Word Embeddings Revisited: Contextual Embeddings

THE UNIVERSITY OF UTAH

Overview

- Word types and tokens
- Training contextual embeddings
- Embeddings from Language Models (ELMo)

Overview

- Word types and tokens
- Training contextual embeddings
- Embeddings from Language Models (ELMo)

How many words…

How many words are in this sentence below?

(Ignoring capitalization and the comma)

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 capitalization and the comma)

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

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 capitalization and the comma)

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

Seventeen words

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

Only nine words

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

How many words…

How many words are in this sentence below?

(Ignoring capitalization and the comma)

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

When we say “words”, which interpretation do we mean?

words

words

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

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

How many words…

How many words are in this sentence below?

(Ignoring capitalization and the comma)

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

When we say “words”, which interpretation do we mean?

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

ask,
do,
do, for, your, country

Word types

Types are abstract and unique objects

- Sets or concepts – e.g. there is only one thing called *laptop*
- Think entries in a dictionary

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

Seventeen words

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

Only nine words

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

Word tokens

Tokens are instances of the types

- Usage of a concept – this *laptop*, my *laptop*, your *laptop*

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

Seventeen words

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

Only nine words

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

The type-token distinction

- A larger philosophical discussion
  - See the Stanford Encyclopedia of Philosophy for a nuanced discussion

- The distinction is broadly applicable and we implicitly reason about it

"We got the same gift”

We got the same gift type vs We got the same gift token

Word embeddings 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

Word embeddings 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
- Can we embed word *tokens* instead?

Word embeddings 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

- Can we embed word *tokens* instead?

- What makes a word token different from a word type?
  - We have the context of the word to inform the embedding
  - We may be able to resolve word sense ambiguity

Overview

- Word types and tokens
- Training contextual embeddings
- Embeddings from Language Models (ELMo)

Word embeddings should…

- Unify superficially different words
  - *bunny* and *rabbit* are similar

Word embeddings 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

Word embeddings 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 embeddings should…

Type embeddings can address the first two requirements

- 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 embeddings 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

Type embeddings can address the first two requirements

Word sense can be disambiguated using the context ⇒ *contextual embeddings*

Type embeddings vs token embeddings

Type embeddings can be thought of as a lookup table

- Map words to vectors independent of any context
- A big matrix

Token embeddings should be functions

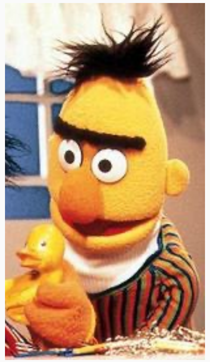
- Construct embeddings for a word on the fly
- There is no fixed “bank” embedding, the usage decides what the word vector is

Contextual embeddings

The big new thing in 2017-18

Two popular models

ELMo
Peters et al 2018



BERT
Devlin et al 2018

Other work in this direction: ULMFit [Howard and Ruder 2018]

Contextual embeddings

The big new thing in 2017-18

ELMo

BERT

We will look at ELMo now. We will visit BERT later in the semster

Overview

- Word types and tokens
- Training contextual embeddings
- Embeddings from Language Models (ELMo)

Embeddings from Language Models (ELMo)

Two key insights

1. The embedding of a word type should depend on its context
   - But the size of the context should not be fixed

     No Markov assumption
   - Need arbitrary context – use an bidirectional RNN

Embeddings from Language Models (ELMo)

Two key insights

1. The embedding of a word type should depend on its context
   - But the size of the context should not be fixed

     No Markov assumption
   - Need arbitrary context – use an bidirectional RNN

2. Language models are already encoding the contextual meaning of words
   - Use the internal states of a language model as the word embedding

The ELMo model

- Embed word types into a vector
  - Can use pre-trained embeddings (GloVe)
  - Can train a character-based model to get a context-independent embedding

The ELMo model

- Embed word types into a vector
  - Can use pre-trained embeddings (GloVe)
  - Can train a character-based model to get a context-independent embedding

- Deep bidirectional LSTM language model over the embeddings
  - Two layers of BiLSTMs, but could be more

The ELMo model

- Embed word types into a vector
  - Can use pre-trained embeddings (GloVe)
  - Can train a character-based model to get a context-independent embedding
- Deep bidirectional LSTM language model over the embeddings
  - Two layers of BiLSTMs, but could be more
- Loss = language model loss
  - Cross-entropy over probability of seeing the word in a context

Specific training/modeling details in the paper

The ELMo model

- Embed word types into a vector
  - Can use pre-trained embeddings (GloVe)
  - Can train a character-based model to get a context-independent embedding
- Deep bidirectional LSTM language model over the embeddings
  - Two layers of BiLSTMs, but could be more

Hidden state of each BiLSTM cell = embedding for the word

- Loss = language model loss
  - Cross-entropy over probability of seeing the word in a context

The ELMo model

- Embed word types into a vector
- Deep bidirectional LSTM language model over the embeddings
  - Two layers of BiLSTMs, but could be more
- Hidden state of each BiLSTM cell = embedding for the word
  - Which one do we use?
- The ELMo answer: All of them

Using ELMo in a task

sentence →

ELMo

→ Multiple embeddings for each word

Using ELMo in a task

sentence →

ELMo

→ Multiple embeddings for each word

Layer 2 hidden state h_{2i} for word i
Layer 1 hidden state h_{1i} for word i
Base word embeddings h_{0i} for word i

Using ELMo in a task

sentence →

ELMo

→ Multiple embeddings for each word

Layer 2 hidden state h_{2i} for word i
Layer 1 hidden state h_{1i} for word i
Base word embeddings h_{0i} for word i

$$\mathrm{ELMo}_i^{task} = \gamma^{task}\left(s_0^{task} h_{0i} + s_1^{task} h_{1i} + s_2^{task} h_{2i}\right)$$

Linear interpolation of all the embeddings

Using ELMo in a task

sentence → 
ELMo → Multiple embeddings for each word

Layer 2 hidden state h_{2i} for word i
Layer 1 hidden state h_{1i} for word i
Base word embeddings h_{0i} for word i

$$\mathrm{ELMo}_i^{task} = \gamma^{task}\left(s_0^{task} h_{0i} + s_1^{task} h_{1i} + s_2^{task} h_{2i}\right)$$

Linear interpolation of all the embeddings

The interpolation term is part of the task parameters

Using ELMo in a task

sentence →

ELMo

→ Multiple embeddings for each word

Layer 2 hidden state h_{2i} for word i
Layer 1 hidden state h_{1i} for word i
Base word embeddings h_{0i} for word i

$$\mathrm{ELMo}_i^{task} = \gamma^{task}\left(s_0^{task} h_{0i} + s_1^{task} h_{1i} + s_2^{task} h_{2i}\right)$$

Linear interpolation of all the embeddings

The interpolation term is part of the task parameters

Also a scaling term that scales the entire ELMo vector

Using ELMo in a task

sentence →

ELMo

→ Multiple embeddings for each word

Layer 2 hidden state h_{2i} for word i
Layer 1 hidden state h_{1i} for word i
Base word embeddings h_{0i} for word i

$$\mathrm{ELMo}_i^{task} = \gamma^{task}\left(s_0^{task} h_{0i} + s_1^{task} h_{1i} + s_2^{task} h_{2i}\right)$$

Linear interpolation of all the embeddings

The interpolation term is part of the task parameters

Also a scaling term that scales the entire ELMo vector

Could optionally fine-tune the entire language model on task data

Evaluating ELMo

General idea

- Pick an NLP task that uses a neural network model
- Replace the context-independent word embeddings with ELMo
  - Or perhaps append to the context independent embeddings
- Train the new model with these embeddings
  - Also train the ELMo parameters: $\gamma, s_j' s$
- Compare using the official metric for the task

ELMo improves a broad range of tasks

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table from Peters et al 2018

Since the paper was published, ELMo showed similar improvements in other tasks as well

Coming up…

We will revisit context dependent embeddings one more time

- BERT – uses the transformer architecture