis-)similar sentences and rank them based on [intuitive similarity](#)
- Obtain contextual embeddings for each embedding type
- Calculate similarity between embedding(reference) and embedding(similar\_sentence)
- Visualizations

### 3. Experiments

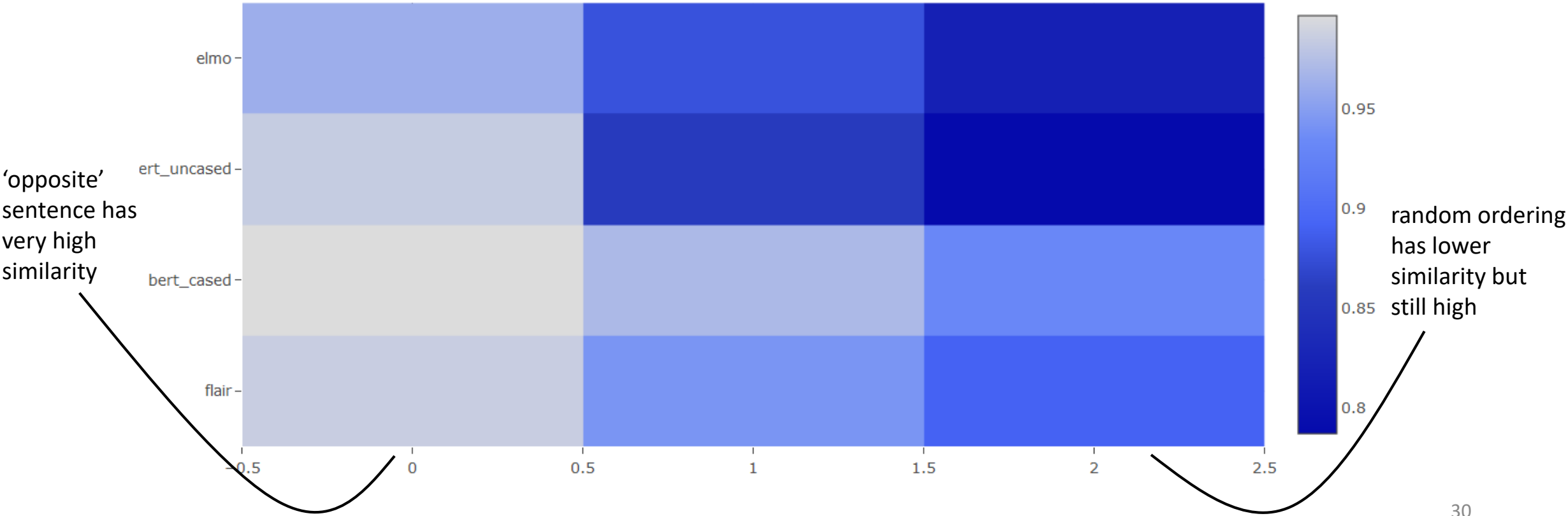
	Input text type	Vocabulary size
<b>ELMo</b>	word	700k words
<b>BERT</b>	subword	30k tokens
<b>BERT</b> multilingual	subword	120k tokens in total for 104 languages
<b>flair</b>	character	~256 characters (depending on the language)

### 3.1. Does word order matter?

<i>the doctor invited the patient for lunch</i>	Change
<i>0. the patient invited the doctor for lunch</i>	Switch subject and object ('opposite' sentence)
<i>1. the lunch invited the doctor for the patient</i>	Change semantic role
<i>2. for invited patient the doctor the lunch</i>	Random ordering

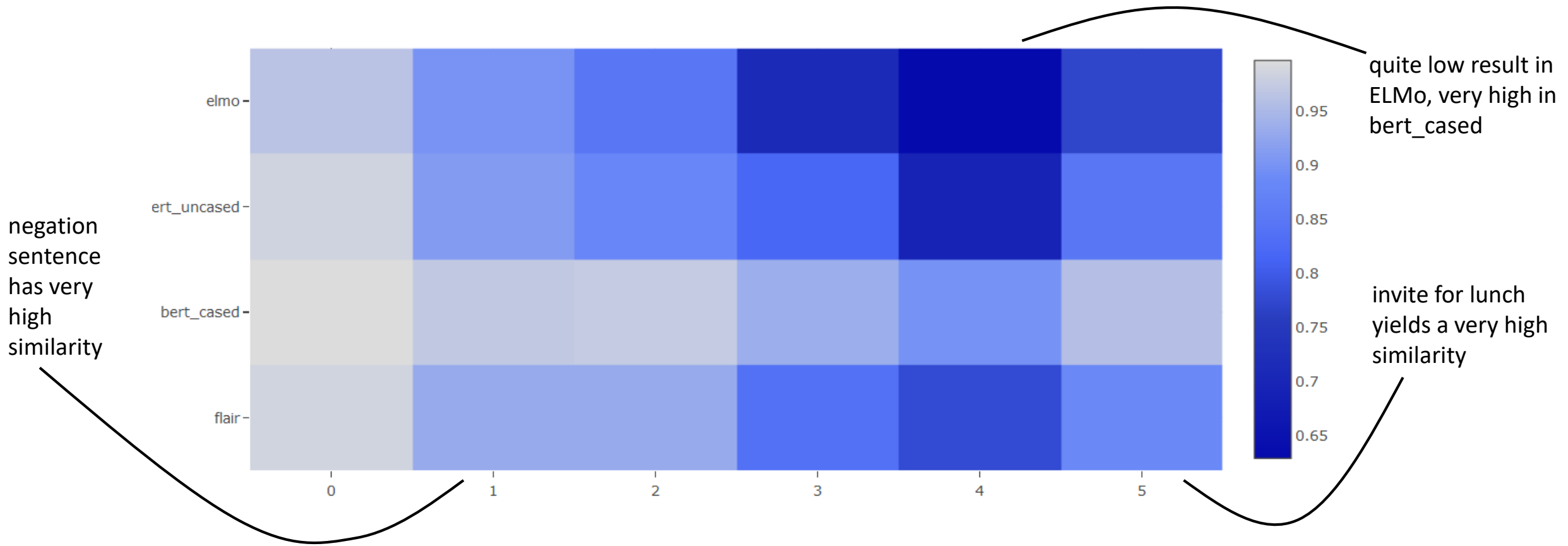
### 3.1. Does word order matter?

<i>the doctor invited the patient for lunch</i>	Change
<i>0. the patient invited the doctor for lunch</i>	Switch subject and object ('opposite' sentence)
<i>1. the lunch invited the doctor for the patient</i>	Change semantic role
<i>2. for invited patient the doctor the lunch</i>	Random ordering



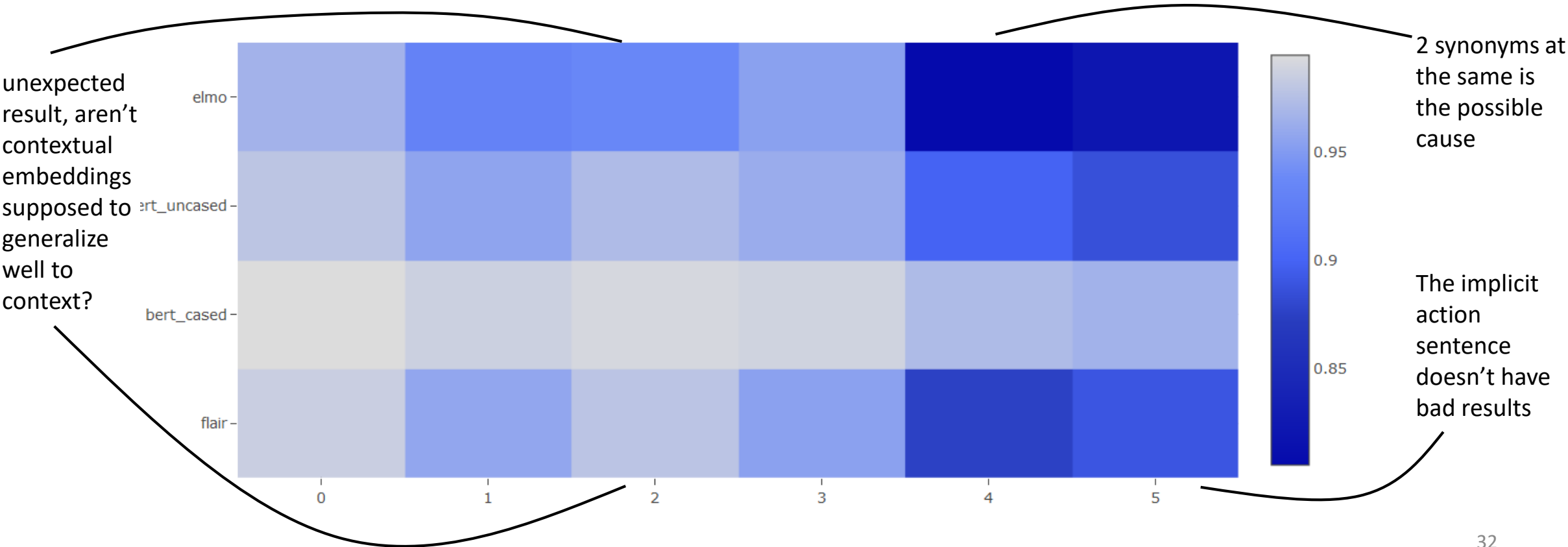
# 3.2. What is the impact of lexical similarity?

<i>the doctor invited the patient for lunch</i>	Change		Change
0. the <i>patient</i> invited the <i>doctor</i> for lunch	'opposite' sentence	3. <i>the doctor</i> told <i>the patient</i> he was a fraud	Subj. and obj. overlap
1. the doctor did <i>not</i> invite the patient for lunch	negation	4. that is a matter between <i>the doctor</i> and <i>the patient</i>	Subj. and obj. overlap
2. the <i>child</i> invited the <i>grandfather</i> for lunch	Subj. and obj. change	5. the child and the grandfather got <i>invited for lunch</i>	Invite, lunch overlap



### 3.3. What is the impact of synonyms?

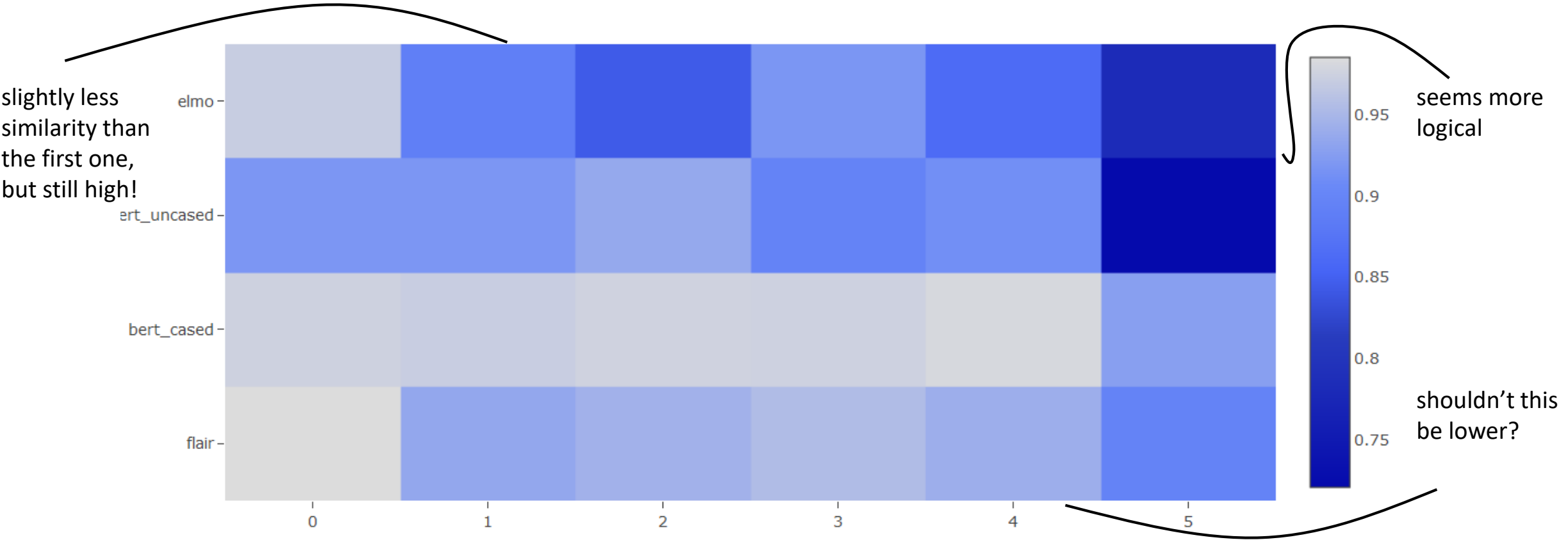
<i>the doctor invited the patient for lunch</i>	Change		Change
0. <i>the <b>surgeon</b> invited the patient for lunch</i>	Synonym of subj.	3. <i>the doctor invited the patient for a <b>meal</b></i>	More general word
1. <i>the doctor invited the <b>sick person</b> for lunch</i>	More general synonym of obj.	4. <i>the doctor <b>took</b> the patient <b>out</b> for <b>tea</b></i>	More general expression
2. <i>the <b>professor</b> invited the patient for lunch</i>	Synonym in different context	5. <i>the doctor <b>paid</b> for the patient's <b>lunch</b></i>	Implicit action





# 3.4. What is the impact of out of vocabulary words?

<i>the doctor invited the patient for lunch</i>	Change		Change
0. the doctor <i>invitted</i> the patient for lunch	Typo	3. the doctor invited the patient for <i>sushi</i>	Specific food
1. the doctor <i>kartoffeled</i> the patient for lunch	Wrong word	4. the doctor invited the <i>patent</i> for lunch	Typo / different word
2. <i>Stefan</i> invited the patient for lunch	Proper name	5. the doctor invited the patent for <i>linch</i>	Typo / different word



### 3. Experiment takeaways

- BERT overall shows extremely high similarities
  - *Mystery*: especially BERT cased: lowest achieved was 85% with random sentences
- Sentences in the same semantic space are evaluated as similar, even negation or opposite sentences
- Word order is not very relevant → bag of word-like approach?
  - In Basque, random ordering has ~75% similarity in ELMo and flair and 95% in BERT
- A lexical overlap of 50% can raise similarity to 90%
  - *Mystery*: ELMo (word-based) shows much lower similarities than flair (character-based)
  - In Basque as well, while BERT shows similarities of 90%+ in all cases
- With synonyms that change context, similarities are still high ('professor invited patient')
- Typos don't affect similarity much

# Plan

## Part 1: Semantic search

- Ontology matching

## Part 2: Similarity ranking

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

## Part 3: Experiments

- Word order
- Lexical similarity
- Synonyms
- Out of vocab words

## Part 4: Conclusion

## 4. Conclusion

- Semantic search → ontology matching → document ranking → experiments on sentence ranking with contextual embeddings
- In my specific examples:
  - word order doesn't seem to be relevant
  - they don't generalize very well to context
  - slightly worse results for basque: because multilingual, low-resource language?
  - typos don't have much impact
- Future work
  - research extreme high similarities of *bert\_cased*
  - research cause of similarity differences in *bert\_cased* and *bert\_uncased*
  - research impact of word ordering and lexical overlap
  - research impact of word-based embeddings vs character-based ones
  - try more phenomena/examples
  - Visualization: [T-SNE](#)

# Semantic search and similarity ranking

Ane Berasategi

18. July 2019

