Introduction to Word Vector Representations

Shweta Bhatt | Data Scientist | ML GDE

Agenda

- What is NLP
    - What are words?
    - Word representations in Traditional NLP
- Neural Probabilistic models
    - Word2vec
    - Other popular embeddings: GloVe, FastText
- Using word embeddings:
    - Sentence similarity task
- State-of-the-art Embeddings
    - Seq-to-seq models, LMs
- Challenges
- Q & A

Agenda

- **What is NLP?**
  - **What are words?**
  - **Word representations in Traditional NLP**
- Neural Probabilistic models
  - Word2vec
  - Other popular embeddings: GloVe, FastText
- Using word embeddings:
  - Sentence similarity task
- State-of-the-art Embeddings
  - Seq-to-seq models, LMs
- Challenges
- Q & A

What is Natural Language Processing (NLP) ?

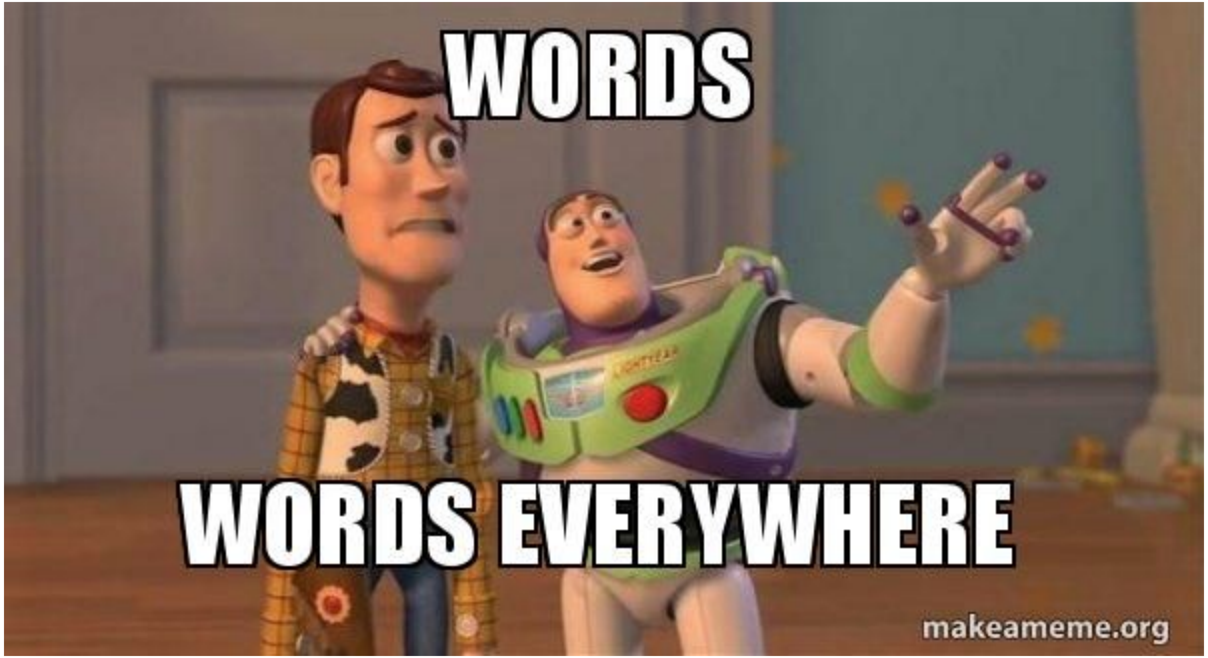


Image source

What is NLP?

- Goal: design algorithms to make machines understand natural language to perform some tasks

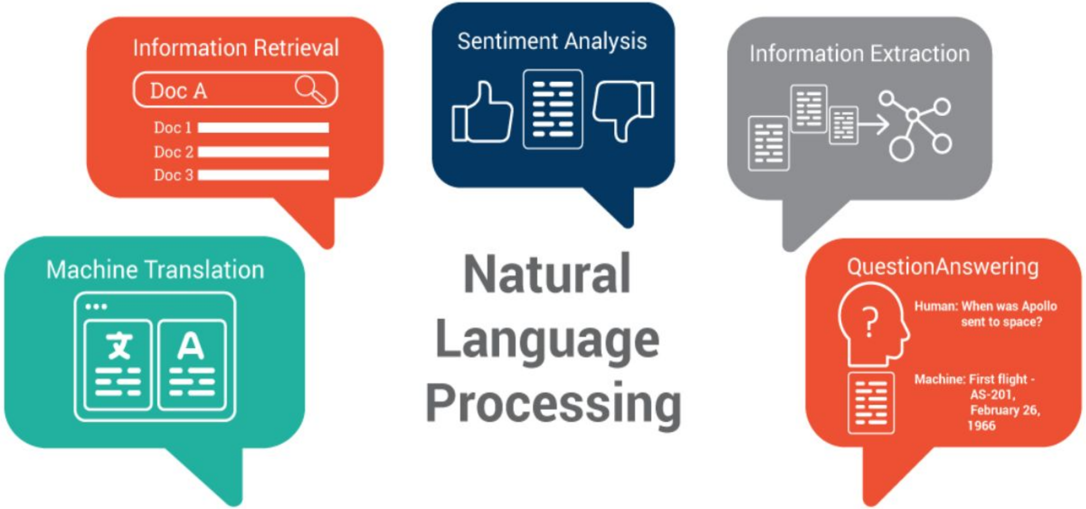


Image source

NLP vs NLU vs ASR

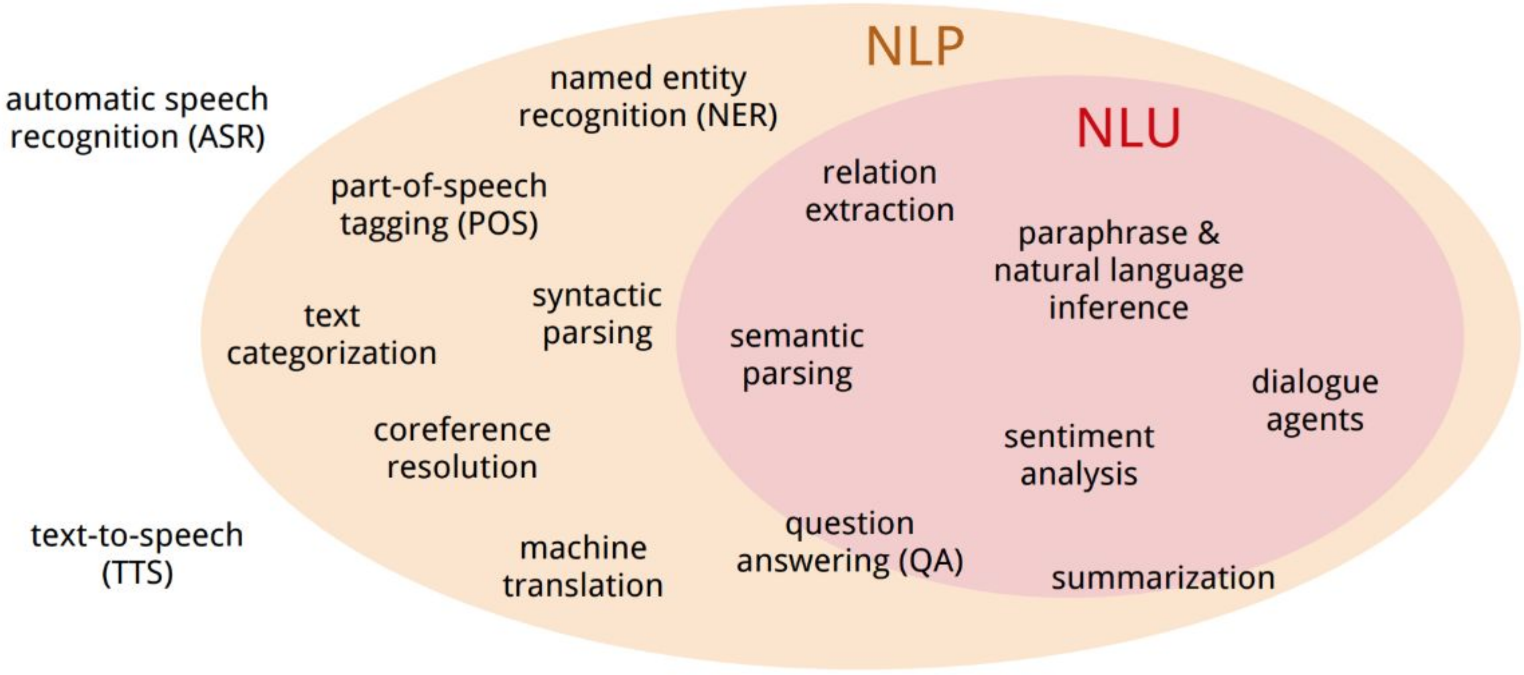


Image source

What are words?

Image source

Word = basic unit of language

Image source

Motivation

Data representation is crucial!

Image source

Localist representations = one-hot vectors

- Discrete/atomic symbols
- Context free
- Long and sparse
- Orthogonal vectors ⇒
- No notion of similarity

$$
\begin{array}{lcl}
 & & \begin{array}{cccccccc} \text{Rome} & \text{Paris} & & & & & & \text{word V} \end{array} \\
\texttt{Rome} & = & [1,\ 0,\ 0,\ 0,\ 0,\ 0,\ \ldots,\ 0] \\
\texttt{Paris} & = & [0,\ 1,\ 0,\ 0,\ 0,\ 0,\ \ldots,\ 0] \\
\texttt{Italy} & = & [0,\ 0,\ 1,\ 0,\ 0,\ 0,\ \ldots,\ 0] \\
\texttt{France} & = & [0,\ 0,\ 0,\ 1,\ 0,\ 0,\ \ldots,\ 0]
\end{array}
$$

How do you define meaning of a word?

X played his last cricket test match in 2010.

The Sri Lankan team gave **X** a proud retirement.

X has 800 total test wickets.

Who is X?

Image source

Most likely ?

Image source

Distributed representations = Dense vectors

Distributional semantics: A word’s meaning is defined by its environment (surrounding or context words)

If A and B have almost similar environments ⇒ they are synonyms

Word vectors = word embeddings = word representations

Agenda

- What is NLP?
  - What are words?
  - Words representation in Traditional NLP: one-hot vectors, tf-idf
- **Neural Probabilistic models**
  - **Word2vec**
  - **Other popular embeddings: GloVe, FastText**
- Using word embeddings:
  - Sentence similarity task
- State-of-the-art Embeddings
  - Seq-to-seq models, LMs
- Challenges
- Q & A

Word2Vec

- **Task** → Given a specific word in the middle of a sentence (**center word**) and the **context** (surrounding) words, train a neural network such that it predicts the **probability of every word in the vocabulary of being the context word** (within a window n)

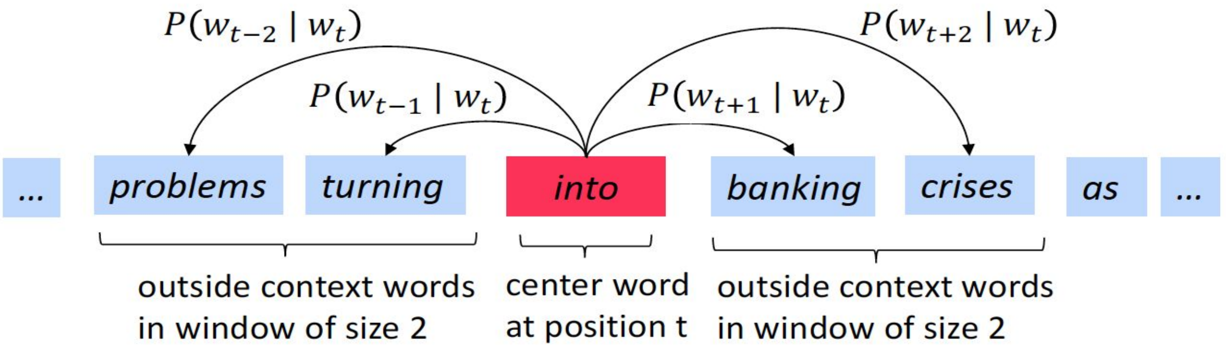


Image source

Neural Network Architecture

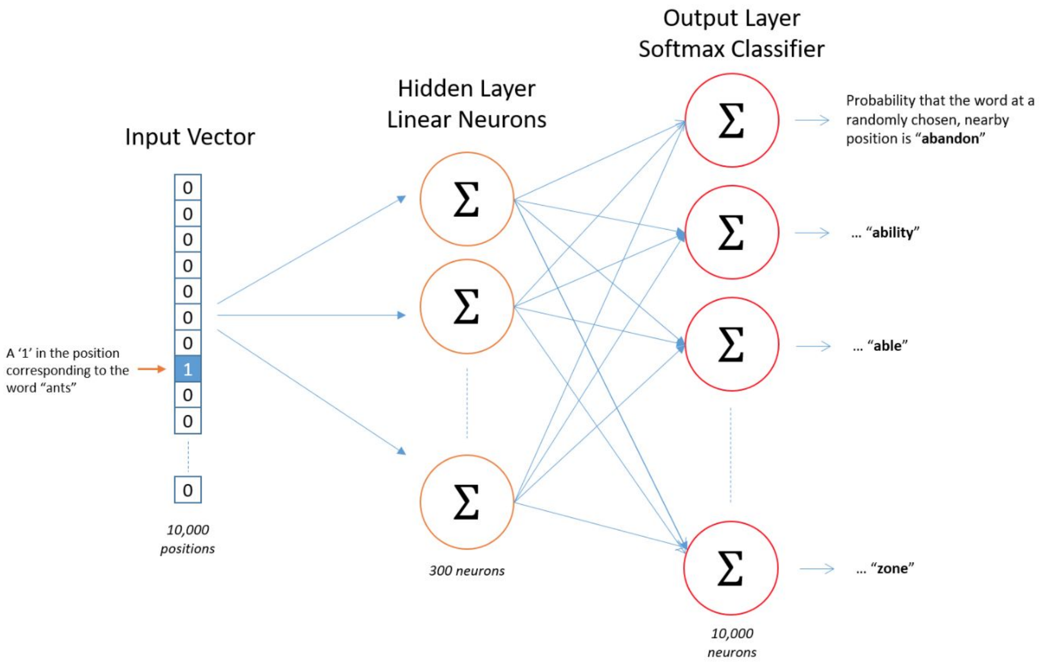


Image source

CBOW vs Skip-gram

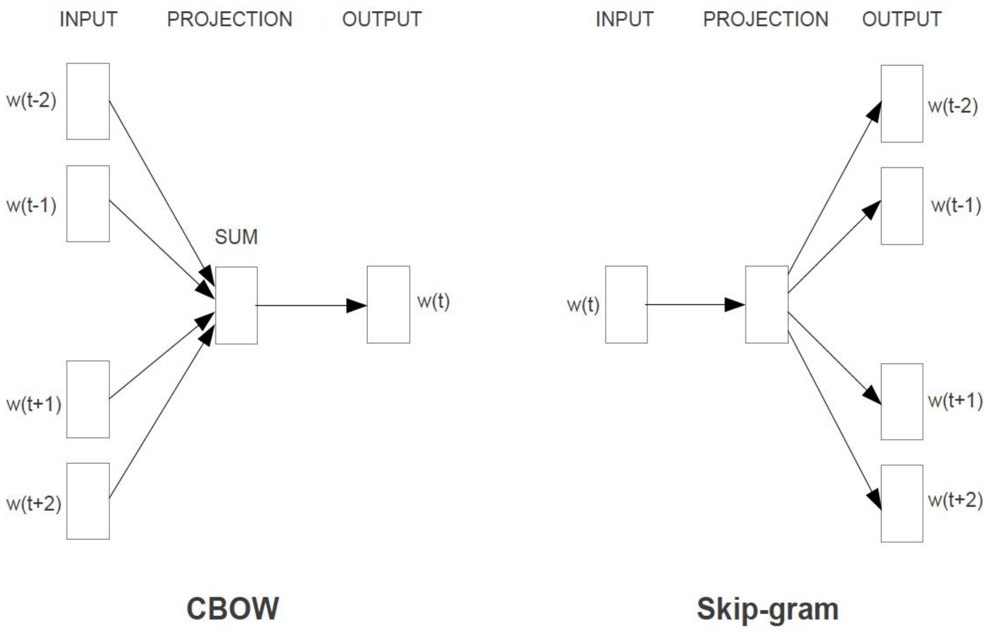


Image source

Gensim Word2Vec API

```
from gensim.models import Word2Vec
model_ted = Word2Vec(sentences=sentences_ted, size=100, window=5, min_count=5, workers=4, sg=0)
```

```
model_ted.wv.most_similar("man")
```

```
[('woman', 0.8443934321403503),
 ('guy', 0.8030357956886292),
 ('lady', 0.7726334929466248),
 ('girl', 0.759391188621521),
 ('boy', 0.7479357123374939),
 ('soldier', 0.7148764133453369),
 ('kid', 0.699984610080719),
 ('gentleman', 0.6899228692054749),
 ('surgeon', 0.6823126077651978),
 ('david', 0.675527632236480 7)]
```

Training requires: Time + Data + Computing power

Image source

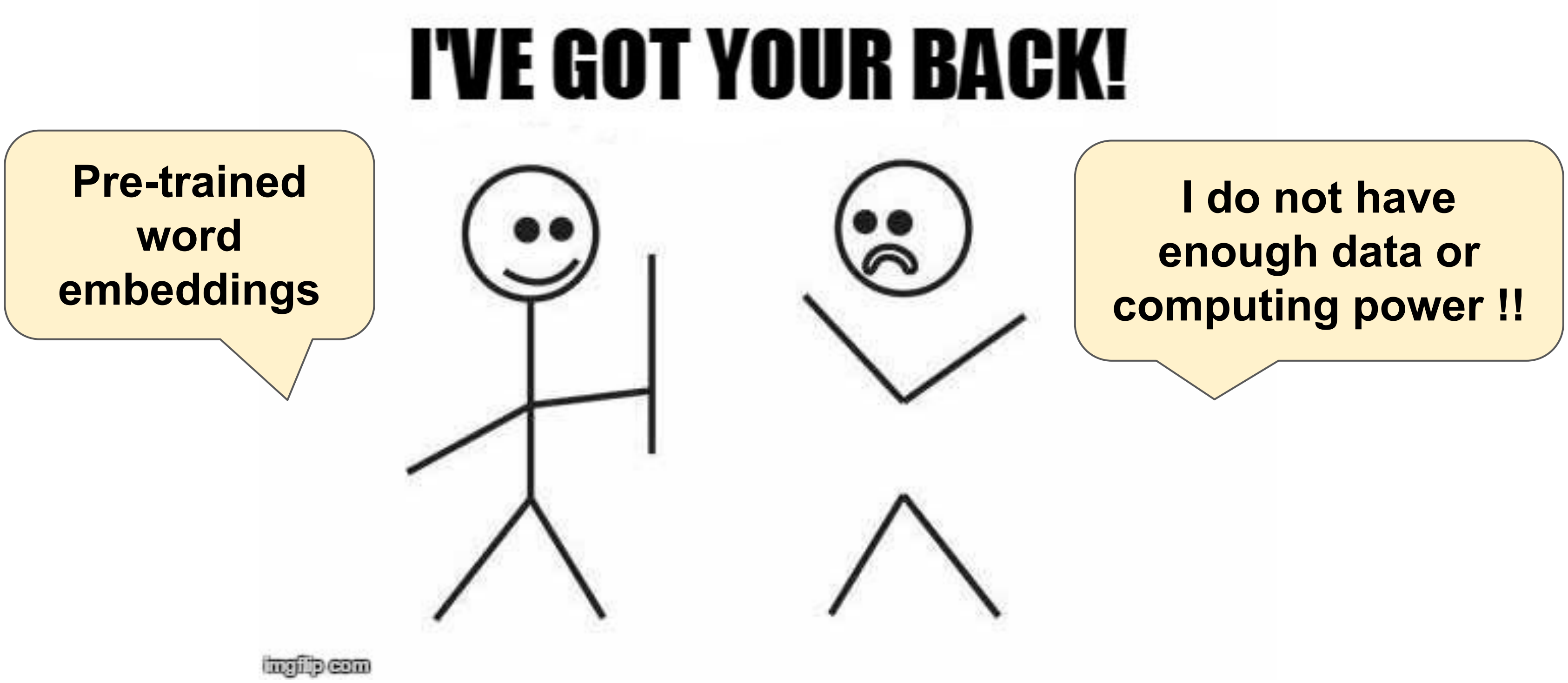


Image source

Loading Google’s pre-trained Word2Vec

```
from gensim.models import KeyedVectors
filename = 'GoogleNews-vectors-negative300.bin'
model = KeyedVectors.load_word2vec_format(filename, binary=True)
```

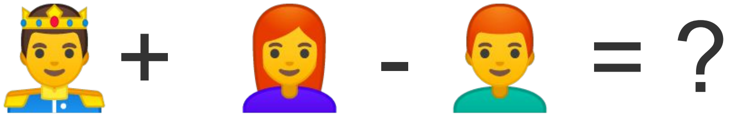
- Trained on Google news data (**~100 billion words**)
- **300** dimensional vectors
- Size of unzipped binary file = **3.4GB**

Semantic and syntactic relations

The very famous,

🤴 + 👩‍🦰 - 👨‍🦰 = ?

Semantic and syntactic relations

The very famous, 

king + woman - man = queen

```
result = model.most_similar(positive=['woman', 'king'], negative=['man'], topn=1)
print(result)
```

```
[('queen', 0.7118192911148071)]
```

biggest + small - big = smallest

```
result = model.most_similar(positive=['biggest', 'small'], negative=['big'], topn=1)
print(result)
```

```
[('smallest', 0.6086567640304565)]
```

Why Word2Vec is not always the answer?

```
model.most_similar("devfest")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-63-4278090bb35a> in <module>
----> 1 model.most_similar("devfest")

//anaconda3/lib/python3.6/site-packages/gensim/models/keyedvectors.py in most_similar(self, positive, negative, topn, restrict_vocab, indexer)
    551                     mean.append(weight * word)
    552                 else:
--> 553                     mean.append(weight * self.word_vec(word, use_norm=True))
    554                     if word in self.vocab:
    555                         all_words.add(self.vocab[word].index)

//anaconda3/lib/python3.6/site-packages/gensim/models/keyedvectors.py in word_vec(self, word, use_norm)
    466             return result
    467         else:
--> 468             raise KeyError("word '%s' not in vocabulary" % word)
    469 
    470     def get_vector(self, word):

KeyError: "word 'devfest' not in vocabulary"
```

Can't handle out-of-vocabulary words

GloVe Embeddings

1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{array}{cc} & \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{array}{c} I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} & \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Idea

- Derive semantic relationships between words using word co-occurrence matrix
- Global + local context
- Matrix Factorization / Latent Semantic Analysis

source

Loading Stanford's pre-trained GloVe Embeddings

```
from gensim.scripts.glove2word2vec import glove2word2vec
glove_input_file = 'glove.6B/glove.6B.100d.txt'
word2vec_output_file = 'glove.6B/glove.6B.100d.txt.word2vec'
glove2word2vec(glove_input_file, word2vec_output_file)
```

```
(400000, 100)
```

```
from gensim.models import KeyedVectors
# load the Stanford GloVe model
filename = 'glove.6B/glove.6B.100d.txt.word2vec'
model = KeyedVectors.load_word2vec_format(filename, binary=False)
```

- Trained on Wikipedia data with 6 billion tokens and 400000 vocabulary
- 4 different models - 50, 100, 200 and 300 dimensional vectors
- Size of unzipped file = 2.63GB

Same disadvantage as Word2Vec

```
model.most_similar('devfest')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-74-260e005f8a53> in <module>
----> 1 model.most_similar('devfest')

//anaconda3/lib/python3.6/site-packages/gensim/models/keyedvectors.py in most_similar(self, positive, negative, topn, restrict_vocab, indexer)
    551                 mean.append(weight * word)
    552             else:
--> 553                 mean.append(weight * self.word_vec(word, use_norm=True))
    554                 if word in self.vocab:
    555                     all_words.add(self.vocab[word].index)

//anaconda3/lib/python3.6/site-packages/gensim/models/keyedvectors.py in word_vec(self, word, use_norm)
    466             return result
    467         else:
--> 468             raise KeyError("word '%s' not in vocabulary" % word)
    469
    470     def get_vector(self, word):

KeyError: "word 'devfest' not in vocabulary"
```

Can't handle out-of-vocabulary words

Word2Vec vs GloVe

Word2Vec	GloVe
Neural network approach	Matrix Factorization technique
Predictive model	Count based approach
Incorporates local context of words	Incorporates both local and global context (word co-occurrence)
Scales with corpus size	Faster training

FastText embeddings

- Extension of word2vec
- Represent words as n-gram of characters
- Example, word = “artificial”, n=3
- Fasttext representation = <ar, art, rti, tif, ifi, fic, ici, cia, ial, al>
- Allows embeddings to understand suffixes and prefixes
- **Huge advantage**

Works well for rare/out-of-vocabulary words

Gensim example

```
from gensim.models import FastText
model_ted = FastText(sentences_ted, size=100, window=5, min_count=5, workers=4,sg=1)
```

```
model_ted.wv.most_similar('devfest')
```

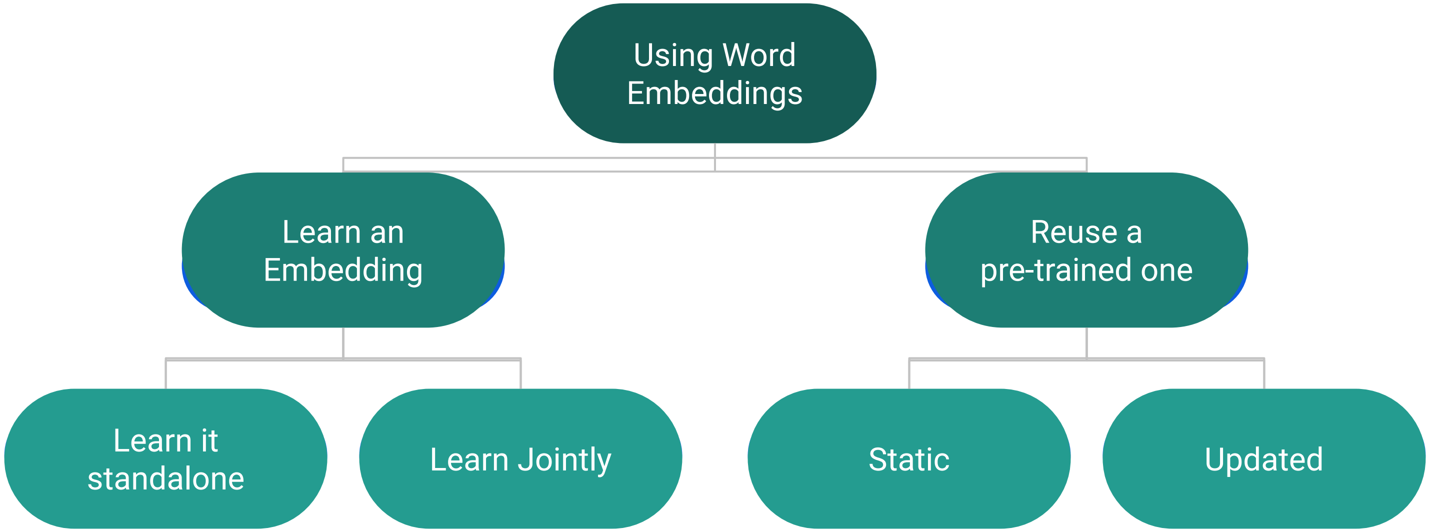
```
[('manifest', 0.6591683030128479),
 ('manifesto', 0.621625423431396 5),
 ('lifestyle', 0.5846694707870483),
 ('impairment', 0.576905131340026 9),
 ('disruptive', 0.571802854537963 9),
 ('develop', 0.5686429738998413),
 ('role', 0.5619493722915649),
 ('endeavor', 0.5553780198097229),
 ('inventive', 0.5544175505638123),
 ('augment', 0.550832331180572 5)]
```

Handles out-of-vocabulary words

Agenda

- What are words?
    - What is NLP?
    - Words representation in Traditional NLP: one-hot vectors, tf-idf
- Neural Probabilistic models
    - Word2vec
    - Other popular embeddings: GloVe, FastText
- **Using word embeddings:**
    - **Sentence similarity task**
- State-of-the-art Embeddings
    - Seq-to-seq models, LMs
- Conclusion
- Q & A

How do we use word embeddings in a task?



Sentence Similarity using word embeddings

Task: Given two english sentences, calculate how similar they are.

```
from gensim.models import KeyedVectors
from gensim.utils import simple_preprocess

def clean_sentence(sentence, vocabulary):
    return [word for word in simple_preprocess(sentence) if word in vocabulary]

def compute_sentence_similarity(sentence_1, sentence_2, model_wv):
    vocabulary = set(model_wv.index2word)
    tokens_1 = clean_sentence(sentence_1, vocabulary)
    tokens_2 = clean_sentence(sentence_2, vocabulary)
    return model_wv.n_similarity(tokens_1, tokens_2)
```

Sentence 1: “ this is a sentence.”

Sentence 2: “this is also a sentence.”

Sentence 3: “today is a sunny day.”

```
filename = 'GoogleNews-vectors-negative300.bin'
model_wv = KeyedVectors.load_word2vec_format(filename, binary=True)
```

```
sim = compute_sentence_similarity('this is a sentence', 'this is also a sentence', model_wv)
print(sim)
```

```
0.95966756
```

```
sim = compute_sentence_similarity('this is a sentence', 'today is a sunny day', model_wv)
print(sim)
```

```
0.42698938
```

Agenda

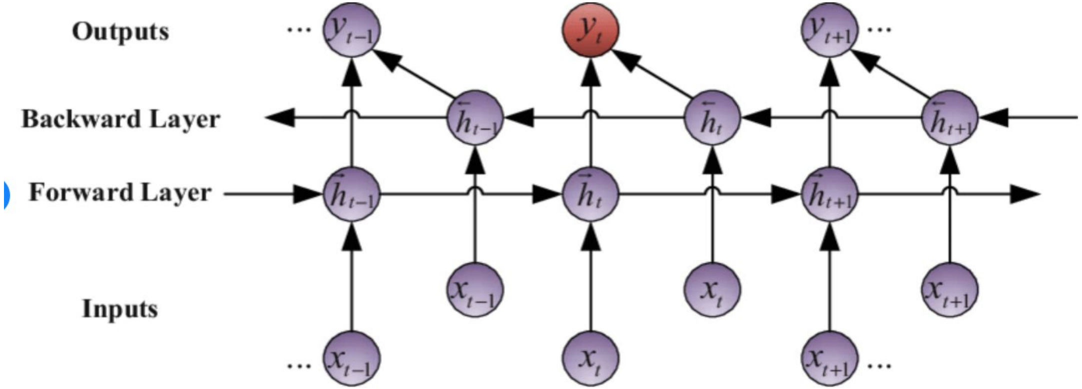
- What is NLP?
  - What are words?
  - Words representation in Traditional NLP
- Neural Probabilistic models
  - Word2vec
  - Other popular embeddings: GloVe, FastText
- Using word embeddings:
  - Sentence similarity task
- **State-of-the-art Embeddings**
  - **Seq-to-seq models, LMs**
- Conclusion
- Q & A

Motivation: Limitations of shallow representations

- **polysemy ⇒** words/phrases with different but related senses
  - Example: **bank**
    - Financial institution
    - A building where a financial institution can offer services
    - A synonym for "**rely upon**". (e.g. "I am your friend, you can bank on me.") Shared theme is security.
- **Out-of-vocabulary** words ⇒ words not present in the training corpus
- **Homonyms** ⇒ Not context specific
  - **Example:**
    - **"I am eating an apple."**
    - **"I have an Apple iphone"**

Deep Pre-trained Language Models

Language modelling



Bidirectional recurrent neural networks

Few popular pre-trained LMs

- ELMo - **E**mbeddings from **L**anguage **Mo**dels
- ULMFiT - **U**niversal **L**anguage **M**odel **Fi**ne-**T**uning
- Open AI GPT - **G**enerative **P**re-training **T**ransformer
- BERT - **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- Flair Embeddings - Contextual String Embeddings

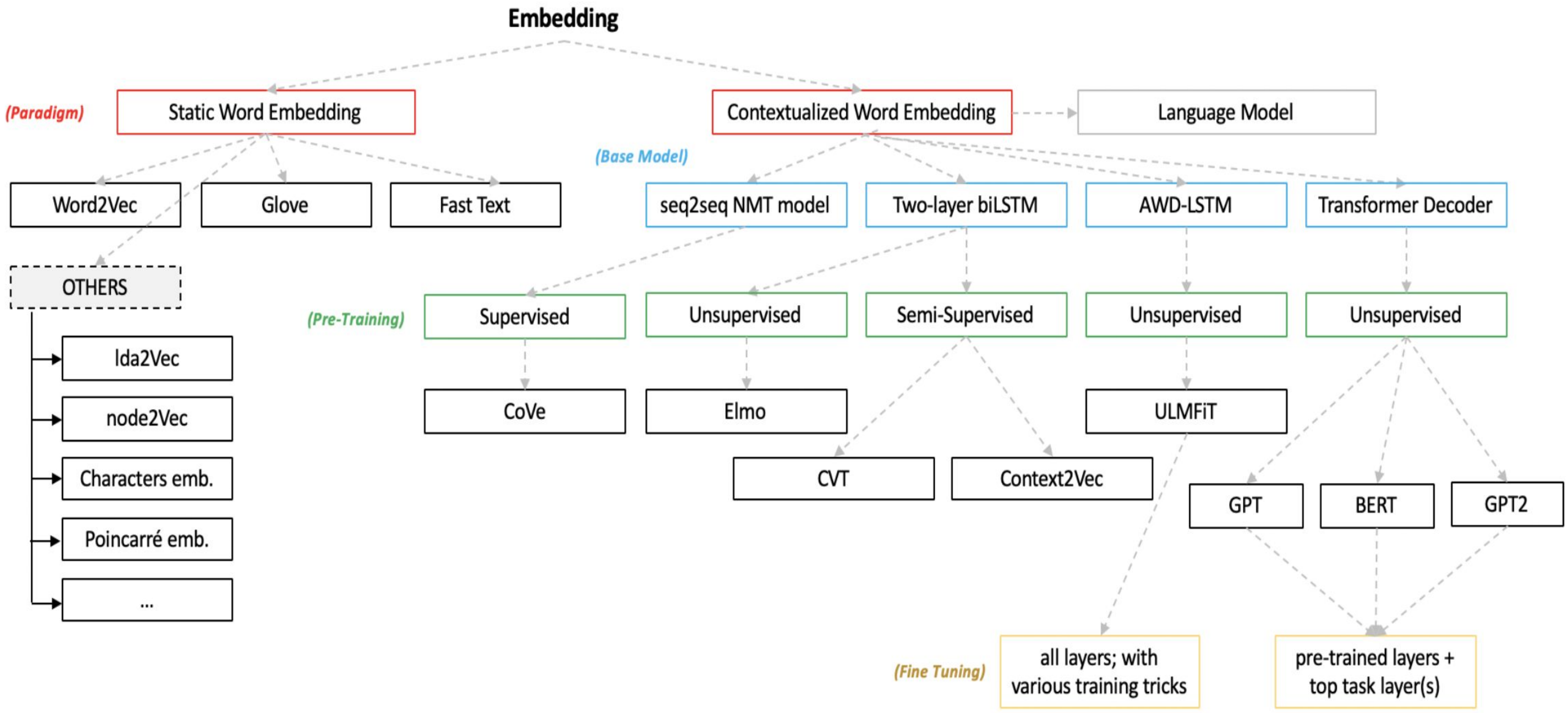
Beyond word embeddings

Image source

Few popular pre-trained models

- InferSent - Facebook
- Universal Sentence Encoders - Google

Conclusion



Embedding
(Paradigm)
Static Word Embedding
Contextualized Word Embedding
Language Model
(Base Model)
Word2Vec
Glove
Fast Text
seq2seq NMT model
Two-layer biLSTM
AWD-LSTM
Transformer Decoder
OTHERS
(Pre-Training)
Supervised
Unsupervised
Semi-Supervised
Unsupervised
Unsupervised
lda2Vec
node2Vec
Characters emb.
Poincarré emb.
...
CoVe
Elmo
ULMFiT
CVT
Context2Vec
GPT
BERT
GPT2
(Fine Tuning)
all layers; with
various training tricks
pre-trained layers +
top task layer(s)

References

- CS 224N, NLP with Deep Learning, Stanford University
- The Illustrated Word2Vec, Jay Alamar
- Gensim


THANK YOU
FOR LISTENING

Questions?

Let's connect!

@shweta_bhatt8

@shweta_bhatt

shwetabhatt08