Short introduction to NLP and neural networks in sequence processing

Paweł Rychlikowski

Instytut Informatyki UWr

18 czerwca 2024

Hello World!

We start!

This lecture, and the rest (?) of our course will be about **natural language processing (NLP)**

Hello World!

We start!

This lecture, and the rest (?) of our course will be about **natural language processing (NLP)**

Our main goals:

- Show some flavour of NLP
- Describe the most important NLP tasks
- Describe some NN architectures in NLP (and generally sequence processing)

Hello World!

We start!

This lecture, and the rest (?) of our course will be about **natural language processing (NLP)**

Our main goals:

- Show some flavour of NLP
- Describe the most important NLP tasks
- Describe some NN architectures in NLP (and generally sequence processing)
- Note: transformers, and many more advanced topics will be postponed to the Language Models course.

Bibliography

This part of the course is mainly based on:

- Natural Language Processing with Deep Learning (CS224N/Ling284), by Chris Manning
- Speech and Language Processing, 3rd edition draft, by D. Juraffsky, H. Martin (https://web.stanford.edu/~jurafsky/slp3/)
- Deep Learning book (Goodfellow, Bengio, Courville)

Bibliography

This part of the course is mainly based on:

- Natural Language Processing with Deep Learning (CS224N/Ling284), by Chris Manning
- Speech and Language Processing, 3rd edition draft, by D. Juraffsky, H. Martin (https://web.stanford.edu/~jurafsky/slp3/)
- Deep Learning book (Goodfellow, Bengio, Courville)

(some slides from CS224N will be just copied...)

Plans for the lectures

1. Word semantics (aka. context free word embedings)
2. 3 simplest NN architectures for NLP (try to guess!)

Plans for the lectures

1. Word semantics (aka. context free word embedings)
2. 3 simplest NN architectures for NLP (try to guess!)
3. Recurent Neural Networks (RNN, LSTM, GRU)

Plans for the lectures

1. Word semantics (aka. context free word embedings)
2. 3 simplest NN architectures for NLP (try to guess!)
3. Recurent Neural Networks (RNN, LSTM, GRU)
4. Machine Translation (maybe with Attention)

What could be treated as a natural language?

Virtually everything!

What could be treated as a natural language?

Virtually everything!

- Computer programs,
- Music score,
- Picture (sequence of pixels/blocks when the resolution is fixed)
- DNA sequences,
- Moves in chess, othello, checkers, ...
- ...


ANNO
LITERARIA INSTRUXIT

Brief history of NLP

- Rule based systems
  - Noam Chomsky published his book, **Syntactic Structures**, in **1957**.
  - In **1964**, ELIZA, a "typewritten" comment and response process, designed to imitate a psychiatrist
- Using **corpora** to statistically model language
  - (for instance IBM automatic speech recognition, approx. **1980+**)
- Machine Learning comes! (late 1980's)
  - NBC, HMM, CRF, Logistic Regression, SVM, ...
  - Feature engineering (binary features like **w** in distance 2, and other)
- **Pretraining + Neural Networks and Deep Learning**

Brief history of NLP

- Rule based systems
  - Noam Chomsky published his book, **Syntactic Structures**, in **1957**.
  - In **1964**, ELIZA, a "typewritten" comment and response process, designed to imitate a psychiatrist
- Using **corpora** to statistically model language
  - (for instance IBM automatic speech recognition, approx. **1980+**)
- Machine Learning comes! (late 1980's)
  - NBC, HMM, CRF, Logistic Regression, SVM, ...
  - Feature engineering (binary features like **w** in distance 2, and other)
- **Pretraining + Neural Networks and Deep Learning**
- **Large Language Models, few shots learning, zero shots learning, reinforcement learning from human feedback**

What is natural language?

What is natural language?

Answer

Language is a set of sequences!

What is natural language?

Answer

Language is a set of sequences!

Many options:

- Characters (ASCII, Extended ASCII, Unicode)
- Bytes (UTF-8)

What is natural language?

Answer

Language is a set of sequences!

Many options:

- Characters (ASCII, Extended ASCII, Unicode)
- Bytes (UTF-8)
- Words, Words++
- (sometimes common phrases, like New York, are treated as words)

What is natural language?

Answer

Language is a set of sequences!

Many options:

- Characters (ASCII, Extended ASCII, Unicode)
- Bytes (UTF-8)
- Words, Words++
- (sometimes common phrases, like New York, are treated as words)
- WordPieces (fixed number, approx. 30K, includes letters, and popular words)

Words++

Remember: we start with a language treated as a sequence of words

Note: the number of **word types** can be huge!

- You will need some technical *pseudowords*: `<out-of-vocab>`, `<begin-of-sentence>`, `<end-of-sentence>`, `<pad>`, ...
- Other option: `<unknown-adj>`, `<unknown-noun>`, `<unknown-verb>`, `<unknown-other>`
- Other option: `<-ology>`, `<-ator>`, `<-ization>`, ...
- (you can just use last K letters for uncommon words)

Tokenization

Look for **tokenize** method in your favourite NLP/NN library!

Tokenization

Look for **tokenize** method in your favourite NLP/NN library!

Simple tokenization of s:

- For every punctuation character c, do
  `s = s.replace(c, ' ' + c + ' ')`
- Return `s.split()` or `s.lower().split()`

Representing words by their context

- Distributional semantics: **A word's meaning is given by the words that frequently appear close-by**
  - *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP!
- When a word *w* appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- We use the many contexts of *w* to build up a representation of *w*

…government debt problems turning into	***banking***	*crises as happened in 2009…*
…saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge…*
…India has just given its	***banking***	*system a shot in the arm…*

These context words will represent ***banking***

Word2Vec

Remark 1

The first robust dense vector representation of words!

Word2Vec

Remark 1

The first robust dense vector representation of words!

Remark 2

The first example of successful **pretraing** in NLP

Word2Vec

Remark 1

The first robust dense vector representation of words!

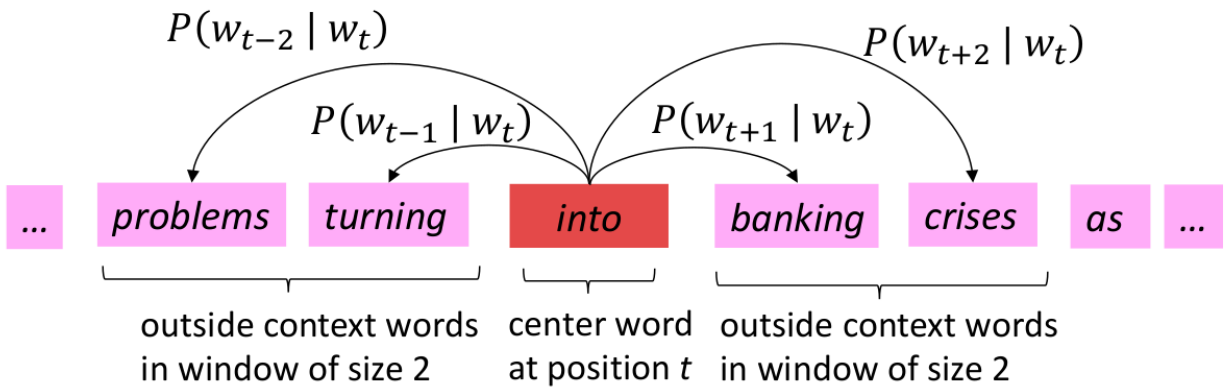
Remark 2

The first example of successful **pretraing** in NLP

- *Efficient Estimation of Word Representations in Vector Space* Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean
- *Distributed representations of words and phrases and their compositionality*, Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg S.; Dean, Jeff (2013).

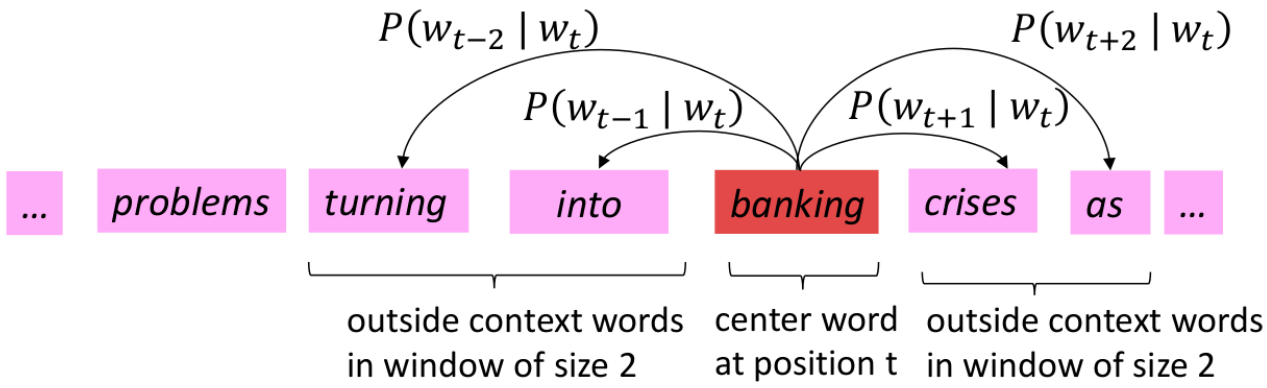
Word2Vec Overview

Example windows and process for computing $P(w_{t+j} \mid w_t)$



Word2Vec Overview

Example windows and process for computing $P(w_{t+j} \mid w_t)$



23

W2V versions

1. Skip-grams: $P(w_{t+j}|w_t)$ $(j \in \{-k, \ldots, k\})$

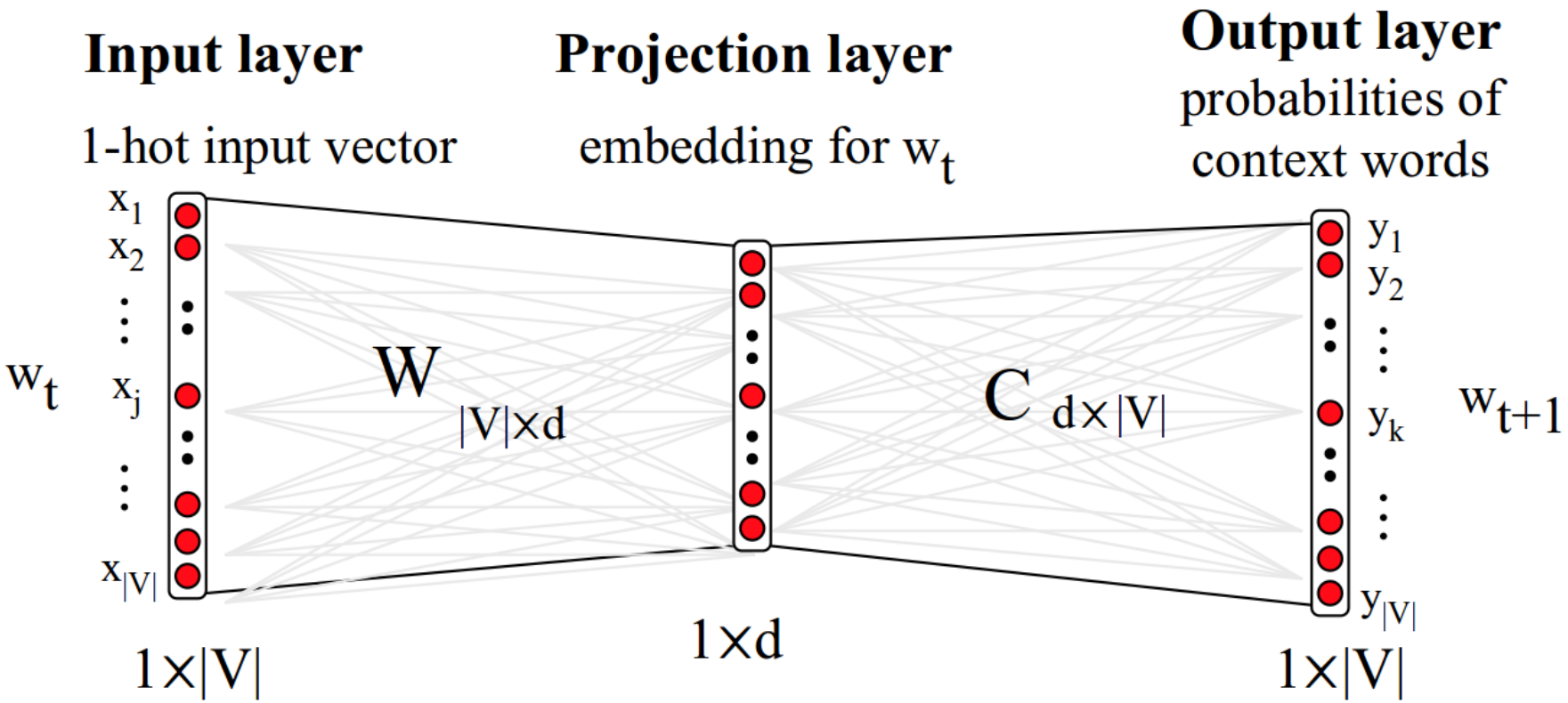
W2V versions

1. Skip-grams: $P(w_{t+j}|w_t)$ $(j \in \{-k, \ldots, k\})$
2. Bigrams: $P(w_{t+1}|w_t)$

W2V versions

1. Skip-grams: $P(w_{t+j}|w_t)$ $(j \in \{-k, \ldots, k\})$
2. Bigrams: $P(w_{t+1}|w_t)$
3. **CBOW** (continuous bag of words)

Word2Vec with skip-grams as Neural Network



Word2vec: objective function

For each position $t = 1, \ldots, T$, predict context words within a window of fixed size m, given center word w_t. Data likelihood:

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

θ is all variables to be optimized

sometimes called a *cost* or *loss* function

The objective function $J(\theta)$ is the (average) negative log likelihood:

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Minimizing objective function $\Leftrightarrow$ Maximizing predictive accuracy

24

Word2vec: prediction function

② Exponentiation makes anything positive

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

① Dot product compares similarity of *o* and *c*.
$u^T v = u.v = \sum_{i=1}^{n} u_i v_i$
Larger dot product = larger probability

③ Normalize over entire vocabulary to give probability distribution

- This is an example of the **softmax function** $\mathbb{R}^n \to (0,1)^n$ ← Open region

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} = p_i$$

- The softmax function maps arbitrary values x_i to a probability distribution p_i
  - "max" because amplifies probability of largest x_i
  - "soft" because still assigns some probability to smaller x_i
  - Frequently used in Deep Learning

But sort of a weird name because it returns a distribution!

27

Problem with large Softmax

Large sum in the gradient!

Problem with large Softmax

Large sum in the gradient!

Solution: hierarchical softmax (decompose the decision to the list of decisions)
(probably) Better solution: negative sampling

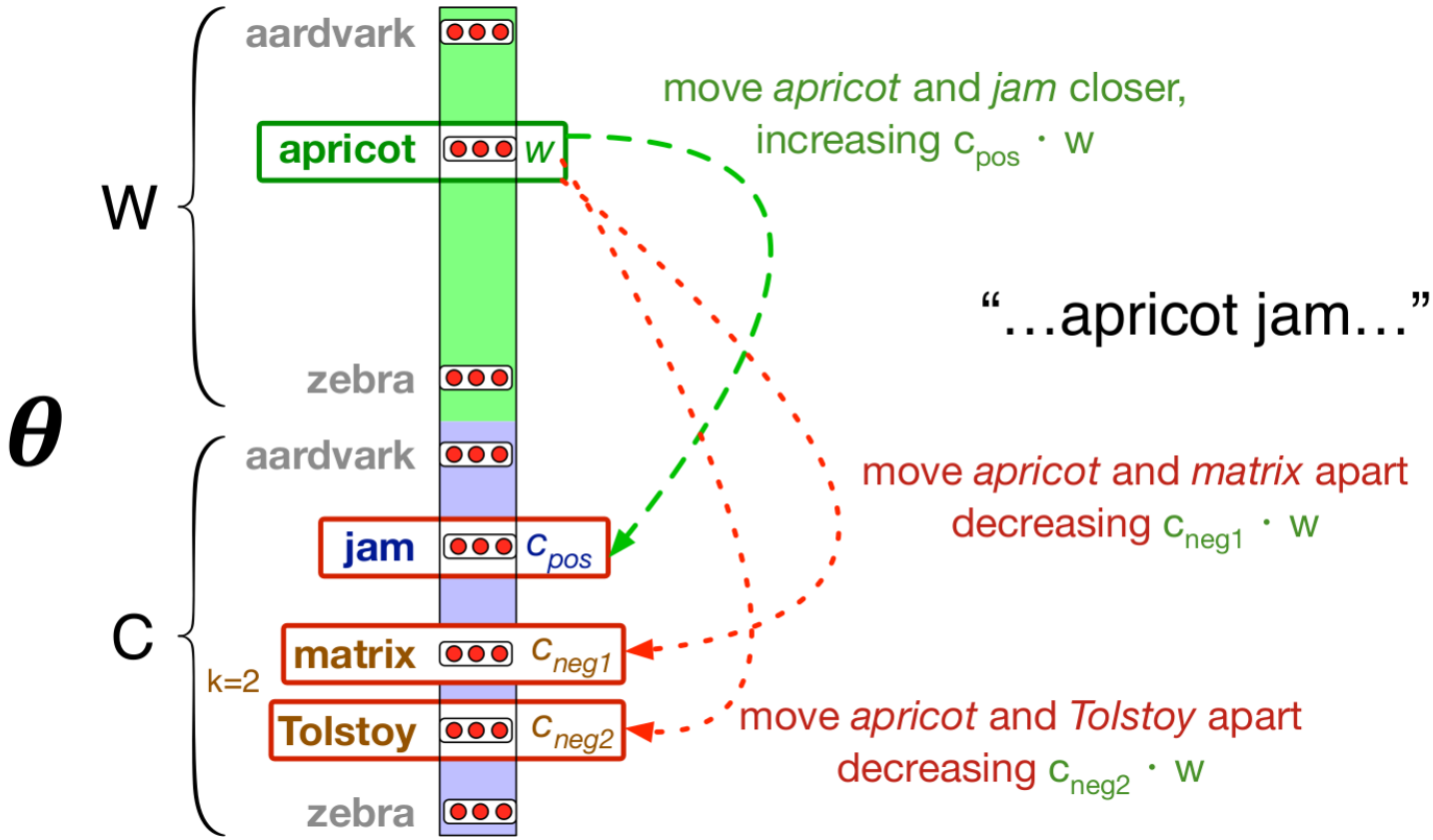
Negative sampling

- We can easily obtain related objects
  - Two consecutive words, sentences, or sentences in one article, ...

Negative sampling

- We can easily obtain related objects
  - Two consecutive words, sentences, or sentences in one article, ...
- We can **sample** unrelated objects

Intuition of one step of gradient descent



The skip-gram model with negative sampling (HW2)

- Notation more similar to class and HW2:

$$J_{neg-sample}(\boldsymbol{u}_o, \boldsymbol{v}_c, U) = -\log \sigma(\boldsymbol{u}_o^T \boldsymbol{v}_c) - \sum_{k \in \{K \text{ sampled indices}\}} \log \sigma(-\boldsymbol{u}_k^T \boldsymbol{v}_c)$$

- We take *k* negative samples (using word probabilities)
- Maximize probability that real outside word appears;
  minimize probability that random words appear around center word

- Sample with $P(w)=U(w)^{3/4}/Z$, the unigram distribution U(w) raised to the 3/4 power (We provide this function in the starter code).
- The power makes less frequent words be sampled more often

11

Word2Vec training details

- Linear learning rate decay
- Window size ≈ 10
  - smaller window – more syntactic relations
  - bigger window – more semantic
- 3-6 epochs
- Starting learning rate $= 0.003$

Word2Vec training details

- Linear learning rate decay
- Window size ≈ 10
  - smaller window – more syntactic relations
  - bigger window – more semantic
- 3-6 epochs
- Starting learning rate $= 0.003$

Sample negative words with $P'(w) \sim \text{cnt}(w)^{0.75}$

Not only words!

We can apply the same algorithm to different objects:

- In Wikipedia: $P(\text{pointer to doc}_j|\text{doc}_i)$
- For characters: $P(c_j|c_i)$ (discovers vowels?)
- For recomendation systems: $P(\text{product}|\text{customer})$

Not only words!

We can apply the same algorithm to different objects:

- In Wikipedia: $P(\text{pointer to doc}_j|\text{doc}_i)$
- For characters: $P(c_j|c_i)$ (discovers vowels?)
- For recomendation systems: $P(\text{product}|\text{customer})$

Quiz

What is the meaning of:

- doc2vec
- node2vec
- import2vec
- code2vec
- dna2vec
- wave2vec

Common phrases

Common phrases can be treated as words!

Common phrases

Common phrases can be treated as words!

In original paper very simple strategy was implemented:

1. Find valuable, common bigrams, replace them by a new word
   - New York $\rightarrow$ New_York
2. Repeat!

Common phrases

Common phrases can be treated as words!

In original paper very simple strategy was implemented:

1. Find valuable, common bigrams, replace them by a new word
   - New York $\rightarrow$ New_York
2. Repeat!

Bigram quality:

$$\text{score}(w_i, w_j) = \frac{count(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}$$

where $\delta \approx 0.5$

Common phrases

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

How to use word2vec?

Best option: use gensim library

How to use word2vec?

Best option: use gensim library

1. Task 1: train vectors
2. Task 2: test vectors
3. Task 3: work with pretrained vectors

Let's test it in a notebook!

The most famous word equation

The most famous word equation

king - man + woman $\approx$ queen

Why it works?

Suppose we have that:

- $w_1, w_2, \ldots, w_k$ are (typical) contexts for women
- $m_1, m_2, \ldots, m_k$ are (typical) contexts for men
- $r_1, r_2, \ldots, r_k$ are (typical) contexts for medievial ruler

Why it works?

Suppose we have that:

- $w_1, w_2, \ldots, w_k$ are (typical) contexts for women
- $m_1, m_2, \ldots, m_k$ are (typical) contexts for men
- $r_1, r_2, \ldots, r_k$ are (typical) contexts for medievial ruler

The training objectives

- Make vec('king') similar to $\sum_i con(m_i) + \sum_i con(r_i)$
- Make vec('queen') similar to $\sum_i con(w_i) + \sum_i con(r_i)$
- Make vec('man') similar to $\sum_i con(m_i)$
- Make vec('woman') similar to $\sum_i con(m_i)$