Using Multi-Sense Embeddings for Named Entity Disambiguation
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:

\[
sim(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}
\]

- \( \sim(x, y) \in [-1, 1] \)

- 0 = no relationship, 1 = same meaning, -1 = opposite meaning

- Robust against different lengths and high dimensionality

<table>
<thead>
<tr>
<th>Word a</th>
<th>Word b</th>
<th>( \text{sim}(a, b) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coffee</td>
<td>Tea</td>
<td>0.68</td>
</tr>
<tr>
<td>Toyota</td>
<td>Freiburg im Breisgau</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Multi-Sense Embeddings

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

<table>
<thead>
<tr>
<th>WORD</th>
<th>p(z)</th>
<th>NEAREST NEIGHBOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>0.33</td>
<td>monty, spamalot, cantsin</td>
</tr>
<tr>
<td></td>
<td>0.42</td>
<td>perl, php, java, c++</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>molurus, pythons</td>
</tr>
<tr>
<td>apple</td>
<td>0.34</td>
<td>almond, cherry, plum</td>
</tr>
<tr>
<td></td>
<td>0.66</td>
<td>macintosh, iifx, iigs</td>
</tr>
</tbody>
</table>

Source: [3]
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 $S = \{s_0, ..., s_a\}$
2. Calculate average of context word embeddings $\bar{c}$
3. Select the sense most similar to $\bar{c}$

\[
s^* = \text{argmax}_i \frac{\bar{c} \cdot s_i}{||\bar{c}|| \cdot ||s_i||}
\]
NEDard models
Overview

Preprocessing ➔ Training ➔ Indexing ➔ Querying
Microsoft’s Windows10 is a widely-used operating system in Malagati

[Microsoft, Windows, is, widely, used, operating, system, in]
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 \textit{tf-idf} weights for each word in the entity labels
- \textbf{Goal}: Give more weight to words that are more descriptive
Training: Sense disambiguation

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

<table>
<thead>
<tr>
<th>Token</th>
<th>Context window (size 5) and without stopwords</th>
<th>Sense embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Florian</td>
<td>[Eichinger, German, film, producer]</td>
<td>Florian#0</td>
</tr>
<tr>
<td>Eichinger</td>
<td>[Florian, German, film, producer]</td>
<td>Eichinger#3</td>
</tr>
</tbody>
</table>
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

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

<table>
<thead>
<tr>
<th>Mention (Inrm)</th>
<th>Wiki link</th>
<th>Relevance</th>
<th>Wikidata id</th>
</tr>
</thead>
<tbody>
<tr>
<td>lnrm__freiburg</td>
<td>Freiburg_im_Breisgau</td>
<td>0.99</td>
<td>Q2833</td>
</tr>
<tr>
<td>lnrm__freiburg</td>
<td>Canton_of_Fribourg</td>
<td>0.01</td>
<td>Q12640</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Article</th>
<th>Article text</th>
<th>Position</th>
<th>Mention</th>
<th>Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>“Windows is a operating system…”</td>
<td>[12:28]</td>
<td>operating system</td>
<td>Q9135</td>
</tr>
<tr>
<td>Windows</td>
<td>“Windows is a operating system…”</td>
<td>[0:7]</td>
<td>Windows</td>
<td>Q1406</td>
</tr>
</tbody>
</table>
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

\[
Accuracy = \frac{\sum_{i=1}^{n} correct(a_i)}{\sum_{i=1}^{n} total(a_i)}
\]

- $correct(a_i) = \text{correct disambiguations in article } a_i$
- $total(a_i) = \text{total disambiguations in article } a_i$
## Evaluation Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Evaluation set</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$X_{2k}$</td>
<td>$X_{ambig}$</td>
<td>$X_{oneword}$</td>
</tr>
<tr>
<td></td>
<td>#disambig.</td>
<td>100,320</td>
<td>20,626</td>
<td>37,732</td>
</tr>
<tr>
<td>Oracle</td>
<td></td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Random entity</td>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Baseline1 (Random candidate)</td>
<td></td>
<td>0.45</td>
<td>0.08</td>
<td>0.15</td>
</tr>
<tr>
<td>Baseline2 (Highest relevance)</td>
<td></td>
<td>0.89</td>
<td>0.61</td>
<td>0.82</td>
</tr>
<tr>
<td>Reference (DBpedia spotlight)</td>
<td></td>
<td>0.87</td>
<td>0.61</td>
<td>0.78</td>
</tr>
<tr>
<td>NEDardv1 (case + candidates)</td>
<td></td>
<td>0.72</td>
<td>0.40</td>
<td>0.51</td>
</tr>
<tr>
<td>NEDardv1 (nocase + candidates)</td>
<td></td>
<td>0.72</td>
<td>0.40</td>
<td>0.52</td>
</tr>
<tr>
<td>NEDardv1 (case + all)</td>
<td></td>
<td>0.29</td>
<td>0.26</td>
<td>0.55</td>
</tr>
<tr>
<td>NEDardv2 (case + candidates)</td>
<td></td>
<td>0.74</td>
<td>0.40</td>
<td>0.57</td>
</tr>
<tr>
<td>NEDardv2 (case + all)</td>
<td></td>
<td>0.31</td>
<td>0.14</td>
<td>0.24</td>
</tr>
</tbody>
</table>
Error analysis
Error Analysis: NEDard

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

<table>
<thead>
<tr>
<th>Mention</th>
<th>Candidates</th>
<th>Cosine distances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Inc.</td>
<td>Q312#0, Q421253#0, Q421253#1</td>
<td>0.0, 0.0, 0.88 to &quot;Apple&quot;</td>
</tr>
<tr>
<td></td>
<td>Apple, <strong>Apple Store</strong>, Apple Store</td>
<td>0.58, 0.58, 0.81 to &quot;Inc.&quot;</td>
</tr>
<tr>
<td>Cupertino</td>
<td>Cupertino, 3x Santa Clara County, Copertino</td>
<td>0.0, 0.4, 0.75, 0.43, 0.77</td>
</tr>
<tr>
<td>California</td>
<td>612 candidates…</td>
<td>Multiple 0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wikidata Id</th>
<th>Cosine distance</th>
<th>Entity label</th>
<th>Correct?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q312</td>
<td>0.0</td>
<td>Apple</td>
<td>Yes</td>
</tr>
<tr>
<td>Q189471</td>
<td>0.0</td>
<td>Cupertino</td>
<td>Yes</td>
</tr>
<tr>
<td>Q3650742</td>
<td>0.0</td>
<td>California Golden Bears football</td>
<td>No</td>
</tr>
</tbody>
</table>
Apple Inc. is an American multinational technology company headquartered in Cupertino, California.

<table>
<thead>
<tr>
<th>Mention</th>
<th>Candidates</th>
<th>Cosine distances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Inc.</td>
<td>Q312, Q421253, Apple, Apple Store</td>
<td>0.21, 0.26 to „Apple Inc.“</td>
</tr>
<tr>
<td>Cupertino</td>
<td>Cupertino, Santa Clara County, Copertino</td>
<td>0.0, 0.5, 0.776</td>
</tr>
<tr>
<td>California</td>
<td>254 candidates…</td>
<td>Multiple 0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wikidata Id</th>
<th>Cosine distance</th>
<th>Entity label</th>
<th>Correct?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q312</td>
<td>0.21</td>
<td>Apple</td>
<td>Yes</td>
</tr>
<tr>
<td>Q189471</td>
<td>0.0</td>
<td>Cupertino</td>
<td>Yes</td>
</tr>
<tr>
<td>Q1134176</td>
<td>0.0</td>
<td>California</td>
<td>No</td>
</tr>
</tbody>
</table>
Demonstration
Sources


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

<table>
<thead>
<tr>
<th>Mention</th>
<th>Named entity</th>
<th>Wikidata ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>Microsoft Windows</td>
<td>Q1406</td>
</tr>
<tr>
<td>operating system</td>
<td>operating system</td>
<td>Q9135</td>
</tr>
<tr>
<td>Jobs</td>
<td>Steve Jobs</td>
<td>Q19837</td>
</tr>
<tr>
<td>Linus</td>
<td>Linus Torvalds</td>
<td>Q34253</td>
</tr>
</tbody>
</table>
Word Embedding Properties

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

\[ vec(\text{king}) - vec(\text{man}) + vec(\text{woman}) \approx vec(\text{queen}) \]
Semantic Modelling

Source: [1]
Semantic Reasoning

\[ \vec{\text{king}} - \vec{\text{man}} + \vec{\text{woman}} \approx \vec{\text{queen}} \]
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
- $\gamma: V \rightarrow \mathbb{R}$ mapping each embedding to its similarity to the ego word
- $C_i = \{vec_1, \ldots, vec_a\}$ 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-occurrence matrix for context window

<table>
<thead>
<tr>
<th></th>
<th>Pizza</th>
<th>Sauerkraut</th>
<th>Frankfurt</th>
<th>Apple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Italy</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Germany</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>USA</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Delicious</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
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

\[ PMI(w, a) = PMI(w, b) \]
\[ vec(w) * vec(a) = vec(w) * vec(b) \]
\[ vec(w) * (vec(a) - vec(b)) = 0 \]

Needs to work for all words \( w \)

\[ vec(a) = vec(b) \]

Word Differences

Semantic reasoning can be describes as vector operations:

\[
\text{vec}(\text{she}) - \text{vec}(\text{he}) = \log[P(\text{w}|a)] - \log[P(\text{w}|b)]
\]

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

- Get a context for each entity inside Wikidata:

<table>
<thead>
<tr>
<th>ID</th>
<th>rdfs:label</th>
<th>schema:description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1000006</td>
<td>Florian Eichinger</td>
<td>German film producer and screenwriter</td>
</tr>
<tr>
<td>Q1000007</td>
<td>IFA S4000</td>
<td>truck</td>
</tr>
<tr>
<td>Q1000008</td>
<td>Neuvireuil</td>
<td>commune in Pas-de-Calais, France</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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

\[
\text{tfidf} = \text{#word_in_label} \times \log \frac{|\text{entities}|}{|\text{entities with word in label}|}
\]
Potential of combination
Linear combination of baseline2 and NEDardv2

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

<table>
<thead>
<tr>
<th>Evaluation set</th>
<th>Baseline2</th>
<th>NEDardv2</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{ambig}$</td>
<td>0.616</td>
<td>0.409</td>
<td>0.650</td>
</tr>
<tr>
<td>$X_{onestore}$</td>
<td>0.822</td>
<td>0.583</td>
<td>0.839</td>
</tr>
<tr>
<td>$X_{2k}$</td>
<td>0.904</td>
<td>0.753</td>
<td>0.913</td>
</tr>
</tbody>
</table>
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

- 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