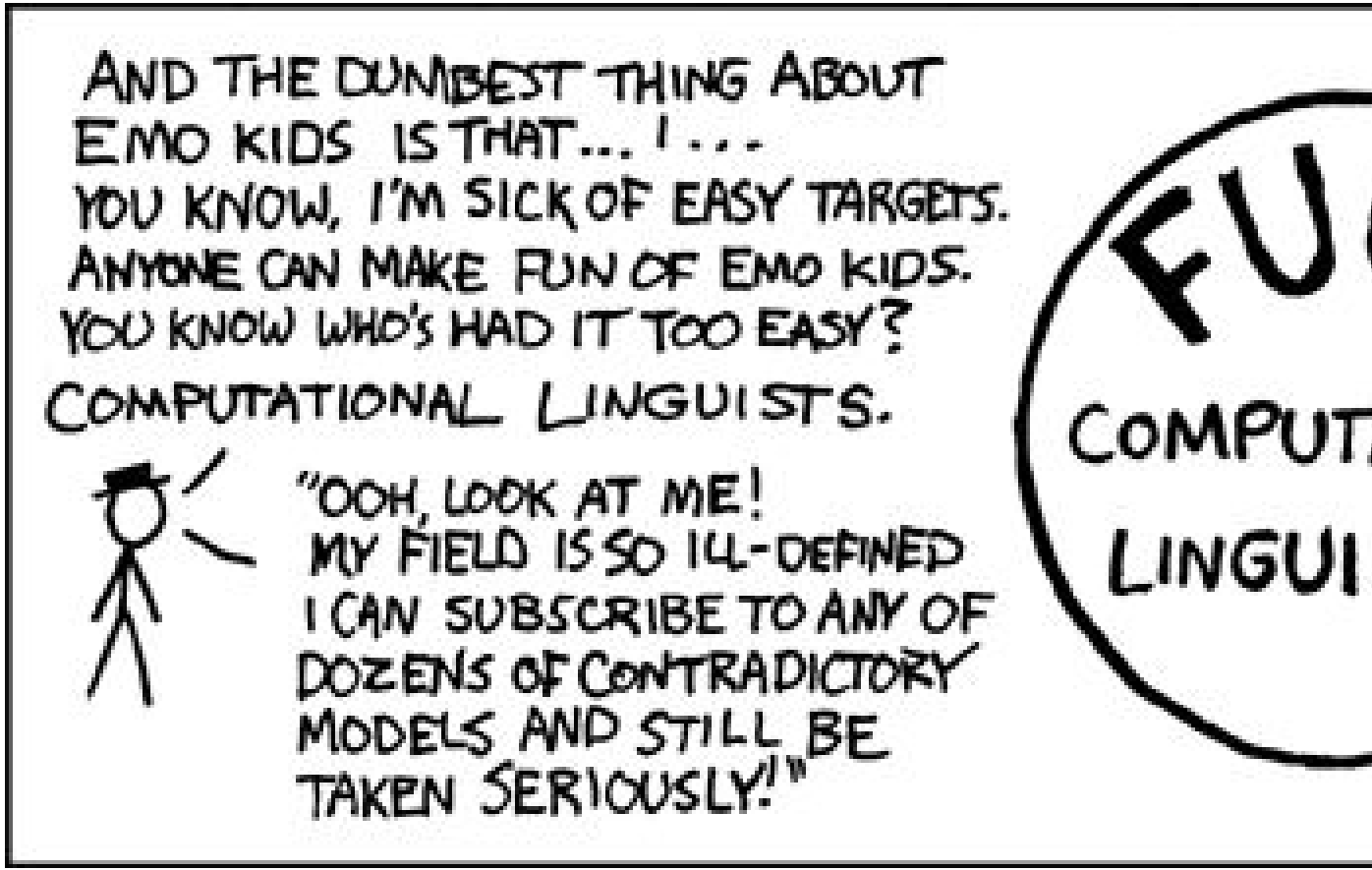
# Обработка естественного языка: введение

## The *Other* NLP*

Антон Алексеев
NativeRoll.TV, ПОМИ РАН
anton.m.alexeyev@gmail.com

* шутка на одном из стикеров DataFest'17


AND THE DUMBEST THING ABOUT EMO KIDS IS THAT... I... YOU KNOW, I'M SICK OF EASY TARGETS. ANYONE CAN MAKE FUN OF EMO KIDS. YOU KNOW WHO'S HAD IT TOO EASY? COMPUTATIONAL LINGUISTS.
"OOH, LOOK AT ME! MY FIELD IS SO ILL-DEFINED I CAN SUBSCRIBE TO ANY OF DOZENS OF CONTRADICTORY MODELS AND STILL BE TAKEN SERIOUSLY!"
FU
COMPUT
LINGUI

# What is NLProc?

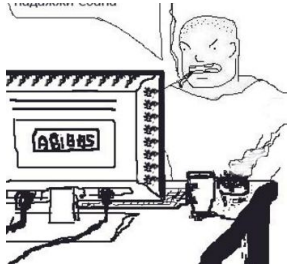
*DISCLAIMER: формулировки спорны, можно долго придираться*

**Computational Linguistics (CompLing)**

область лингвистики, в которой методы математики (и в частности, информатики) применяются для анализа и синтеза языка и речи; интересует **устройство и моделирование естественного языка**

**Natural Language Processing (NLP, NLProc)**

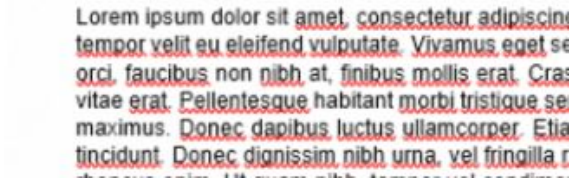
набор методов для обеспечения **машинной обработки** (анализ, синтез, ...) **текстов** на естественном языке

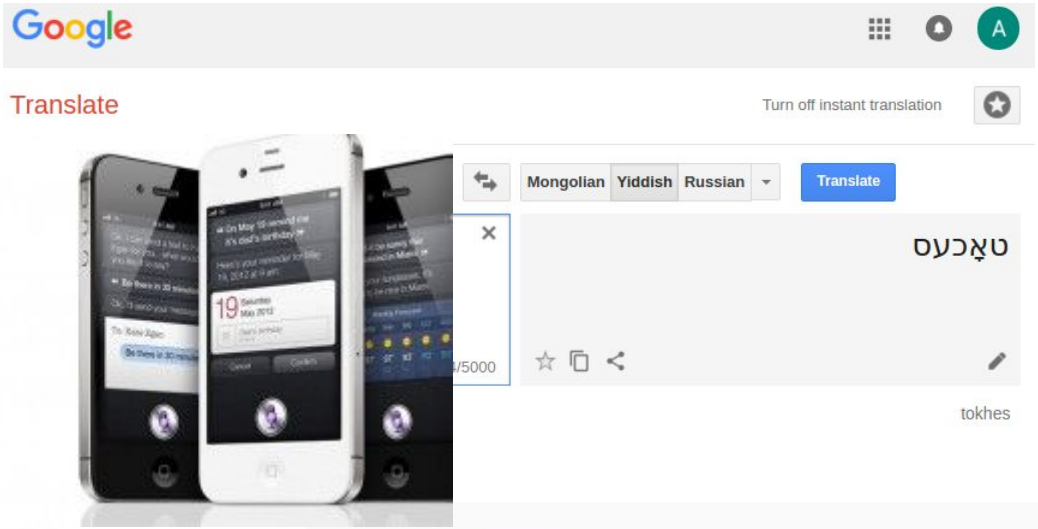


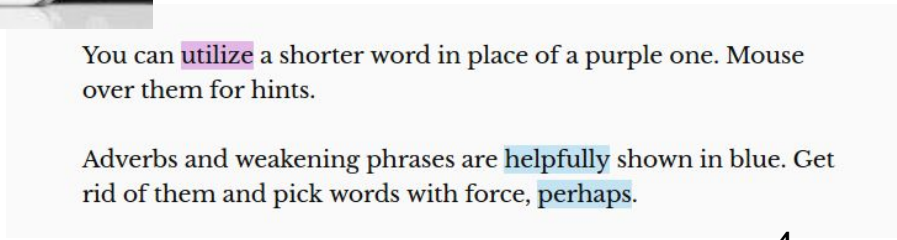
Пожалуйста, воздержитесь от шуток о том, что
*NLP = нейролингвистическое программирование*
Это глупо и всем надоело ^_^

# NLProc в нашей жизни

?

# Задачи NLP (неполный список)

Language modeling
Part-of-speech tagging
**Named entity recognition**
**Text (topic) classification**
**Keyword extraction**
**Spelling checking**
Syntax parsing
Dependency parsing
Machine translation
**Stemming**
**Lemmatization**
**Wikification**
**QA systems, dialogue systems**
Plagiarism detection

Morpheme analysis
Grammar check
Hyphenation
Relation extraction
Entity linking
**Sentiment analysis**
**Topic modeling**
Text summarization
Semantic role labeling
**Distributional semantics**
**Text generation**
**Text clustering**
***Information retrieval***
*Speech Recognition / Synthesis*

# Как будет построен рассказ

**Disclaimer 1**: никаких претензий на полноту изложения
**Disclaimer 2**: никаких претензий на уровень *Advanced*
**Цель**: рассказать, **что бывает**
**НО:** расстрою хипстеров — **без нейронных сетей**

- Конечно, всё будет с оглядкой на Statistical NLP
- От наиболее простых и важных с практической точки зрения — к более эзотерическим задачам
- 0-2 простых/классических/известных подходов на каждую задачу
- Вопросы можно задавать после любого слайда

Keyword extraction

Duplicate detection

Semantic role labeling

Machine translation

# План

1. Как превращать текст в числа
2. Задачи NLProc
    a. Классификация текстов
    b. Кластеризация текстов
    c. Тематическое моделирование
    d. Дистрибутивная семантика
    e. Поиск опечаток
    f. Выделение именованных сущностей
    g. Частеречная разметка
    h. Синтаксический разбор
    i. Машинный перевод
3. Книги, курсы, инструменты

# Как превращать текст в числа [0]

- Текст = String
- ...а мы умеем работать только с числами:
  векторами и иногда последовательностями векторов

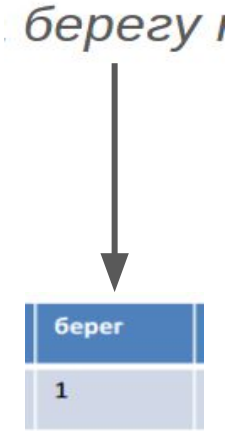
**Способ #1, Bag-of-words: one hot (BOW; мешок слов)**
~ one-hot-encoding / dummy coding: много интерпретируемых фич
“*На берегу пустынных волн...*”

| абакан | абырвалг | азкабан | аксакал | аксолоть | аксон | берег | волна | заноза | ... |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | ... |

**Bag-of-words: word counts** (sklearn: CountVectorizer)
вместо единиц — частоты / относительные частоты

# Отступление: нормализация

Вы заметили? Это разные формы слова, и умение приводить к нормальной — отдельная задача




- **Стемминг**: “отрезаем” от слов куски (псевдоокончания, псевдосуффиксы) так, чтобы “котик” и “кот” машина восприняла как одно слово

- **Лемматизация**: приводим слова к нормальной форме

BTW: “берегу” может быть и глаголом!

# Стеммер Портера

все рассказывают, и я расскажу

- Варианты для нескольких языков
- Последовательно применяя ряд правил, отсекает окончания и суффиксы
- Работает быстро, **далеко не всегда** безошибочно


C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. New models in probabilistic information retrieval. London: British Library. (British Library Research and Development Report, no. 5587).

# Как превращать текст в числа [1]

Не все слова одинаково полезны!
Примеры: служебные части речи, слишком редкие слова...

**Bag-of-words: weird numbers (sklearn: TfIdfVectorizer)**
вместо единиц — TF-IDF или *другие оценки “значимости”*
терма **t** для данного документа **d** из данной коллекции **D**

**df** - в скольких документах встречается терм
**tf** - сколько раз терм встречается в данном документе

Однако есть варианты!

**Variants of term frequency (TF) weight**

| weighting scheme | TF weight |
|---|---|
| binary | $0, 1$ |
| raw count | $f_{t,d}$ |
| term frequency | $f_{t,d} / \sum_{t' \in d} f_{t',d}$ |
| log normalization | $1 + \log(f_{t,d})$ |
| double normalization 0.5 | $0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$ |
| double normalization K | $K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$ |

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

# TF-IDF: term frequency - inverse document frequency

## TF-IDF$(q, d)$ — мера релевантности документа $d$ запросу $q$

$n_{dw}$ (term frequency) — число вхождений слова $w$ в текст $d$;
$N_w$ (document frequency) — число документов, содержащих $w$;
$N$ — число документов в коллекции $D$;

$N_w/N$ — оценка вероятности встретить слово $w$ в документе;
$(N_w/N)^{n_{dw}}$ — оценка вероятности встретить его $n_{dw}$ раз;
$P(q,d) = \prod_{w \in q} (N_w/N)^{n_{dw}}$ — оценка вероятности встретить в документе $d$ слова запроса $q = \{w_1, \ldots, w_k\}$ *чисто случайно*;

Оценка релевантности запроса $q$ документу $d$:

$$-\log P(q,d) = \sum_{w \in q} \underbrace{n_{dw}}_{\mathrm{TF}(w,d)} \underbrace{\log(N/N_w)}_{\mathrm{IDF}(w)} \to \max.$$

$\mathrm{TF}(w,d) = n_{dw}$ — term frequency;
$\mathrm{IDF}(w) = \log(N/N_w)$ — inverted document frequency.

# BM is for Best Match

Нельзя не упомянуть **Okapi BM25** — семейство idf-подобных функций, которые долго жили в продакшеновых поисковиках

Q — запрос
D — документ
avgdl — средняя длина документа

Остальное — свободные параметры

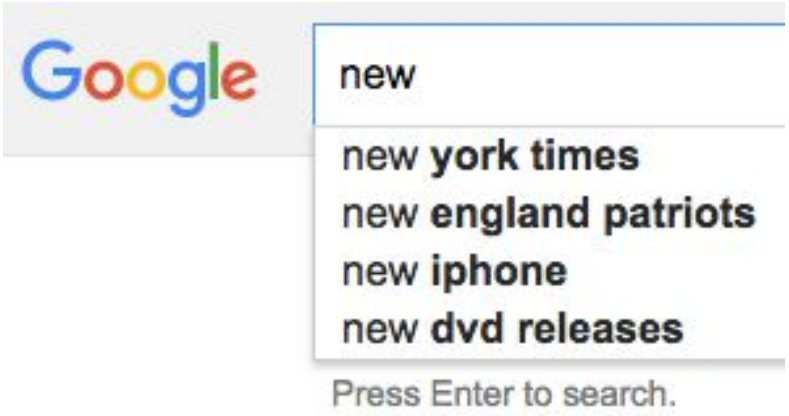
$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})},$$

please see https://en.wikipedia.org/wiki/Okapi_BM25

# Как превращать текст в числа [2]

Очевидно, с порядком слов в тексте мы теряем много информации (см. рис.), но есть средство для бедных!

**Bag-of-ngrams (sklearn vectorizers поддерживают)**
вместо отдельных термов наборы из n подряд идущих в тексте термов.



# Как превращать текст в числа [3]

https://twitter.com/stanfordnlp/status/399551909595344896

**Итоги** по **bag of words:**
много разреженных фич, можем столкнуться с проблемами больших размерностей, поэтому:

- учимся **фильтровать и наказывать термы весами**;
  частотные, редкие и т. д.- это есть из коробки есть в sklearn;
  кроме того — фильтрация по словарям
  (в т. ч. **stopwords**: словарь “вредных слов”)

- выбираем модели для работы **с большим числом разреженных признаков**,

- обязательно экспериментируем **с числом N в ngram-max** и вариациями one-hot/count/tf-idf/...

# Как превращать текст в числа [3*]

**Итоги** по **bag of words:**
много разреженных фич, можем столкнуться с проблемами больших размерностей, поэтому:

- учимся **фильтровать и наказывать термы весами**;
  частотные, редкие и т. д.- это есть из коробки есть в sklearn;
  кроме того — фильтрация по словарям
  (в т. ч. **stopwords**: словарь “бессмысленных слов”)

- выбираем модели для работы **с большим числом разреженных признаков**,

- обязательно экспериментируем **с числом N в ngram-max** и вариациями one-hot/count/tf-idf/...

https://twitter.com/stanfordnlp/status/399551909595344896

# Как превращать текст в числа [4]

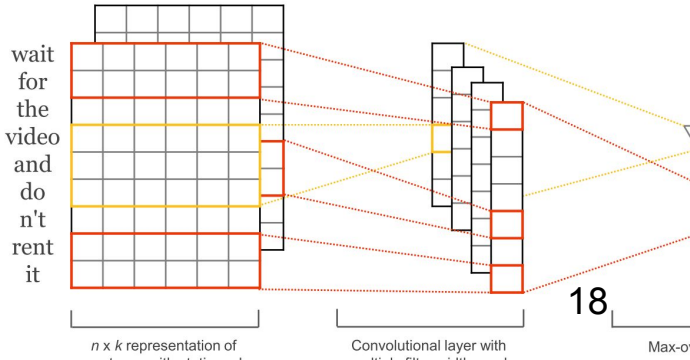
**SAY word2vec ONE MORE TIME!**

# Как превращать текст в числа [5]

- Слову можно сопоставить вектор, в который как-то будет зашит его “смысл”, мысль не нова (1950е)
- Гипотеза: некоторые “семантические свойства” слова можно получить из “контекстов”, в которых оно употребляется в достаточно большом и репрезентативном корпусе
- Этим занимается дистрибутивная семантика

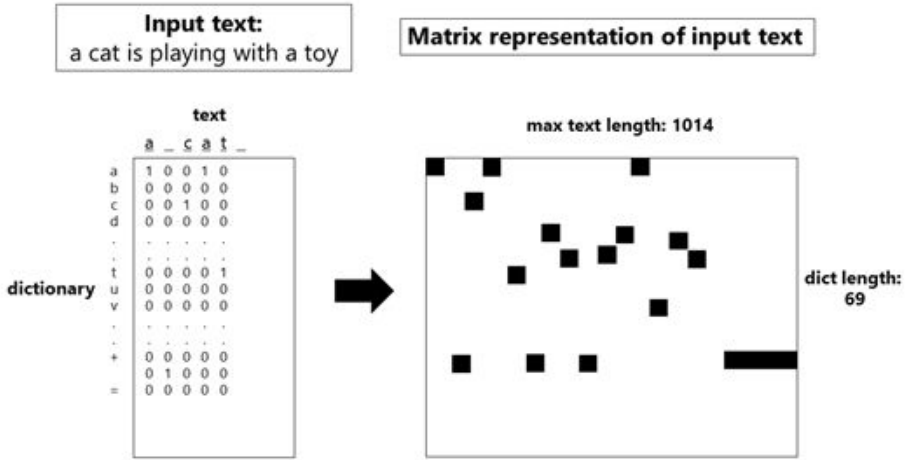
**Способ #2** взять word vectors (напр., word2vec) всех слов короткого текста - и

- сложить?
- сложить с весами, пропорциональными TF-IDF?
- *“склеить” в матрицу и подать на вход RNN*
- *“склеить” в матрицу, обрезать и подать на вход CNN*
- ваш вариант?



# Как превращать текст в числа [6]

Стало модно представлять текст на “подсловном” уровне —
как последовательность символов, морфем или иных языковых единиц
уровнем “ниже” слова




e.g. see: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

# Классификация текстов

Формулировка задачи классификации — стандартная:
даны объекты (тексты) с проставленными классами (метками, категориями)

$$X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

требуется построить функцию, которая сможет верно классифицировать произвольный объект

$$a: X \to Y$$

# Классификация текстов, дефолтный подход

Набросок:

1. разбили текст на токены, привели слова в нормальную форму
2. построили векторы с tf-idf bag-of-ngrams
3. разбили на обучение, контроль и валидацию
4. подобрали лучшие параметры для предобработки и для вашей любимой модели
   (не любимой, а той, которая работает с редкими фичами)
5. ???????
6. PROFIT!!!


Да, так можно, так и делаю :)

# Классификация текстов, naive Bayes [0]

Хотим оценить

$$p(C_k \mid x_1, \ldots, x_n)$$

Из определения условной вероятности

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

Chain rule

$$\begin{aligned} p(C_k, x_1, \ldots, x_n) &= p(x_1, \ldots, x_n, C_k) \\ &= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2, \ldots, x_n, C_k) \\ &= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2 \mid x_3, \ldots, x_n, C_k) p(x_3, \ldots, x_n, C_k) \\ &= \ldots \\ &= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2 \mid x_3, \ldots, x_n, C_k) \ldots p(x_{n-1} \mid x_n, C_k) p(x_n \mid C_k) p(C_k) \end{aligned}$$

А теперь допустим, что все слова условно независимы

$$\begin{aligned} p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) \\ &\propto p(C_k)\, p(x_1 \mid C_k)\, p(x_2 \mid C_k)\, p(x_3 \mid C_k) \cdots \\ &\propto p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,. \end{aligned}$$

# Классификация текстов, “наивно” [1]

Условные и априорные вероятности оцениваем как частоты в обучающем множестве, и вуаля

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

Удивительно, но иногда неплохо работало!

Tips:

- умножение на ноль — это сильно; надо сглаживать!
- иногда лучше складывать логарифмы, чем перемножать сами вероятности

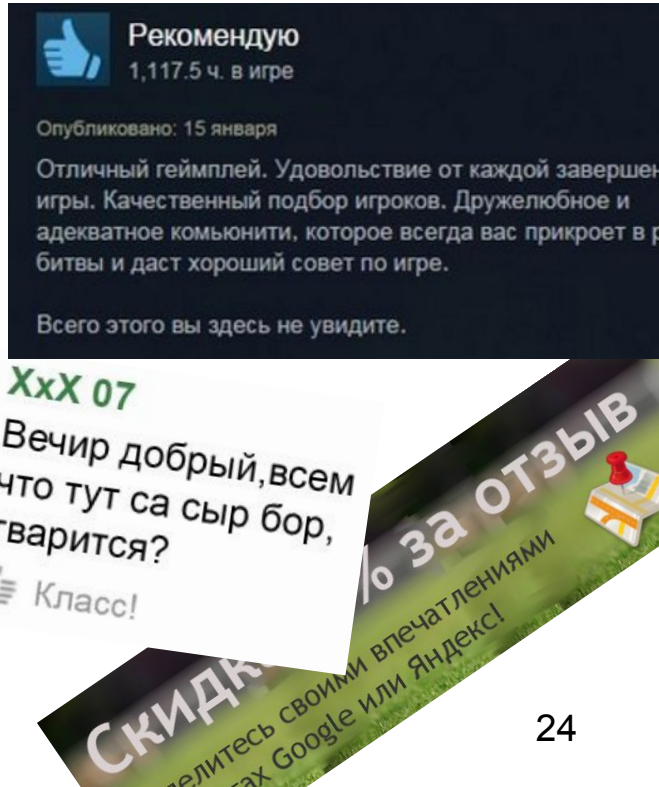
# Sentiment analysis (анализ тональности текста)

- Всего лишь один из вариантов задач классификации текстов?
- Да, но есть несколько трюков и интересных подходов
    - “словарь положительных слов”, собрав твиты с радостными смайликами (аналогично с “отрицательной окраской”)
    - понять, “положительное” словосочетание или нет, сделав запрос в altavista вместе со словами Excellent и Poor; у кого больше результатов в выдаче, тот и победил

Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews

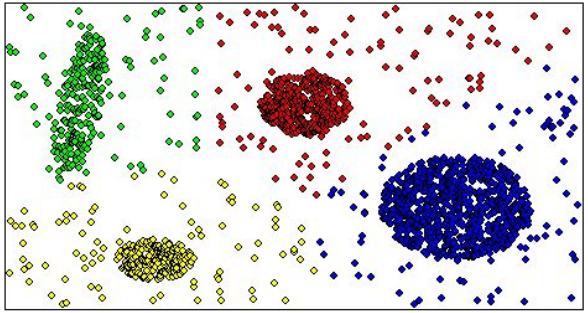
// Proceeding ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics Pages 417-424

- А ведь ещё есть сарказм!
- Словом, специфика очень важна



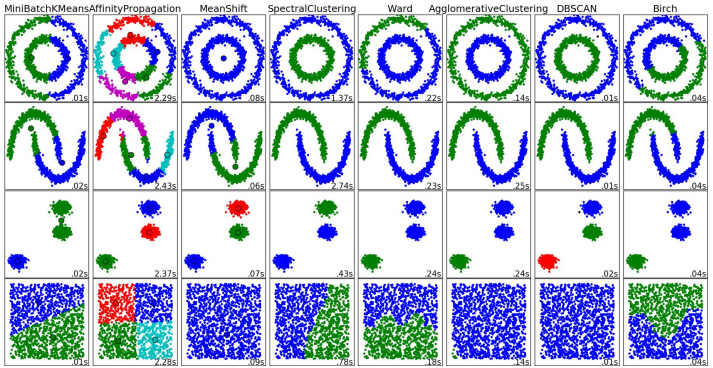
# Кластеризация текстов (~ тематики?) [0]

**Грубо и спорно**: дано множество объектов, мы умеем считать расстояния между ними, и требуется разбить его (чётко/нечётко) на подмножества-кластеры так, чтобы ***похожие*** объекты оказались в одном кластере, а ***непохожие*** — в разных.



**Пример**: Яндекс.Новости, Quora contest :)

- **методы** бывают разные
- **метрики** бывают разные

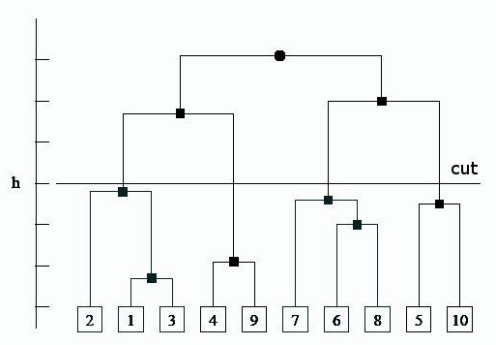



*import-sklearn*-подход тоже сойдёт

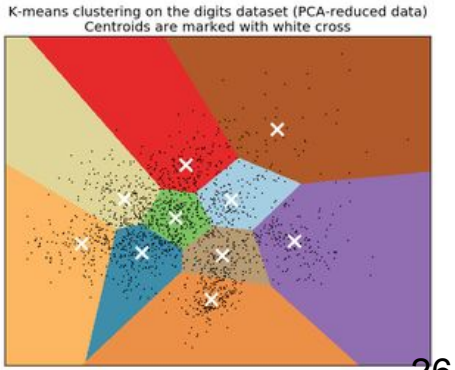
Все видели эту картинку?

# Кластеризация текстов (~ тематики?)[1]

- Пример подхода для работы с bag-of-words
  a. Посчитать матрицу косинусных  расстояний между соответствующими текстами
  b. Каждый текст изначально сам себе кластер
  c. Последовательно сливаем наиболее близко расположенные друг к другу кластеры
  d. Построенное дерево разрезаем, как нам нужно




- Если у нас **dense** vectors (неразреженные) — скажем, word2vec-и, можно попробовать и **KMeans**; иногда работает

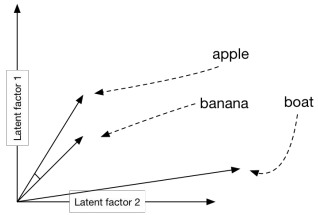


# Отступление: важные расстояния

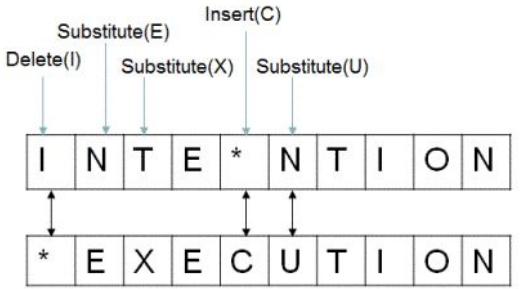
- Для векторов

  wiki: similarity and distance measures
  - Cosine distance
  - Euclidean distance
  - Euclidean distance ‘generalization’: Minkowski distance
  - Manhattan distance
  - ...

$$sim(U_i, U_j) = \cos(U_i, U_j) = \frac{U_i \cdot U_j}{|U_i| \cdot |U_j|} = \frac{\sum_{k=1}^{n} u_{ik} \times u_{jk}}{\sqrt{\sum_{k=1}^{n} u_{ik}^2} \sqrt{\sum_{k=1}^{n} u_{jk}^2}}$$




- Для строк
  - расстояние Левенштейна
  - Longest common subsequence
  - Hamming distance
  - Jaro-Winkler distance
  - ...

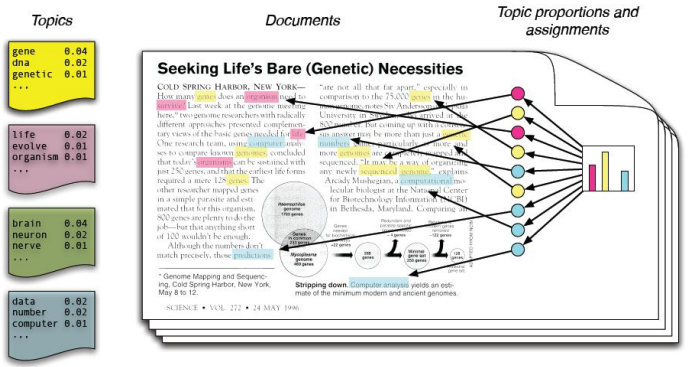


# Topic modeling (тематическое моделирование)[0]

- Есть неразмеченная коллекция текстов, верим, что в ней K тематик (это необязательно, но пусть так)

- Хотим “выделить” эти тематики
  - Для каждого текста — степень, с которой он удовлетворяет каждой тематике
  - Для каждой тематики — степень важности каждого терма для неё

(тоже довольно спорная постановка задачи)



# Topic modeling (тематическое моделирование)[1]

**Зачем нужно?**

- Много неразмеченных текстов — частая ситуация :)
  “Разведочный поиск”: понять о чём и как устроена коллекция текстов
  Например, чтобы принять решение, какие классы ввести для последующего обучения классификатора
- Это тоже word vectors! И обычно их можно интерпретировать!

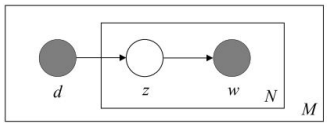
| Topic 1 | | Topic 2 | | Topic 3 | |
|---|---|---|---|---|---|
| **term** | **weight** | **term** | **weight** | **term** | **weight** |
| game | 0.014 | space | 0.021 | drive | 0.021 |
| team | 0.011 | nasa | 0.006 | card | 0.015 |
| hockey | 0.009 | earth | 0.006 | system | 0.013 |
| play | 0.008 | henry | 0.005 | scsi | 0.012 |
| games | 0.007 | launch | 0.004 | hard | 0.011 |

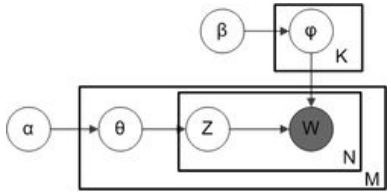
# Topic modeling (тематическое моделирование)[2]

- Вероятностный подход: относимся к тексту как к выводу генеративного процесса
  - для каждого документа сэмплируем тематику из распределения тематик
  - затем сэмплируем слово из распределения слов в тематике

- Задача **обучения**: подобрать параметры распределений так, чтобы максимизировать likelihood генерации заданной коллекции

- pLSA, LDA, Tag-LDA, LDA-*, ARTM, Pachinko allocation … - много их



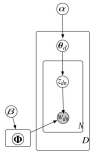



Figure 1: LDA model.

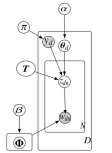


Figure 2: DiscLDA.

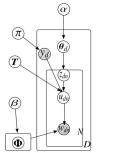
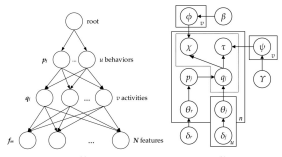
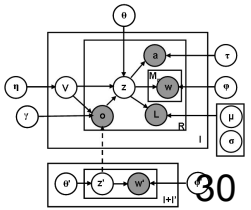


Figure 3: DiscLDA with auxiliary variable $u$.



# Topic modeling: pLSA [3]

**pLSA: T** — тематики, **W** — термы, **D** — документы

**Допущение**: генерация слова зависит только от темы, но не от конкретного документа

$$p(w \mid d, t) = p(w \mid t)$$

Если расписать p(w, d), получим

$$p(w \mid d) = \sum_{t \in T} p(t \mid d)\, p(w \mid t)$$

Выходит, что это разложение нормализованной term-document-матрицы **D x T**

в произведение стохастических матриц **P(w|t)** и **P(t|d)**!

Есть свои проблемы, есть свои методы, есть способы регуляризации…

В общем, goto K. Vorontsov за интуицией и EM-алгоритмом

# Дистрибутивная семантика (distributional semantics) [0]

**Distributional Hypothesis** [Firth, 1957]: words with similar distributional properties (i.e. that co-occur regularly) tend to share some aspect of semantic meaning

- Не word2vec-ом единым! Например, pLSA  (как и LSA, LDA, etc.) тоже даёт векторное представление слова (как распределения над темами)
- Простой старый вариант: HAL (hyperspace analogue to language)
  C. Burgess, K. Livesay, K. Lund. Explorations in context space: Words, sentences, discourse. *Discourse Processes 25 (2-3), 211-257*

Идём по тексту с “окном” ***K*** и наполняем трёхмерный массив **n**

The HAL weighting for a term $t$ and any other term $t'$ is given by:

$$HAL(t'|t) = \sum_{k=1}^{K} w(k) n(t, k, t') \qquad (1)$$

$$w(k) = K - k + 1$$

Тогда HAL(t’|t) задаёт что-то вроде “семантической близости”.

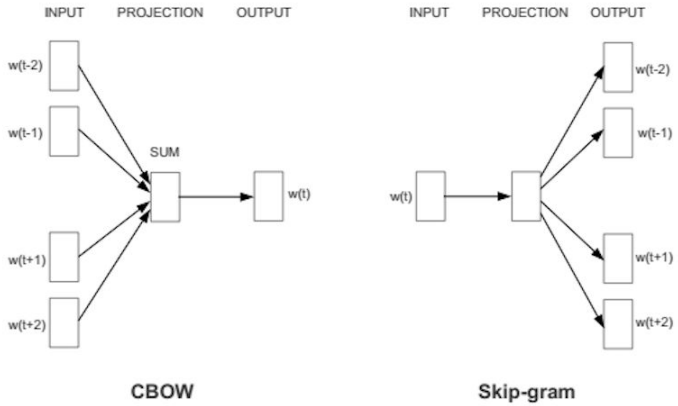
Ещё есть pHAL

# Дистрибутивная семантика (distributional semantics) [1]

word2vec немного похож, но строит другое пространство и “работает лучше”

Это семейство алгоритмов (2-слойных нейронных сетей):

- по контексту восстанавливаем слово
- по слову восстанавливаем контекст

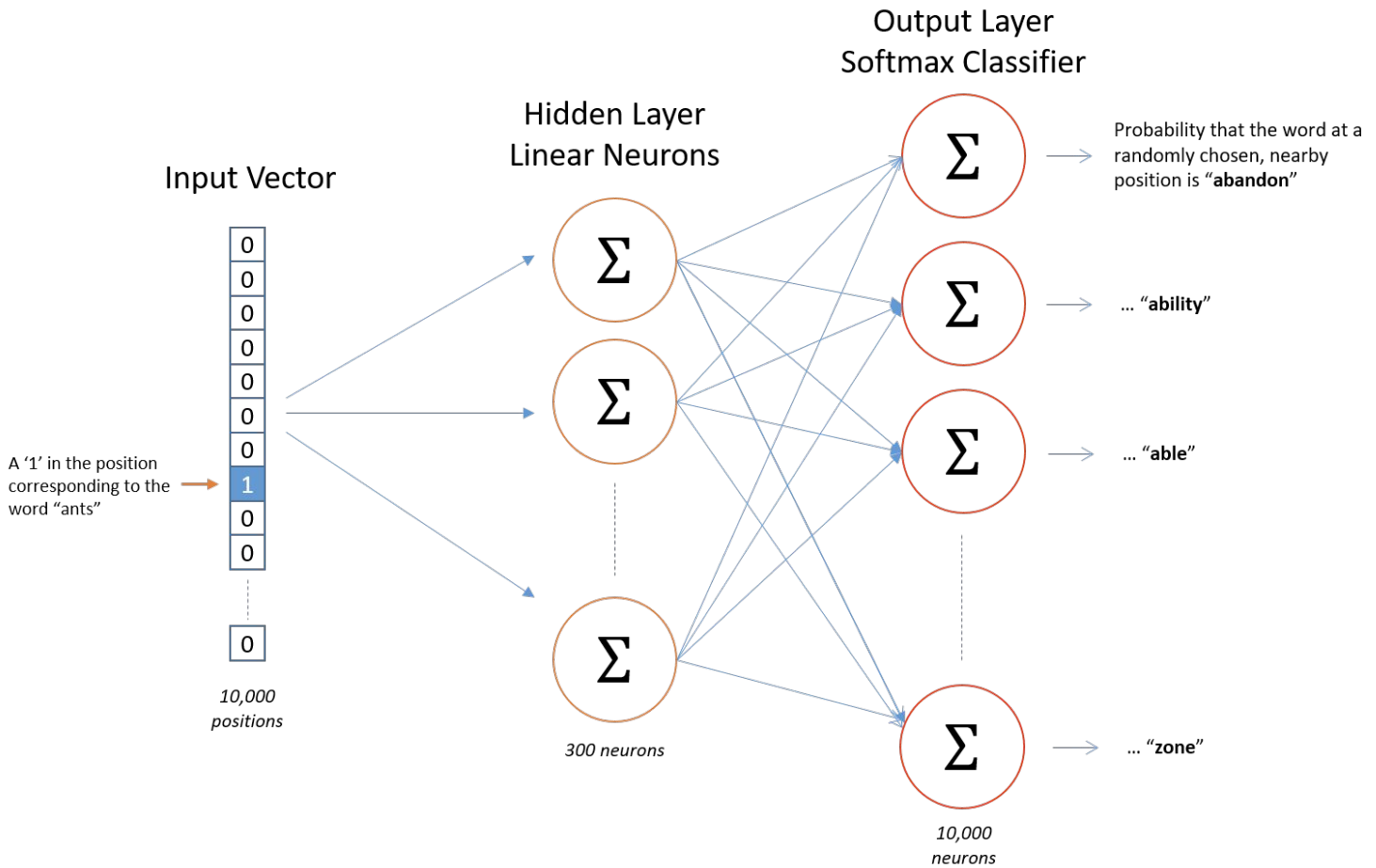



Контекст — окно соседних слов без учёта их порядка.

Секрет успеха word2vec в том числе в трюках, которые используются при обучении:

- **Negative sampling**: обновляем веса далеко не у всех “неправильных контекстов”
- **Sub-sampling** слов с высокой частотой встречаемости (чем выше частота, тем больше шансы, что выкинем)

Много хороших релевантных ссылок на статьи и разборы можно найти на вики: https://en.wikipedia.org/wiki/Word2vec Не надо стесняться.

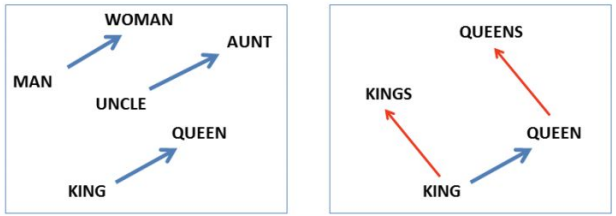



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

# Дистрибутивная семантика (distributional semantics) [2]

Так а что все так шумят?

- действительно хорошие представления (в смысле **семантической близости** и **аналогии**)
- “смысловая арифметика”, которой специально векторы никто не учил (однако есть свои “но”)
- успешно используют как “входные данные” для других алгоритмов машинного обучения




(Mikolov et al., NAACL HLT, 2013)

# Дистрибутивная семантика (distributional semantics) [3]

**все *ахнули*, когда узнали, что такое на самом деле word2vec!!!**

Рекомендуемый вариант — skip-gram with negative sampling…

> We later found out that SGNS is implicitly factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant:
> Neural Word Embeddings as Implicit Matrix Factorization

$$\mathrm{pmi}(x;y) \equiv \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(y|x)}{p(y)}.$$

То есть — **ничего нового**! Просто работает очень хорошо.

https://levyomer.wordpress.com/2014/09/10/neural-word-embeddings-as-implicit-matrix-factorization/

# Дистрибутивная семантика (distributional semantics) [4]

Так всё-таки — что когда использовать?

**T. Mikolov**

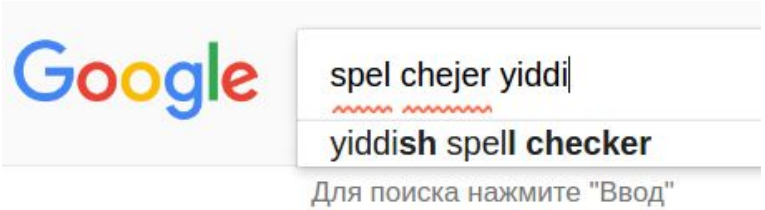
**Skip-gram:** works well with small amount of the training data, represents well even rare words or phrases.

**CBOW:** several times faster to train than the skip-gram, slightly better accuracy for the frequent words

# Spellchecker (поиск опечаток и ошибок) [0]

- Есть текст, надо
  - проверить, нет ли опечаток/ошибок
  - исправить их, если возможно

**Вариант 1 (quick-and-dirty)**
отсутствие слова в словаре и расстояние Левенштейна (edit distance) до ближайшего слова в словаре




Also see: Peter Norvig’s half-page of code
http://norvig.com/spell-correct.html

# Spellchecker (поиск опечаток и ошибок) [1]

**Вариант 2: context-aware spell-checking**

1. Обнаружили слово, которого **нет в словаре**
2. Находим в корпусе слова, которые встречаются в **окружении таких же слов** (размер окна нужно подобрать)
3. Найденные сортируем по любимому расстоянию для слов-как-последовательностей-символов;
   например, **edit distance**, n-gram Jaccard distance
4. Берём **топ**, если нам нравится расстояние (thresholding).

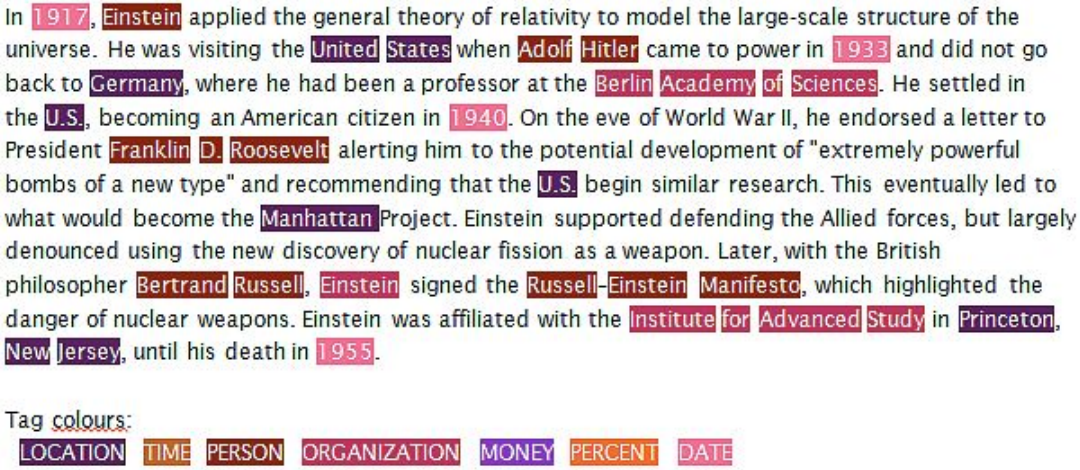

Ссылку на статью никак не могу найти :(

...А ещё можно пользоваться сторонними сервисами и тулами, у которых большие словари и собрано много статистики по ошибкам

# Named entity recognition, NER (выделение именованных сущностей)

На картинке указаны типы именованных сущностей — они зашиты в метки.

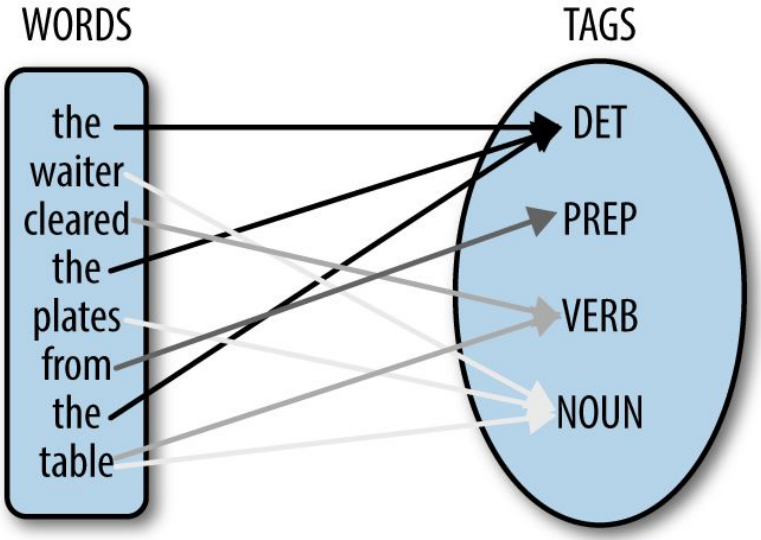
Видов разметки много.

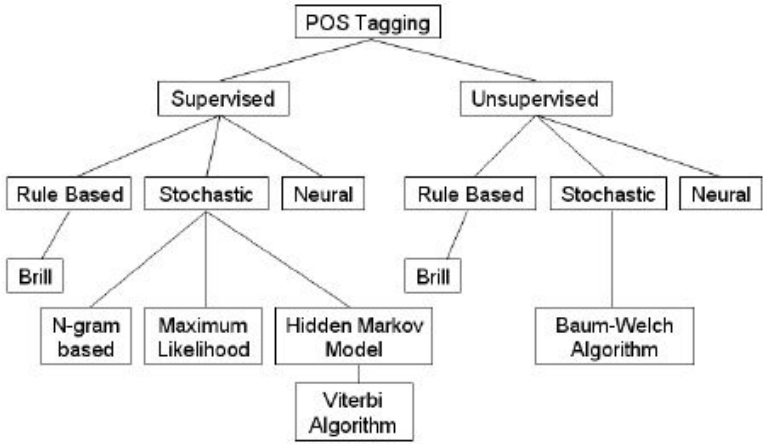



Для сущностей из нескольких слов отмечаем первое слово отдельной меткой:

Кто/**O** такой/**O** Андрей/**B_NAME** Александрович/**I_NAME** Фильченков/**I_NAME**?

# Part-of-Speech (POS) tagging (частеречная разметка)






Img src: https://www.researchgate.net/publication/228947716_Comparison_of_unigram_bigram_hmm_and_brill%27s_pos_tagging_approaches_for_some_south_asian_languages

# POS tagging, NER

- Зачем нужен NER — объяснять не надо :)

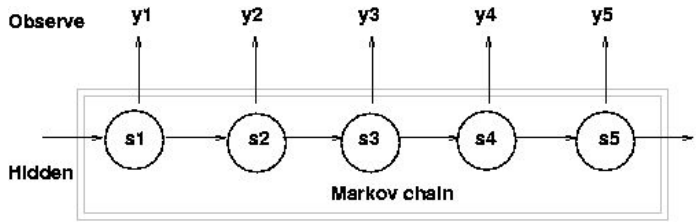
- POS tagging помогает, например, сглаживать фичи (слово можно заменить на метку), выделять определённые части речи (полезно для sentiment analysis и вообще фильтрации слов), ...

- Как решать, что и на чём обучать?
  - Эвристические подходы
  - Модные нейронные сети
  - Структурные модели машинного обучения: HMM, StructPerceptron, ...

# Hidden Markov Models [0]

- **Даны**: последовательности слов с соответствующими метками
- **Модель**: состояния = метки, наблюдения = слов

  Вектор вероятностей начальных состояний и 2 матрицы:
  - вероятности перехода из состояния в состояние p(s_i|s_i-1)
  - вероятности генерации наблюдения p(o_i|s_i)
- **Задачи**
  - оценить вероятность последовательности наблюдений
  - восстановить **наиболее вероятную последовательность состояний** (меток!)




L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257-286, Feb 1989.

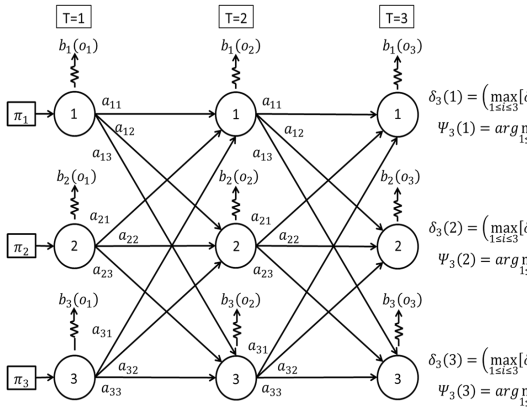
# Hidden Markov Models [1]

## Алгоритм Витерби: динамический, идея

Даны наблюдения **o_1,...,o_N** состояния **s_1,...,s_N**

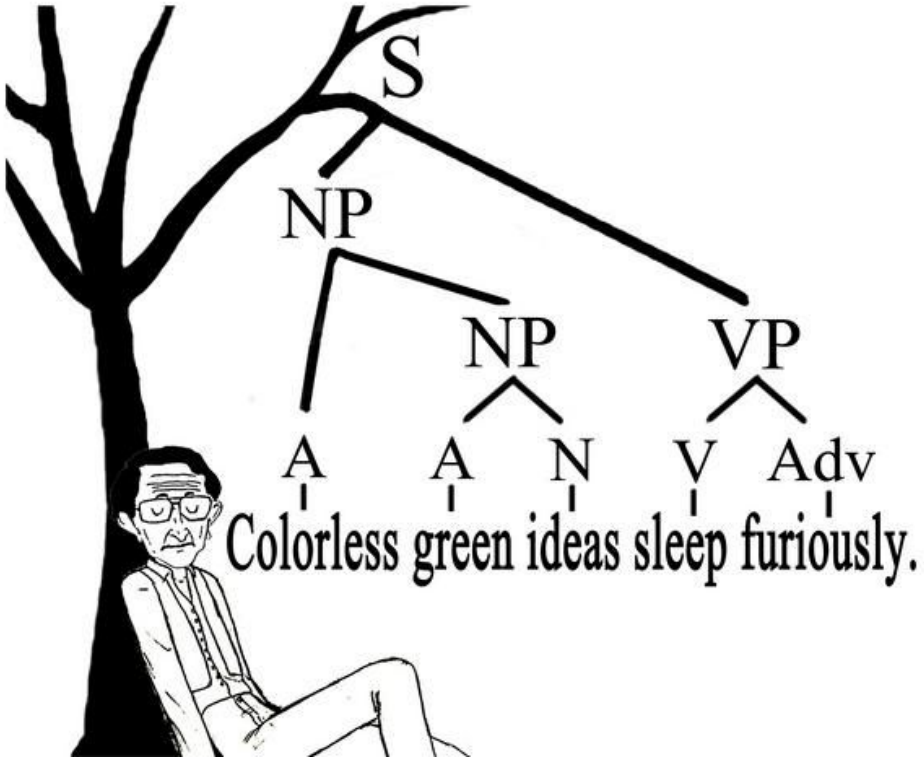
1. Идём последовательно по “времени” — от первых шагов к последним
2. На t-м шаге ищем самую большую вероятность оказаться в состоянии s_t (среди всех цепочек — но тут мы жадно смотрим на предыдущий шаг). Пишем в матрицу M.
3. Также в отдельный массив будем сохранять указатели на состояния на предыдущем шаге, на которых был достигнут максимум

$$M_{state,t} = p(o_t|state) * \max_{k \in S} p(state|k) * M_{k,t-1}$$




Когда матрицы будет заполнены, мы сможем восстановить по указателям **наиболее вероятную последовательность** состояний!

# Syntax parsing (синтаксический разбор)[0]



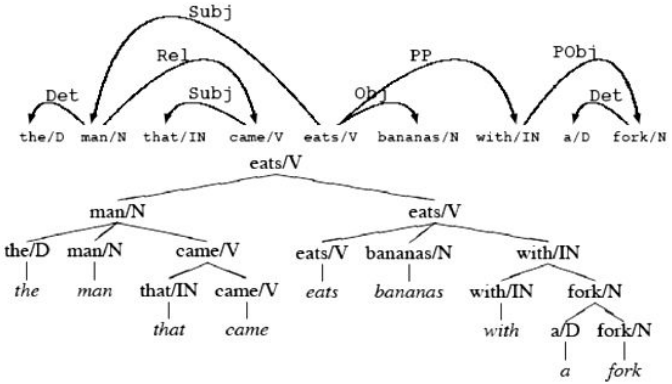
# Syntax parsing (синтаксический разбор)[1]

Язык - это словарь и **грамматика**, то есть правила, по которым мы отделяем “правильный” текст от “неправильного”

Есть несколько подходов, согласно которым “связи” между словами должны образовывать определённые структуры

- грамматика с **фразовой структурой**
  (PSG; Хомский; см. картинку на пред. слайде)
  ~ грамматика непосредственных составляющих

- грамматика зависимостей (DG)
  строй предложения в виде **иерархии** слов
  (как в школе стрелочками показывали, что от чего зависит)
  похожая вещь - link grammar, но там может не быть направления зависимости




https://bop.unibe.ch/linguistik-online/article/view/789/1357

Есть несколько форматов и видов разметки.

# Syntax parsing (синтаксический разбор)[2]

Как парсить? Говорят, не всегда достаточно КС-грамматик, но для примера —

Алгоритм **Кока-Янгера-Касами** (CYK):

- КС-грамматика размера **G**
  (продукции д. б.даны в нормальной форме Хомского:
  **слева** только один литерал, **справа** не более двух)
- текст длины **n**

Тогда разбирает строку для заданной PSG за **O(n^3 * G)**

# Syntax parsing (синтаксический разбор)[3]

Пример из Википедии

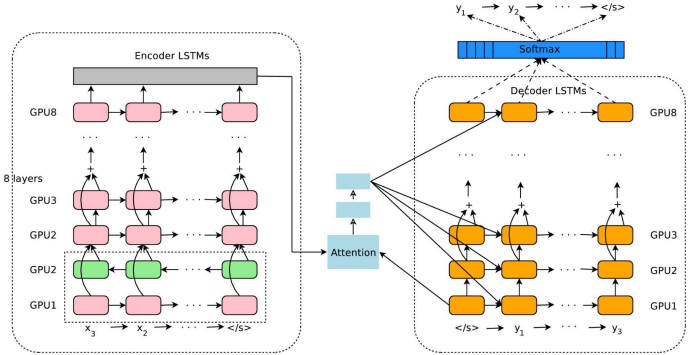
$$
\begin{aligned}
S &\rightarrow NP\ VP \\
VP &\rightarrow VP\ PP \\
VP &\rightarrow V\ NP \\
VP &\rightarrow eats \\
PP &\rightarrow P\ NP \\
NP &\rightarrow Det\ N \\
NP &\rightarrow she \\
V &\rightarrow eats \\
P &\rightarrow with \\
N &\rightarrow fish \\
N &\rightarrow fork \\
Det &\rightarrow a
\end{aligned}
$$

**CYK table**

| | | | | | | |
|---|---|---|---|---|---|---|
| **S** | | | | | | |
| | VP | | | | | |
| | | | | | | |
| **S** | | | | | | |
| | VP | | | PP | | |
| **S** | | NP | | | NP | |
| NP | V, VP | Det. | N | P | Det | N |
| she | eats | a | fish | with | a | fork |

# Машинный перевод

- С этой задачи началась компьютерная лингвистика, и эта задача остаётся по прежнему одной из самых сложных
- Много подходов — в том числе нестатистические
- Для статистических — нужен параллельный корпус
- Очень круты в последнее время нейронные сети, гугл вот выкатил в продакшен уже давно



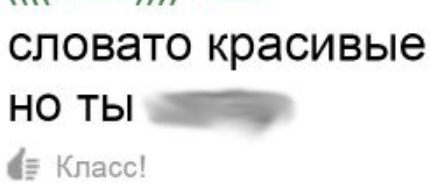

Рассказывать, как делается статистический машинный перевод, *я, конечно, не буду*; но всё не так сложно, как может казаться!

Чтобы в этом убедиться, стоит начать с IBM alignment models 1 и 2

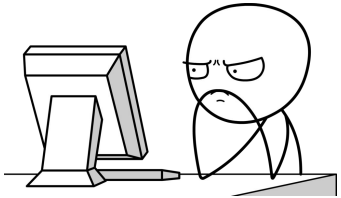
# to be continued…

можно было бы поболтать о диалоговых системах, привести несколько сгенерированных “чисто символьной” рекуррентной нейронной сетью “порошков”, поговорить о сексизме word2vec-а, спеть песни “Нейронной обороны” и зачитать перлы гоп-стоп-бота... ~~но тут Шахразаду застигло утро, и она прекратила дозволенные речи~~

# FINAL REMARKS




- На деле часто мало что работает так, как вам нужно, “из коробки”; стоит проверять глазами и для практических целей придумывать эвристики — *не надо стесняться*!



- Особенность: в проде бывают нужны *очень специфичные* данные и трюки по работе с *малыми/грязными* данными (**crowdsourcing** rules!)

- Очень многое зависит от источника и размера текста

- Осторожнее с лицензиями!
  А когда кегглите — с любыми внешними данными

# Как научиться, где искать [0]

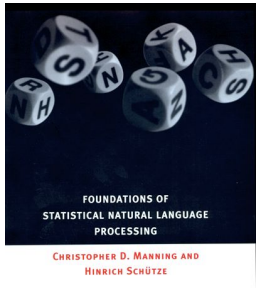
1. **Speech and Language Processing**,
   by Daniel Jurafsky, James H. Martin

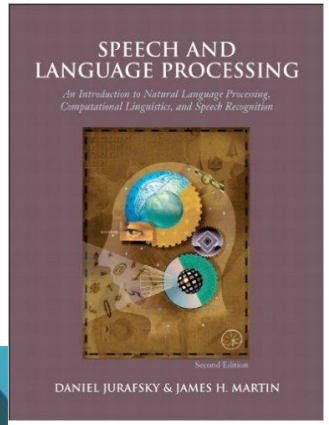
2. **Foundations of Statistical Natural Language Processing**,
   by Chris Manning, Hinrich Schütze

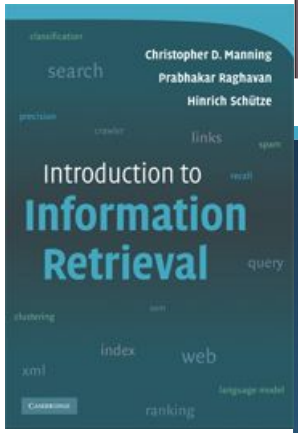
3. **Introduction into Information Retrieval**
   by Manning, Schutze, Raghavan - FREE (and lovely)! + перевод “от Яндекса”
   https://nlp.stanford.edu/IR-book/

4. **A Primer on Neural Network Models for NLProc** by Yoav Goldberg
   http://u.cs.biu.ac.il/~yogo/nnlp.pdf

5. **Есть и другие книжки**, *тысячи их*:
   книжка об **NLTK**, NLP на Хаскеле etc.




# Как научиться, где искать [1]

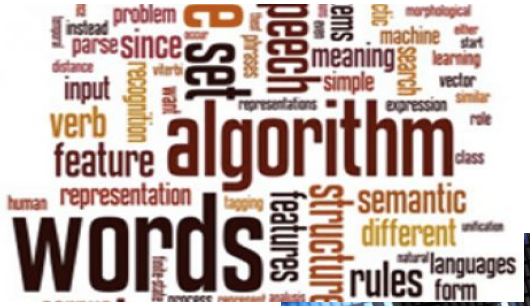
1. **Онлайн-курсы** -- google it!
   Michael Collins; Manning-Jurafsky; Manning; Socher
   (DL in NLProc))

2. **Конференции**
   **ACL** conf, **EACL**, **EMNLP**, ...
   NIPS, ICLR и прочая нейроерунда
   Russia: Dialog, AINL, AIST, …

3. Хипстерские ***митапы***
   DataFest, SPbDSM, AI-whatever hackathon-ы, kaggle club

4. NLPub.Ru
   **Кто хочет спросить об инструментах - вам туда!**
   Отличный информационный **вики-ресурс!**

5. *Паблики, твиттеры, подписки*
   **opendatascience.slack.com** (!)





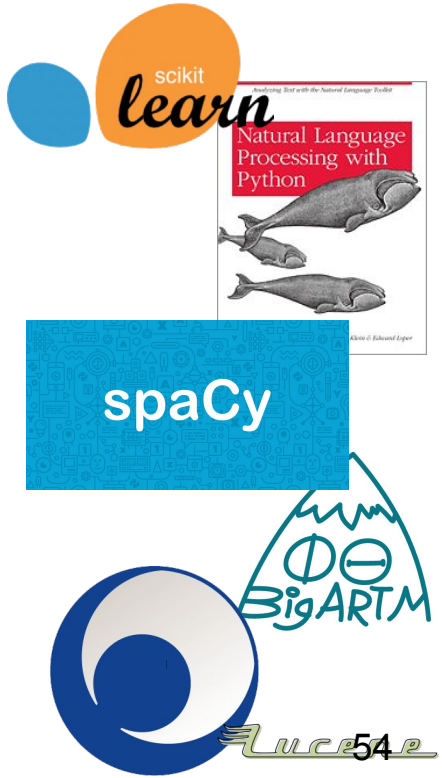
Open Data Science

NLPub

The Association for Computational Linguistics

# Попсовые инструменты для Python

- **scikit-learn**: предобработка (векторизаторы), классификация, кластеризация
- **scipy**: кластеризация
- **pystruct, seqlearn, hmmlearn**: структурные модели
- **bigartm, lda**: тематическое моделирование
- **gensim**: тематическое моделирование, дистрибутивная семантика, в т.ч. doc2vec
- **nltk, spacy.io**: “швейцарские ножи” академического и индустриального происхождения соответственно; идеально, когда надо что-то попробовать;
  *обратите внимание на доступ к WordNet*
- **%ваш любимый фреймворк для нейросетей%**
- **xapian, pylucene:** информационный поиск
- никто не отменял ваши любимые библиотеки с **edit distance, radix trees, longest common substrings, regular expressions**



а ещё NLProc и CompLing преподают
в высших образовательных учреждениях

Спасибо.
Вопросы?

# Acknowledgements

Благодарю

- Евгения Путина за приглашение
- Дениса Кирьянова за проверку презентации на одном из этапов её развития
- Остальных людей, без пинков которых эта лекция бы не состоялась

- Ещё раз — всех за внимание!

# Обработка естественного языка: введение

## The *Other* NLP*

Антон Алексеев
NativeRoll.TV, ПОМИ РАН
anton.m.alexeyev@gmail.com

* шутка на одном из стикеров DataFest'17

# Bonus

- Б. Орехов
  - Поэтический генератор
  - Векторные модели и русская литература +
  - Монография, написанная LSTM, обученной на текстах В. А. Плунгяна
- word2vec и сексизм (а потом всерьёз)
- DopeLearning: подбор рэп-строчек из корпуса нейронной сетью
- Нейронная оборона (на Я.Музыке)
- …?


src: Образовач