Practical AI:
NLP. Semantics

Stanislav Protasov for
Harbour.Space University

CLIPS time!

- Computer has no brain shortcuts
- Computer interpret facts and rules EXACTLY as they are written

```
(defrule weapon-wins
   (weapon ?item)
   (range ?item ?type)
   (distance ?type)
   =>
   (assert (wins ?item))
)

(defrule battle
   (warrior ?person)
   (has ?person ?weapon)
   (wins ?weapon)
   =>
   (assert (wins ?person))
)
```

```
(deffacts inital-facts
   (warrior harry)
   (has harry bow)
   (weapon bow)
   (range bow large)
   (distance small)
)
```

Agenda

- Step aside: vectors, matrices
- Distributional semantics
- Building inverted index
- Text to vector, today’s search engines

Vectors, space, distance

What is **scalar**?

What is **vector**?

Why do applies math likes vectors?

What is **space**? Vector space?

How to we **draw** a vector?

What is vector’s **length/norm**?

How do we multiply **vector by scalar**? Normalized vector?

How do we define **distance** between two vectors?

That is **dot product**?

Matrix and matrix operations

What is **matrix**?

What is **matrix** multiplied **by scalar**?

What is **vector** multiplied **by matrix**? And vice versa?

What is **matrix by matrix**?

What is **identity matrix**?

What is **inverse matrix**?

Distributional semantics

Distributional hypothesis: *linguistic items with similar distributions have similar meanings*.

! distributions in **big** *corpora*

! [semantic] **similarity/distance** is the main object of research (of texts, words, authors, ...)

- Detect if the text is authored by ...
- Detect shift in word usage ...

! cannot generate new *meaningful* texts [yet]

- *Try iTap in your a phone*

Distributional semantics: task

- For each text/term you provide a vector
  - Latent space (of embeddings)
- Other operations are done with vectors
  - Euclidean distance
  - Cosine similarity
  - …
- E.g.
  - Search for synonyms or duplicate texts
  - Cluster texts in topics (news)
  - Use vectors as input of other models

From text to vector.
Let the journey begin!

Stemming

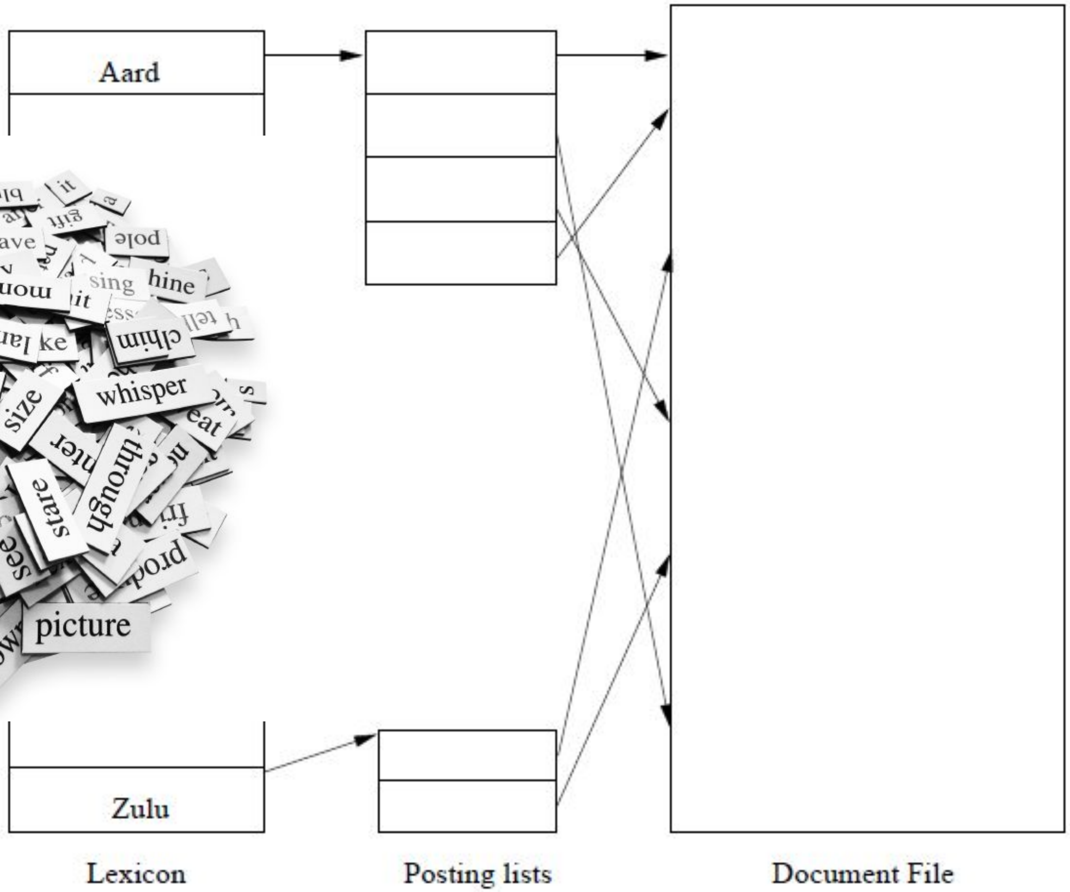
Word **stem** holds lexical meaning

Word **affixes** hold grammatical form

Lab #1. Let’s perform stemming

1) Given a text parse this text into **words**
2) Convert these words into **stems**
3) Answer:
    a) What is the size of lexicon for “the old man and the sea”? For “The Men of Good Will?”
    b) Build classes of words that were tokenized into the same stem

Lexicon and inverted index




```
девочек 562 3811 3995 6172 3933 495
кубического 4127
славно 2288 2864 802 5634 2665
бархатной 2562 616 5674
бархатного 6260
бархатном 6259 71 5070
avertir 1121
пустым 357 1224
```

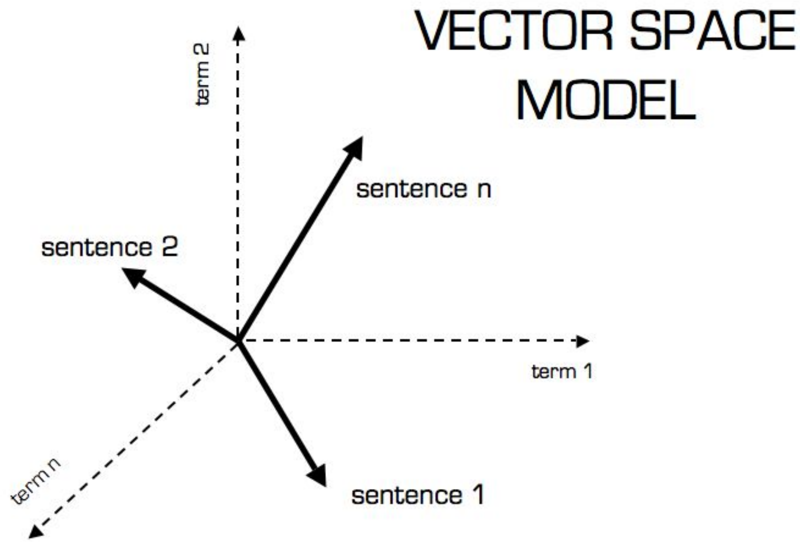
Lab #2. Let’s implement search engine

1) Use result of Lab #1. Convert sentences into *bags of words* (stems).
2) Implement search engine that **maximizes intersection cardinality** for query and document bags.

TF-IDF

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Search engine



$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Hometask. Let’s improve search engine!

1) Use results of lab #1 and lab #2.
2) Store lexicon as a vector.
3) Create TDM (list of vectors) representing all sentences 1-by-1. Instead of bags of words (0/1) use **vectors of TF-IDF values** of words on i^{th} position.
4) Write ranking search engine that maximized **cosine similarity.**

NB: for details go to github

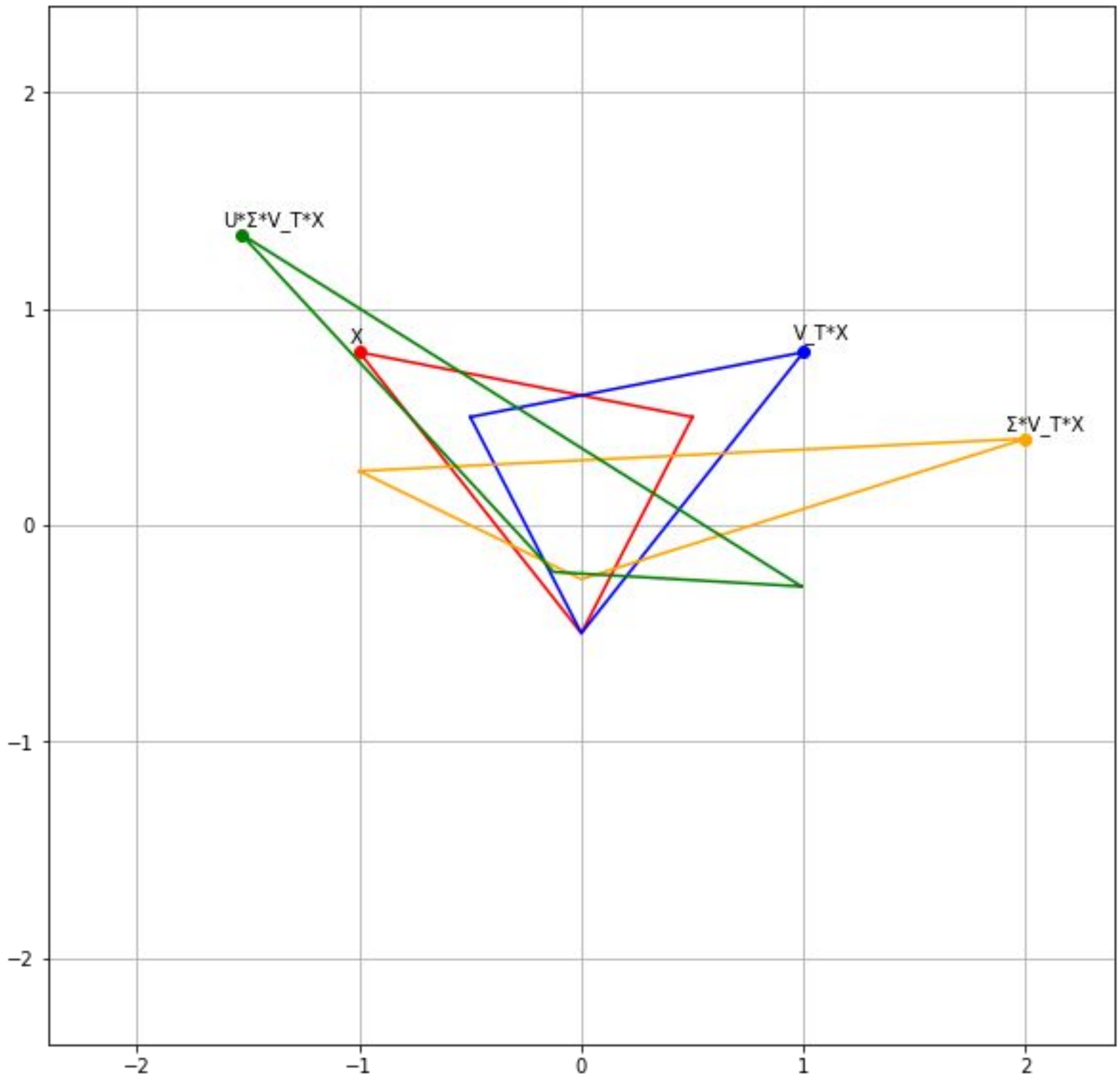
Enough for Friday!

Documents to vectors (LSA)

1) Build terms-document matrix
2) Reduce dimensions (LSA) preserving similarity measure
3) Profit!

Latent semantic analysis can be easily performed using **PCA** (principal component analysis) which can be performed using **SVD** (singular value decomposition) of terms-document matrix. Or other algorithm :)

$$\underset{m \times n}{\overset{X}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}}} = \underset{m \times r}{\overset{U}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}}} \underset{r \times r}{\overset{S}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}}} \underset{r \times n}{\overset{V^{\mathsf{T}}}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}}$$



U*Σ*V_T*X
X
V_T*X
Σ*V_T*X
2
1
0
−1
−2
−2
−1
0
1
2

Lab #4. Document to smaller vector

Study this example
https://github.com/str-anger/hsu.ai/blob/master/code/05.%20SVD%20and%20PCA%20magic.ipynb

Apply provided techniques to reduce number of dimensions in term-document matrix.

What do you need to run search engine?

Considering context: word2vec

CBoW (Continuous bag of words) - predict a word given a context

N-skip-grams - predict a context given a word

In both models word order doesn’t matter.

This models are trained in **reduced** space.

Word2vec is a tool to train such models.

Deep Structured Semantic Model (DSSM) - cooler version of semantic analysis from Microsoft.

There are also **sent2vec**, **text2vec**, ...

Homework #1: replace PCA in your search engine with doc2vec

PCA considers **text as a bag of words**. For short texts this works ok, but for longer texts it doesn’t catch the difference between “A killed B” and “B killed A”, although it encodes the fact of murder.

*2vec methods consider word appearances in relatively small surrounding, that brings order into context. Advances methods like DSSM also work with 3-trams.

Your hometask is to sum up results of today’s labs and build **search engine** powered with **doc2vec** technology.

Machine translation today

Companies move from distributional models to more accurate **semantic models**.

Semantics is **shared among languages**.

See example to try machine translation.

Hometask #2: Speak with AI in your language

Summarize experience of this week.

Ask questions in you **native language** and get answers from AI services.

1) Solve problems and answer requests with **Wolfram Alpha.**
2) If possible, **solve** problems **locally.**
3) Create **Q&A database** and search your previous questions and answers to avoid additional computations.