Textual Feature Representation

CPE 393: Text Analytics

Dr. Sansiri Tarnpradab

Department of Computer Engineering
King Mongkut's University of Technology Thonburi

Announcement

Reminder

Midterm <u>next week</u>

- On paper
- Coverage: From 1st lecture 😬
- Duration: 2 Hrs (18.00 - 20.00)
- Closed-book
- Question Types:
    - Short-to-medium length
    - Multiple choice
    - Multiple selection
    - True/False
- Full scores: 50 or 100 🤔

Intro

Pattern Matching

Text Visualization

Web Scraping

Text Preparation

Text Feature Representation

Text Classification

Text Clustering

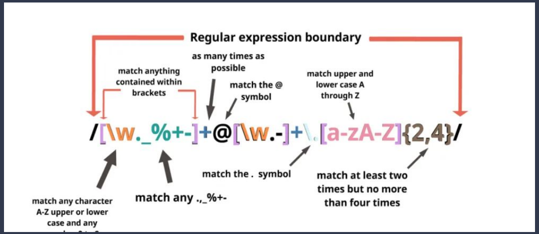
Topic Modeling

Extractive Summarization

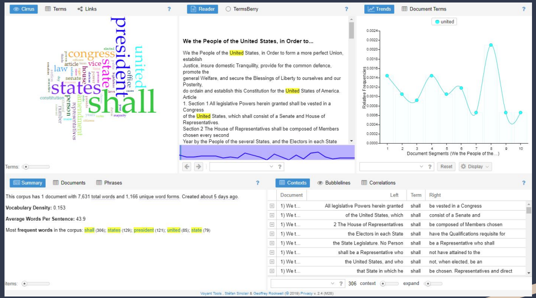
Abstractive Summarization

???

So far ...



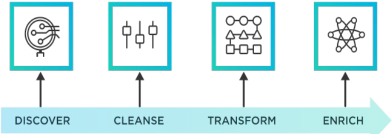
Ref: https://dev.to/mconner89/regular-expressions-grouping-and-string-methods-3ijn



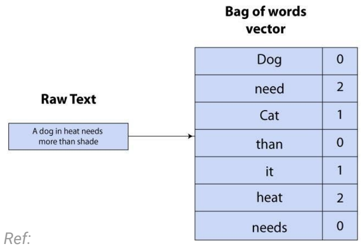
Ref: https://www.flerlagetwins.com/2019/09/text-analysis.html

Ref: https://shiftproxy.io/blog/post/web-scraping-101-what-why-and-how-350e1136

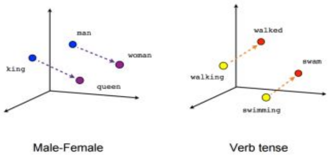
- Pattern Matching
- Textual Data Visualization
- API & Web Scraping
- Textual Data Preparation
- Textual Feature Representation



Ref: https://www.tibco.com/reference-center/what-is-data-preparation



Ref: https://www.analyticssteps.com/blogs/an-optimum-approach-towards-the-bag-of-words-with-code-illustration-in-python

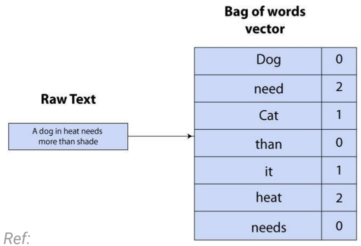


Ref: https://aylien.com/blog/word-embeddings-and-their-challenges

Outline

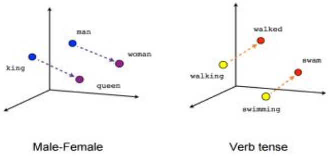
Text Feature Representation

- Discrete



Ref: https://www.analyticssteps.com/blogs/an-optimum-approach-towards-the-bag-of-words-with-code-illustration-in-python

- Continuous



Ref: https://aylien.com/blog/word-embeddings-and-their-challenges

Necessity

Why do we need Text Representation?

From: https://unsplash.com/photos/0alh2MojUuk

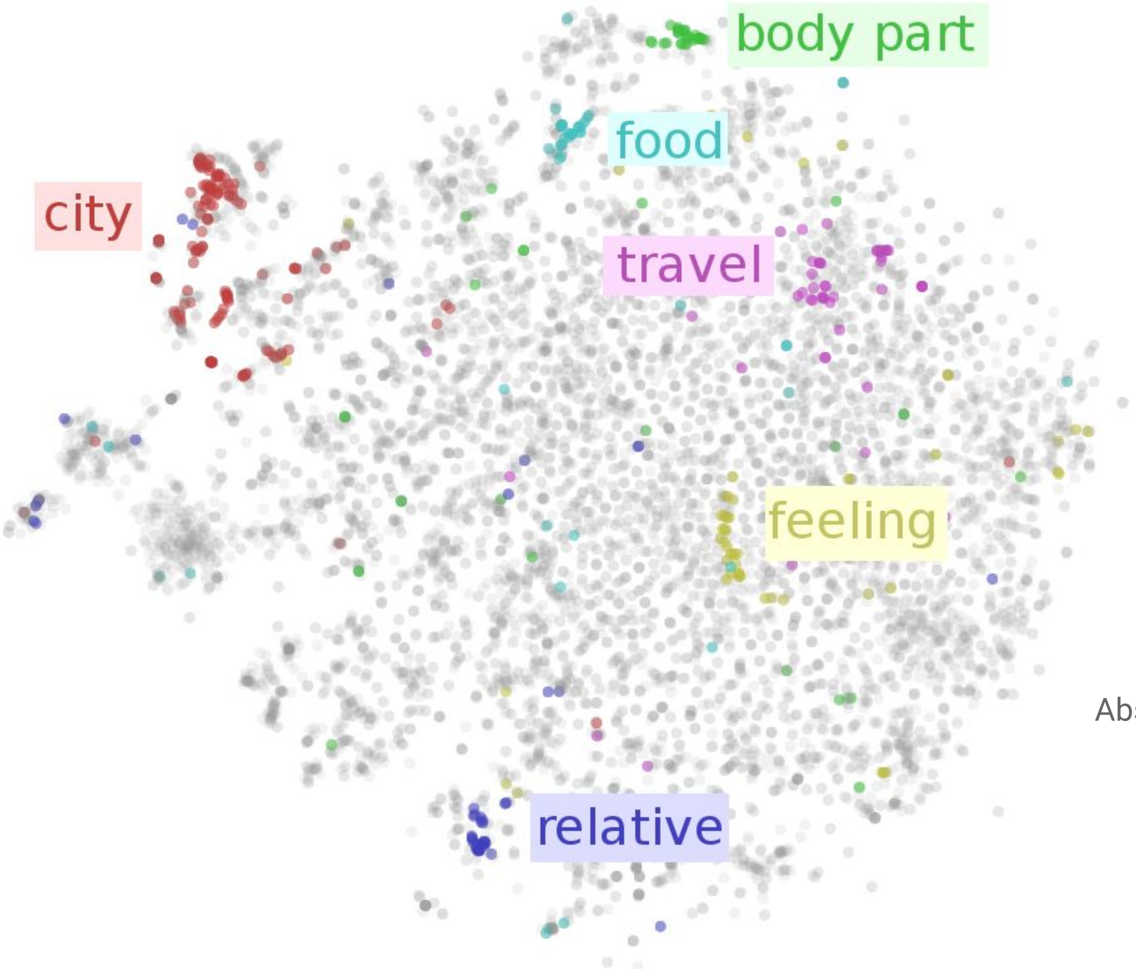
For **Human**: characters, words, sentences, …

For **Machine**: ?????????????????????????????

In a *nutshell*

Text Feature Representation is:

- A way to convert text in its natural form to vector form
- A format that is understandable by machine
- Numbers/vectors form



Word Embeddings

Embedding Space

or

Latent Space

Latent = Hidden

Abstract multi-dimensional space that encodes a meaningful internal representation of externally observed events [2]

Refs:
[1] Embedding space figure
[2] Short description of latent

Types of representation

Discrete text representations

- One-Hot encoding
- Bag-of-words (BOW)

Distributed text representations

- Co-Occurrence matrix
- Word2Vec
- GloVe

Discrete

Text Representation

Discrete

Text Representation

ONE-HOT ENCODING

- One-hot vector
- Traditional NLP
- Every element in a vector is assigned a value 0, <u>except for one</u> that is assigned 1

Example

- A 4-word sentence (s) is represented as a vector of 4 elements $\mathbb{R}^{|s|}$

- $w_1 \rightarrow$ [**1** 0 0 0]
- $w_2 \rightarrow$ [0 **1** 0 0]
- $w_3 \rightarrow$ [0 0 **1** 0]
- $w_4 \rightarrow$ [0 0 0 **1**]
- s $\rightarrow$ [**1**,0,0,0, 0,**1**,0,0, 0,0,**1**,0, 0,0,0,**1**]

Simple & Easy to understand

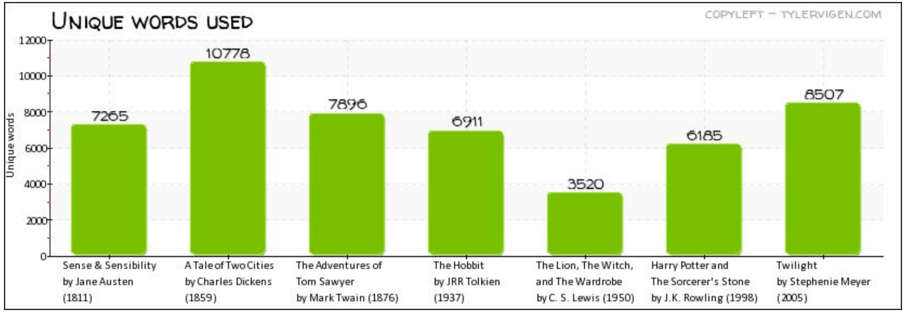
But…

Discrete

Text Representation

ONE-HOT ENCODING

- In reality, a document has (wayyy) more than 5 words
- Length of an array of word depends on the **vocabulary size**
- Vector dimension = number of words in the vocabulary



PROS:

- Simple & Easy to understand

CONS:

- Explosion in feature space
- Memory & computationally expensive
- Can only measure the word's existence, not its importance
- Cannot determine relationship between different words
- Out-of-vocabulary (OOV) case

Discrete
Text Representation

BAG-OF-WORDS

- **CountVectorizer**
- **TF-IDF Vectorizer**

Discrete

Text Representation

BAG-OF-WORDS

CountVectorizer

- Words put in a bag
- **Frequency** of each is counted
- Not take into account the **word order**
- Not take into account a **structure of words** in the document

Example

Excerpt from Adore You by Harry Styles

"Oh, honey, I would walk through fire for you"

"Just let me adore you"

"Like it is the only thing I will ever do"

Steps of CountVectorizer:

1. Tokenization
2. Build a vocabulary of unique words
3. Construct a DTM (document-term matrix)
4. Sparse representation (non-zero entries are stored)
5. Output matrix (Voila!)

```
   adore  do  ever  fire  for  honey  is  it  just  let  ...  me  oh  only  \
0      0   0     0     1    1      1   0   0     0    0  ...   0   1     0
1      1   0     0     0    0      0   0   0     1    1  ...   1   0     0
2      0   1     1     0    0      0   1   1     0    0  ...   0   0     1

   the  thing  through  walk  will  would  you
0    0      0        1     1     0      1    1
1    0      0        0     0     0      0    1
2    1      1        0     0     1      0    0

[3 rows x 21 columns]
```

Discrete

Text Representation

[[0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1]

[1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1]

[0 1 1 0 0 0 1 1 0 0 1 0 0 1 1 1 0 0 1 0 0]]

BAG-OF-WORDS

CountVectorizer

PROS:

- Gives frequency of words which One-hot encoding doesn’t
- Length of the encoded vector = length of the dictionary

CONS:

- Ignores the location information of the word (context-free)
- High-frequency words have higher importance... but, **stopwords**?

Discrete

Text Representation

BAG-OF-WORDS

CountVectorizer

TF-IDF Vectorizer

Discrete
Text Representation

BAG-OF-WORDS

Tf-Idf Vectorizer

- TF = Term frequency
- IDF = Inverse document frequency
- Tf-Idf is a product of the above 2 factors

$$TFIDF = TF(w,d) * IDF(w)$$

$$IDF(w) = log(\frac{N}{df(w)})$$

where ***TF(w, d)*** is frequency of word w in document d;

For ***IDF(w), N*** is total number of documents, ***df(w)*** is the frequency of documents containing the word w.

The **weight** assigned to each word not only depends on the frequency, but also how frequent that particular word is in the entire corpora.

```
The text:  ['Oh, honey, I would walk through fire for you', 'Just let me adore you', 'Like it is the only thing I will ever do']
{'adore': 1.6931471805599454, 'do': 1.6931471805599454, 'ever': 1.6931471805599454, 'fire': 1.6931471805599454, 'for': 1.6931471805599454, 'honey': 1.6931471805599454, 'is': 1.6931471805599454, 'it': 1.6931471805599454, 'just': 1.6931471805599454, 'let': 1.6931471805599454, 'like': 1.6931471805599454, 'me': 1.6931471805599454, 'oh': 1.6931471805599454, 'only': 1.6931471805599454, 'the': 1.6931471805599454, 'thing': 1.6931471805599454, 'through': 1.6931471805599454, 'walk': 1.6931471805599454, 'will': 1.6931471805599454, 'would': 1.6931471805599454, 'you': 1.2876820724517808}
```

Discrete

Text Representation

BAG-OF-WORDS

Tf-Idf Vectorizer

PROS:

- Simple and easy to implement
- Address the flaw of CountVectorizer
- Reduce noise

CONS:

- Positional information of the word is not captured
- Highly depend on a corpus.

```
The text:  ['Oh, honey, I would walk through fire for you', 'Just let me adore you', 'Like it is the only thing I will ever do']
{'adore': 1.6931471805599454, 'do': 1.6931471805599454, 'ever': 1.6931471805599454, 'fire': 1.6931471805599454, 'for':
1.6931471805599454, 'honey': 1.6931471805599454, 'is': 1.6931471805599454, 'it': 1.6931471805599454, 'just': 1.6931471805599454,
'let': 1.6931471805599454, 'like': 1.6931471805599454, 'me': 1.6931471805599454, 'oh': 1.6931471805599454, 'only':
1.6931471805599454, 'the': 1.6931471805599454, 'thing': 1.6931471805599454, 'through': 1.6931471805599454, 'walk':
1.6931471805599454, 'will': 1.6931471805599454, 'would': 1.6931471805599454, 'you': 1.2876820724517808}
```

Discrete

Text Representation

SUMMARY

- Words are converted into a numerical format based on:
    - Existence
    - Frequency
    - Weighted frequency

PROS:

- Simple
- Easy to implement

CONS:

- Proportional to vocab size
- Explosion in feature space
- All words are independent of each other
- Do not capture context and semantics of the word

Distributed

Text Representation

Distributed

Text Representation

CO-OCCURRENCE MATRIX

- The matrix is generated from a co-occurrence of entities nearby each other.
- Capture association between words in a corpus
- **Words that are similar to each other will tend to co-occur together**

Co-occurrence

For a given set of documents, the co-occurrence of a pair of words is equal to the frequency the two words have appeared together in a **context** window.

Context window

- 1-gram, 2-gram, phrase that are nearby

Distributed

Text Representation

CO-OCCURRENCE MATRIX

1-gram

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

2-gram

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

Distributed
Text Representation

CO-OCCURRENCE MATRIX

Example

I'm riding in my car to the beach.
I'm riding in my jeep to the beach.
My car is jeep.
My jeep is a car.
I ate a banana yesterday.
I ate a peach yesterday.

	ate	banana	beach	car	jeep	peach	riding	yesterday
ate	2	1	0	0	0	1	0	2
banana	1	1	0	0	0	0	0	1
beach	0	0	2	1	1	0	2	0
car	0	0	1	3	2	0	1	0
jeep	0	0	1	2	3	0	1	0
peach	1	0	0	0	0	1	0	1
riding	0	0	2	1	1	0	2	0
yesterday	2	1	0	0	0	1	0	2

Distributed
Text Representation

CO-OCCURRENCE MATRIX

Windows & Scaling

	Even	the	smallest	person	can	change	the	course	of	history
windows: 3	5	4	3	2	1	0	1	2	3	4
scaling: flat	0	0	1	1	1	1	1	1	1	0
scaling: 1/n	0	0	1/3	1/2	1/1	1	1/1	1/2	1/3	0

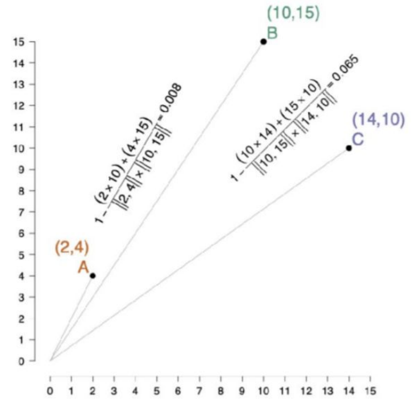
- Different window sizes capture more or less info
- Larger window captures more semantic info
- With the scaling of 1/n, a word is far from the target word will be assigned lesser values.

How to measure similarity?

Distributed

Text Representation

SIMILARITY MEASURE



Two common methods to measure a distance between vectors in a vector space.

Euclidean Distance

$$euclidean(u, v) = \sqrt{\sum_{i=1}^{n} |u_i - v_i|^2}$$

Cosine Similarity

Dot product of the vectors divided by the product of their magnitudes.

$$\cos(\theta) = \frac{u \cdot v}{\|u\|\|v\|} = \frac{\sum_{i=1}^{n} u_i \times v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

Distributed
Text Representation

CO-OCCURRENCE MATRIX

PROS:

- Include word association information
- Take order of words into consideration

CONS:

- Sparse matrix
- Not storage efficient

- Not all word associations can be understood

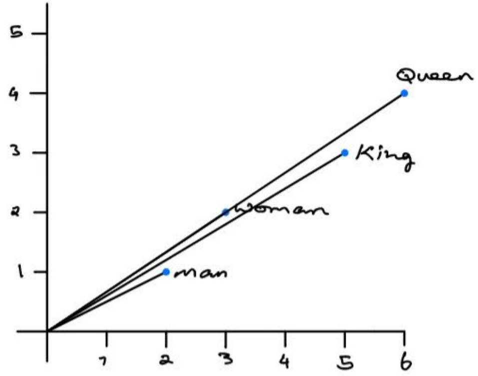
Distributed
Text Representation

WORD2VEC

- By Mikolov et al. 2013
- Is a framework for learning word vectors
- Its effectiveness is from the ability to group together vectors of similar words.

Classic Example

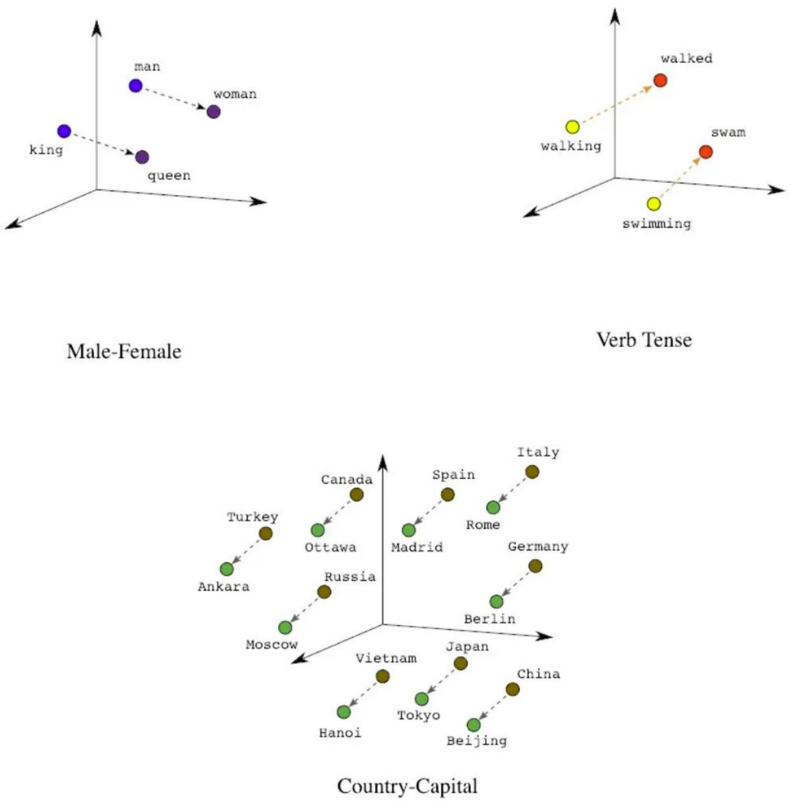
King, Queen, Man, Woman



King	-	Man	+	Woman	=	Queen
[5,3]	-	[2,1]	+	[3, 2]	=	[6,4]

Distributed
Text Representation

WORD2VEC

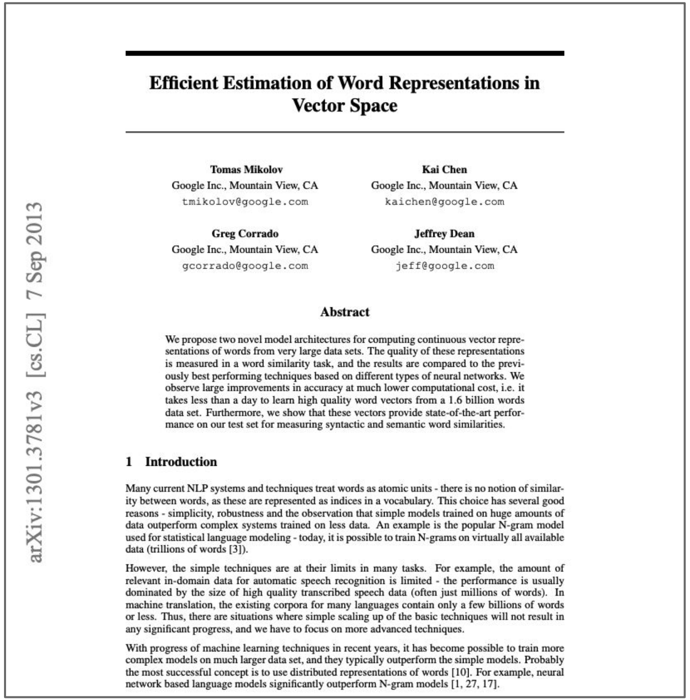


Ref: https://medium.com/@towardsautonomy/word2vec-part1-5436c846c3fa

Distributed

Text Representation

WORD2VEC



arXiv:1301.3781v3 [cs.CL] 7 Sep 2013

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1, 27, 17].

Download

Two architectures that contribute to word2vec. (Either one could be used)

- Continuous Bag of Words (CBOW)
- Continuous Skip-Gram

Distributed
Text Representation

WORD2VEC

The idea in a nutshell

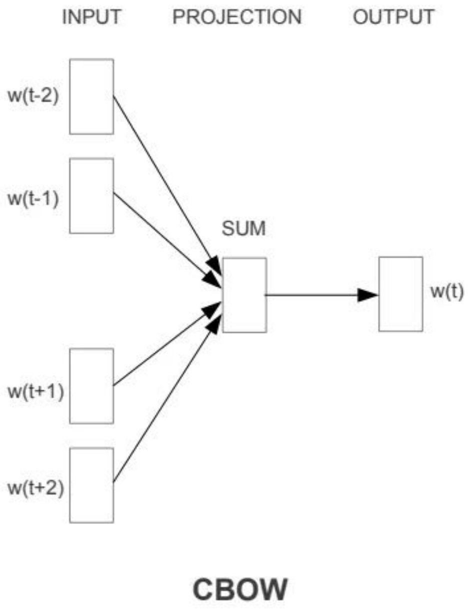
- Have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position `t` in the text, which has a center word `c` and context word `o`
- Use the similarity of the word vectors for `c` and `o` to calculate the probability of `o` given `c` (or vice versa)
- Keep adjusting the word vectors to maximize this probability

Distributed
Text Representation

WORD2VEC

CBOW model

- Architecture similar to feed forward neural network
- **Goal**: to predict a target word from a list of context words
- **How**: take the distributed representations of the context words and predict the target word



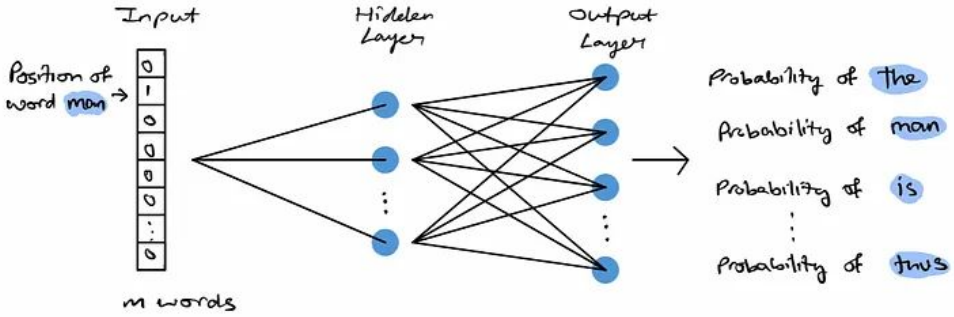
- S. Tarnpradab -

Distributed
Text Representation

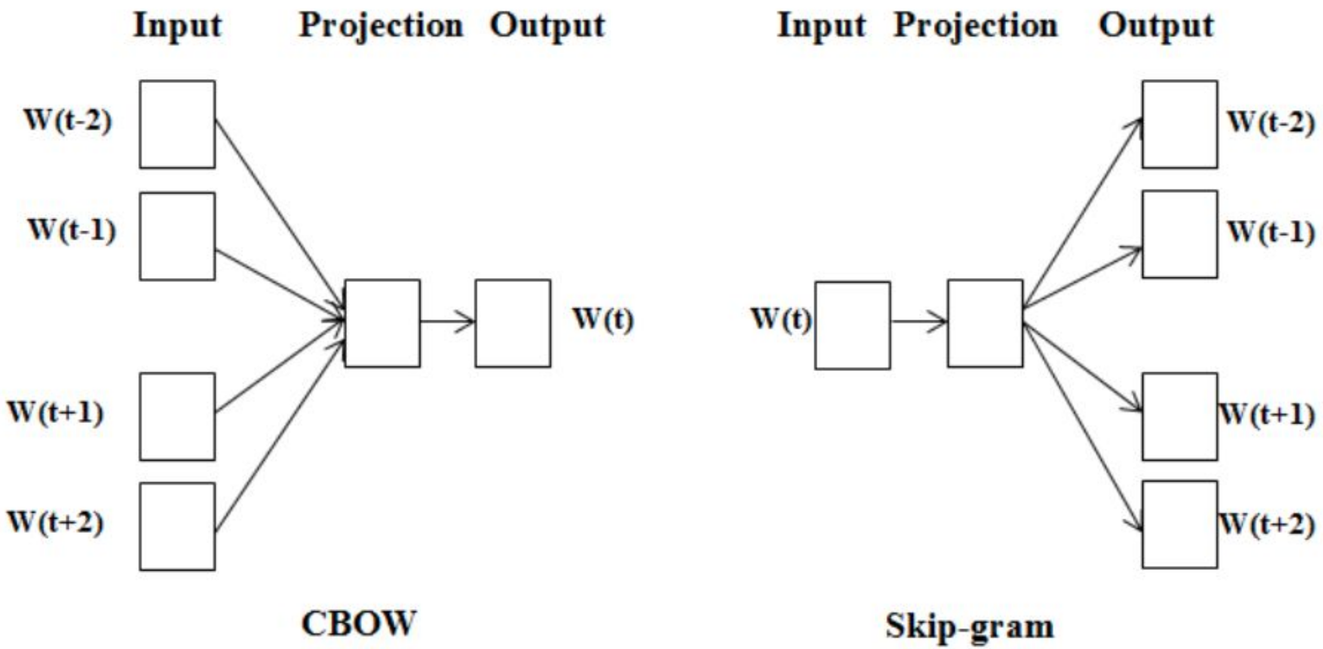
WORD2VEC

Continuous Skip-Gram model

- Opposite of CBOW model
- **Goal**: Simple neural network with one hidden layer trained to predict the probability of a given word being present when an input word is present.
- **How**: the model takes the current word as an input and tries to predict the words before and after the current word. (learn to predict the context words around the input word)



Ref: https://towardsdatascience.com/word2vec-explained-49c52b4ccb71



CBOW vs Skip-gram side by side

Distributed
Text Representation

GLOVE

- word2vec relies on local statistics – the local context information of words.
- GloVe also incorporates global statistics –word co-occurrence – to obtain word vectors

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

Abstract

Recent methods for learning vector space representations of words have succeeded in capturing fine-grained semantic and syntactic regularities using vector arithmetic, but the origin of these regularities has remained opaque. We analyze and make explicit the model properties needed for such regularities to emerge in word vectors. The result is a new global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. Our model efficiently leverages statistical information by training only on the nonzero elements in a word-word co-occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus. The model produces a vector space with meaningful substructure, as evidenced by its performance of 75% on a recent word analogy task. It

the finer structure of the word vector space by examining not the scalar distance between word vectors, but rather their various dimensions of difference. For example, the analogy "king is to queen as man is to woman" should be encoded in the vector space by the vector equation *king* − *queen* = *man* − *woman*. This evaluation scheme favors models that produce dimensions of meaning, thereby capturing the multi-clustering idea of distributed representations (Bengio, 2009).

The two main model families for learning word vectors are: 1) global matrix factorization methods, such as latent semantic analysis (LSA) (Deerwester et al., 1990) and 2) local context window methods, such as the skip-gram model of Mikolov et al. (2013c). Currently, both families suffer significant drawbacks. While methods like LSA efficiently leverage statistical information, they do relatively poorly on the word analogy task, indicating a sub-optimal vector space structure. Methods like skip-gram may do better on the analogy task, but they poorly utilize the statistics of the corpus since they train on separate local context windows instead of on global co-occurrence counts.

Download

Distributed
Text Representation

GLOVE

Example from Glove

Word embedding for the word “king” trained on Wikipedia.

```
[ 0.50451 , 0.68607 , -0.59517 , -0.022801,
0.60046 , -0.13498 , -0.08813 , 0.47377 ,
-0.61798 , -0.31012 , -0.076666, 1.493 ,
-0.034189, -0.98173 , 0.68229 , 0.81722 ,
-0.51874 , -0.31503 , -0.55809 , 0.66421 ,
0.1961 , -0.13495 , -0.11476 , -0.30344 ,
0.41177 , -2.223 , -1.0756 , -1.0783 ,
-0.34354 , 0.33505 , 1.9927 , -0.04234 ,
-0.64319 , 0.71125 , 0.49159 , 0.16754 ,
0.34344 , -0.25663 , -0.8523 , 0.1661 ,
0.40102 , 1.1685 , -1.0137 , -0.21585 ,
-0.15155 , 0.78321 , -0.91241 , -1.6106 ,
-0.64426 , -0.51042 ]
```

Distributed Text Representation

WORD2VEC
VS
GLOVE

W2V

PROS: Able to capture relationships between different words including their syntactic & semantic relationships

CONS: OOV case, w2v merely assigns a random vector representation for those OOV words. It also relies on local information.

GLOVE

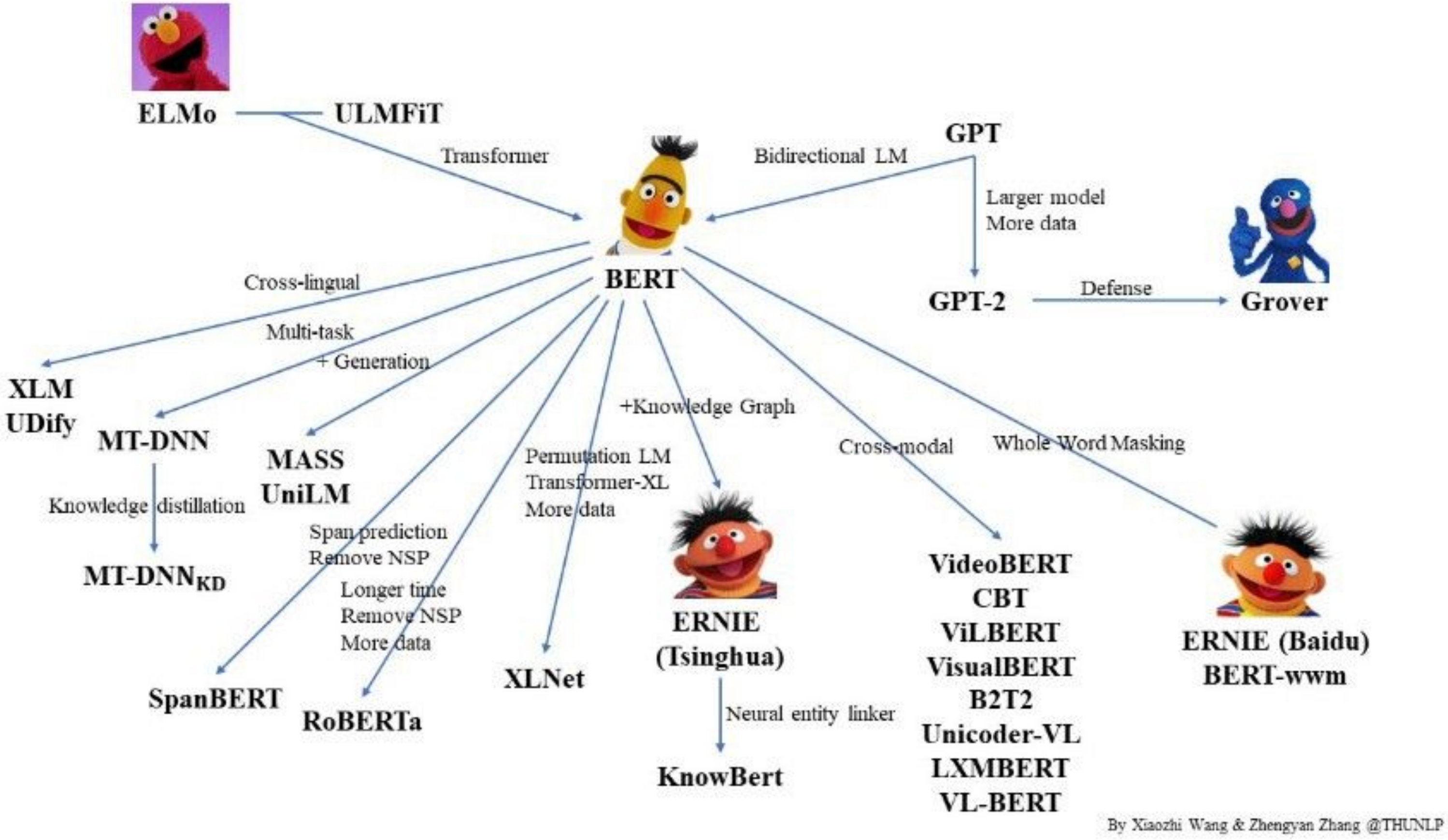
PROS: Address the limitation of w2v – also use the global information

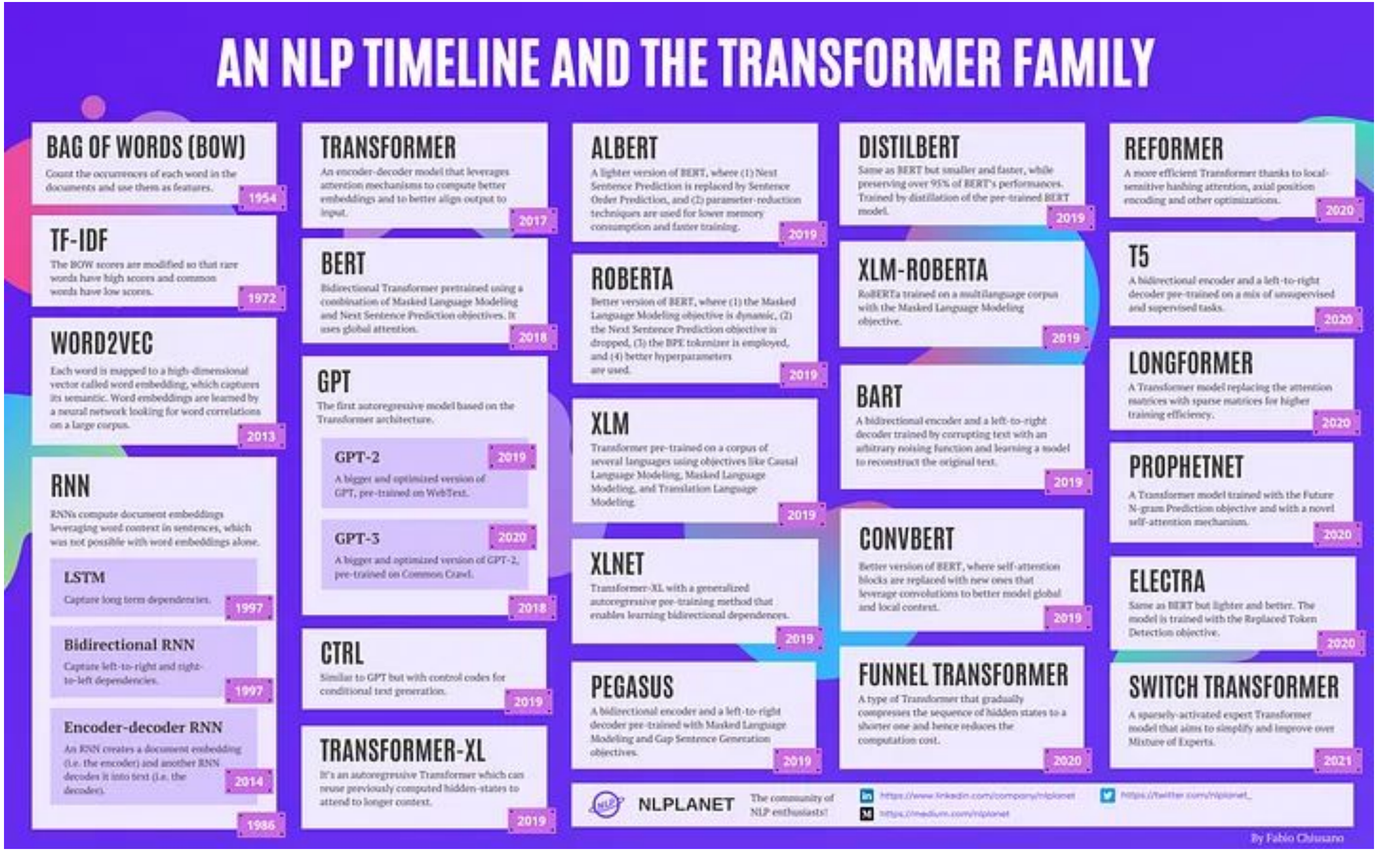
CONS: Using a co-occurrence matrix and global information requires more memory in GloVe than word2vec.

Other methods:

- FastText
- ELMO
- BERT (and many variations of BERT)
- GPT

- S. Tarnpradab -


ELMo
ULMFiT
Transformer
BERT
Bidirectional LM
GPT
Larger model
More data
GPT-2
Defense
Grover
Cross-lingual
Multi-task
+ Generation
XLM
UDify
MT-DNN
MASS
UniLM
Knowledge distillation
MT-DNN$_{KD}$
Span prediction
Remove NSP
Longer time
Remove NSP
More data
SpanBERT
RoBERTa
Permutation LM
Transformer-XL
More data
XLNet
+Knowledge Graph
ERNIE
(Tsinghua)
Neural entity linker
KnowBert
Cross-modal
VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMBERT
VL-BERT
Whole Word Masking
ERNIE (Baidu)
BERT-wwm
By Xiaozhi Wang & Zhengyan Zhang @THUNLP



AN NLP TIMELINE AND THE TRANSFORMER FAMILY
BAG OF WORDS (BOW)
Count the occurrences of each word in the documents and use them as features.
1954
TF-IDF
The BOW scores are modified so that rare words have high scores and common words have low scores.
1972
WORD2VEC
Each word is mapped to a high-dimensional vector called word embedding, which captures its semantic. Word embeddings are learned by a neural network looking for word correlations on a large corpus.
2013
RNN
RNNs compute document embeddings leveraging word context in sentences, which was not possible with word embeddings alone.
LSTM
Capture long term dependencies.
1997
Bidirectional RNN
Capture left-to-right and right-to-left dependencies.
1997
Encoder-decoder RNN
An RNN creates a document embedding (i.e. the encoder) and another RNN decodes it into text (i.e. the decoder).
2014
1986
TRANSFORMER
An encoder-decoder model that leverages attention mechanisms to compute better embeddings and to better align output to input.
2017
BERT
Bidirectional Transformer pretrained using a combination of Masked Language Modeling and Next Sentence Prediction objectives. It uses global attention.
2018
GPT
The first autoregressive model based on the Transformer architecture.
GPT-2
2019
A bigger and optimized version of GPT, pre-trained on WebText.
GPT-3
2020
A bigger and optimized version of GPT-2, pre-trained on Common Crawl.
2018
CTRL
Similar to GPT but with control codes for conditional text generation.
2019
TRANSFORMER-XL
It's an autoregressive Transformer which can reuse previously computed hidden-states to attend to longer context.
2019
ALBERT
A lighter version of BERT, where (1) Next Sentence Prediction is replaced by Sentence Order Prediction, and (2) parameter-reduction techniques are used for lower memory consumption and faster training.
2019
ROBERTA
Better version of BERT, where (1) the Masked Language Modeling objective is dynamic, (2) the Next Sentence Prediction objective is dropped, (3) the BPE tokenizer is employed, and (4) better hyperparameters are used.
2019
XLM
Transformer pre-trained on a corpus of several languages using objectives like Causal Language Modeling, Masked Language Modeling, and Translation Language Modeling.
2019
XLNET
Transformer-XL with a generalized autoregressive pre-training method that enables learning bidirectional dependencies.
2019
PEGASUS
A bidirectional encoder and a left-to-right decoder pre-trained with Masked Language Modeling and Gap Sentence Generation objectives.
2019
DISTILBERT
Same as BERT but smaller and faster, while preserving over 95% of BERT's performances. Trained by distillation of the pre-trained BERT model.
2019
XLM-ROBERTA
RoBERTa trained on a multilanguage corpus with the Masked Language Modeling objective.
2019
BART
A bidirectional encoder and a left-to-right decoder trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text.
2019
CONVBERT
Better version of BERT, where self-attention blocks are replaced with new ones that leverage convolutions to better model global and local context.
2019
FUNNEL TRANSFORMER
A type of Transformer that gradually compresses the sequence of hidden states to a shorter one and hence reduces the computation cost.
2020
REFORMER
A more efficient Transformer thanks to local-sensitive hashing attention, axial position encoding and other optimizations.
2020
T5
A bidirectional encoder and a left-to-right decoder pre-trained on a mix of unsupervised and supervised tasks.
2020
LONGFORMER
A Transformer model replacing the attention matrices with sparse matrices for higher training efficiency.
2020
PROPHETNET
A Transformer model trained with the Future N-gram Prediction objective and with a novel self-attention mechanism.
2020
ELECTRA
Same as BERT but lighter and better. The model is trained with the Replaced Token Detection objective.
2020
SWITCH TRANSFORMER
A sparsely-activated expert Transformer model that aims to simplify and improve over Mixture of Experts.
2021
NLPLANET
The community of NLP enthusiasts!
https://www.linkedin.com/company/nlplanet
https://twitter.com/nlplanet_
https://medium.com/nlplanet
By Fabio Chiusano

Data

Download customer complaint data via this link.

Conclusion

Text feature representation

- Discrete
  - One-Hot encoding
  - Bag-of-words (BOW)
- Continuous
  - Co-Occurrence matrix
  - Word2Vec
  - GloVe

Q & A