Big Data and Automated Content Analysis Part I+II

Week 13 – Wednesday
»Word Embeddings«

Damian Trilling

d.c.trilling@uva.nl
@damian0604
www.damiantrilling.net

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

8 May 2019

Today

1. From word counts to word vectors
2. Training word embeddings
3. Using word embeddings to improve models
4. (Ab-)using word embeddings to detect biases
5. AEM: An application from our own research
6. Next meetings

From word counts to word vectors

Our BOW approach until now

Representing a document by word frequency counts

Result of preprocessing and vectorizing:

0. `He took the dog for a walk to the dog playground`

$\Rightarrow$ `took dog walk dog playground`

$\Rightarrow$ `'took':1, 'dog': 2, walk: 1, playground: 1`

Consider these other sentences

Our BOW approach until now

Representing a document by word frequency counts

Result of preprocessing and vectorizing:

0. `He took the dog for a walk to the dog playground`

⇒ `took dog walk dog playground`

⇒ `'took':1, 'dog': 2, walk: 1, playground: 1`

Consider these other sentences

1. He took the doberman for a walk to the dog playground

Our BOW approach until now

Representing a document by word frequency counts

Result of preprocessing and vectorizing:

0. `He took the dog for a walk to the dog playground`

⇒ `took dog walk dog playground`

⇒ `'took':1, 'dog': 2, walk: 1, playground: 1`

Consider these other sentences

1. He took the doberman for a walk to the dog playground
2. He took the cat for a walk to the dog playground

Our BOW approach until now

Representing a document by word frequency counts

Result of preprocessing and vectorizing:

0. `He took the dog for a walk to the dog playground`

⇒ `took dog walk dog playground`

⇒ `'took':1, 'dog': 2, walk: 1, playground: 1`

Consider these other sentences

1. He took the doberman for a walk to the dog playground
2. He took the cat for a walk to the dog playground
3. He killed the dog on his walk to the dog playground

Our BOW approach until now

Representing a document by word frequency counts

Result of preprocessing and vectorizing:

0. `He took the dog for a walk to the dog playground`

⇒ `took dog walk dog playground`

⇒ `'took':1, 'dog': 2, walk: 1, playground: 1`

Consider these other sentences

1. He took the doberman for a walk to the dog playground
2. He took the cat for a walk to the dog playground
3. He killed the dog on his walk to the dog playground

The vectorized representations of these sentences have a "distance" (dissimilarity) of 1 each, but arguably, sentences 0 and 1 should be "closer" than others

Our BOW approach until now

- Our vectorizers gave a random ID to each word
- What if we instead would represent each word by another vector representing its meaning?
- For, instance, ‘doberman’ and ‘dog’ should be represented by vectors that are close to each other in space, while ‘kill’ and ‘walk’ should be far from each other.

Our BOW approach until now

- Our vectorizers gave a random ID to each word
- What if we instead would represent each word by another vector representing its meaning?
- For, instance, ‘doberman’ and ‘dog’ should be represented by vectors that are close to each other in space, while ‘kill’ and ‘walk’ should be far from each other.

⇒ That’s the idea behind word embeddings!

Our BOW approach until now

- Our vectorizers gave a random ID to each word
- What if we instead would represent each word by another vector representing its meaning?
- For, instance, ‘doberman’ and ‘dog’ should be represented by vectors that are close to each other in space, while ‘kill’ and ‘walk’ should be far from each other.

$\Rightarrow$ That’s the idea behind word embeddings!
Or, more broadly: Can computers understand meanings, semantic relationships, different types of contexts?

Training word embeddings

Continous Bag of Words (CBOW) vs skipgram

Example sentence: "the quick brown fox jumped over the lazy dog"

CBOW: Predict a word given its context

Dataset:

`([the, brown], quick), ([quick, fox], brown),`
`([brown, jumped], fox), ...`

Continous Bag of Words (CBOW) vs skipgram

Example sentence: “the quick brown fox jumped over the lazy dog”

CBOW: Predict a word given its context

Dataset:
`([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...`

skipgram: Predict the context given the word

`(quick, the), (quick, brown), (brown, quick), (brown, fox), ...`

Example taken from here: `https://medium.com/explore-artificial-intelligence/word2vec-a-baby-step-in-deep-learning-but-a-giant-leap-towards-natural-language-processing-40fe4e8602ba`

Continous Bag of Words (CBOW) vs skipgram

- CBOW is faster
- skipgram works better for infrequent words
- Both are often used
- Usually, we use larger window sizes (e.g, 5)
- We need to specify the number of dimensions (typically 100–300)

Continous Bag of Words (CBOW) vs skipgram

- CBOW is faster
- skipgram works better for infrequent words
- Both are often used
- Usually, we use larger window sizes (e.g, 5)
- We need to specify the number of dimensions (typically 100–300)

*In any event, as a result of the prediction task, we end up with a {100|200|300}-dimensional vector representation of each word.**

* If that makes you think of PCA/SVD, that's not completely crazy, see Levy, O., Goldberg, Y., & Dagan, I. (2018). Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, *3*, 211–225. doi:10.1162/tacl_a_00134

"...a word is characterized by the company it keeps..." (Firth, 1957)

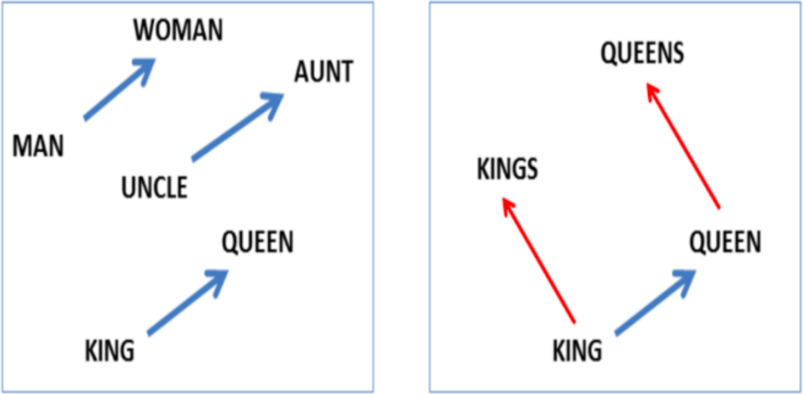
Word embeddings . . .

- help capturing the meaning of text
- are low-dimensional vector representations that capture semantic meaning
- are state-of-the-art in NLP...

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. Studies in linguistic analysis.

You can literally calculate with words!

And answer questions such as “Man is to woman as king is to _____?”



semantic relationships vs. syntactic relationships

Using word embeddings to improve models

In supervised machine learning

- Modify CountVectorizer or TfIdfVector such that for each term, you do not only count how often it occurs, but also multiply with its embedding vector

In supervised machine learning

- Modify CountVectorizer or TfIdfVector such that for each term, you do not only count how often it occurs, but also multiply with its embedding vector
- Often, pre-trained embeddings (e.g., trained on the whole wikipedia) are used

In supervised machine learning

- Modify CountVectorizer or TfIdfVector such that for each term, you do not only count how often it occurs, but also multiply with its embedding vector
- Often, pre-trained embeddings (e.g., trained on the whole wikipedia) are used
- Thus, our supervised model will be able to deal with synonyms and related words!

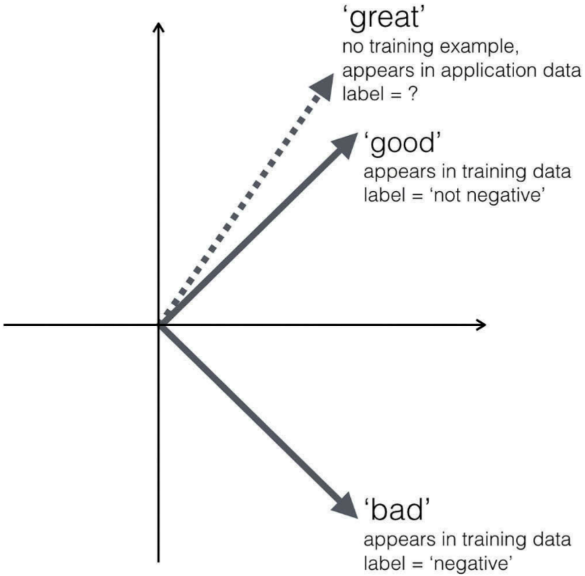
In supervised machine learning

- Modify CountVectorizer or TfIdfVector such that for each term, you do not only count how often it occurs, but also multiply with its embedding vector
- Often, pre-trained embeddings (e.g., trained on the whole wikipedia) are used
- Thus, our supervised model will be able to deal with synonyms and related words!

In supervised machine learning

- Modify CountVectorizer or TfIdfVector such that for each term, you do not only count how often it occurs, but also multiply with its embedding vector
- Often, pre-trained embeddings (e.g., trained on the whole wikipedia) are used
- Thus, our supervised model will be able to deal with synonyms and related words!

Let's look at an example for using supervised sentiment analysis (i.e., what we did with IMDB-data before).



Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, Š., & Sedlmair, M. (2018). More than Bags of Words: Sentiment Analysis with Word Embeddings. *Communication Methods and Measures, 12*(2–3), 140–157. doi:10.1080/19312458.2018.1455817

It’s not always black/white. . .

Sometimes, BOW may be just fine (for very negative sentences, it doesn’t matter). But especially in less clear cases (’slightly negative’), embeddings increased performance.

Table 1. Precision, recall, and F1 score for the bag of words approach.

	Actual	Predicted	Precision	Recall	F1 Score
not/slightly negative	524.3	205.6	0.33	0.83	0.47
negative	805.7	1188.7	0.71	0.48	0.57
very negative	730	665.7	0.53	0.58	0.56

Table 2. Precision, recall, and F1 score for the Word Embeddings approach.

	Actual	Predicted	Precision	Recall	F1 Score
not/slightly negative	522.4	575	0.65	0.59	0.61
negative	799.2	771.6	0.52	0.53	0.53
very negative	739.4	714.4	0.55	0.57	0.56

Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, Š., & Sedlmair, M. (2018). More than Bags of Words: Sentiment Analysis with Word Embeddings. *Communication Methods and Measures, 12*(2–3), 140–157. doi:10.1080/19312458.2018.1455817

In document similarity calculation

Use cases

- plagiarism detection
- Are press releases/news agency copy/...taken over?
- Event detection

In document similarity calculation

Use cases

- plagiarism detection
- Are press releases/news agency copy/. . . taken over?
- Event detection

Traditional measures

- Levenshtein distance (How many characters|words do I need to change to transform string A into string B?)
- Cosine similarity ("correlation" between the BOW-representations of string A and string B)

BUT: This only works for literal overlap. What if the writer chooses synonyms?

BUT: This only works for literal overlap. What if the writer chooses synonyms?

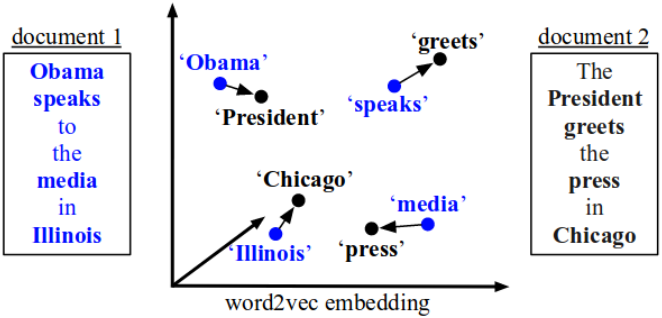


Figure 1. An illustration of the *word mover's distance*. All non-stop words (***bold***) of both documents are embedded into a *word2vec* space. The distance between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2. (Best viewed in color.)

Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From Word Embeddings To Document Distances. *Proceedings of The 32nd International Conference on Machine Learning* (Vol. 37, pp. 957–966)

There are several approaches

- word mover's distance
- soft cosine similarity

In common: we use pre-trained embeddings to replace words (that otherwise would just have a random identifier and be unrelated) with vectors representing their meaning, when calculating our measure of interest

(Ab-)using word embeddings to detect biases

Biased embeddings

- word embeddings are trained on large corpora
- As the task is to learn how to predict a word from its context (CBOW) or vice versa (skip-gram), biased texts produce biased embeddings
- If in the training corpus, the words "man" and "computer programmer" are used in the same context, then we will learn such a gender bias

Bolukbasi, T., Chang, K.-W., Zou, J., Saligrama, V., & Kalai, A. (2016). Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings, 1–25. Retrieved from http://arxiv.org/abs/1607.06520

Biased embeddings

Usually, we do not want that (and it has a huge potential for a shitstorm)

Biased embeddings

Usually, we do not want that (and it has a huge potential for a shitstorm)

unless. . .

Biased embeddings

Usually, we do not want that (and it has a huge potential for a shitstorm)

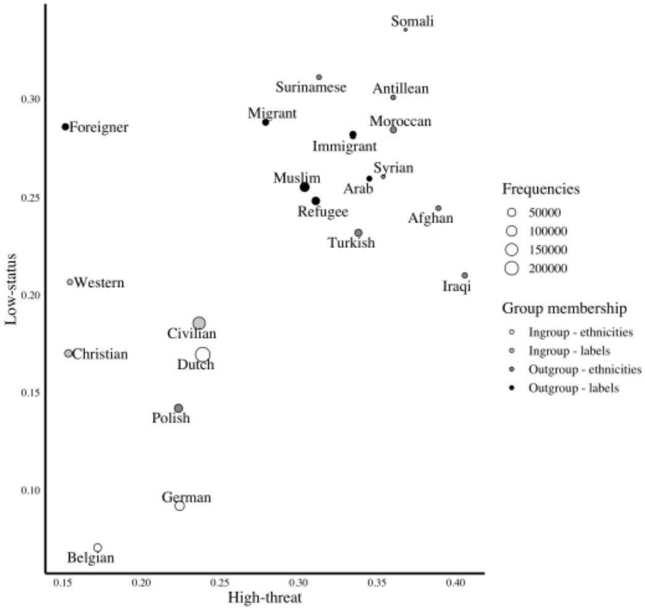
unless. . .

we actually want to chart such biases.

An exmaple from our research

We trained word embeddings on 3.3 million Dutch news articles. Are vector representations of outgroups (Maroccans, Muslims) closer to representations of negative stereotype words than ingroups?

Kroon, A.C., Van der Meer, G.L.A., Jonkman, J.G.F., &Trilling, D. (manuscript in prepration): Guilty by Association: Using Word Embeddings to Measure Ethnic Stereotypes in News Coverage


Somali
Surinamese
Antillean
Migrant
Moroccan
Foreigner
Immigrant
Syrian
Muslim
Arab
Refugee
Afghan
Turkish
Iraqi
Western
Civilian
Christian
Dutch
Polish
German
Belgian
0.30
0.25
0.20
0.15
0.10
Low-status
0.15
0.20
0.25
0.30
0.35
0.40
High-threat
Frequencies
50000
100000
150000
200000
Group membership
Ingroup - ethnicities
Ingroup - labels
Outgroup - ethnicities
Outgroup - labels

We can use pre-trained embeddings – but can we make even better ones? **The Amsterdam Embedding Model (AEM)**

Anne Kroon, Antske Fokkens, Damian Trilling, Felicia Loecherbach, Judith Moeller, Mariken A. C. G. van der Velden, Wouter van Atteveldt

Why do this?

- Embedding models are of great interest to communication scholars
- yet... Most publicly available models represent **English** language
- The preparation of good-performing embedding models require a significant amount of **time** and **access to a large amount of data sets**
- Few Dutch embedding models are available, but trained on ordinary human language from the World Wide Web.
- These models do not capture the specifics of news article data and are therefore less suitable to study and understand dynamics of this domain
- $\Rightarrow$ No model is available trained on Dutch news data

Project's Aim

Aim of the current project

1. Develop and evaluate a high-quality embedding model
2. Assess performance in downstream tasks of interest to Communication Science (such as topic classification of newspaper data).
3. Facilitate distribution and use of the model
4. Offer clear methodological recommendations for researchers interested using our Dutch embedding model

Training data

Training data set

- Dataset: diverse print and online news sources
- Preprocessing: duplicate sentences were removed
- Telegraaf (print & online), NRC Handelsblad (print & online), Volkskrant (print & online), Algemeen Dabldad (print & online), Trouw (print & online), nu.nl , nos.nl
- # words: 1.18b (1181701742)
- # sentences: 77.1M (77151321)

Training model

Training model

- We trained the model using Gensim's Word2Vec package in Python
- Skip-gram with negative sampling, window size of 5, 300-dimensional word vectors

Evaluation

Evaluation of the Amsterdam Embedding Model

Evaluation

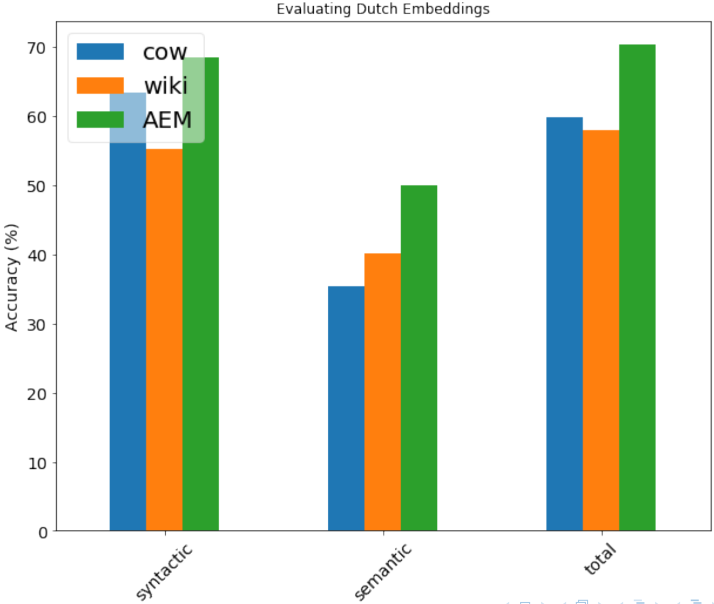
Evaluation methods

- To evaluate the model, we compare it to two other publicly available embedding models
  - $\Rightarrow$ **'Wiki'**: Embedding model trained on Wikipedia data (FastText)
  - $\Rightarrow$ **'Cow'**: Embedding model trained on diverse .nl and .be sites (Schafer & Bildhauer, 2012; Tulkens et al., 2016)
  - $\Rightarrow$ **'AEM'**: Amsterdam Embedding Model

Evaluation data

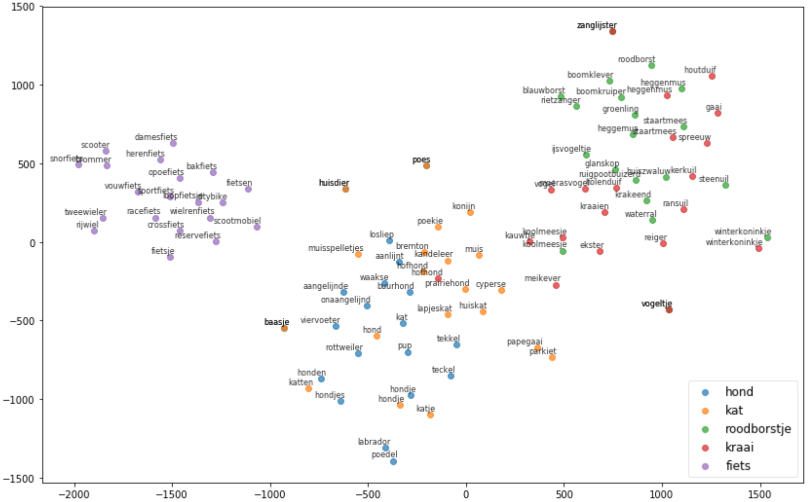
Evaluation data

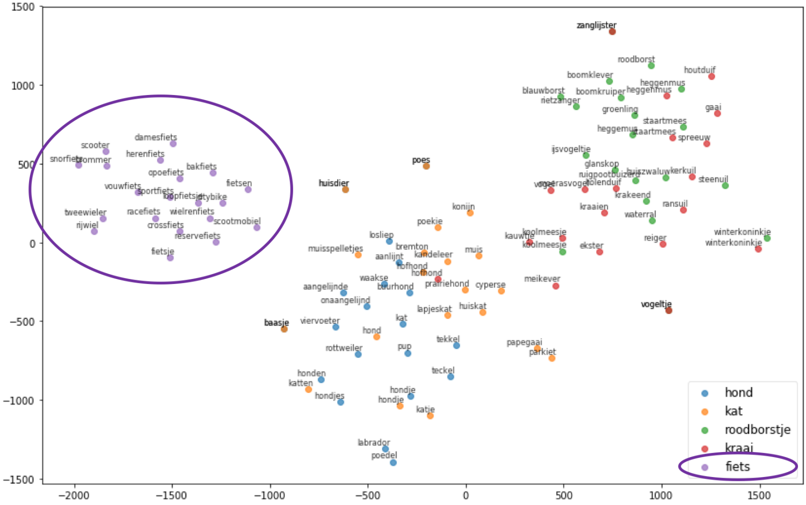
1. 'relationship'/ analogy-task (Tulkens et al., 2016)
   - **syntatic relationships**: dans dansen loop *[lopen]*
   - **semantic relationships**: denemarken kopenhagen noorwegen *[oslo]*
2. 5806 relationship tasks


Evaluating Dutch Embeddings
cow
wiki
AEM
Accuracy (%)
0
10
20
30
40
50
60
70
syntactic
semantic
total

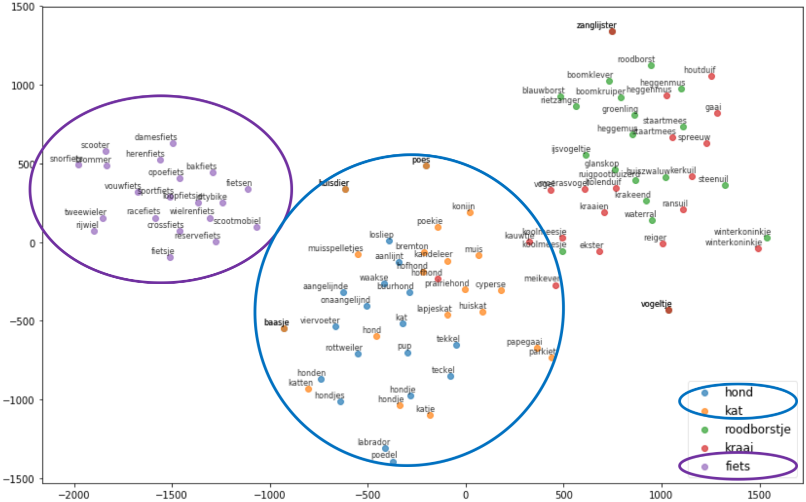
Illustration

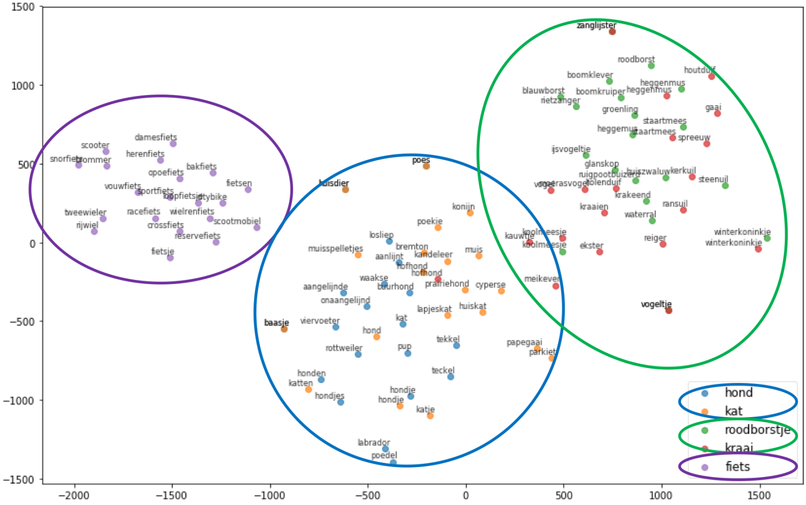
Illustration - Using the Amsterdam Embedding Model


hond
kat
roodborstje
kraai
fiets
zanglijster
roodborst
houtduif
boomklever
heggenmus
blauwborst
boomkruiper
rietzanger
groenling
gaai
staartmees
heggemus
spreeuw
ijsvogeltje
poes
damesfiets
scooter
herenfiets
snorfiets
brommer
bakfiets
opoefiets
glanskop
huiszwaluw
kerkuil
steenuil
huisdier
fietsen
vouwfiets
sportfiets
loopfiets
citybike
krakeend
tweewieler
racefiets
wielrenfiets
konijn
kraaien
ransuil
waterral
rijwiel
crossfiets
scootmobiel
poekie
reservefiets
losliep
koolmeesje
kauwtje
reiger
winterkoninkje
fietsje
muisspelletjes
bremton
muis
ekster
aanlijnt
kandeleer
hofhond
meikever
waakse
prairiehond
cyperse
aangelijnde
buurhond
onaangelijnd
vogeltje
lapjeskat
huiskat
baasje
viervoeter
kat
hond
tekkel
papegaai
parkiet
rottweiler
pup
teckel
honden
katten
hondjes
hondje
katje
labrador
poedel


zanglijster
roodborst
houtduif
boomklever
heggenmus
blauwborst
boomkruiper
heggenmus
rietzanger
groenling
gaai
staartmees
heggemus
staartmees
spreeuw
ijsvogeltje
glanskop
huiszwaluw
kerkuil
steenuil
ruigpootbuizerd
krakeend
kraaien
ransuil
waterral
koolmeesje
kauwtje
koolmeesje
ekster
reiger
winterkoninkje
winterkoninkje
meikever
vogeltje
damesfiets
scooter
herenfiets
snorfiets
brommer
bakfiets
opoefiets
fietsen
vouwfiets
sportfiets
loopfietsje
citybike
tweewieler
racefiets
wielrenfiets
scootmobiel
rijwiel
crossfiets
reservefiets
fietsje
poes
huisdier
konijn
poekie
losliep
bremton
muisspelletjes
aanlijnt
kandeleer
muis
hofhond
waakse
buurhond
prairiehond
cyperse
aangelijnde
onaangelijnd
lapjeskat
huiskat
kat
baasje
viervoeter
hond
tekkel
papegaai
parkiet
rottweiler
pup
teckel
honden
katten
hondjes
hondje
hondje
katje
labrador
poedel
1500
1000
500
0
−500
−1000
−1500
−2000
−1500
−1000
−500
0
500
1000
1500
hond
kat
roodborstje
kraai
fiets


damesfiets
scooter
herenfiets
brommer
bakfiets
opoefiets
fietsen
vouwfiets
sportfiets
tweewieler
racefiets
wielrenfiets
rijwiel
scootmobiel
crossfiets
reservefiets
fietsje


1500
1000
500
0
−500
−1000
−1500
−2000
−1500
−1000
−500
0
500
1000
1500
hond
kat
roodborstje
kraai
fiets
zanglijster
roodborst
houtduif
boomklever
heggenmus
blauwborst
boomkruiper
rietzanger
groenling
gaai
staartmees
heggenmus
spreeuw
ijsvogeltje
glanskop
huiszwaluw
kerkuil
steenuil
krakeend
kraaien
waterral
ransuil
reiger
winterkoninkje
ekster
meikever
vogeltje
damesfiets
scooter
herenfiets
snorfiets
bakfiets
opoefiets
fietsen
vouwfiets
sportfiets
tweewieler
racefiets
wielrenfiets
scootmobiel
rijwiel
crossfiets
reservefiets
fietsje
poes
huisdier
konijn
poekje
losliep
bremton
muisspelletjes
aanlijnt
kandeleer
muis
waakse
buurhond
prairiehond
cyperse
aangelijnde
onaangelijnd
lapjeskat
huiskat
baasje
viervoeter
kat
hond
tekkel
papegaai
parkiet
rottweiler
pup
teckel
honden
katten
hondjes
hondje
hondje
katje
labrador
poedel


1500
1000
500
0
−500
−1000
−1500
−2000
−1500
−1000
−500
0
500
1000
1500
hond
kat
roodborstje
kraai
fiets
zanglijster
roodborst
houtduif
boomklever
heggenmus
blauwborst
boomkruiper
rietzanger
groenling
gaai
staartmees
heggemus
staartmees
spreeuw
ijsvogeltje
glanskop
huiszwaluw
kerkuil
ruigpootbuizerd
steenuil
holenduif
krakeend
kraaien
ransuil
waterral
winterkoninkje
winterkoninkje
reiger
kauwtje
koolmeesje
koolmeesje
ekster
meikever
vogeltje
damesfiets
scooter
herenfiets
snorfiets
brommer
bakfiets
opoefiets
fietsen
vouwfiets
sportfiets
loopfiets
citybike
tweewieler
racefiets
wielrenfiets
rijwiel
crossfiets
scootmobiel
reservefiets
fietsje
poes
huisdier
konijn
poekie
losliep
bremton
muisspelletjes
aanlijnt
kandeleer
muis
hofhond
waakse
buurhond
prairiehond
cyperse
aangelijnde
onaangelijnd
lapjeskat
huiskat
baasje
viervoeter
kat
hond
tekkel
papegaai
parkiet
rottweiler
pup
honden
katten
teckel
hondjes
hondje
hondje
katje
labrador
poedel

Re-usability

Re-usability of the Amsterdam Embedding Model

Re-usability

Reusing model and data

1. The final Amsterdam Embedding Model will be made available at `https://github.com/annekroon/AEM`
2. Open access to all the code

Next meetings

Friday

Working with word embeddings in gensim. (Fun tutorial, if you want to: `https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial`)

Wednesday: Wrapping up and moving on

What did we not cover? What are future directions?