Using Multi-Sense Embeddings for Named Entity Disambiguation

Bachelor Thesis
Felix Jablonski
26.04.2019

Problem statement

- Named Entity Disambiguation (NED)
- Given a text containing mentions of Wikidata entities:
  - Disambiguate each mention to the correct named entity
  - E.g., disambiguate between the different meanings of „Apple“
- Mentions are already provided by a NER-tagger
- Evaluate the performance against a reference and two baselines

Motivation

- Evaluate a novel way of linking mentions to knowledgebase entities
- Tasks that benefit from NED:
  - Enriching text with details about contained entities
  - Extract features from entities in text for search indexing

Overview

- Main idea:
  - Learn representations for words
  - Extract the meaning of a word given its context
    - multi-sense embeddings
  - Learn representations for each entity in the knowledge base
  - Compute a representation for each mention in the text
  - For each mention representation select the best matching entity

Prelimiaries

Word Embeddings

- How to encode words?
- Vector representations for words:
  - Dense
  - High-dimensional (100-500, most often 300)
  - Comparable
  - Encode information

Word2Vec (Mikolov et al. 2013)

- *„You shall know a word by the company it keeps”* (Firth, J. R. 1957:11)
- For each word learn to predict its context with a neuronal network

„What words are used alongside with ‘Apple’?”

- Input: Unlabeled text corpus
- Output: Embeddings for each word in the corpus

- **CBOW:** Predict the central word given a context
- **Skip-Gram:** Predict the context given a center word

Word Embedding Similarity

- Compare word embeddings using cosine similarity:

$$sim(x, y) = \frac{x \; * y}{||x|| * ||y||}$$

- $sim(x, y) \in [-1, 1]$
- 0 = no relationship, 1 = same meaning, -1 = opposite meaning
- Robust against different lengths and high dimensionality

Word a	Word b	sim(a, b)
Coffee	Tea	0.68
Toyota	Freiburg im Breisgau	0.12

Multi-Sense Embeddings

- Word embeddings have a single representation per word
- „Apple“ is learned with a fruit and a computer context

WORD	$p(z)$	NEAREST NEIGHBOURS
python	0.33	monty, spamalot, cantsin
	0.42	perl, php, java, c++
	0.25	molurus, pythons
apple	0.34	almond, cherry, plum
	0.66	macintosh, iifx, iigs

Source: [3]

- Solved by learning an embedding per word meaning
- Number of meanings can be fixed or dynamic

- E.g.: **AdaGram** (Bartunov et al. 2015) or **SenseGram** (Pelvina et al. 2016)

SenseGram Sense Disambiguation

- NEDard models use the **word sense disambiguation** performance
- Find the correct sense of a word given its context

- 1. For a word w get all known senses $S = \{s_0, \ldots, s_a\}$
- 2. Calculate average of context word embeddings $\bar{c}$
- 3. Select the sense most similar to $\bar{c}$

$$s^* = argmax_i \frac{\bar{c} * s_i}{||\bar{c}|| * ||s_i||}$$

NEDard models

Overview

Preprocessing → Training → Indexing → Querying

Preprocessing: Example

Microsoft‘s Windows10 is a widely-used **operating system** in **Malagati**

[**Microsoft**, **Windows**, is, widely, used, **operating**, **system**, in]

Mention tokens	**Start (inclusive)**	**Stop (exclusive)**
‚Microsoft‘ + ‚Windows‘	0	2
‚operating‘ + ‚system‘	5	7

Multi-Word Mentions

- The embedding model only knows single words
- Ideas for dealing with multi-word mentions and entities:
  - Use every sense embedding on its own: **NEDard**
  - Calculate the weighted average over the sense embeddings: **NEDardv2**

Training: Learning Weights (NEDardv2)

- Learn *tf-idf* weights for each word in the entity labels
- **Goal**: Give more weight to words that are more descriptive

Training: Sense disambiguation

ID	rdfs:label	schema:description
Q1000006	Florian Eichinger	German film producer and screenwriter

- Perform word sense disambiguation on **„Florian“** and **„Eichinger“**
  - **German**
  - **Film industry**
  - **Producer**

Token	Context window (size 5) and without stopwords	Sense embedding
Florian	[Eichinger, German, film, producer]	Florian#0
Eichinger	[Florian, German, film, producer]	Eichinger#3

Training: Index Storing

- Found sense embeddings: **Florian#0** and **Eichinger#3**
- **NEDard:**
  - Store both embeddings as distinct entity embeddings **Q1000006#0** and **Q1000006#1**
- **NEDardv2:**
  - Calculate *tf-idf* weighted average of the sense embeddings
  - Store the average embedding as entity embedding **Q1000006**

Querying: NEDard

„computer“ sense of „**Microsoft**“ and „**Windows**“ → „**Microsoft**“ **+** „**Windows**“ Q1406

Index

Mention „**Windows**“ in „computer“ context

Get the closest entity-word embedding. E.g. the „**Windows**“ in „**Microsoft Windows**“.

„computer“ mention of „**Windows**“ is resolved to „**Microsoft Windows**“ Q1406

Querying: NEDardv2

„computer“ sense of „**Microsoft**“ and „**Windows**“ → „**Microsoft Windows**“ Q1406

Index

+

Mention „**Windows**“ in „computer“ context

Get the closest entity embedding. E.g. „**Microsoft Windows**“ Q1406

„computer“ mention of „**Windows**“ is resolved to „**Microsoft Windows**“ Q1406

Advantages

- Unsupervised learning
- Pretrained embedding model can be used
- Adding new entities on the fly

Disadvantages

- Fuzziness
- Depends on quality of embeddings

Baselines and reference

Candidates

- Restrict the considered entities to candidates
- The candidate set is taken from Niklas Baumert‘s bachelor thesis[6]
  - For each link text in Wikipedia remember the articles it is linked to
  - Relevance: The fraction of times this article is linked to

Mention (lnrm)	**Wiki link**	**Relevance**	**Wikidata id**
lnrm__freiburg	Freiburg_im_Breisgau	0.99	Q2833
lnrm__freiburg	Canton_of_Fribourg	0.01	Q12640

Baselines

- Oracle performance
- Baseline 1 – Random candidate
- Baseline 2 – Candidate with highest relevance
- Reference – DBPedia spotlight
  - Uses entity-context matrix based on Wikipedia

Evaluation

Evaluation Sets

- 3 evaluation sets with different characteristics
- Extracted from Wikipedia inter-article links:
  - Link text as mention
  - Wikidata id of target article as entity
  - Wikipedia articles with no Wikidata entity are ignored

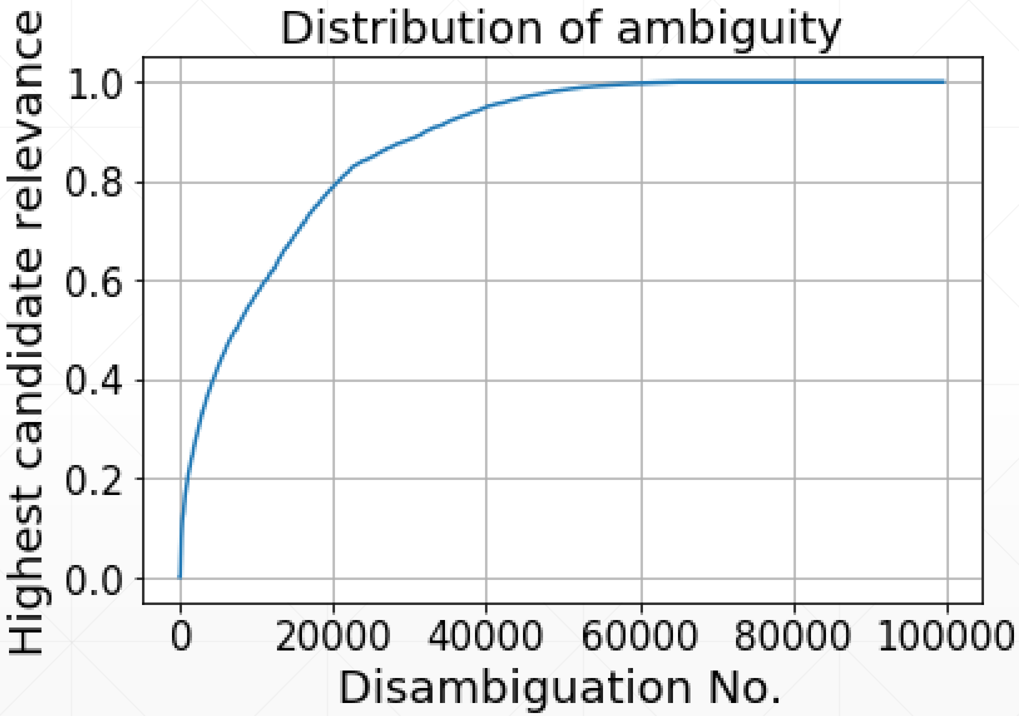
Article	Article text	Position	Mention	Entity
Windows	„Windows is a operating system…“	[12:28]	operating system	Q9135
Windows	„Windows is a operating system…“	[0:7]	Windows	Q1406

Evaluation Set 1: Wiki-2k

- NEDard is trained on first 5 million Wikidata entities
  - 3,661,533 could be learned by NEDard
- First **2,000 Wikipedia articles** including **100,320 disambiguations**

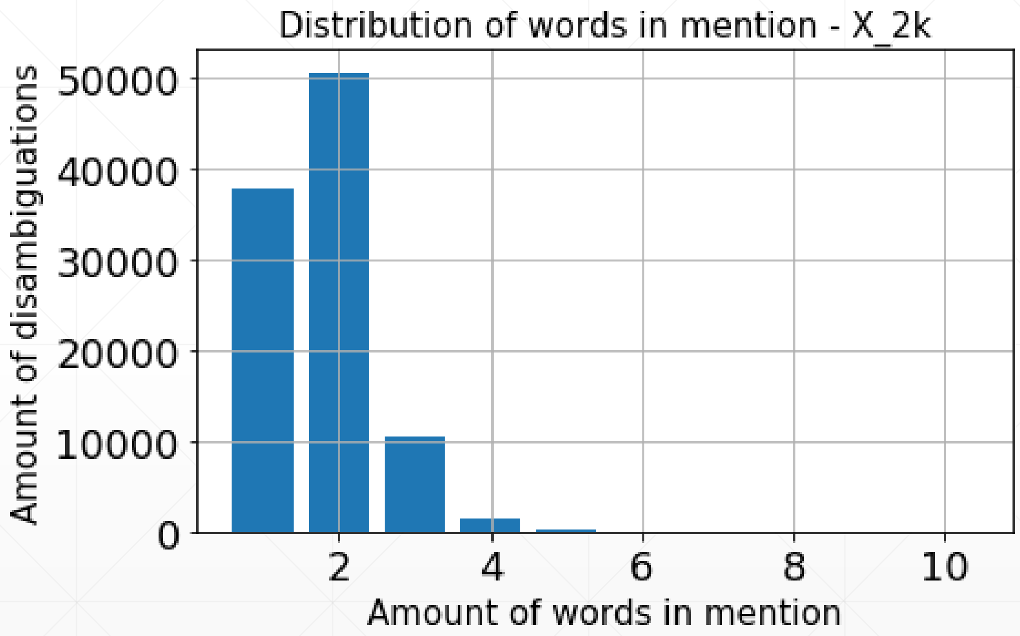
Evaluation Set 2: Wiki-ambig

- Subset of Wiki-2k
- Disambiguations with multiple strong candidates
- Highest candidate relevance < 0.8
- More difficult evaluation set
- **20,626 disambiguations**



Evaluation Set 3: Wiki-oneword

- Only single-word mentions
- Subset of Wiki-2k
- **37,732 disambiguations**



Evaluation Run

- Input: Text and mentions of n articles
- The correct entities are not known to the model

$$Accuracy = \frac{\sum_i^n correct(a_i)}{\sum_i^n total(a_i)}$$

- $correct(a_i) = correct\ disambiguations\ in\ article\ a_i$
- $total(a_i) = total\ disambiguations\ in\ article\ a_i$

Evaluation Results

Accuracy	Evaluation set #disambig.		
Model	X_{2k} 100,320	X_{ambig} 20,626	$X_{oneword}$ 37,732
Oracle	0.99	0.99	0.99
Random entity	0.00	0.00	0.00
Baseline1 (Random candidate)	0.45	0.08	0.15
Baseline2 (Highest relevance)	**0.89**	**0.61**	**0.82**
Reference (DBpedia spotlight)	0.87	0.61	0.78
NEDardv1 (case + candidates)	0.72	0.40	0.51
NEDardv1 (nocase + candidates)	0.72	0.40	0.52
NEDardv1 (case + all)	0.29	0.26	0.55
NEDardv2 (case + candidates)	0.74	0.40	0.57
NEDardv2 (case + all)	0.31	0.14	0.24

Error analysis

Error Analysis: NEDard

Apple Inc. is an American multination technology company headquartered in **Cupertino, California**.

Mention	Candidates	Cosine distances
Apple Inc.	Q312#0, Q421253#0, Q421253#1 Apple, **Apple** Store, Apple **Store**	0.0, 0.0, 0.88 to „**Apple**“ 0.58, 0.58, 0.81 to „**Inc.**“
Cupertino	Cupertino, 3x Santa Clara County, Copertino	0.0, 0.4, 0.75, 0.43, 0.77
California	612 candidates…	Multiple 0.0

Wikidata Id	Cosine distance	Entity label	Correct?
Q312	0.0	Apple	Yes
Q189471	0.0	Cupertino	Yes
Q3650742	0.0	California Golden Bears football	**No**

Error Analysis: NEDardv2

Apple Inc. is an American multination technology company headquartered in **Cupertino, California**.

Mention	Candidates	Cosine distances
Apple Inc.	Q312, Q421253 Apple, Apple Store	0.21, 0.26 to „**Apple Inc.**“
Cupertino	Cupertino, Santa Clara County, Copertino	0.0, 0.5, 0.776
California	254 candidates…	Multiple 0.0

Wikidata Id	Cosine distance	Entity label	Correct?
Q312	0.21	Apple	Yes
Q189471	0.0	Cupertino	Yes
Q1134176	0.0	California	**No**

Demonstration

Sources

1. https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/
2. https://www.slideshare.net/ChristopherMoody3/word2vec-lda-and-introducing-a-new-hybrid-algorithm-lda2vec-57135994
3. Bartunov et al. “Breaking Sticks and Ambiguities with Adaptive Skip-gram”. In: (2015). URL: https://arxiv.org/abs/1502.07257
4. Pelevina et al. “Making Sense of Word Embeddings”. In: (2016). URL: http: //aclweb.org/anthology/W/W16/W16-1620.pdf
5. Thomas Mikolov. “Efficient Estimation of Word Representations in Vector Space”. In: (2013). URL: https://arxiv.org/abs/1301.3781
6. Niklas Baumert. “Web-scalable Named-entity Recognition and Linking with a Wikipedia-backed Knowledge Base”. In: (2018). URL: http://ad-publications.informatik.uni-freiburg.de/theses/Bachelor_Niklas_Baumert_2018.pdf

Appendix

NED Example

Windows is a widely used **operating system**, but **Jobs** and **Linus** wanted to build their own.

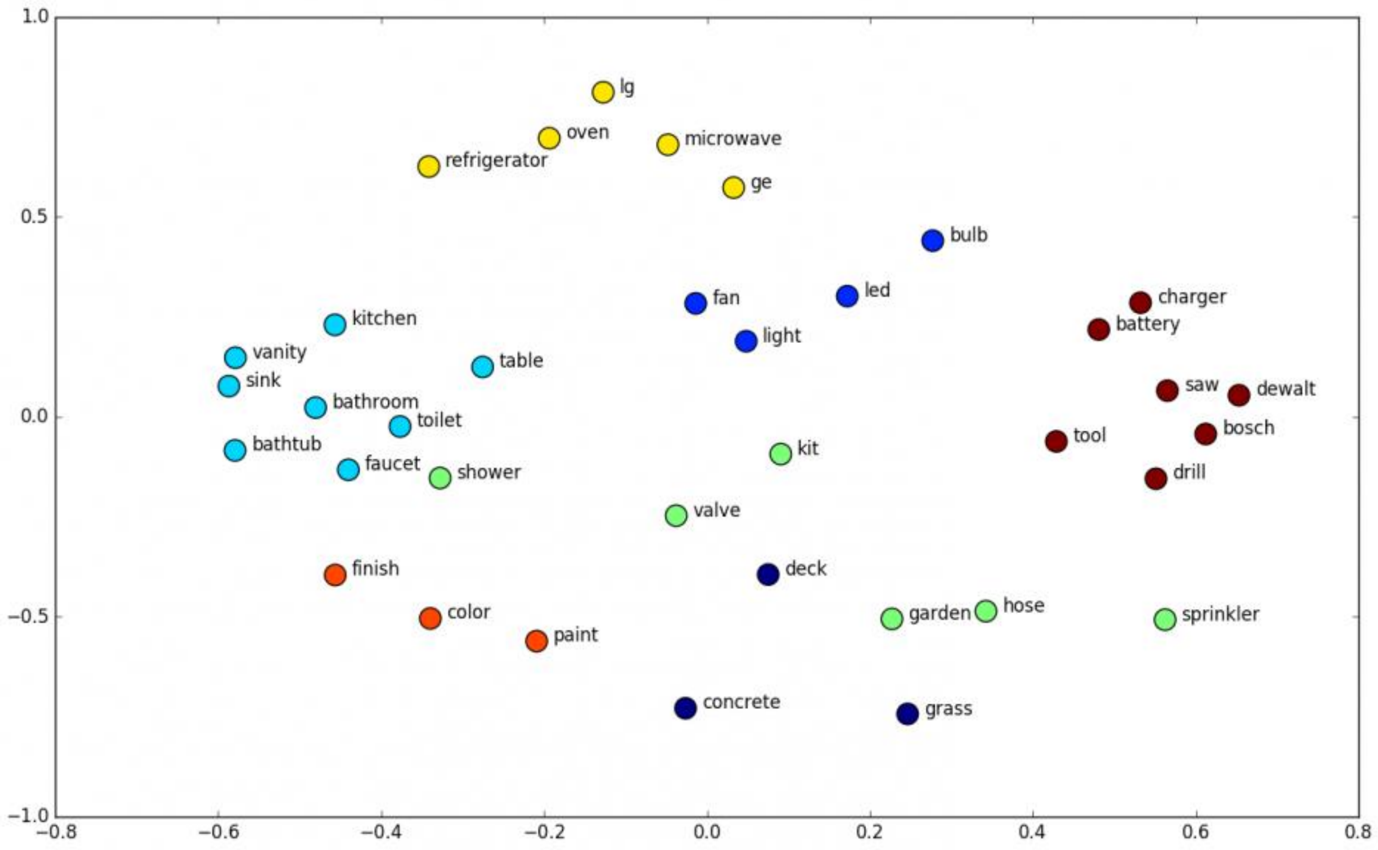
Mention	Named entity	Wikidata ID
Windows	Microsoft Windows	Q1406
operating system	operating system	Q9135
Jobs	Steve Jobs	Q19837
Linus	Linus Torvalds	Q34253

Word Embedding Properties

- Semantic modelling
  - Related words are more similar in embedding space
- Semantic reasoning
  - Embeddings support arithmetic operations

$$vec(king) - vec(man) + vec(woman) \approx vec(queen)$$

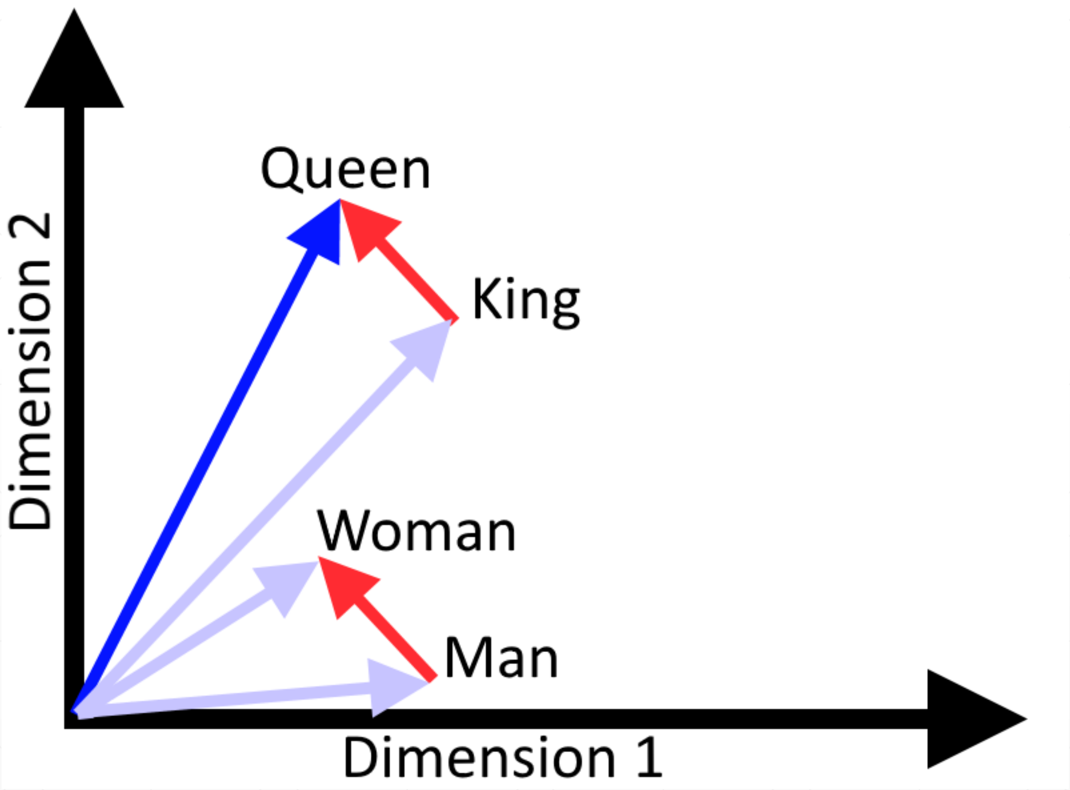
Semantic Modelling



Source: [1]

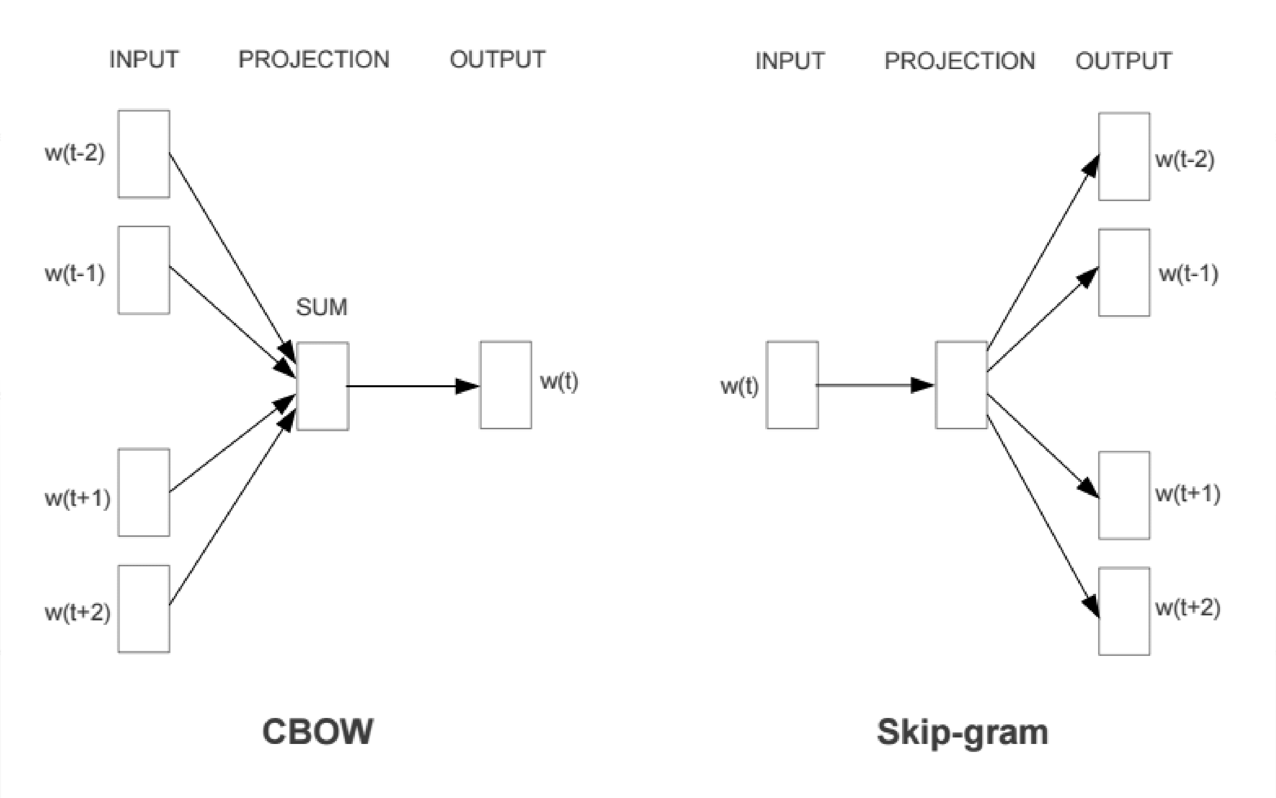
Semantic Reasoning

$$vec(king) - vec(man) + vec(woman) \approx vec(queen)$$



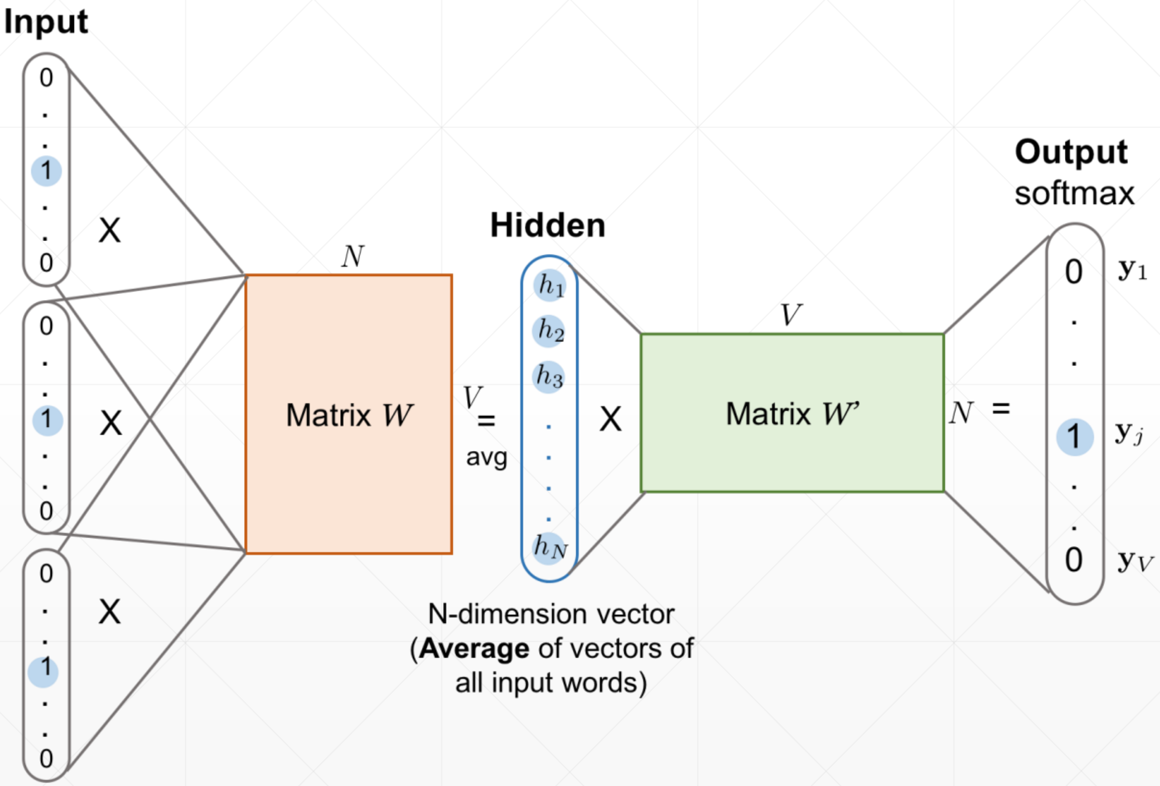
Adapted from: [2]

Word2Vec Architecture



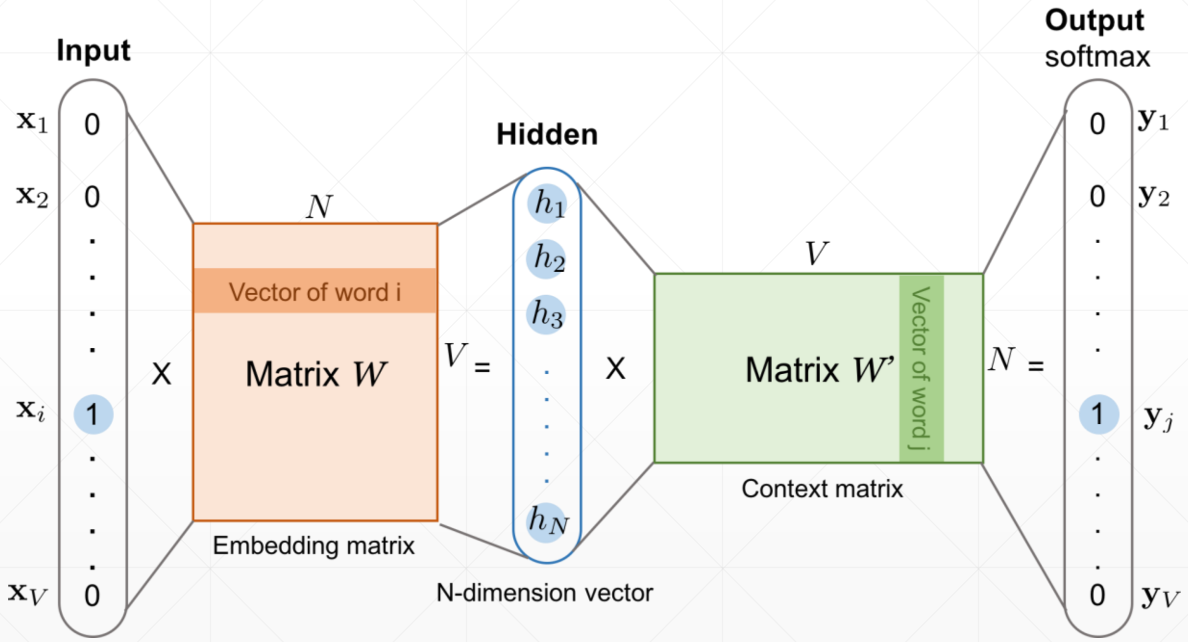
Source: [5]

Word2Vec Architecture: CBOW



Source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Word2Vec Architecture: Skip-Gram



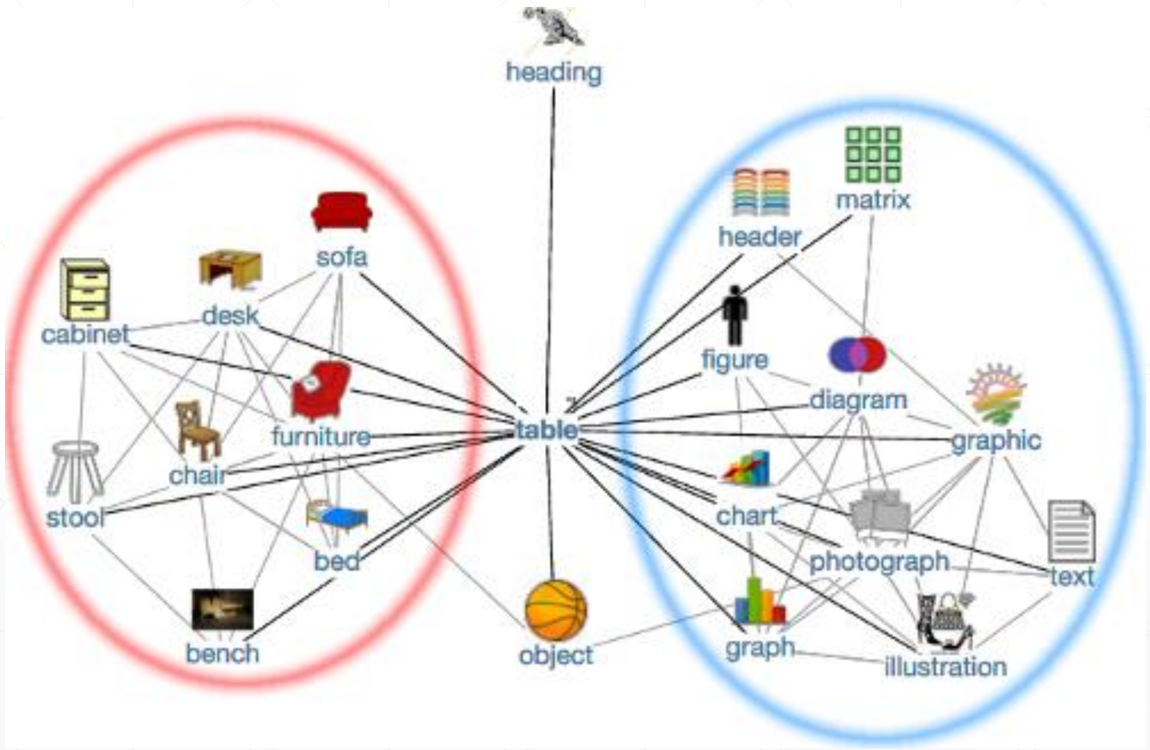
Source: https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html

Chinese Whispers (Biemann 2006)

- Graph clustering algorithm for NLP
- Knowledge-free and unsupervised
- Complexity: Linear in number of edges
- Grouping together based on word similarity
  - Similarity is the co-occurrence significance of two words
- Every node (word) starts with own label
- Iterative updating of group based on evidence from connected nodes
- Termination: Max iterations or no more changes

SenseGram Training

- 1. Take an existing word embedding
- 2. Get the 200 most similar embeddings
- 3. Build an ego-network graph out of the similar words
- 4. Connect words if they are similar themself
- 5. Perform graph clustering with Chinese Whispers



Source: [4]

SenseGram Training

- For each cluster compute the similarity-weighted average
- $\boldsymbol{\gamma}$: $\boldsymbol{V} \rightarrow \mathbb{R}$ mapping each embedding to its similarity to the ego word
- $C_i = \{vec_1, \dots, vec_a\}$ denoting each cluster

$$s_i = \frac{\sum_{k=1}^{a} \gamma(vec_k) vec_k}{\sum_{k=1}^{a} \gamma(vec_k)}$$

- These are the **sense embeddings** of the ego word

Word-Context Matrix

- Co-occurence matrix for context window

	Pizza	Sauerkraut	Frankfurt	Apple
Italy	4	0	0	1
Germany	0	4	5	2
USA	2	1	2	3
Delicious	4	3	0	4

Pointwise Mutual Information

- Measure of association used in information theory.
  - How much more likely we get a pair than if it were at random?

$$PMI(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- They occur more together than at random = high PMI
- *Tf-idf* is used to normalize term document matrices
- PMI is used to normalize word-context matrices
- Approximation by vector multiplication: $PMI(a,b) = vec(a) \cdot vec(b)$

Word Similarity

$$PMI(w, a) = PMI(w, b)$$
$$vec(w) * vec(a) = vec(w) * vec(b)$$
$$vec(w) * \left(vec(a) - vec(b)\right) = 0$$

Needs to work for all words w

$$vec(a) = vec(b)$$

Source: https://p.migdal.pl/2017/01/06/king-man-woman-queen-why.html

Word Differences

Semantic reasoning can be describes as vector operations:

$$vec(she) - vec(he)$$
$$vec(w) * \big(vec(a) - vec(b)\big) = \log[P(w|a)] - \log[P(w|b)]$$

This is the relative occurrence of a word within different contexts.

Source: https://p.migdal.pl/2017/01/06/king-man-woman-queen-why.html

Training: Input

- Get a context for each entity inside Wikidata:

ID	rdfs:label	schema:description
Q1000006	Florian Eichinger	German film producer and screenwriter
Q1000007	IFA S4000	truck
Q1000008	Neuvireuil	commune in Pas-de-Calais, France
…		

- Take one entity

TF-IDF

- Term frequency (tf):
  - count of word in entity label
- Inverse document frequency (idf):
  - Importance of word over all entity labels

$$tfidf = \#word\_in\_label \ * \ \log \frac{|entities|}{|entities\ with\ word\ in\ label|}$$

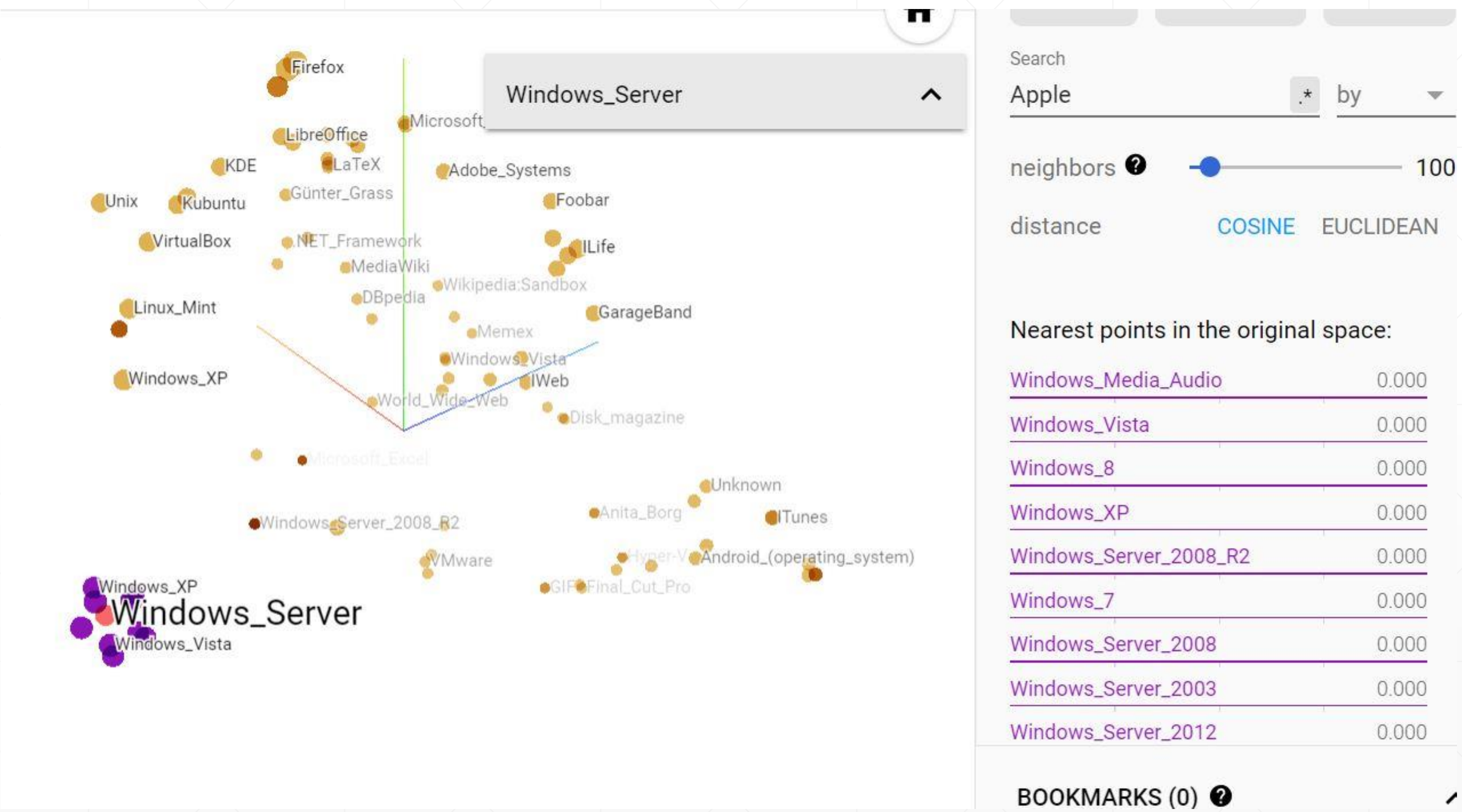
Potential of combination

Linear combination of baseline2 and NEDardv2

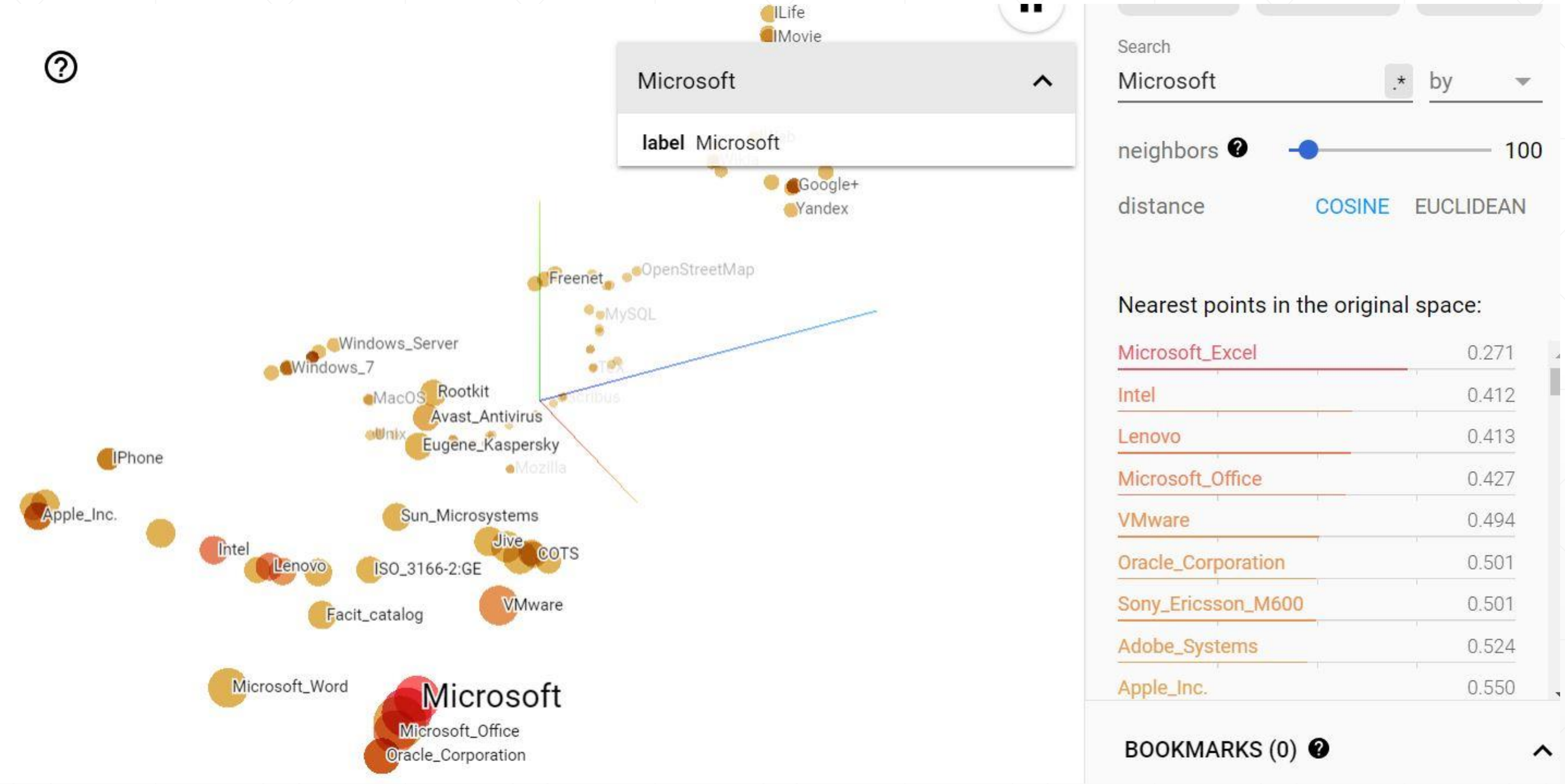
- Split evalution set into 70% train and 30% test
- Learn to combine NEDardv2 score and relevance
- Run logistic regression to predict likelihood for each candidate

Evaluation set	Baseline2	NEDardv2	Combined
X_{ambig}	0.616	0.409	0.650
$X_{oneword}$	0.822	0.583	0.839
X_{2k}	0.904	0.753	0.913

Visualization: NEDard



Visualization: NEDardv2



Error Analysis: General

- Same word with different meanings
  - „Apple“ as company vs. as fruit
- Same real-world object in different contexts
  - „Berlin“ and „Alt-Berlin“. Current vs. Historic.
- Same mention with similar context but different entity
  - „French revolution“ as TV series or real event.
- Too specific Wikidata entity
  - „Jews“ -> History of the Jews in Kazakhstan (Q2067366)
- Noise in candidate set and Wikidata entities