Tutorial 4
Word Embedding

Xi Chen
E-mail: xichen7@link.cuhk.edu.cn

Outline

- Recap
- TF-IDF Example
- Word2vec Example
- BERT Example

Recap

vector semantics, which instantiates this linguistic vector

semantics embeddings hypothesis by learning representations of the meaning of words, called embeddings, directly from their distributions in texts.

- static embeddings
  - Word2Vec, Glove, etc.
- contextualized embeddings
  - BERT, etc.

TF-IDF Example

TF-IDF is used by search engines to better understand the content that is undervalued. For example, when you search for “Coke” on Google,

Google may use TF-IDF to figure out if a page titled “COKE” is about:

a) Coca-Cola.
b) Cocaine.
c) A solid, carbon-rich residue derived from the distillation of crude oil.
d) A county in Texas.

TF-IDF Example

For a term *t* in document *d*, the weight *Wt,d* of term *t* in document *d* is given by:

- $Wt,d = TFt,d * log\ (N/DFt)$

Where:

- *TFt,d* is the number of occurrences of *t* in document *d*.
- *DFt* is the number of documents containing the term *t*.
- *N* is the total number of documents in the corpus.

TF-IDF Example

How to compute TF-IDF?

Suppose we are looking for documents using the query Q and our database is composed of the documents D1, D2, and D3.

- Q: The cat.
- D1: The cat is on the mat.
- D2: My dog and cat are the best.
- D3: The locals are playing.

TF-IDF Example

- Let's compute the TF scores of the words "the" and "cat" (i.e. the query words) with respect to the documents D1, D2, and D3.

TF("the", D1) = 2
TF("the", D2) = 1
TF("the", D3) = 1
TF("cat", D1) = 1
TF("cat", D2) = 1
TF("cat", D3) = 0

TF-IDF Example

- Let's compute the IDF scores of the words "the" and "cat"

  IDF("the") = log(3/3) = log(1) = 0
  IDF("cat") = log(3/2) = 0.18

- Multiplying TF and IDF gives the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

  TF-IDF("the", D1) = 2 * 0 = 0
  TF-IDF("the, D2) = 1 * 0 = 0
  TF-IDF("the", D3) = 1 * 0 = 0
  TF-IDF("cat", D1) = 1 * 0.18= 0.18
  TF-IDF("cat, D2) = 1 * 0.18= 0.18
  TF-IDF("cat", D3) = 0 * 0 = 0

TF-IDF Example

- Order the documents according to the TF-IDF scores of their words.

Average TF-IDF of D1 = (0 + 0.18) / 2 = 0.09
Average TF-IDF of D2 = (0 + 0.18) / 2 = 0.09
Average TF-IDF of D3 = (0 + 0) / 2 = 0

As a conclusion, when performing the query “The cat” over the collection of documents D1, D2, and D3, the ranked results would be:

D1=D2>D3

TF-IDF Example

- Implement

```
# coding:utf-8
import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

# corpus
corpus=["The cat is on the mat.",
        "My dog and cat are the best.",
        "The locals are playing."]

vectorizer = CountVectorizer(stop_words=None)

#compute the term frequency
X = vectorizer.fit_transform(corpus)

#get all words in bag
word = vectorizer.get_feature_names_out()
df_word = pd.DataFrame(X.toarray(),columns=vectorizer.get_feature_names_out())

transformer = TfidfTransformer(smooth_idf=True,norm='l2',use_idf=True)

#compute tfidf
tfidf = transformer.fit_transform(X)

#print idf
df_word_idf = pd.DataFrame(list(zip(vectorizer.get_feature_names_out(),transformer.idf_)),columns=['word','idf'])
print(df_word_idf.T)

df_word_tfidf = pd.DataFrame(tfidf.toarray(),columns=vectorizer.get_feature_names_out())
print(df_word_tfidf.T)
```

Word2vec Example

The first really influential dense word embeddings

- **Main idea:**

Use a classifier to predict which words appear in the context of (i.e. near) a target word (or vice versa)
This classifier induces a dense vector representation of words (embedding)

Words that appear in **similar contexts** (that have high distributional similarity) will have very **similar vector representations.**

Word2vec Example

The first really influential dense word embeddings

- **Main idea:**

Use a classifier to predict which words appear in the context of (i.e. near) a target word (or vice versa) This classifier induces a dense vector representation of words (embedding)

Words that appear in **similar contexts** (that have high distributional similarity) will have very **similar vector representations.**

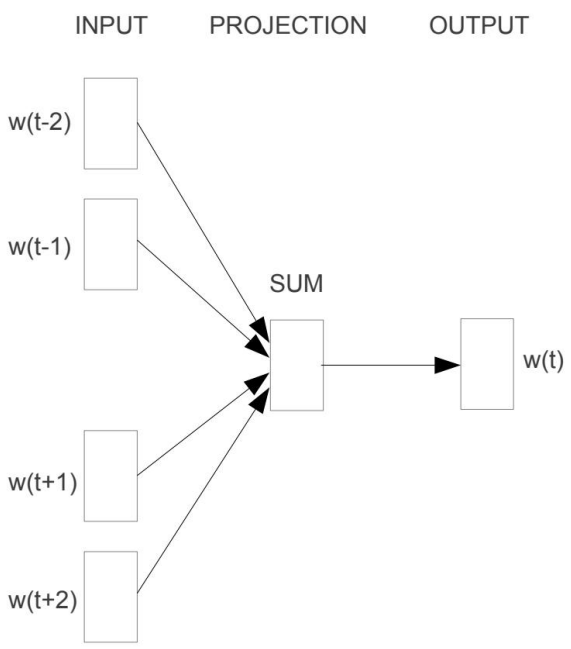
Word2vec Example

- Two ways to think about Word2Vec:
  - a simplification of neural language models
  - a binary logistic regression classifier

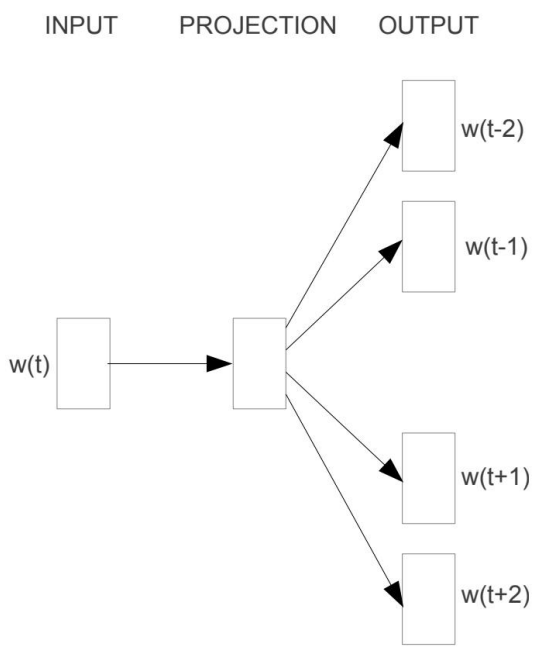
- **Variants of Word2Vec**
  - CBOW(Continuous Bag of Words)
  - Skip-Gram

Word2vec Example

- CBOW/Skip-Gram Architectures



CBOW



Skip-gram

Word2vec Example

- CBOW: predict target from context

Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1 c2 t c3 c4

Given the surrounding context words (tablespoon, of, jam, a), predict the target word (apricot).

Input: each context word is a one-hot vector
Projection layer: map each one-hot vector down to a dense D-dimensional vector
Output: predict the target word with softmax

Word2vec Example

- **Skipgram:** predict context from target

Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

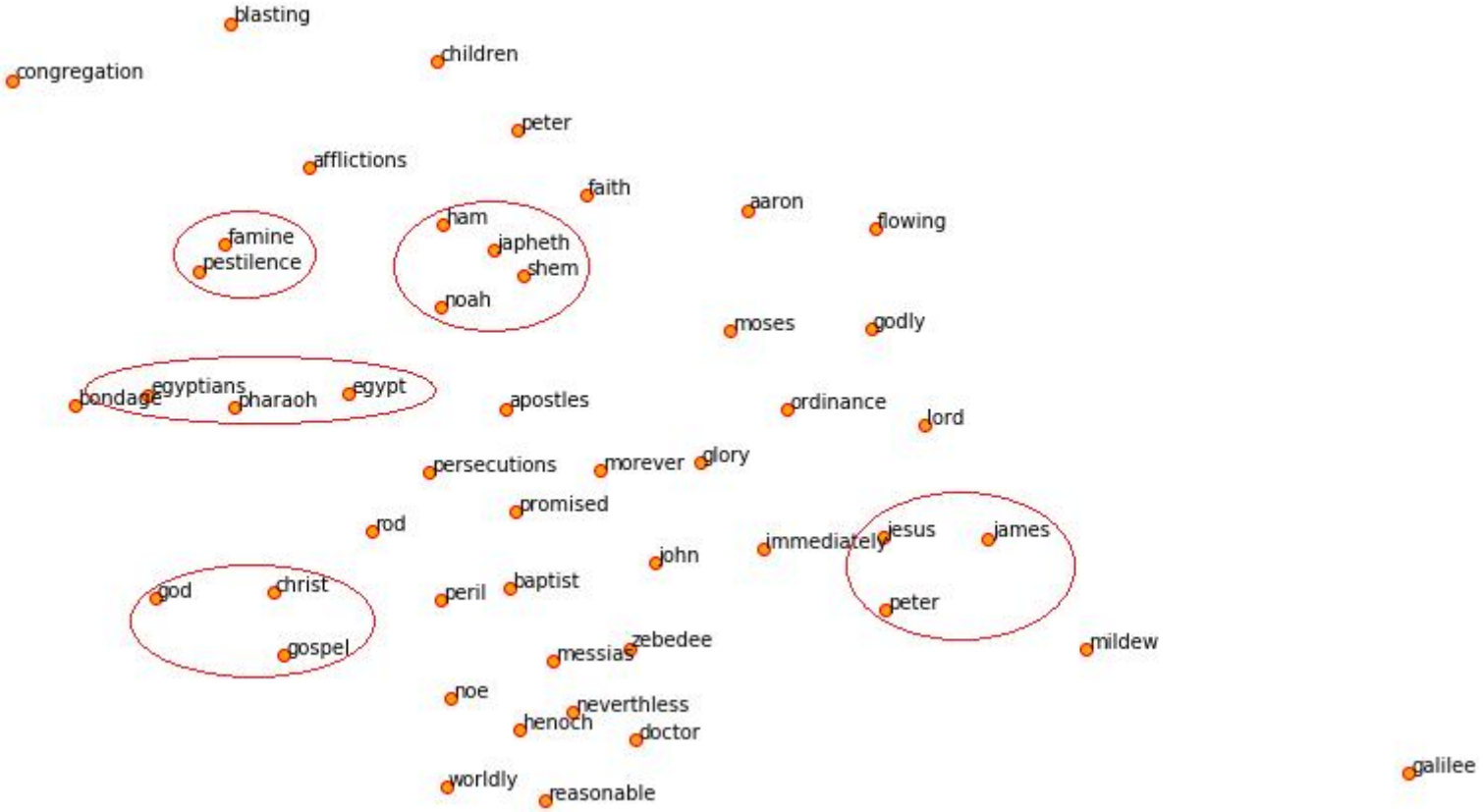
c1 c2 t c3 c4

Given the target word (apricot), predict the surrounding context words (tablespoon, of, jam, a),

Input: each target word is a one-hot vector
Projection layer: map each one-hot vector down to a dense D-dimensional vector
Output: predict the context word with softmax

Word2vec Example

- Visualize the word2vec



BERT Example

What is BERT?

BERT (Bidirectional Encoder Representations from Transformers), released in late 2018. BERT is a method of pretraining language representations that was used to create models that NLP practicioners can then download and use for free. You can either use these models to extract high quality language features from your text data, or you can fine-tune these models on a specific task (classification, entity recognition, question answering, etc.) with your own data to produce state of the art predictions.

Why BERT embeddings?

- useful for keyword/search expansion, semantic search and information retrieval
- these vectors are used as high-quality feature inputs to downstream models.

https://www.youtube.com/watch?v=xI0HHN5XKDo&ab_channel=CodeEmporium

BERT Example

For example, given two sentences:

"The man was accused of robbing a **bank**."

"The man went fishing by the **bank** of the river."

Word2Vec would produce the same word embedding for the word "**bank**" in both sentences, while under BERT the word embedding for "**bank**" would be different for each sentence.

Aside from capturing obvious differences like polysemy, the context-informed word embeddings capture other forms of information that result in more accurate feature representations, which in turn results in better model performance.

BERT Example

Contextually dependent vectors:

"After stealing money from the **bank vault**, the **bank robber** was seen fishing on the Mississippi **river bank**."

Bert embeddings:

```
[ ]  print('First 5 vector values for each instance of "bank".')
     print('')
     print("bank vault   ", str(token_vecs_sum[6][:5]))
     print("bank robber  ", str(token_vecs_sum[10][:5]))
     print("river bank   ", str(token_vecs_sum[19][:5]))
```

```
First 5 vector values for each instance of "bank".

bank vault    tensor([ 3.3596, -2.9805, -1.5421,  0.7065,  2.0031])
bank robber   tensor([ 2.7359, -2.5577, -1.3094,  0.6797,  1.6633])
river bank    tensor([ 1.5266, -0.8895, -0.5152, -0.9298,  2.8334])
```

BERT Example

Let's calculate the cosine similarity between the vectors to make a more precise comparison.

```
[ ]  from scipy.spatial.distance import cosine

     # Calculate the cosine similarity between the word bank
     # in "bank robber" vs "river bank" (different meanings).
     diff_bank = 1 - cosine(token_vecs_sum[10], token_vecs_sum[19])

     # Calculate the cosine similarity between the word bank
     # in "bank robber" vs "bank vault" (same meaning).
     same_bank = 1 - cosine(token_vecs_sum[10], token_vecs_sum[6])

     print('Vector similarity for  *similar*  meanings:  %.2f' % same_bank)
     print('Vector similarity for *different* meanings:  %.2f' % diff_bank)
```

```
Vector similarity for  *similar*  meanings:  0.94
Vector similarity for *different* meanings:  0.69
```

Implement:

https://colab.research.google.com/drive/1TRQyU5MtWn9DTuFCnKjbI1cjxh2mh_KW?usp=sharing

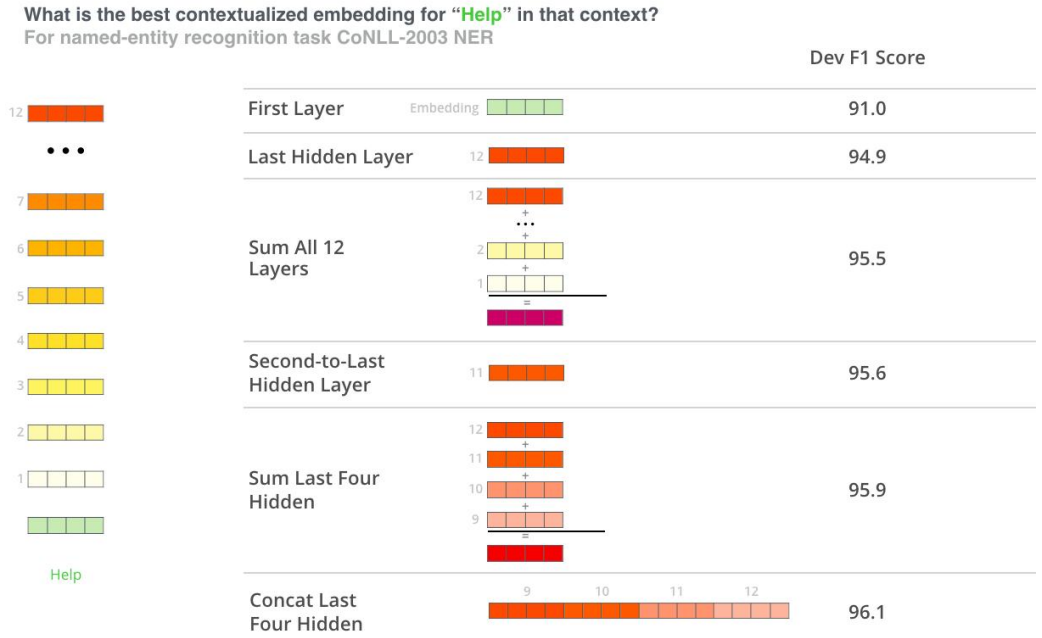
BERT Example:Visualize the Bert Embedding

vocabulary embeddings

BERT Example

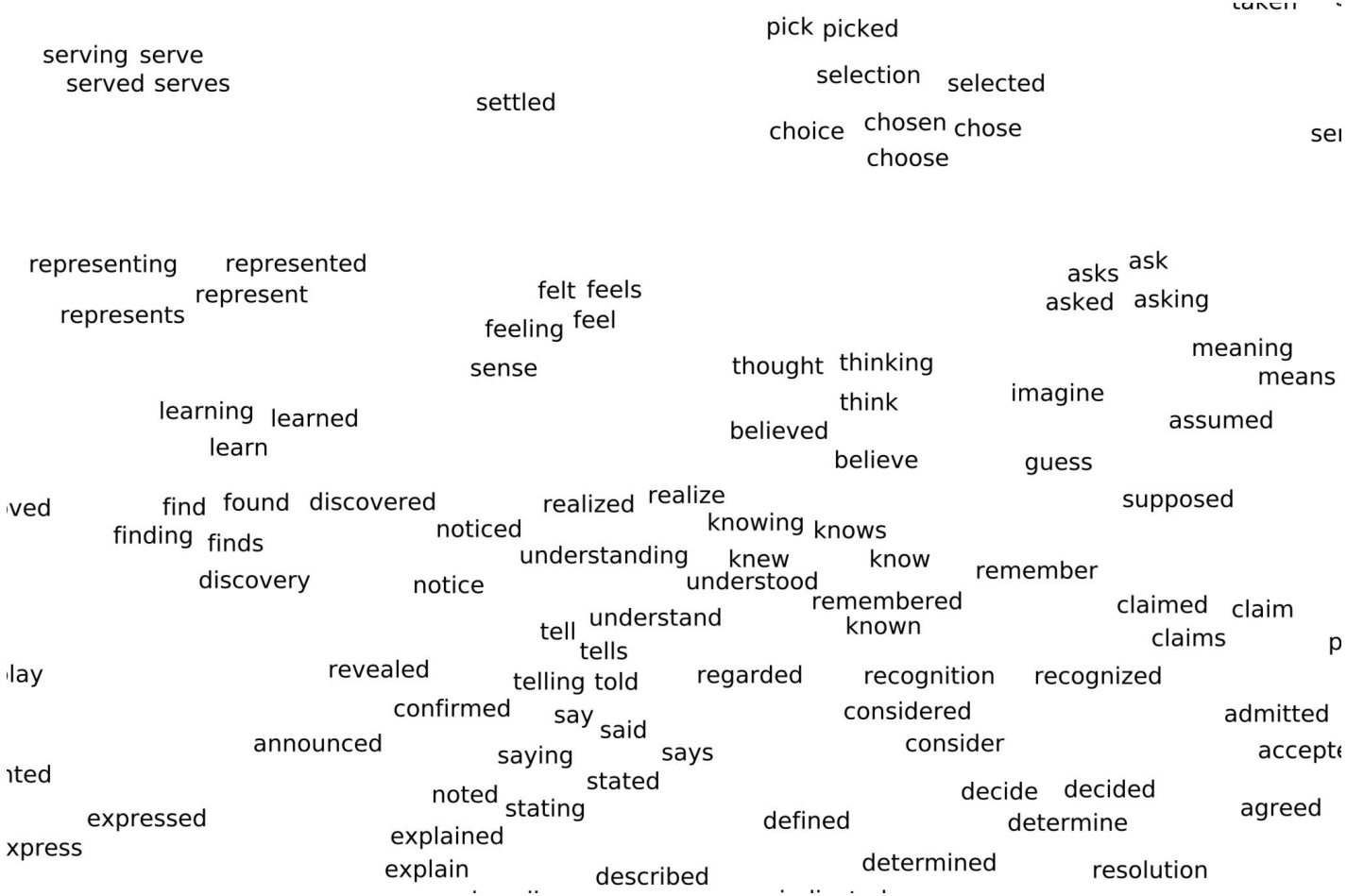
What’s the best contextualized embedding for “help” in that context?

The BERT authors tested word-embedding strategies by feeding different vector combinations as input features to a BiLSTM used on a named entity recognition task and observing the resulting F1 scores.



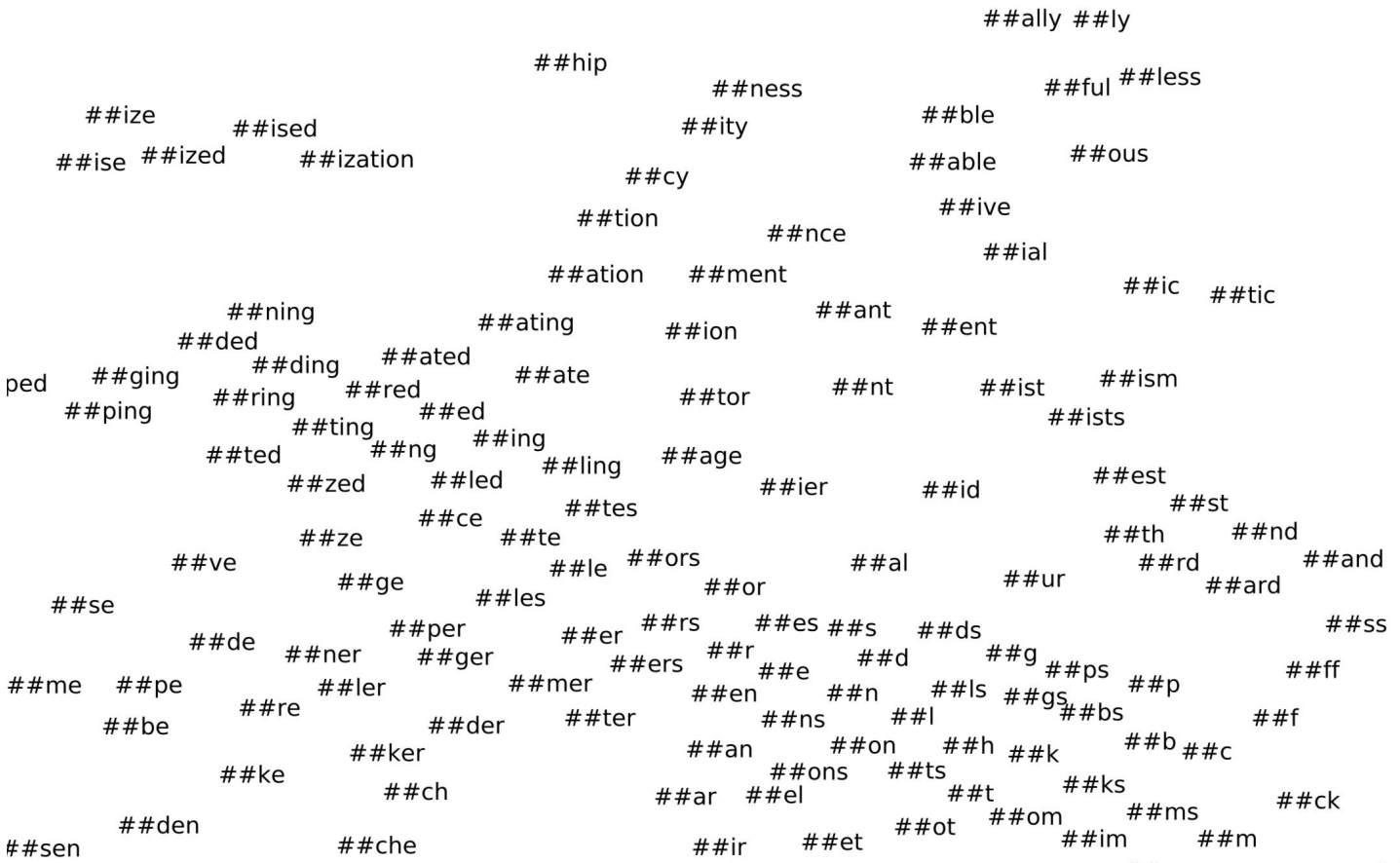
BERT Example:Visualize the Bert Embedding

vocabulary embeddings



BERT Example:Visualize the Bert Embedding

vocabulary embeddings



BERT Example:Visualize the Bert Embedding

context dependent embeddings: values

BERT Example:Visualize the Bert Embedding

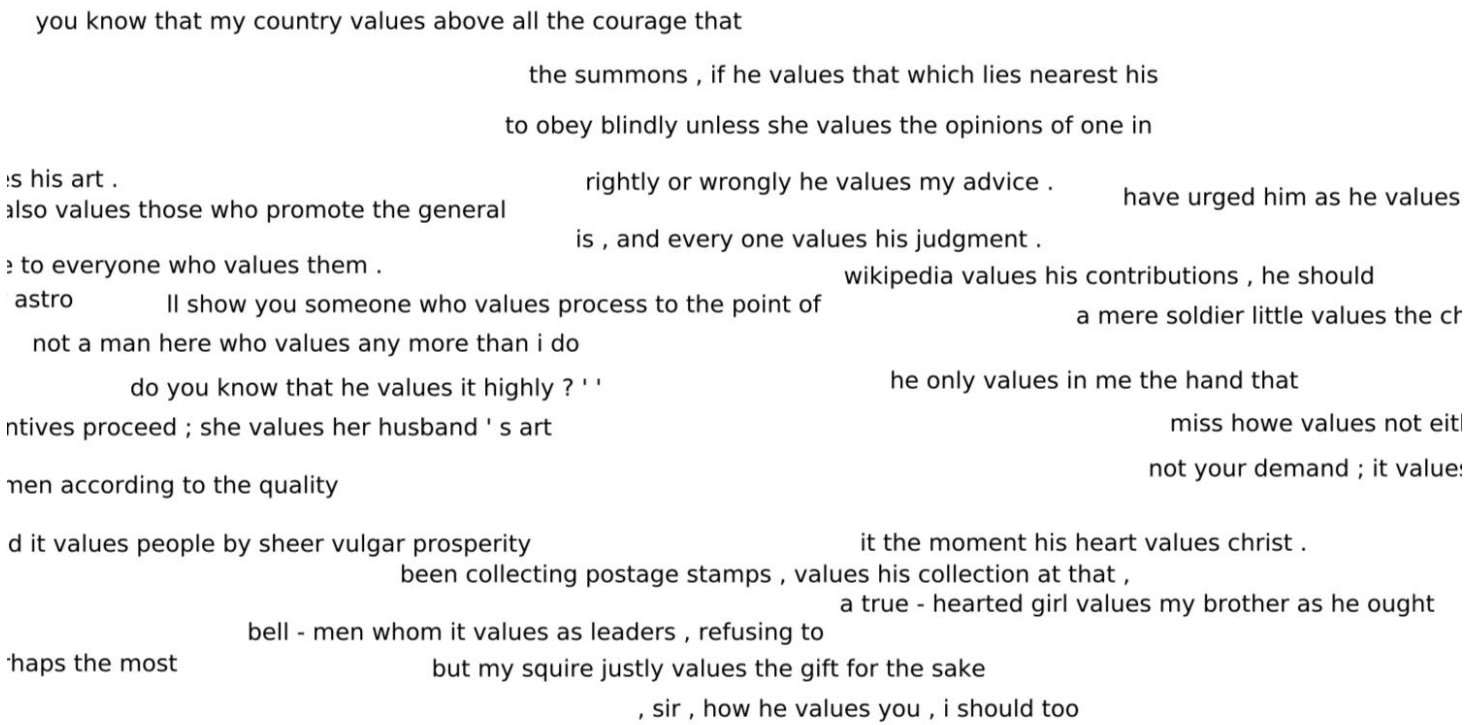
context dependent embeddings: values

Now, we use BERT to embed 15,000 instances of values in sentences drawn from Wikipedia and Project Gutenberg, then run t-SNE on the embeddings taken from the final layer.

BERT Example:Visualize the Bert Embedding

context dependent embeddings: values

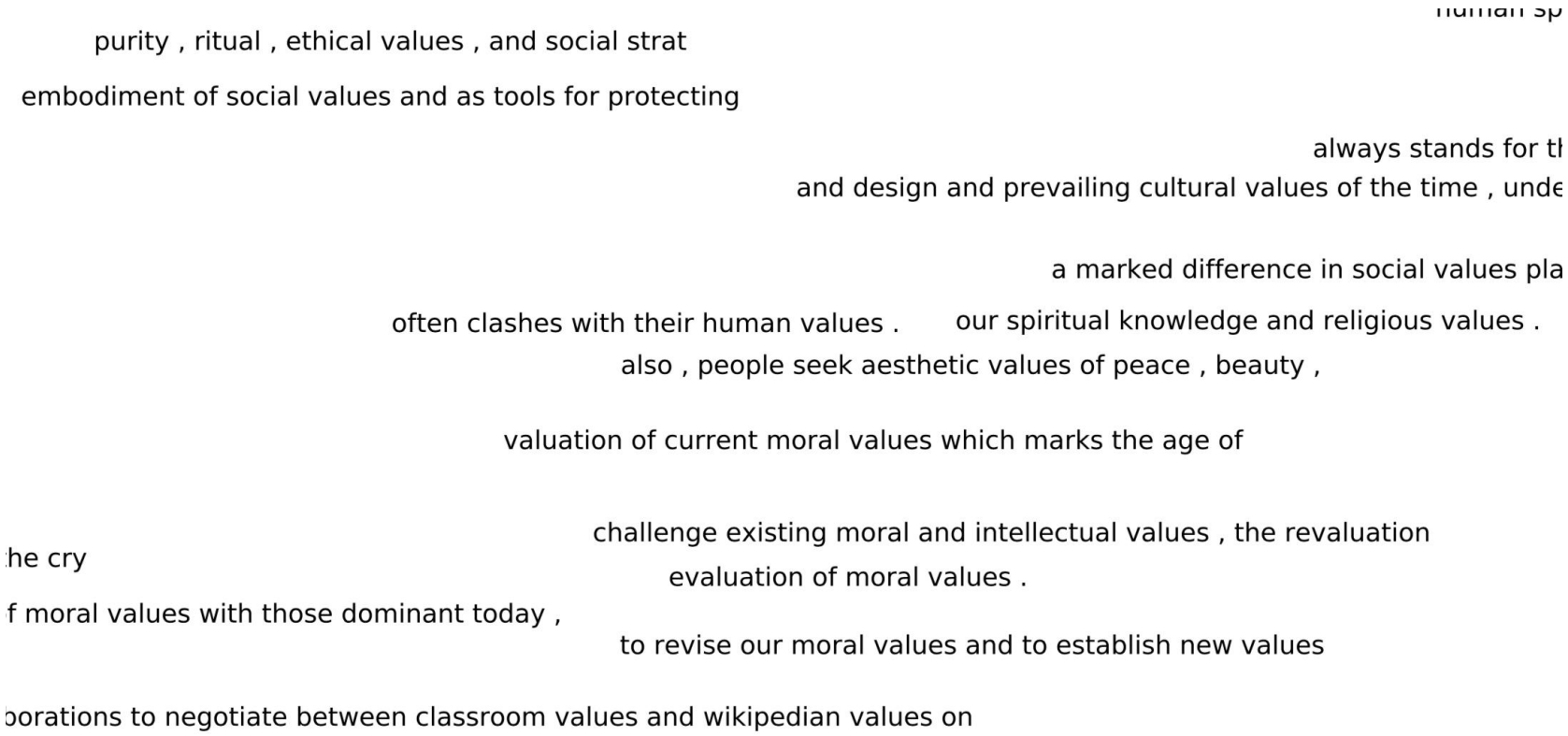
Zooming in, we find different senses of the word in different areas of the visualization. The cluster in the lower left corresponds to verbal uses:



BERT Example:Visualize the Bert Embedding

Context dependent embeddings: values

The remaining are mostly nominal uses. On the left are uses of the sense related to principles or standards:



BERT Example:Visualize the Bert Embedding

Context dependent embeddings: values

To the right we find scientific and mathematical uses; the following shows the lower right corner:

which is true for all values of the algebraic symbols which
of the gradient , as
f the electromagnetic constants
the cone angle .
we will consider different values of .
ions of observed values of the dependent variable from
n is restricted to the values of 1 , 2 ,
s
s it easy to determine what values of λ and κ give
ger values of both parameters result in
r values of this ratio imply that
solutions for y for some values of x , and no
we are interested in odd values of n because any even
ferent values of theta , θ .
ize this approach to higher values of k .
for larger values of n the instanton
for some values of n the (n

Thanks