

IN2110: Språkteknologiske metoder

Ordvektorer

Egil Rønningstad

Språkteknologigruppen (LTG)

13. februar, 2024



- ▶ Focus on **words** rather than *documents*.
- ▶ Lexical semantics = the study of word meaning.
- ▶ **Distributional models** of semantics.
- ▶ **Word vectors**
- ▶ Example tasks for evaluating word vectors
- ▶ And more.



- ▶ So far we've been assuming **BoW features** for **representing documents**.
- ▶ Often also used for representing other units of texts, like **sentences**.
- ▶ Many **sentence-classification** tasks in NLP.

- ▶ So far we've been assuming **BoW features** for **representing documents**.
- ▶ Often also used for representing other units of texts, like **sentences**.
- ▶ Many **sentence-classification** tasks in NLP.
- ▶ Example: polarity classification (part of sentiment analysis).

I was impressed, this was not bad!



{was, was, !, not, I, impressed, bad, this}

- ▶ What is missing with a BoW representation?

I was impressed, this was not bad!

≠

I was not impressed, this was bad!

- ▶ Will have the same BoW representation! :(

I was impressed, this was not bad!

≠

I was not impressed, this was bad!

- ▶ Will have the same BoW representation! :(
- ▶ A simplistic but much-used approximation to capture ordering constraints: *n*-grams (typically bigrams and trigrams).
- ▶ Ordered sub-sequences of *n* words.

{was, was, !, not, I, impressed, bad, this}

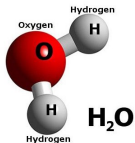
vs.

{‘I was’, ‘was impressed’ ... ‘was not’, ‘not bad’, ‘bad, !’}

- ▶ No information sharing between features.
 - ▶ All features are equally distinct.
- ▶ The pizza was great
 - ▶ The margeritha was awesome
 - ▶ The dog was sick
- ▶ Would be nice if our BoW representations knew that *pizza* and *margeritha* are similar to each other (but not to *dog*).

- ▶ A **sense** or concept is the meaning component of a word.
- ▶ Machines can not (yet?) hope to capture the actual **reference**...
- ▶ We focus instead on modeling the **relationships** between different words and their senses.
- ▶ There are many aspects of meaning...

- ▶ A relation that can be defined both on the word-level or sense-level:
- ▶ Holds between word **senses** whose meanings are (nearly) identical.
- ▶ Holds between **words** that are interchangeable without changing the *truth conditions* of a proposition.
- ▶ E.g.: *couch & sofa, big & large, automobile & car, water & H₂O*
- ▶ The linguistic **principle of contrast**:
any difference in *form* marks a difference in *meaning*.
- ▶ Arguably, **perfect** synonymy does not exist.



=



?

- ▶ True synonymy too strict.
- ▶ We will instead focus on words with **similar** or **related** meanings.
- ▶ Words that share important elements of meaning.
- ▶ Useful knowledge for many NLP tasks

- ▶ True synonymy too strict.
- ▶ We will instead focus on words with **similar** or **related** meanings.
- ▶ Words that share important elements of meaning.
- ▶ Useful knowledge for many NLP tasks
- ▶ **Similar** to *car*: *train, bicycle, cab, truck, vehicle, airplane, buss*

- ▶ True synonymy too strict.
- ▶ We will instead focus on words with **similar** or **related** meanings.
- ▶ Words that share important elements of meaning.
- ▶ Useful knowledge for many NLP tasks
- ▶ **Similar** to *car*: *train, bicycle, cab, truck, vehicle, airplane, buss*
- ▶ **Related** to *car*: *road, fuel, wheel, traffic, driver, drive, license*
- ▶ Relatedness more broadly defined than similarity: association by domain or semantic field.

- ▶ Sometimes interested in more narrow kinds of word similarity.
- ▶ E.g. **affective meanings** or **connotations**:
- ▶ Aspects of a word's meaning related to a writer/reader's emotions, sentiment, opinions, or evaluations.
- ▶ Positive/negative connotation: *happy/sad*
- ▶ Positive/negative evaluation: *great/terrible*
- ▶ Important for NLP tasks like sentiment analysis, stance detection, argumentation mining, hate-speech detection, etc.

- ▶ Can a program automatically learn which words have similar meanings?
 - ▶ Just by looking at data of actual language use?
 - ▶ Without any prior knowledge?
- ▶ How can we represent word meaning in a mathematical model?

- ▶ Can a program automatically learn which words have similar meanings?
 - ▶ Just by looking at data of actual language use?
 - ▶ Without any prior knowledge?
- ▶ How can we represent word meaning in a mathematical model?

Concepts

- ▶ Distributional semantics
- ▶ Semantic spaces:
- ▶ Vector space models of word meaning

AKA the contextual theory of meaning

- *Meaning is use.* (Wittgenstein, 1953)
- *You shall know a word by the company it keeps.* (Firth, 1957)
- *The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities.* (Harris, 1968)

AKA the contextual theory of meaning

- *Meaning is use.* (Wittgenstein, 1953)
- *You shall know a word by the company it keeps.* (Firth, 1957)
- *The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities.* (Harris, 1968)

He was hungover after drinking too many shots of **retawerif** at the party last night.



- ▶ **The hypothesis:** If two words share similar contexts, we can assume that they have similar meanings.
- ▶ Comparing **meaning** reduced to comparing **context**,
 - no need for prior knowledge!



- ▶ **The hypothesis:** If two words share similar contexts, we can assume that they have similar meanings.
- ▶ Comparing **meaning** reduced to comparing **context**,
 - no need for prior knowledge!
- ▶ Given the processing power of modern computers and the availability of vast amounts of electronic texts. . .
- ▶ . . . we can now implement in practice the classic empiricist claims of Firth, Harris, Wittgenstein, et al.

- ▶ Record **contexts** of words across a large corpus.
 - ▶ Each word is represented by a set of **features**.
 - ▶ Each feature records some property of the observed contexts.
 - ▶ Words that are found to have **similar features** are expected to also have **similar meaning**.
 - ▶ Features can be represented in a **vector space** model, with similarity modeled as geometrical **distance**.
-
- ▶ Some design decisions;
 - ▶ How do we define '*context*'?
 - ▶ How do we define a '*word*'?

Term-document matrix



- Recall our **term-document matrix**,
- We have seen how **columns** represent **document-vectors**.
- Can let **rows** represent **word-vectors**.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- ▶ Recall our **term-document matrix**,
- ▶ We have seen how **columns** represent **document-vectors**.
- ▶ Can let **rows** represent **word-vectors**.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- ▶ Recall our **term-document matrix**,
- ▶ We have seen how **columns** represent **document-vectors**.
- ▶ Can let **rows** represent **word-vectors**.

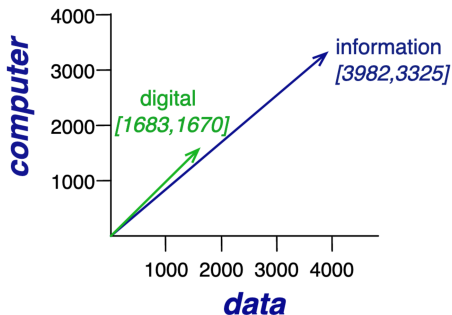
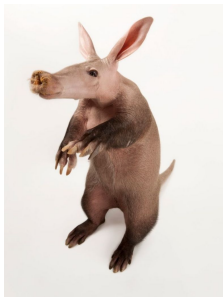
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- ▶ But documents are a bit coarse as context for modeling word meaning.
- ▶ Rather want to use a **word co-occurrence matrix**
- ▶ (aka **term-term** or **term-context matrix**)

Term-term matrix



	aardvark	...	computer	data	result	pie	sugar
cherry	0	...	2	8	9	442	25
strawberry	0	...	0	0	1	60	19
digital	0	...	1670	1683	85	5	4
information	0	...	3325	3982	378	5	13



- ▶ To extract the feature vectors, we record all words co-occurring with the target word in its contexts across a corpus.
- ▶ But what is 'context'?
- ▶ For BoW: Document, sentence, or window ($\pm k$ words left/right).

- ▶ To extract the feature vectors, we record all words co-occurring with the target word in its contexts across a corpus.
- ▶ But what is 'context'?
- ▶ For BoW: Document, sentence, or window ($\pm k$ words left/right).
- ▶ Can also use n -grams or grammatical relations.
- ▶ *Broader* context tend to capture *relatedness*.
- ▶ *Narrower* context tend to capture *similarity*.
- ▶ Like for document vectors, the raw counts are typically **weighted**;
- ▶ using e.g TF-IDF or pointwise mutual information, or some other association measure.

Raw: "The programmer's programs had been programmed."

- ▶ **Tokenization**: Splitting a text into sentences and words or other units.
- ▶ Different levels of abstraction and morphological **normalization**:
 - ▶ What to do with case, numbers, punctuation, compounds, ...?
 - ▶ Full-form words vs. lemmas vs. stems ...
- ▶ Stop-list: filter out closed-class words or function words.
 - ▶ The idea is that only *content words* provide relevant context.

Recap: what is a word?



Raw: "The programmer's programs had been programmed."
Tokenized: the programmer 's programs had been programmed .

- ▶ **Tokenization**: Splitting a text into sentences and words or other units.
- ▶ Different levels of abstraction and morphological **normalization**:
 - ▶ What to do with case, numbers, punctuation, compounds, ...?
 - ▶ Full-form words vs. lemmas vs. stems ...
- ▶ Stop-list: filter out closed-class words or function words.
 - ▶ The idea is that only *content words* provide relevant context.

Recap: what is a word?



Raw: "The programmer's programs had been programmed."
Tokenized: the programmer 's programs had been programmed .
Lemmatized: the programmer 's program have be program .

- ▶ **Tokenization**: Splitting a text into sentences and words or other units.
- ▶ Different levels of abstraction and morphological **normalization**:
 - ▶ What to do with case, numbers, punctuation, compounds, ...?
 - ▶ Full-form words vs. lemmas vs. stems ...
- ▶ Stop-list: filter out closed-class words or function words.
 - ▶ The idea is that only *content words* provide relevant context.

Recap: what is a word?



Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .
Lemmatized:	the programmer 's program have be program .
W/ stop-list:	programmer program program

- ▶ **Tokenization:** Splitting a text into sentences and words or other units.
- ▶ Different levels of abstraction and morphological **normalization**:
 - ▶ What to do with case, numbers, punctuation, compounds, ...?
 - ▶ Full-form words vs. lemmas vs. stems ...
- ▶ Stop-list: filter out closed-class words or function words.
 - ▶ The idea is that only *content words* provide relevant context.

Recap: what is a word?



Raw:	"The programmer's programs had been programmed."
Tokenized:	the programmer 's programs had been programmed .
Lemmatized:	the programmer 's program have be program .
W/ stop-list:	programmer program program
Stemmed:	program program program

- ▶ **Tokenization:** Splitting a text into sentences and words or other units.
- ▶ Different levels of abstraction and morphological **normalization**:
 - ▶ What to do with case, numbers, punctuation, compounds, ...?
 - ▶ Full-form words vs. lemmas vs. stems ...
- ▶ Stop-list: filter out closed-class words or function words.
 - ▶ The idea is that only *content words* provide relevant context.

- ▶ A **word embedding** is a vector with some particular properties:
 - ▶ Low-dimensional (short, $|V|$ = tens or hundreds)
 - ▶ dense (most elements are non-zero)
 - ▶ continuous / distributed
- ▶ In contrast to **traditional count-based vectors** that are:
 - ▶ high-dimensional (long, $|V|$ = thousands or millions)
 - ▶ sparse (most elements are zero)
 - ▶ discrete
- ▶ Can be created by **dimensionality reduction** of the original space.
- ▶ But often estimated through **prediction** tasks rather than **corpus counts**.

Word embeddings



'cake'
vector:

1	2	3	4
0	1	0	0

'bake'

high-dimensional

100k

3	0
---	---

sparse

discrete

'birthday'

'cake'
embedding:

1	2	3	4
0.13	0.02	0.43	0.31

?

low-dimensional

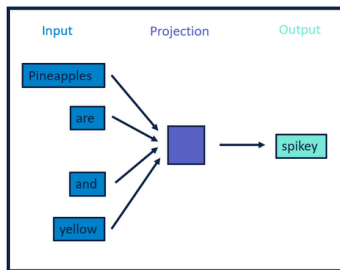
100

0.17

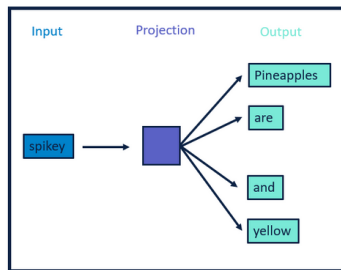
dense

? distributed

- ▶ **Language modeling**; aims to predict the next word in a sequence.
- ▶ *'It was a dark and stormy ...'*
- ▶ The next observed word in the sequence provides the gold answer.
- ▶ No need for hand-labeled data; use **running text as training data**.
- ▶ 'Self-supervision'
- ▶ Classifiers like logistic regression (next lecture!) **train a vector of weights** to make a prediction.
- ▶ Can take the learned classifier **weights** as the word embeddings! (We here don't actually care about the prediction itself.)



CBOW



Skip-gram

- ▶ Example of approach that aim to estimate word embeddings.
- ▶ Implements two algorithms: **CBOW** and **Skip-Gram**
- ▶ **Instead of counting** how often each word w occurs near *pineapple*, **train a classifier to predict**: Is w likely to show up near *pineapple*?
- ▶ A word s seen near *pineapple* acts as gold **positive examples**.
- ▶ Randomly sample other words in the lexicon as **negative examples**.
- ▶ Use the learned weights as the embeddings.

- ▶ Many alternative algorithms and toolkits exists:
- ▶ word2vec, fastText, Glove, ELMo, BERT, etc.
- ▶ Typically based on **artificial neural networks**.
- ▶ A shift from **feature engineering** to **representation learning**.
- ▶ The standard input representation to neural methods.
- ▶ An important part of **IN5550 – Neural Methods in NLP**



- ▶ Several standard ways to evaluate the quality of word vectors.
- ▶ Mainly based on **similarity tests** and **analogical reasoning**.

- ▶ Compare to human judgements of word similarity.
- ▶ Several standard test sets available.
- ▶ Example from SimLex-999 (Hill et al., 2015):

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3



- ▶ Example from the Google analogy test set (Mikolov et al. 2013)
- ▶ Q: *man* is to *woman* as *king* is to ???



- ▶ Example from the Google analogy test set (Mikolov et al. 2013)
- ▶ *Q: man is to woman as king is to ??? (A: queen)*



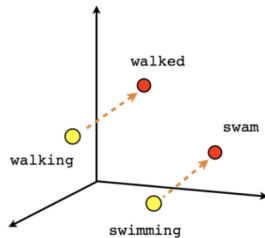
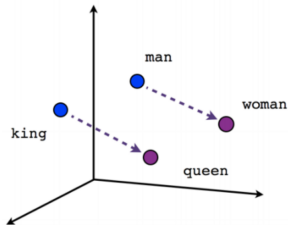
- ▶ Example from the Google analogy test set (Mikolov et al. 2013)
- ▶ Q: *man* is to *woman* as *king* is to ??? (A: *queen*)
- ▶ Q: *walking* is to *walked* as *swimming* is to ???

- ▶ Example from the Google analogy test set (Mikolov et al. 2013)
- ▶ Q: *man* is to *woman* as *king* is to ??? (A: *queen*)
- ▶ Q: *walking* is to *walked* as *swimming* is to ??? (A: *swam*)

Evaluating embeddings (2)



- ▶ Example from the Google analogy test set (Mikolov et al. 2013)
- ▶ Q: *man* is to *woman* as *king* is to ??? (A: *queen*)
- ▶ Q: *walking* is to *walked* as *swimming* is to ??? (A: *swam*)
- ▶ A surprising property of word embeddings is that word **analogies** can often be solved with vector arithmetic.
- ▶ $\text{vec}(\text{'king'}) - \text{vec}(\text{'man'}) + \text{vec}(\text{'woman'}) \approx \text{vec}(\text{'queen'})$
- ▶ $\text{vec}(\text{'swimming'}) - \text{vec}(\text{'walking'}) + \text{vec}(\text{'swim'}) \approx \text{vec}(\text{'swam'})$

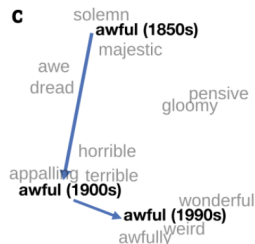
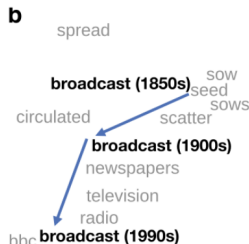


Examples from Google analogy test set



	Relation type	# Questions	Word pair 1		Word pair 2	
Semantic	Common capital city	506	Athens	Greece	Baghdad	Iraq
	All capital cities	4,524	Abuja	Nigeria	Accra	Ghana
	Currency	866	Algeria	dinar	Angola	kwanza
	City-in-state	2,467	Chicago	Illinois	Houston	Texas
	Man–woman	506	boy	girl	brother	sister
Syntactic	Adjective-to-adverb	992	amazing	amazingly	apparent	apparently
	Opposite	812	acceptable	unacceptable	aware	unaware
	Comparative	1,332	bad	worse	big	bigger
	Superlative	1,122	bad	worst	big	biggest
	Present participle	1,056	code	coding	dance	dancing
	Nationality adjective	1,599	Albania	Albanian	Argentina	Argentinean
	Past tense	1,560	dancing	danced	decreasing	decreased
	Plural nouns	1,332	banana	bananas	bird	birds
	Plural verbs	870	decrease	decreases	describe	describes

- ▶ Another example of use-case or evaluation task:
- ▶ Embeddings can help study word history!
- ▶ Train embeddings on corpora from different time periods.
- ▶ Trace how meaning changes over time by comparing changes in vector distances and neighbor relations.



Conflation of word senses

- ▶ Single vector for each word.
- ▶ Problem for polysemy and homonymy.

Conflation of word senses

- ▶ Single vector for each word.
- ▶ Problem for polysemy and homonymy.

Bias

- ▶ Human biases in the training corpus will be reflected word vectors.
- ▶ E.g. associations between gender-terms and stereotypical occupations.

Conflation of word senses

- ▶ Single vector for each word.
- ▶ Problem for polysemy and homonymy.

Bias

- ▶ Human biases in the training corpus will be reflected word vectors.
- ▶ E.g. associations between gender-terms and stereotypical occupations.

Antonyms

- ▶ Senses that are **opposites**.
- ▶ *long/short, fast/slow, dark/light, up/down, rise/fall, hot/cold, in/out*
- ▶ Difficult to distributionally distinguish from synonyms.



- ▶ Word embeddings provide a good representation of words.
- ▶ Let us model **word similarity**.
- ▶ What about **sentence similarity** or **document similarity**?
- ▶ From word embeddings to text embeddings.

- ▶ Word embeddings provide a good representation of words.
- ▶ Let us model **word similarity**.
- ▶ What about **sentence similarity** or **document similarity**?
- ▶ From word embeddings to text embeddings.
- ▶ Consider sentiment analysis as an **example**:
- ▶ We want to train and apply a sentence-level classifier to predict positive / negative polarity of sentences.

Recap: information sharing between features



- ▶ BoW representations of sentences:
 - ▶ No information sharing between features.
 - ▶ All features are equally distinct.
- ▶ The pizza was great
 - ▶ The margherita was awesome
 - ▶ The dog was sick
- ▶ Would be nice if our BoW representations knew that *pizza* and *margherita* are similar to each other (but not to *dog*).

- ▶ Can represent documents by concatenating or averaging word embeddings.
- ▶ For a **sentence** $s = w_1, \dots, w_n$ with **word embeddings** $x_1, \dots, x_n$, we can compute the **sentence embedding** as $\text{vec}(s) = \frac{1}{n} \sum_{i=1}^n x_i$
- ▶ Can then use $\text{vec}(s)$ as input to a sentence classifier.
- ▶ Benefit; similarity relations between features.
- ▶ Allows us to '**pre-train**' word representations on **unlabeled** data, before training e.g. a sentence-classifier on **labeled** data.



- ▶ Both traditional word vectors and more modern embeddings allow us **model word similarity using raw text**; no labeled data needed.
- ▶ **Semantic similarity** modeled as **distributional similarity**, which is in turn computed as **vector distance**.
- ▶ In modern NLP, embeddings have a very central role!
- ▶ Standard input representation to neural (deep learning) classifier.