Reusable Embeddings

Advanced ML with TensorFlow on GCP

End-to-End Lab on Structured Data ML

Production ML Systems

Image Classification Models

Sequence Models

Recommendation Systems

Quiz: Which of these were techniques for dealing with data scarcity?

A. Data augmentation
B. Ensembling
C. Transfer learning
D. AutoML

Quiz: Which of these were techniques for dealing with data scarcity?

A. Data augmentation

B. Ensembling

C. Transfer learning

D. AutoML

Learn how to...

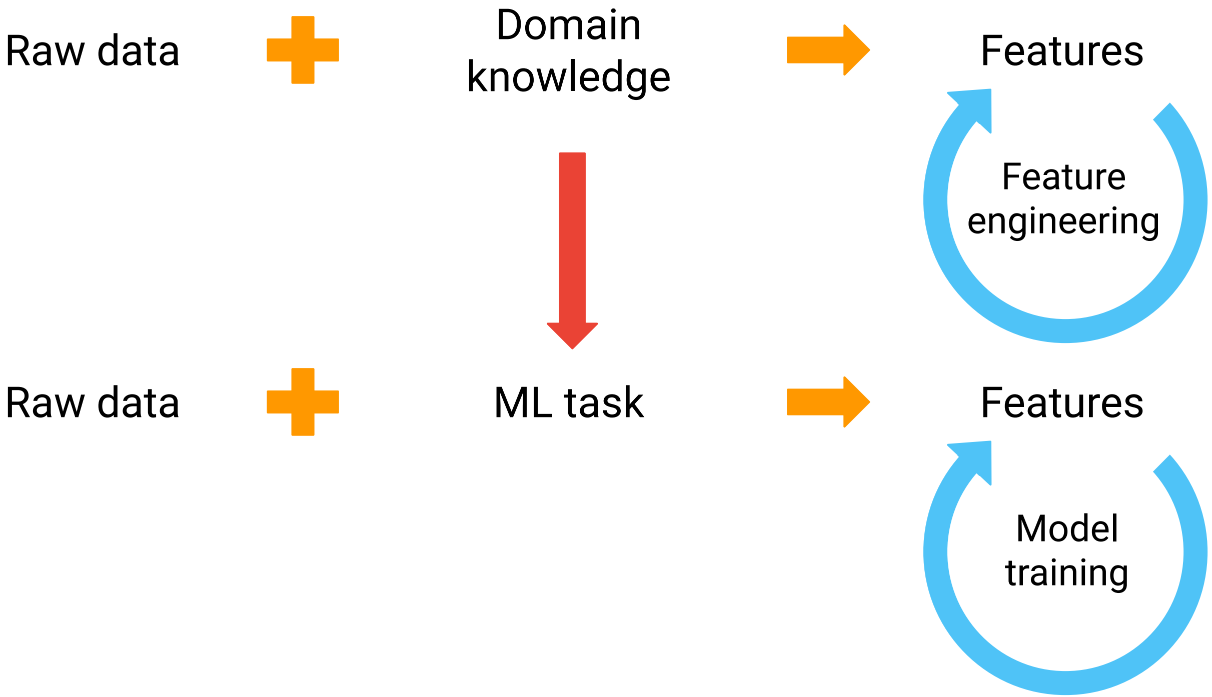
Construct word embeddings using historical techniques

Construct word embeddings using GloVe and Word2Vec

Make use of pre-trained embeddings on TensorFlow Hub

Make your Hub embeddings trainable

Two similar stories



Mapping meaning

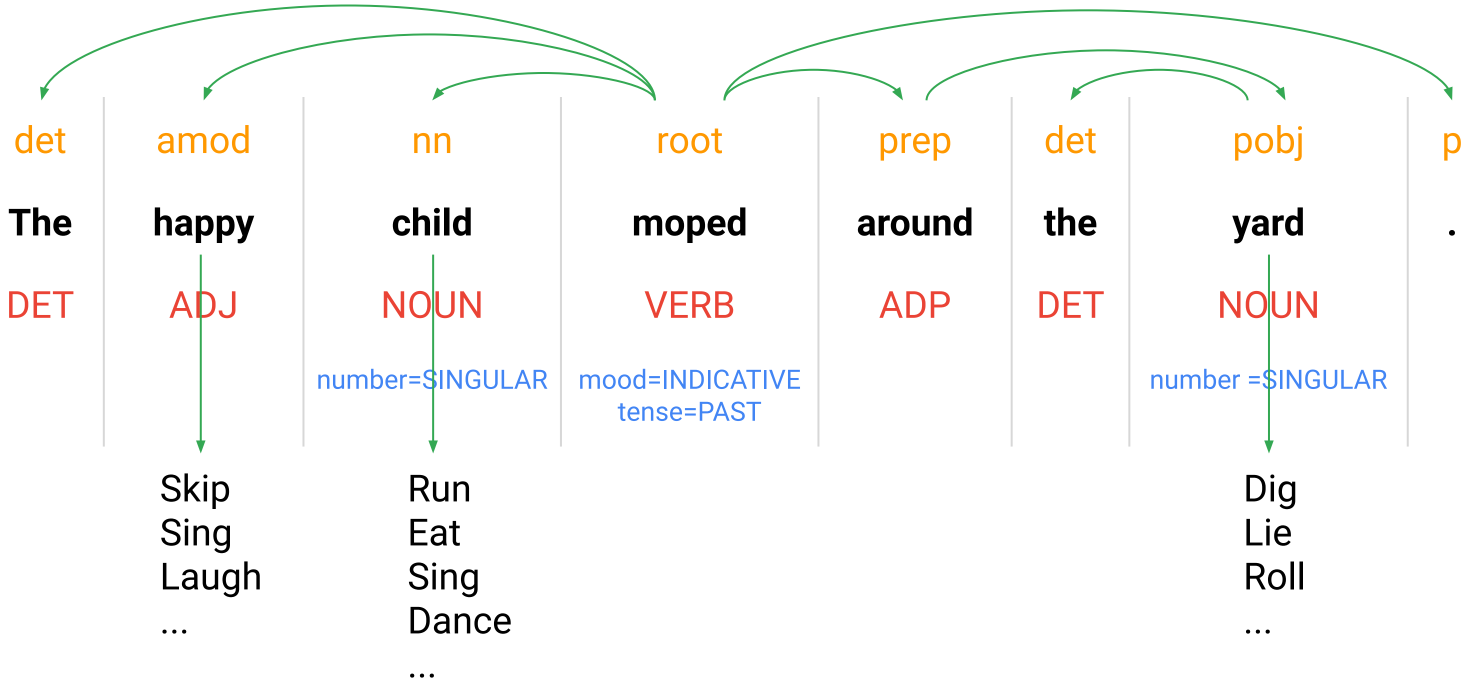
Average ratings for “polite”

Dimension	Avg Rating
Angular-Rounded	4.9
Weak-Strong	6.1
...	...
Fresh-Stale	1.9

Embedding for “polite”

$$[4.1 \;\; 6.1 \;\; \ldots \;\; 1.9]$$

The distributional hypothesis



Latent semantic analysis

	Document 1	Document 2	...	Document N
Term 1	1	0		1
Term 2	0	1		0
...				
Term M	1	1		1

Quiz: Why are term vectors poor word embeddings?

A. The resulting term vectors weren’t of high enough quality
B. The resulting term vectors weren’t useful enough
C. Both
D. Neither

Quiz: Why are term vectors poor word embeddings?

A. The resulting term vectors weren't of high enough quality

B. The resulting term vectors weren't useful enough

C. Both

D. Neither

Term vectors weren’t of high enough quality

	Document 1	Document 2	Document 3
Term 1	1	0	1
Term 2	0	1	1

`Term 1 . Term 2 = 0` With only documents 1 and 2

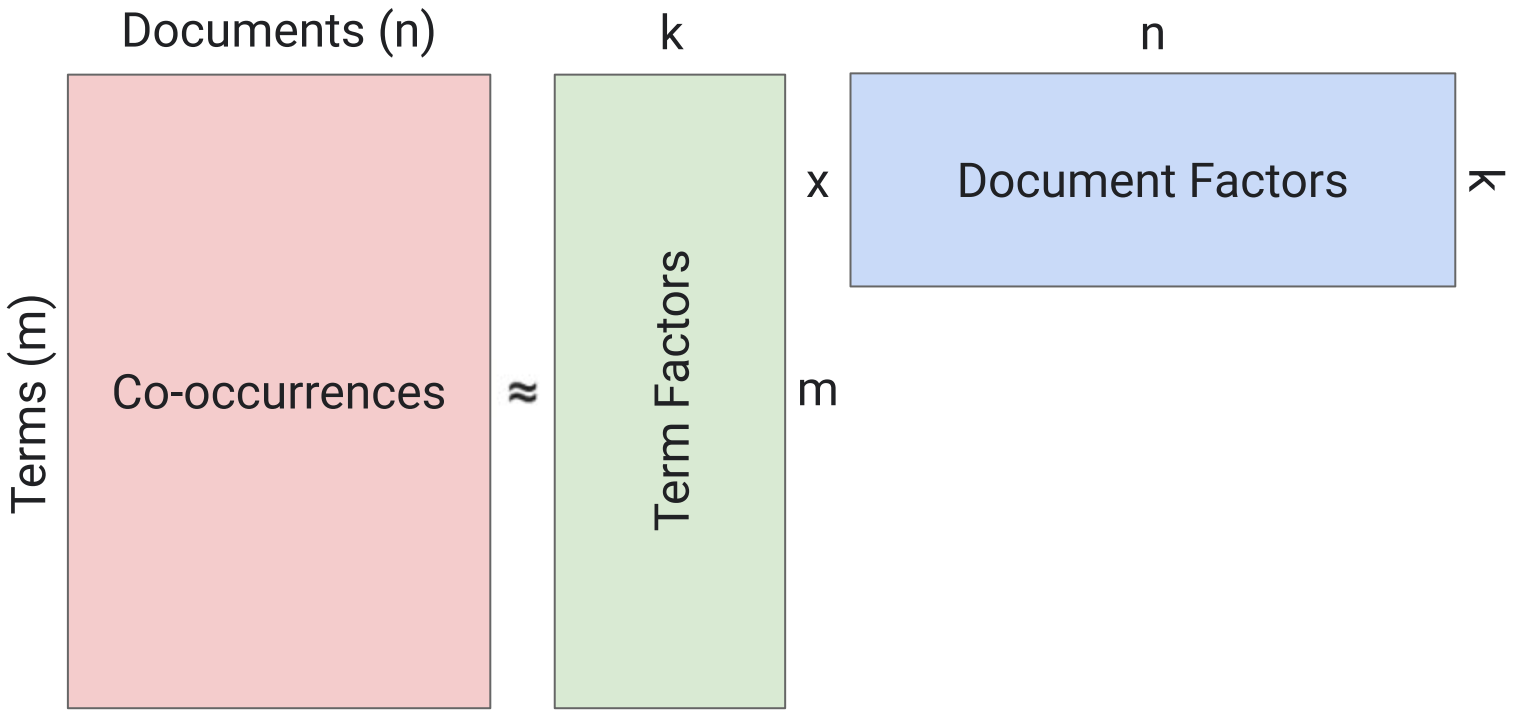
`Term 1 . Term 2 = 1` With all documents

Matrix factorization key ideas

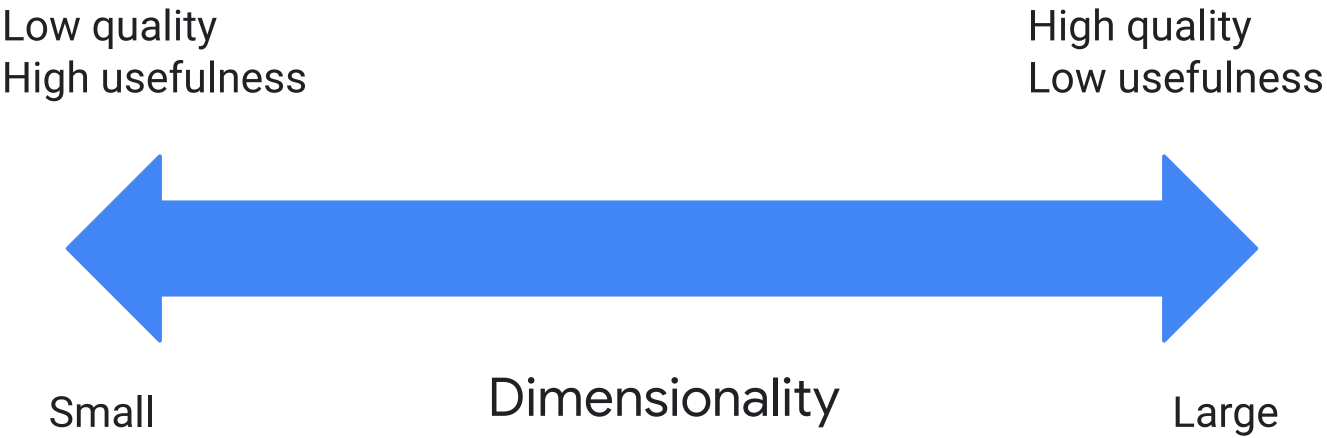
1. Creates smaller embeddings row and column domains.

2. Widely used in machine learning.

Matrix factorization minimizes reconstruction error



Trading off quality and usefulness with dimensionality



From documents to terms

```
The quick brown fox jumps
over the lazy dog
```

	Term 1	**Term 2**	**Term 3**
Term 1	1	0	1
Term 2	0	1	1

Sliding windows but no co-occurrence matrix

```
The quick brown fox jumps
over the lazy dog
```

	Term 1	Term 2	Term 3
Term 1	1	0	1
Term 2	0	1	1

Sliding windows but no co-occurrence matrix

```
The quick brown fox jumps
over the lazy dog
```

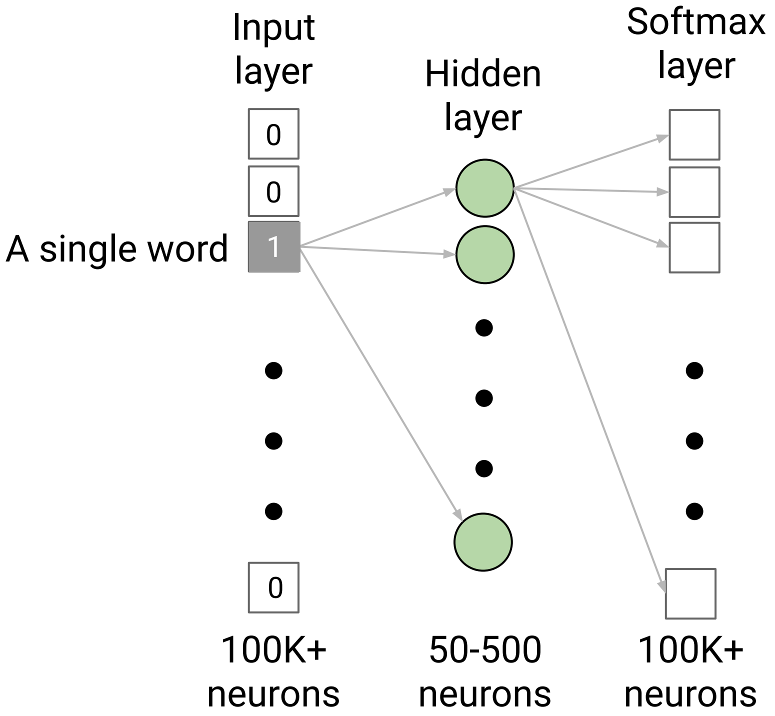
4 words prior	3 words prior	2 words prior	1 word prior	Label
Mary	had	a	little	lamb
...	...	...	...	...

Predicting context given the central word

```
The quick brown fox jumps
over the lazy dog
```

2 words prior	1 word prior	Word	1 word ahead	2 words ahead
Mary	had	**a**	little	lamb
...	...	**...**	...	...

Word2Vec's network architecture



Quiz: Why was normal cross-entropy impractical in this case?

A. Because of problems with numerical precision

B. Because this is actually a regression task

C. Because this is a multi-class classification task

D. Because of the number of classes

Quiz: Why was normal cross-entropy impractical in this case?

A. Because of problems with numerical precision

B. Because this is actually a regression task

C. Because this is a multi-class classification task

D. Because of the number of classes

$$p(y = j|\mathbf{x}) = \frac{exp(\mathbf{w}_j^T \mathbf{x} + b_j)}{\sum_{k \in K} exp(\mathbf{w}_k^T \mathbf{x} + b_k)}$$

Negative sampling makes softmax less expensive

$$p(y=j|\mathbf{x}) = \frac{exp(\mathbf{w}_j^T\mathbf{x}+b_j)}{\sum_{k\in \cancel{K}} exp(\mathbf{w}_k^T\mathbf{x}+b_k)}$$

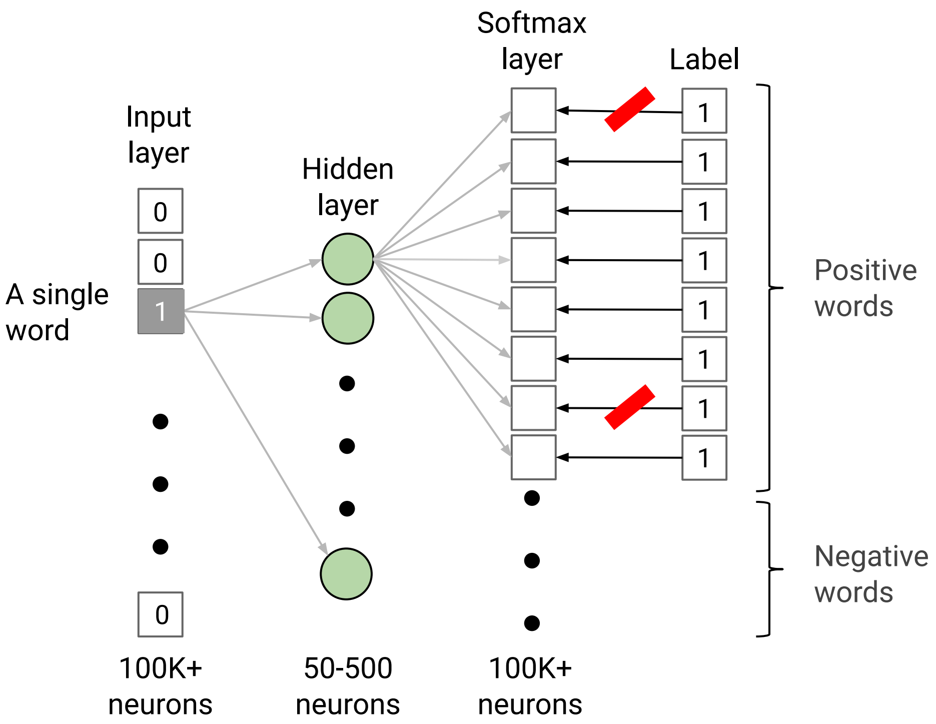
Subset

Negative sampling makes softmax less expensive

Window size < Vocabulary Size

Removing some common words can also reduce cost

```
A little red
fluffy dog
wanders down
the road
```

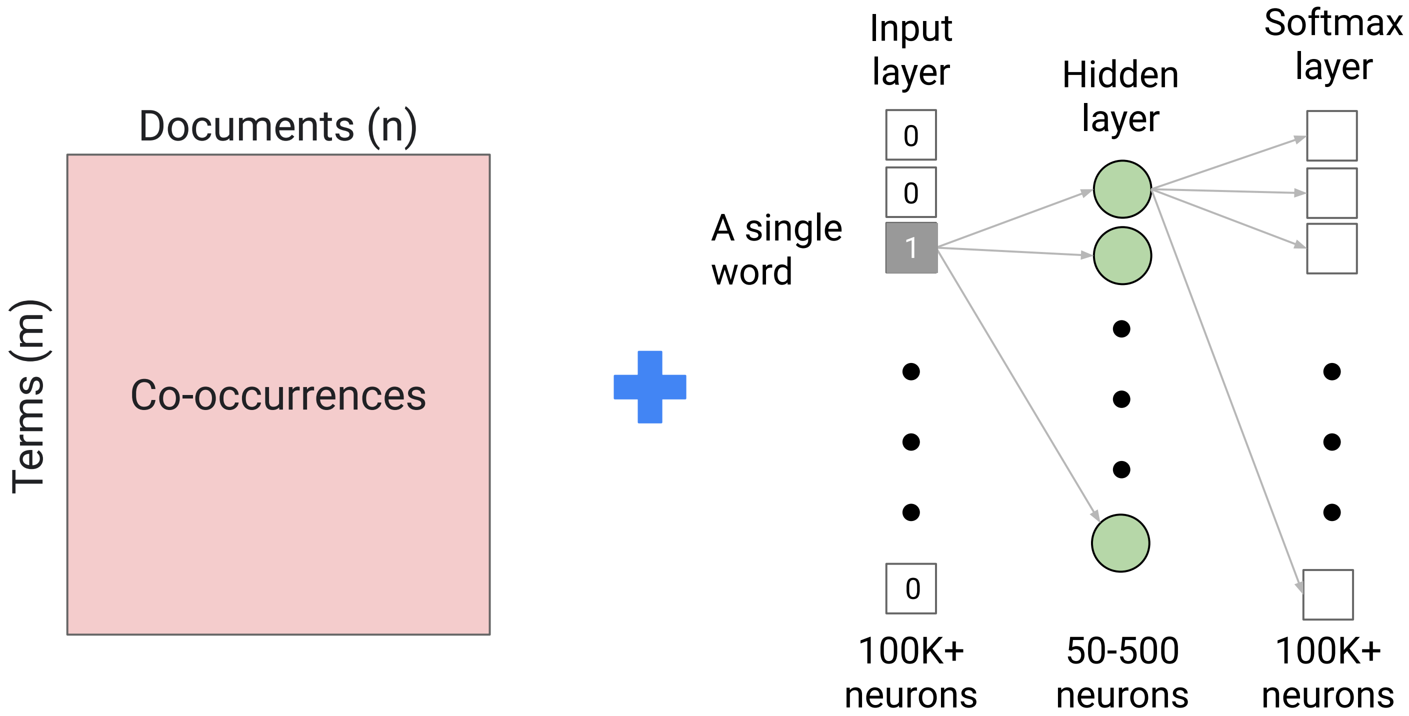


Word2Vec embeddings are compositional

Czech + currency	Vietnam + capital	French + actress
koruna	Hanoi	Juliette Bincohe
Check crown	Ho Chi Minh City	Vanessa Paradis
Polish zolty	Viet Nam	Charlotte Gainsbourg
CTK	Vietnamese	Cecile De

Is it possible to have the powerful properties of Word2Vec for the entire set of co-occurrence statistics?

GloVe is a hybrid between matrix factorization and window-based methods



Co-occurrence ratios are semantically important

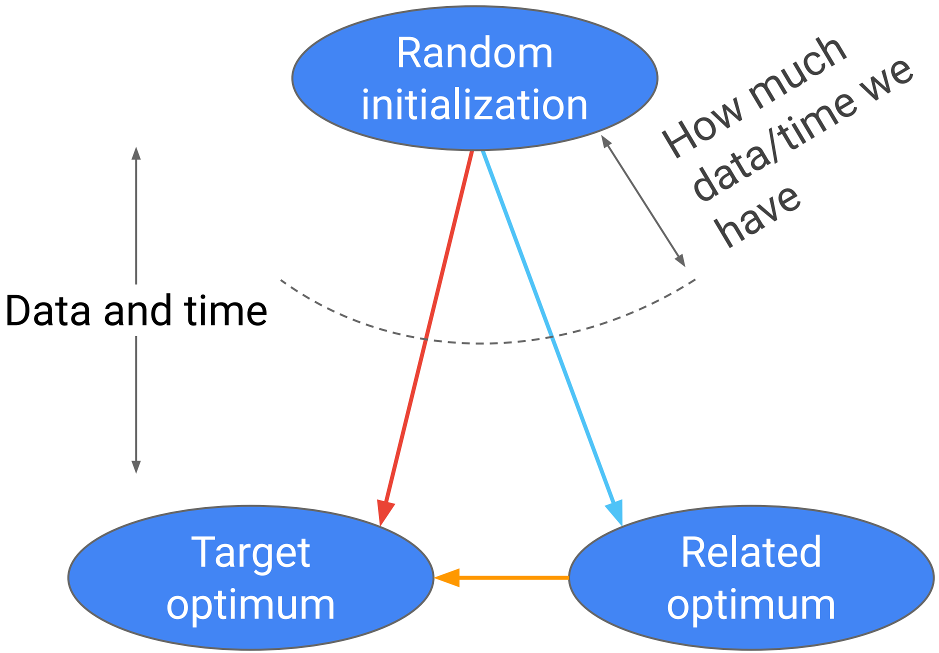
Probability and ratio	solid	gas	water	fashion
P(k|ice)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(k|steam)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k|ice) / P(k|steam)	8.9	8.5×10^{-2}	1.36	0.96

Reverse engineering GloVe's loss function

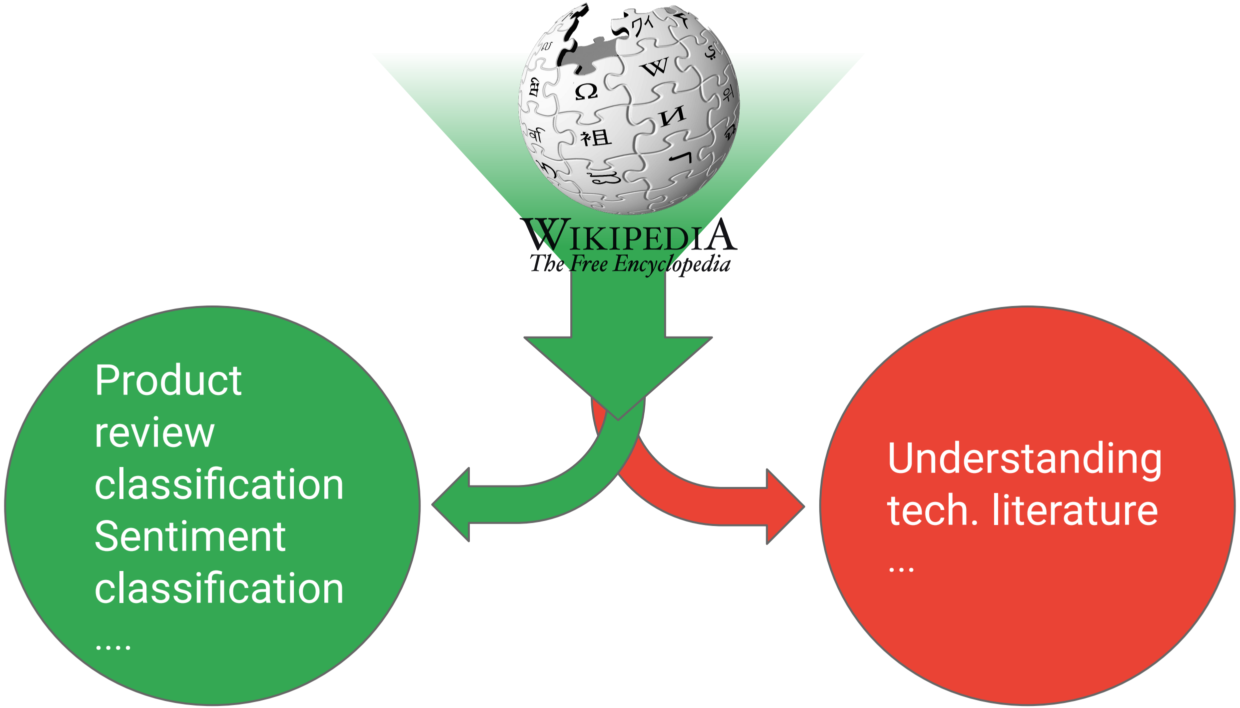
$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ij}}{P_{jk}}$$

$$F(w_i, w_j, \tilde{w}_k) - \frac{P_{ij}}{P_{jk}} = 0$$

Other choices are more important than GloVe versus Word2Vec



Pre-trained embeddings work for general-purpose tasks



To freeze or not to freeze?

Small

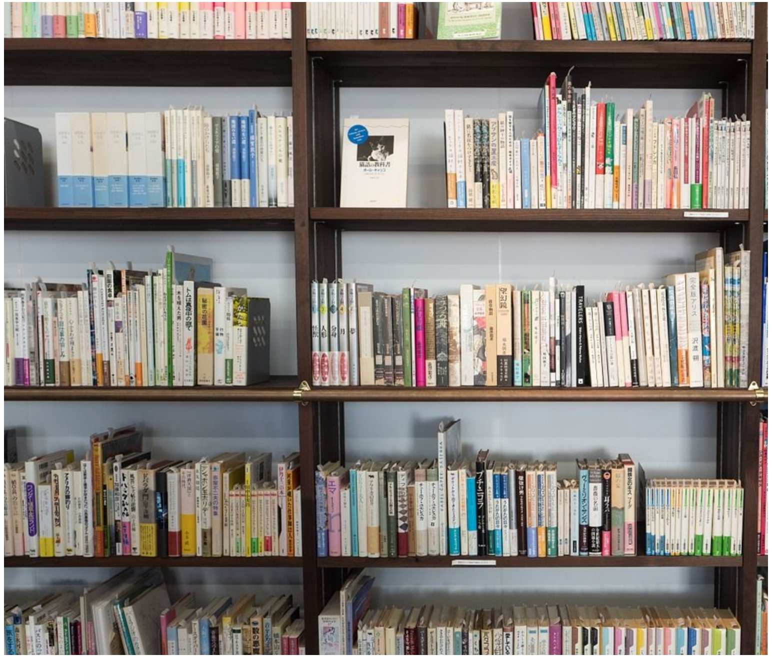
Large

Labeled dataset size

Freeze the source model.

Let the source model train.

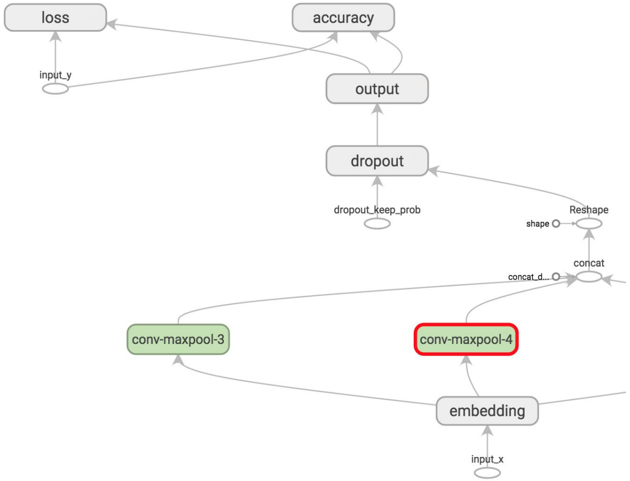
TensorFlow Hub makes using pre-trained embeddings easy



Simply pull from a library of pre-trained embeddings.

Modules are executable portions of a TensorFlow graph

```
embed = hub.Module(module_url)
embeddings = embed(["A long sentence.", "single-word",
                    "http://example.com"])
```



Graphs require sessions to be executed

```
import tensorflow as tf
import tensorflow_hub as hub

with tf.Graph().as_default():
  module_url = "https://tfhub.dev/path/to/module"
  embed = hub.Module(module_url)
  embeddings = embed(["A long sentence.",
                      "single-word",
                      "http://example.com"])

  with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

    print(sess.run(embeddings))
```

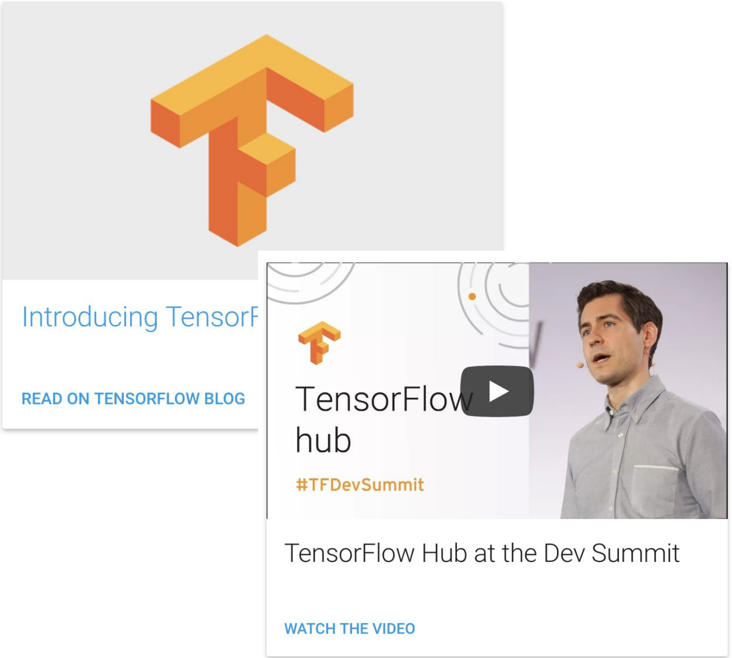
Lab

Create reusable embeddings with TensorHub

Lab Steps

1. Create an embedding using the NNLM model.
2. Assess the embeddings informally.
3. Compare naive methods of sentence embedding against RNNs.
4. Assess the embedding formally.

Relevant TensorFlow Hub functions




```
text_embedding_column

image_embedding_column
```

Text embedding column as an input to a canned estimator

```
embedded_text_feature_column = hub.text_embedding_column(
    key="sentence",
    module_spec="https://tfhub.dev/google/nnlm-en-dim128/1",
    trainable=False)

estimator = tf.estimator.DNNClassifier(
    hidden_units=[500, 100],
    feature_columns=[embedded_text_feature_column],
    n_classes=2,
    optimizer=tf.train.AdagradOptimizer(learning_rate=0.003))
```

cloud.google.com