Word Embeddings

MICHAEL WOLLOWSKI

1

Knowledge Representation in Classic AI

- In order to understand the power of NN-based KR, it is enlightening to understand the limitations of classic KR.
- A key KR mechanism are frames.
- Another one are semantic nets.
- Eventually, both frames and semantic nets incorporated the benefits of the other mechanism into their own
- Another one are rules with certainty factors.

2

KR: Frames

- Frames were developed in 1974
- They are a compact way to represent information that might be considered a concept.
- They were a response to the helter-skelter way of storing information in a logical theorem prover based reasoning system.

A frame includes:

1. *Frame identification information.*
2. *Relationship of this frame to other frames.* The "hotel phone" might be a special instance of "phone," which might be an instance of a "communication device."
3. *Descriptors of requirements for a frame.* A chair, for instance, has its seat between 20 and 40 cm from the floor, its back higher than 60 cm, etc. These requirements may be used to determine when new objects fit the stereotype defined by the frame.
4. *Procedural information on use of the structure described.* An important feature of frames is the ability to attach procedural code to a slot.
5. *Frame default information.* These are slot values that are taken to be true when no evidence to the contrary has been found. For instance, chairs have four legs, telephones are pushbutton, or hotel beds are made by the staff.
6. *New instance information.* Many frame slots may be left unspecified until given a value for a particular instance or when they are needed for some aspect of problem solving. For example, the color of the bedspread may be left unspecified.

Source: Luger: Artificial Intelligence, 6th Ed.

3

KR: Frames

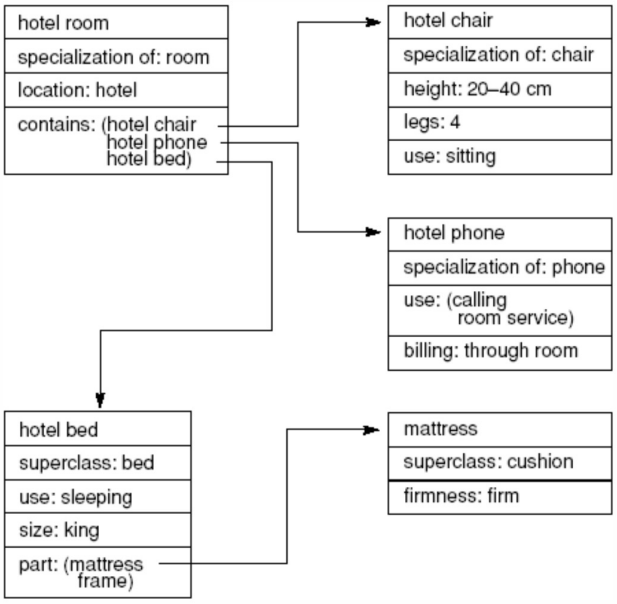
Frames are like objects.

They have fields.

Some fields have pointers to other objects

Some fields have Strings or simple types as values.

When determining whether two Strings are the same, one has to use string comparison.



Source: Luger: Artificial Intelligence, 6th Ed.

4

Frames in IBM's Watson Jeopardy!© Player

The IBM Watson Jeopardy!© Player used them to represent information in a quickly accessible format.

Here is a sample frame, used to present the sentence: "In 1921 Albert Einstein won the Nobel prize for his work on the photoelectric effect."

Frame01	
verb	receive
subj	Einstein
type	PERSON / SCIENTIST
obj	Nobel prize
mod_vprep	in
objprep	1921
type	YEAR
mod_vprep	for
objprep	Frame02

Frame02	
noun	work
mod_ndet	his / Einstein
mod_nobj	on
objprep	effect

Table 2: Frames extracted from Dependency Parse in Figure 2

Source: Fan, Kalyanpur, Gondek, Ferrucci: Automatic knowledge extraction from documents. IBM J. Res. & Sys. Vol 56. No ¾. 2012.

5

Frames in IBM's Watson Jeopardy!© Player

Watson was a high-water mark for NLP, before NN took over.

In some ways, it still has not been surpassed, think hallucinations.

Jeopardy!:

- Is an open domain quiz show
- Requires a breadth of knowledge
- Requires confidence in one's knowledge
- Players get penalized for incorrect answers.
- The level of language is very high.

Relation	Description/Example
subj	subject
obj	direct object
iobj	indirect object
comp	complement
pred	predicate complement
objprep	object of the preposition
mod_nprep	*Bat Cave in Toronto is a tourist attraction.*
mod_vprep	*He made it to Broadway.*
mod_nobj	the object of a nominalized verb
mod_ndet	*City's budget was passed.*
mod_ncomp	*Tweet is a word for microblogging.*
mod_nsubj	*A poem by Byron*
mod_aobj	*John is similar to Steve.*
isa	subsumption relation
subtypeOf	subsumption relation

Table 1: Relations used in a frame and their descriptions

Source: Fan, Kalyanpur, Gondek, Ferrucci: Automatic knowledge extraction from documents. IBM J. Res. & Sys. Vol 56. No ¾. 2012.

6

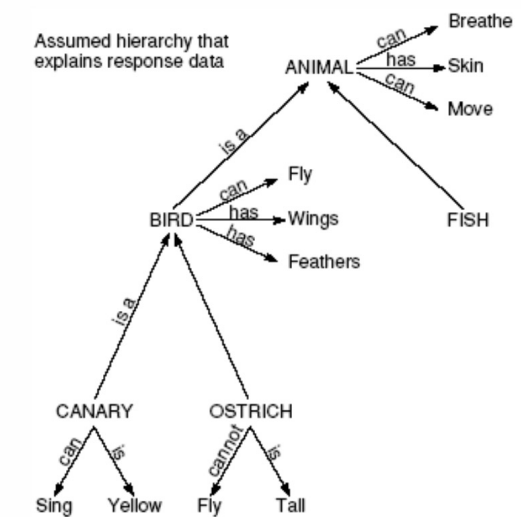
Semantic Networks

Semantic networks were developed as early as 1960.

They are a compact way of representing information.

Objects inherit information from those higher up in the hierarchy.

The nodes in the network typically represent an ontology.



Source: Luger: Artificial Intelligence, 6th Ed.

7

Rules

Expert System were developed during the 70's.

They were big during the 80's

They used rules and confidence factors.

The reasoning in those systems used the confidence factors to determine the certainty for the conclusions it reached.

Rules were developed by domain experts

They were fine-tuned with the help of knowledge engineers.

RULE037

IF: 1) The identity of the organism is not known with certainty, and
2) The stain of the organism is gramneg, and
3) The morphology of the organism is rod, and
4) The aerobicity of the organism is aerobic
THEN: There is strongly suggestive evidence (.8) that the class of the organism is enterobacteriaceae

RULE145

IF: 1) The therapy under consideration is one of: cephalothin clindamycin erythromycin lincomycin vancomycin, and
2) Meningitis is an infectious disease diagnosis for the patient
THEN: It is definite (1) the therapy under consideration is not a potential therapy for use against the organism

RULE060

IF: The identity of the organism is bacteroides
THEN: I recommend therapy chosen from among the following drugs:
1 - clindamycin (.99)
2 - chloramphenicol (.99)
3 - erythromycin (.57)
4 - tetracycline (.28)
5 - carbenicillin (.27)

Source: Buchanan and Shortliffe: Rule-Based Expert Systems

8

Word Embeddings

When talking about NN and NLP, we need to know about word embeddings.

Basically, a word embedding is a mapping of a word into a very high dimensioned vector space.

200-300 dimensions are not unusual.

Each word is mapped into the vector space and as such is represented by a multi-dimensional vector.

9

Unique IDs vs Strings

Glove word embedding for “dog”

dog 0.11008 -0.38781 -0.57615 -0.27714 0.70521 0.53994 -1.0786 -0.40146 1.1504 -0.5678 0.0038977 0.52878 0.64561 0.47262 0.48549 -0.18407 0.1801 0.91397 -1.1979 -0.5778 -0.37985 0.33606 0.772 0.75555 0.45506 -1.7671 -1.0503 0.42566 0.41893 -0.68327 1.5673 0.27685 -0.61708 0.64638 -0.076996 0.37118 0.1308 -0.45137 0.25398 -0.74392 -0.086199 0.24068 -0.64819 0.83549 1.2502 -0.51379 0.04224 -0.88118 0.7158 0.38519

10

Word Embeddings

Semantic similarity.
- Words with similar meanings have vectors that are close together in the embedding space.
- For example, "king" and "queen" would have similar vectors, as would "cat" and "dog."

Syntactic relationships.
- Embeddings capture grammatical patterns and word relationships.
- Words that play similar grammatical roles tend to cluster together.
- For example, like verbs with verbs, adjectives with adjectives.

Algebraic properties.
- Perhaps most famously, embeddings exhibit meaningful vector arithmetic.
- The classic example is: vector("king") - vector("man") + vector("woman") ≈ vector("queen").
- These analogies work because the embeddings capture relational structure.

Compositionality.
- Word vectors can sometimes be combined to represent phrases or sentences.

11

Word Embeddings: Semantic Similarity

Consider the following visualization of number related terms.

This is an excerpt of a larger space (see below for the image source of the entire space)

The number words are in their little partition of the word vector space.

Image source: http://metaoptimize.s3.amazonaws.com/cw-embeddings-ACL2010/embeddings-mostcommon.EMBEDDING_SIZE=50.png

12

Word Embeddings: Semantic Similarity

Word embeddings can be quite useful in language translation.

Consider the following overlay of English and Chinese word embeddings.

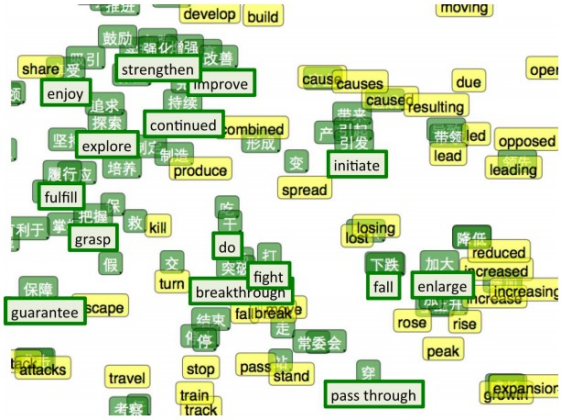


Image source: http://ai.stanford.edu/~wzou/emnlp2013_ZouSocherCerManning.pdf

13

Word Embeddings: Algebraic Properties

king = [0.5, 0.8, 0.1]
queen = [0.5, 0.8, -0.9]
man = [0.1, 0.3, 0.1]
woman = [0.1, 0.3, -0.9]

We want to solve: **"king" - "man" + "woman" ≈ ?**

Step 1: king - man [0.5, 0.8, 0.1] - [0.1, 0.3, 0.1] = [0.4, 0.5, 0.0]

Step 2: (king - man) + woman [0.4, 0.5, 0.0] + [0.1, 0.3, -0.9] = [0.5, 0.8, -0.9]

Result: [0.5, 0.8, -0.9] = queen ✓

14

Word Embeddings: Algebraic Properties

Paris = [0.2, 0.9, 0.3] (capital, Western Europe, large city)
France = [0.0, 0.9, 0.5] (country, Western Europe, medium)
Italy = [0.0, 0.85, 0.4] (country, Western/South Europe, medium)
Rome = [0.2, 0.85, 0.25] (capital, Southern Europe, large city)

Calculation: **Paris - France + Italy**
[0.2, 0.9, 0.3] - [0.0, 0.9, 0.5] + [0.0, 0.85, 0.4] =
[0.2, 0.0, -0.2] + [0.0, 0.85, 0.4] =
[0.2, 0.85, 0.2] ≈ [0.2, 0.85, 0.25] = Rome ✓

15

Word Embeddings: Compositionality

Simple Vector Addition

- vector("red apple") ≈ vector("red") + vector("apple")

Or with averaging:

- vector("red apple") ≈ [vector("red") + vector("apple")] / 2

What happens: The resulting vector should capture both the color concept and the fruit concept.

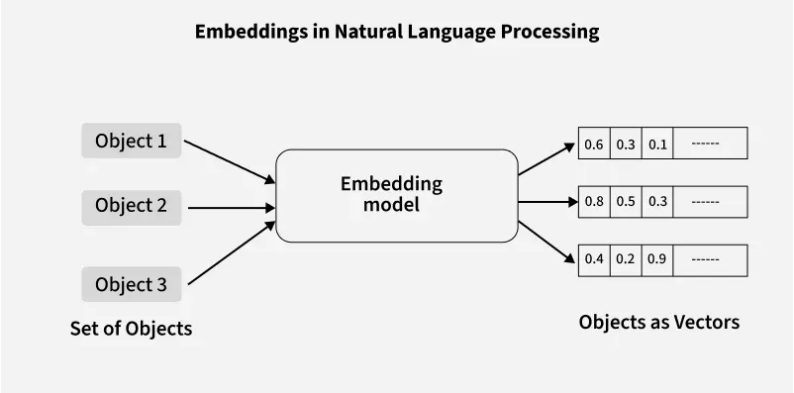
If you query for the nearest neighbors of this combined vector in the embedding space, you might find words like:

- "strawberry" (another red fruit)
- "cherry" (another red fruit)
- "tomato" (red and round)
- "crimson" (shade of red)

16

Word Embeddings

- The input, say an English language sentence is translated into a sequence of vectors.
- This vector sequence can be seen as a different language.
- The biggest difference to classic KR is that each input token is translated to a unique, corresponding system token.
- There is not need for String comparison.
- Any processing is automatically being performed on the same token.



Source: https://www.geeksforgeeks.org/nlp/word-embeddings-in-nlp/

17

Unique IDs vs Strings

Frames (in classic AI)

```
+-----------------------------------------+
| Book Frame                              |
+-----------------------------------------+
| Slots:                                  |
| Title: "To Kill a Mockingbird"          |
| Author: "Harper Lee"                    |
| Publication Year: 1960                  |
| ISBN: "978-0-06-112008-4"               |
| Genre: "Fiction"                        |
+-----------------------------------------+
| Facets:                                 |
| Publication Year:                       |
| - Type: Integer                         |
| - Range: 1450 to current year           |
| ISBN:                                   |
| - Format: 13-digit number               |
+-----------------------------------------+
| Default Values:                         |
| Genre: "Unknown"                        |
+-----------------------------------------+
```

Slot	Value	Type
ALEX	_	(This Frame)
NAME	Alex	(key value)
ISA	Boy	(parent frame)
SEX	Male	(inheritance value)
AGE	IF-NEEDED: Subtract(current,BIRTHDATE) ;	(procedural attachment)
HOME	100 Main St.	(instance value)
BIRTHDATE	8/4/2000	(instance value)
FAVORITE_FOOD	Spaghetti	(instance value)
CLIMBS	Trees	(instance value)
BODY_TYPE	Wiry	(instance value)
NUM_LEGS	1	(exception)

18

Common Sense

X is at location A

If you move X from A to B, then X will be at B

X is now at location B

If X is a block and Y is a block on top of it, then when you move X, both X and Y move

UNLESS you are clumsy

UNLESS Y and X are well affixed to each other.

19

One-Hot Encoding

- Suppose the following vocabulary: {'mat', 'the', 'bird', 'hat', 'on', 'in', 'cat', 'tree', 'dog'}
- Word to Index Mapping: {'mat': 0, 'the': 1, 'bird': 2, 'hat': 3, 'on': 4, 'in': 5, 'cat': 6, 'tree': 7, 'dog': 8}
- One-Hot encoded Matrix:

```
cat:    [0, 0, 0, 0, 0, 0, 1, 0, 0]
in:     [0, 0, 0, 0, 0, 1, 0, 0, 0]
the:    [0, 1, 0, 0, 0, 0, 0, 0, 0]
hat:    [0, 0, 0, 1, 0, 0, 0, 0, 0]
dog:    [0, 0, 0, 0, 0, 0, 0, 0, 1]
on:     [0, 0, 0, 0, 1, 0, 0, 0, 0]
the:    [0, 1, 0, 0, 0, 0, 0, 0, 0]
mat:    [1, 0, 0, 0, 0, 0, 0, 0, 0]
bird:   [0, 0, 1, 0, 0, 0, 0, 0, 0]
in:     [0, 0, 0, 0, 0, 1, 0, 0, 0]
the:    [0, 1, 0, 0, 0, 0, 0, 0, 0]
tree:   [0, 0, 0, 0, 0, 0, 0, 1, 0]
```

- It is like a ”characteristic function.”

Source: https://www.geeksforgeeks.org/nlp/word-embeddings-in-nlp/

20

Embeddings

- A NN-based approach for generating word embeddings.
- Trained on a large corpus
- Every word is assigned a vector.
- Aim: Capture the semantic relationships between words by mapping them to high-dimensional vectors.
- Words with similar meanings should have similar vector representations.
- A dimensional reduction over one-hot encodings.
- Word2Vec and GloVe: pre-trained, i.e. you can download it and use it.
- BERT etc.: dynamic, built into LLM

21

Word2Vec

- We start with either a random or one-hot vector for each token
- Use a Skip-Gram model
- The main objective of Skip-Gram is to predict context words (words surrounding a target word) given a target word.

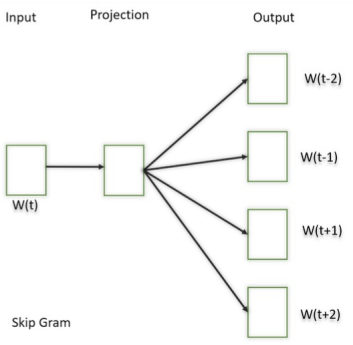


Image source: https://www.geeksforgeeks.org/nlp/word-embeddings-in-nlp/

22

Word2Vec

- The input is the center word.
- The predictions are the context words.

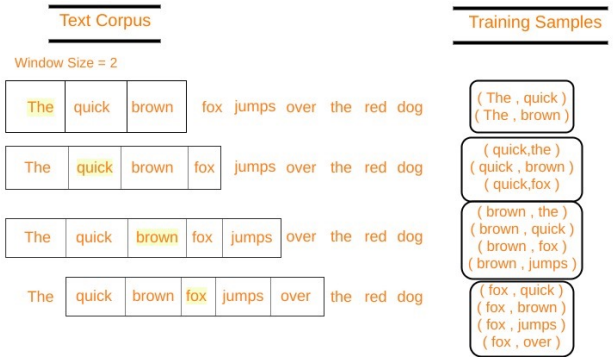


Image source: https://www.geeksforgeeks.org/python/implement-your-own-word2vecskip-gram-model-in-python/

23

GLoVe

GloVe is trained on global word co-occurrence statistics.

Word2Vec only captures the local context of words, i.e. its close neighbors

GloVe creates a large matrix that can capture the co-occurrence of words within the overall corpus.

24

GLoVe

Vectors for each word is assigned randomly.

Take two pairs of vectors and see closeness in space.

If they occur together more often or have a higher value in the cooccurrence matrix and are far apart in space then they are brought close to each other.

If they are close to each other but are rarely or not frequently used together then they are moved further apart in space.

25

GLoVe

Our large data set:

- I love programming.
- Programming is fun.
- I love coding.

Tokenize the text: Split the sentences into words
["I", "love", "programming", "Programming", "is"
"fun", "I", "love", "coding"]

Normalize the text: Convert words to lowercase
ensure consistency:
["i", "love", "programming", "programming", "is"
"fun", "i", "love", "coding"]

Filter out rare words: Keep only meaningful wor
(ignore very rare or irrelevant words, like
stopwords, if needed).

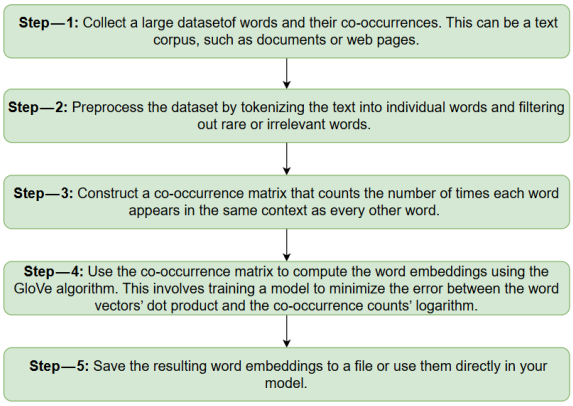


Image source: https://medium.com/@abhishekjainindore24/glove-global-vector-an-extension-to-word2vec-embedding-technique-359ce4289908

26

GLoVe

I love programming.
Programming is fun.
I love coding.

- Construct a Co-occurrence Matrix
- Define a context window (e.g., 2 words to the left and right).
- Count how often each word co-occurs with every other word within this window.

	i	love	programming	is	fun	coding
i	0					
love	2	0				
programming	1	1	0			
is	0	0	1	0		
fun	0	0	1	1	0	
coding	1	1	0	0	0	0

27

GLoVe

GloVe trains word vectors so that: dot_product(word_i, word_j) ≈ log(co-occurrence_count)

This means words that co-occur frequently will have similar embeddings

The process is repeated for all pairs in the co-occurrence matrix.

The model optimizes the word vectors by minimizing the difference between the dot product of word vectors and the expected co-occurrence probabilities.

28

GLoVe

Examples from our matrix:

- 'love' and 'i' co-occur 2.00 times
  → GloVe wants: embedding('love') · embedding('i') ≈ log(2.00) ≈ 0.69
- 'love' and 'programming' co-occur 1.00 times
  → GloVe wants: embedding('love') · embedding('programming') ≈ log(1.00) = 0
- 'programming' and 'coding' co-occur 0 times (different sentences)
  → But they'll still be similar because they share context! → Both co-occur with 'love' and 'i'

29

GLoVe

=== Learned Word Vectors (2D) ===

i : [0.0291, 0.6395]

love : [-0.0491, 0.6378]

programming : [-0.0087, -0.2524]

is : [0.0337, 0.1650]

fun : [-0.2818, 0.0130]

coding : [-0.0026, -0.4990]

=== Word Similarities (Cosine) ===

programming ↔ coding : 0.9996

love ↔ i : 0.9925

programming ↔ love : -0.9938

coding ↔ love : -0.9966

is ↔ fun : -0.1544

30

BERT

A transformer-based model that learns contextualized embeddings for words.

It considers the entire context of a word by considering both left and right contexts, resulting in embeddings that capture rich contextual information.

The context window can be quite large: 512 tokens, about a page.

Processes text from both directions i.e. left to right and right to left.

BERT can effectively analyze the context of each word by considering all the words in the sentence.

More about BERT later, once we covered transformers.

31

GLoVe vs. BERT

GloVe: Produces *static* embeddings

- each word gets the same vector regardless of context.
- "Bank" has the same representation whether it means a financial institution or a river bank.
- Uses matrix factorization on word co-occurrence statistics from a corpus.
- It learns by analyzing how often words appear together globally.
- Typically, 50-300 dimensions, lightweight and fast

BERT: Generates *contextual* **embeddings**

- the same word gets different vectors based on surrounding words.
- "Bank" will have different representations in "savings bank" vs "river bank."
- Uses transformer architecture with bidirectional self-attention.
- 768 dimensions (base) or 1024 (large), much more computationally intensive

32