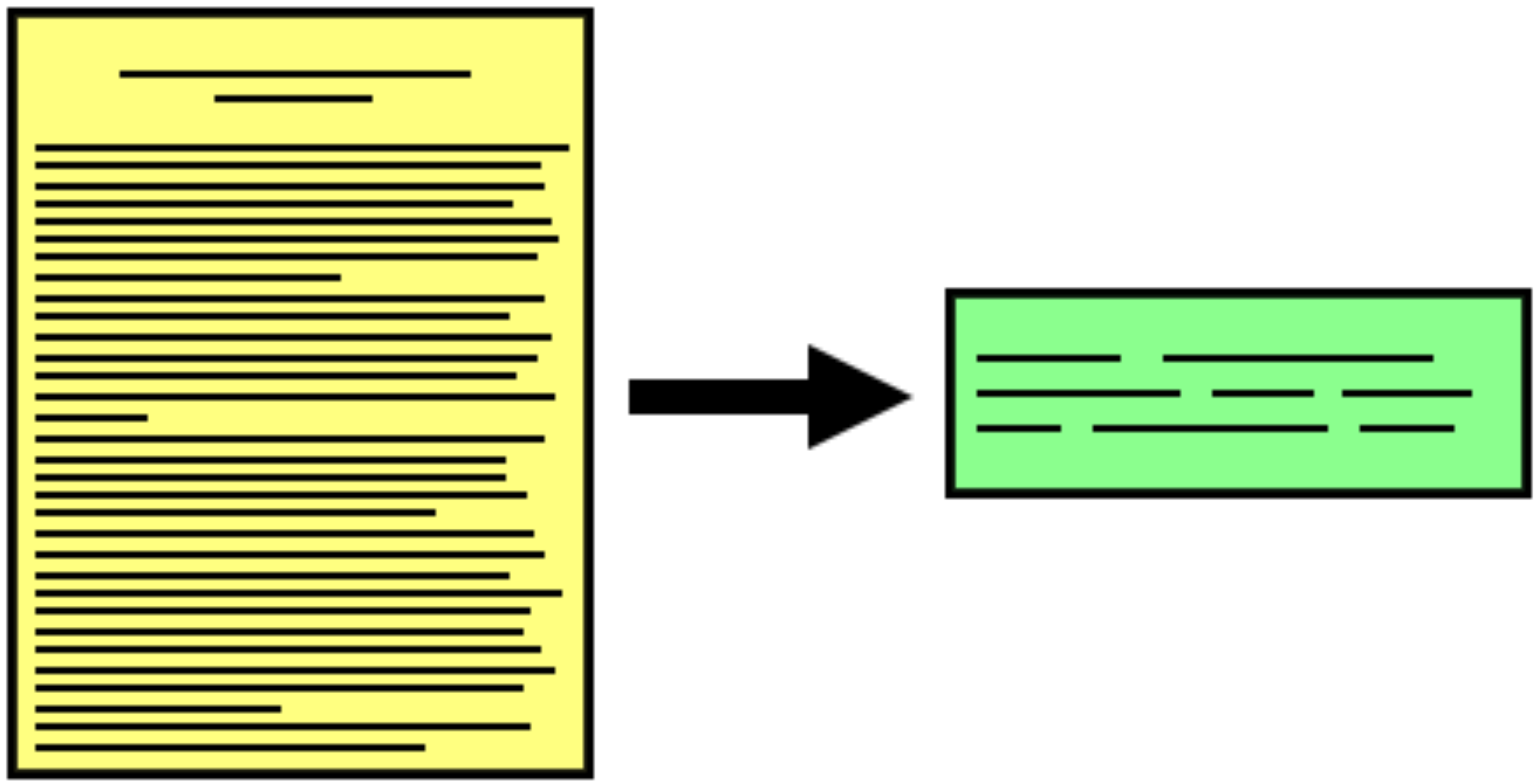
magpie
keyword extraction

Jan Stypka

Outline of the talk

1. Problem description
2. Initial approach and its problems
3. A neural network approach (and its problems)
4. Potential applications
5. Demo & Discussion

Initial project definition



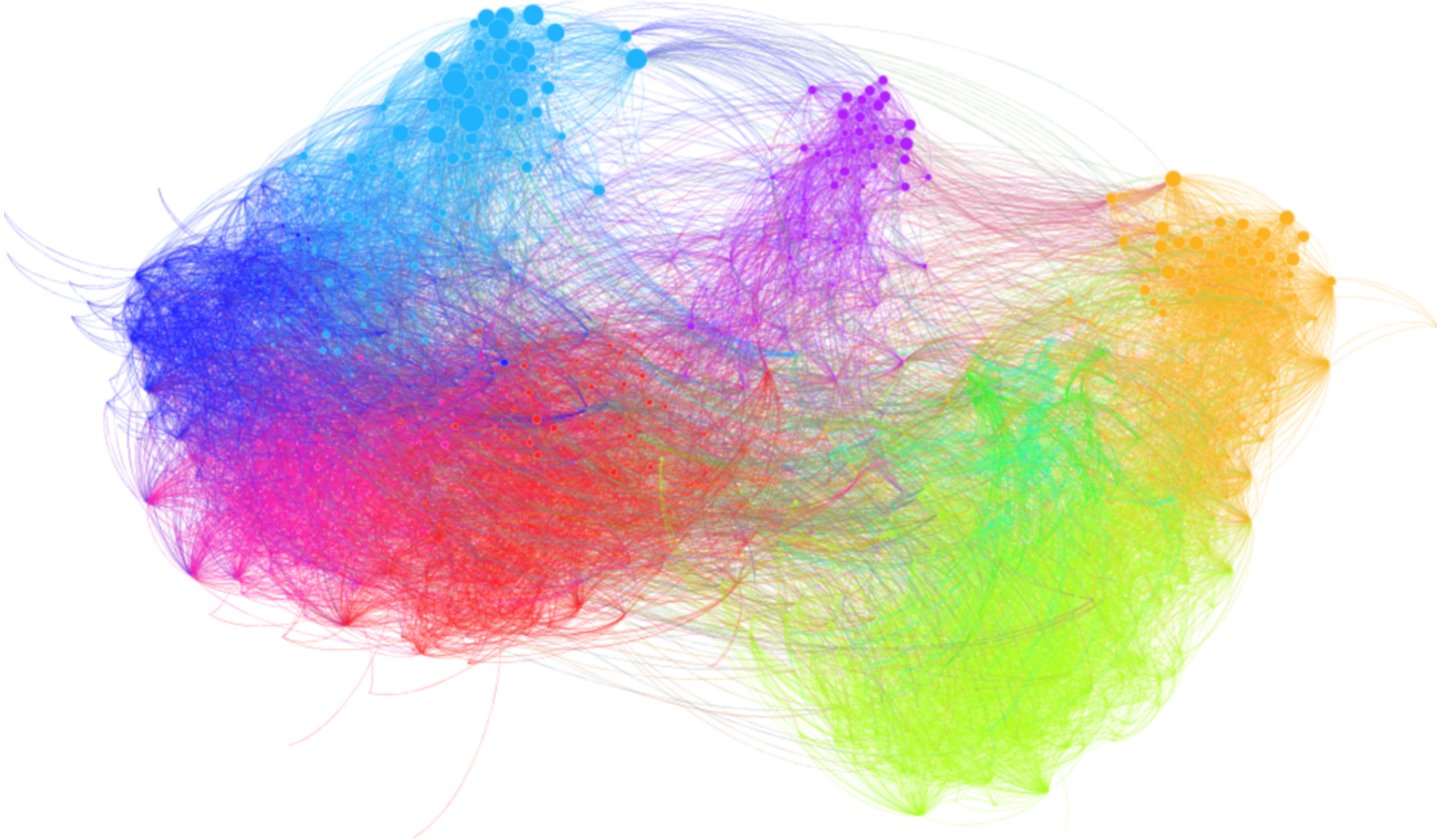
“Extracting keywords from HEP publication abstracts”

Problems with keyword extraction

- What is a keyword?
- When is a keyword relevant to a text?
- What is the ground truth?

Ontology

- all possible terms in HEP
- connected with relations
- ~60k terms altogether
- ~30k used more than once
- ~10k used in practice



Large training corpus

- ~200k abstracts with manually assigned keywords since 2000
- ~300k if you include the 1990s and papers with automatically assigned keywords (invenio-classifier)

Approaches to keyword extraction

- statistical (invenio-classifier)
- linguistic
- unsupervised machine learning
- **supervised machine learning**

Traditional ML approach

- using ontology for candidate generation
- hand engineering features
- a simple linear classifier for binary classification

Candidate generation

- surprisingly difficult part
- matching all the words in the abstract against the ontology
- composite keywords, alternative labels, permutations, fuzzy matching
- including also the neighbours (walking the graph)

Fast Radio Bursts are bright, unresolved, non-repeating, broadband, millisecond flashes, found primarily at high Galactic latitudes, with dispersion measures much larger than expected for a Galactic source[1-8]. The inferred all-sky burst rate[9] is comparable to the core-collapse supernova rate[10] out to redshift 0.5. If the observed dispersion measures are assumed to be dominated by the intergalactic medium, the sources are at cosmological distances with redshifts[11,12] of 0.2 to 1. These parameters are consistent with a wide range of source models[13-18]. One fast radio burst[6] showed circular polarization [21(7)%] of the radio emission, but no linear polarization was detected, and hence no Faraday rotation measure could be determined. Here we report the examination of archival data revealing Faraday rotation in a newly detected burst—FRB 110523. It has radio flux at least 0.6 Jy and dispersion measure 623.30(5) pc cm^{-3}. Using Galactic contribution 45 pc cm^{-3} and a model of intergalactic electron density[11], we place the source at a maximum redshift of 0.5. The burst has rotation measure –186.1(1.4) rad m^{-2}, much higher than expected for this line of sight through the Milky Way and the intergalactic medium, indicating magnetization in the vicinity of the source itself or within a host galaxy. The pulse was scattered by two distinct plasma screens during propagation, which requires either a dense nebula associated with the source or a location within the central region of its host galaxy. Keeping in mind that there may be more than one type of fast radio burst source, the detection in this instance of source-local magnetization and scattering favours models involving young stellar populations such as magnetars over models involving the mergers of older neutron stars, which are more likely to be located in low density regions of the host galaxy.

Feature extraction

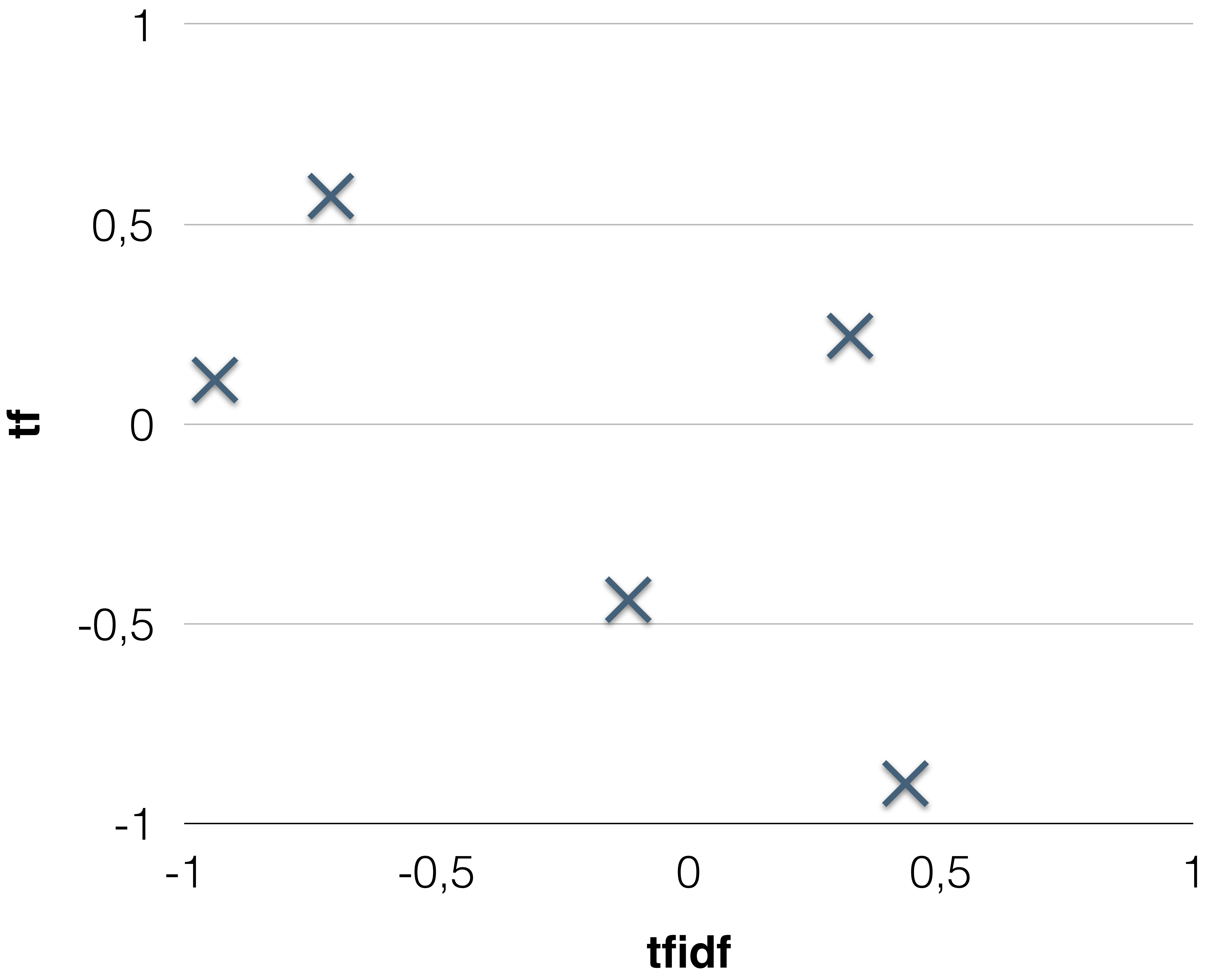
- term frequency (number of occurrences in this document)
- document frequency (how many documents contain this word)
- tf-idf $$tfidf(w, d, D) = \frac{tf(w, d)}{df(w, D)}$$
- first occurrence in the document (position)
- number of words

Feature extraction

	tf	df	tfidf	1st occur	# of words
quark	0.22	-0.12	0.32	0.03	-0.21
neutrino/tau	0.57	0.60	-0.71	-0.30	-0.59
Higgs: coupling	-0.44	-0.41	-0.12	0.89	-0.28
elastic scattering	-0.90	0.91	0.43	-0.43	0.79
Sigma0: mass	0.11	-0.77	-0.94	0.46	0.17

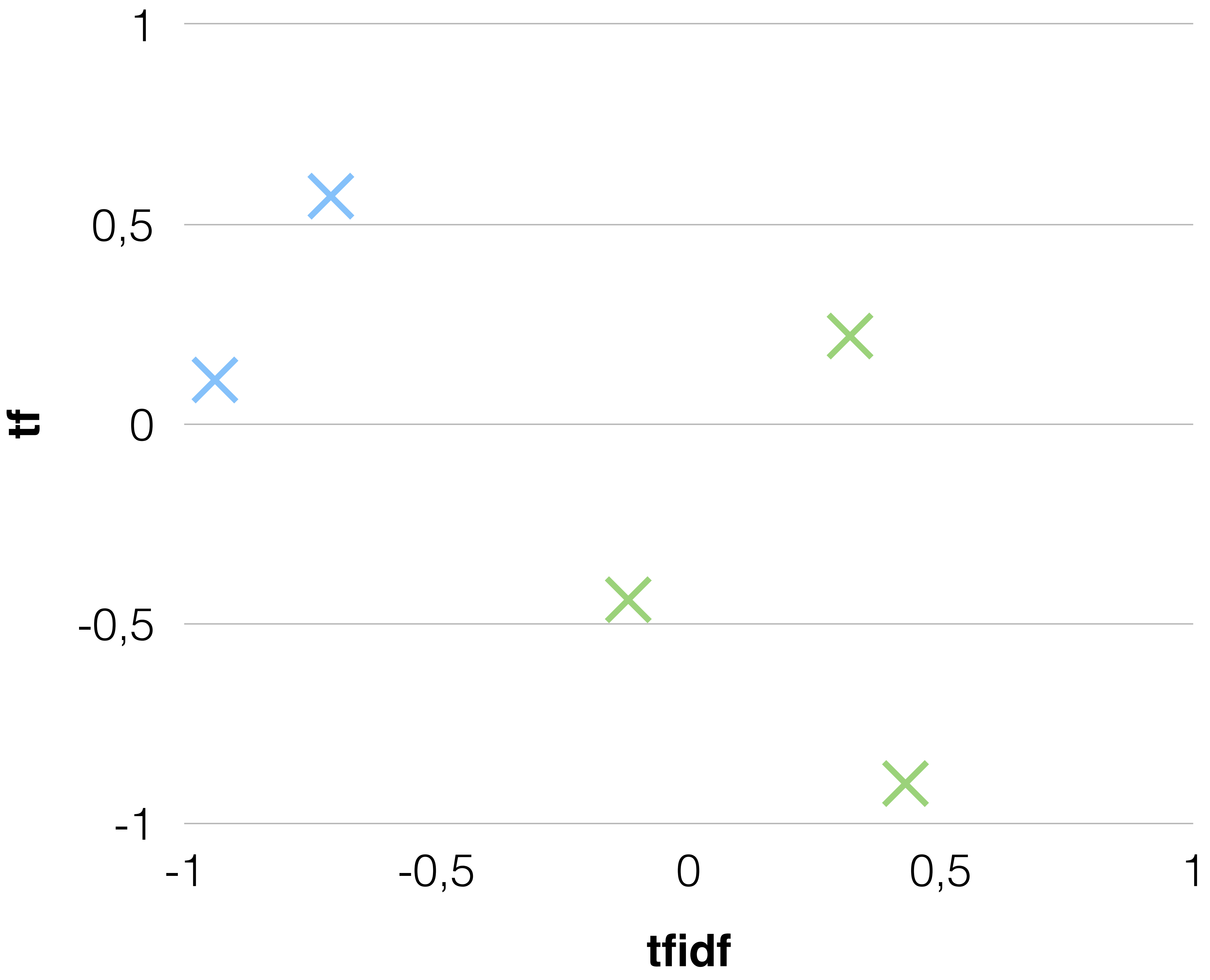
Keyword classification

	tf	tfidf
quark	0.22	0.32
neutrino/tau	0.57	-0.71
Higgs: coupling	-0.44	-0.12
elastic scattering	-0.90	0.43
Sigma0: mass	0.11	-0.94



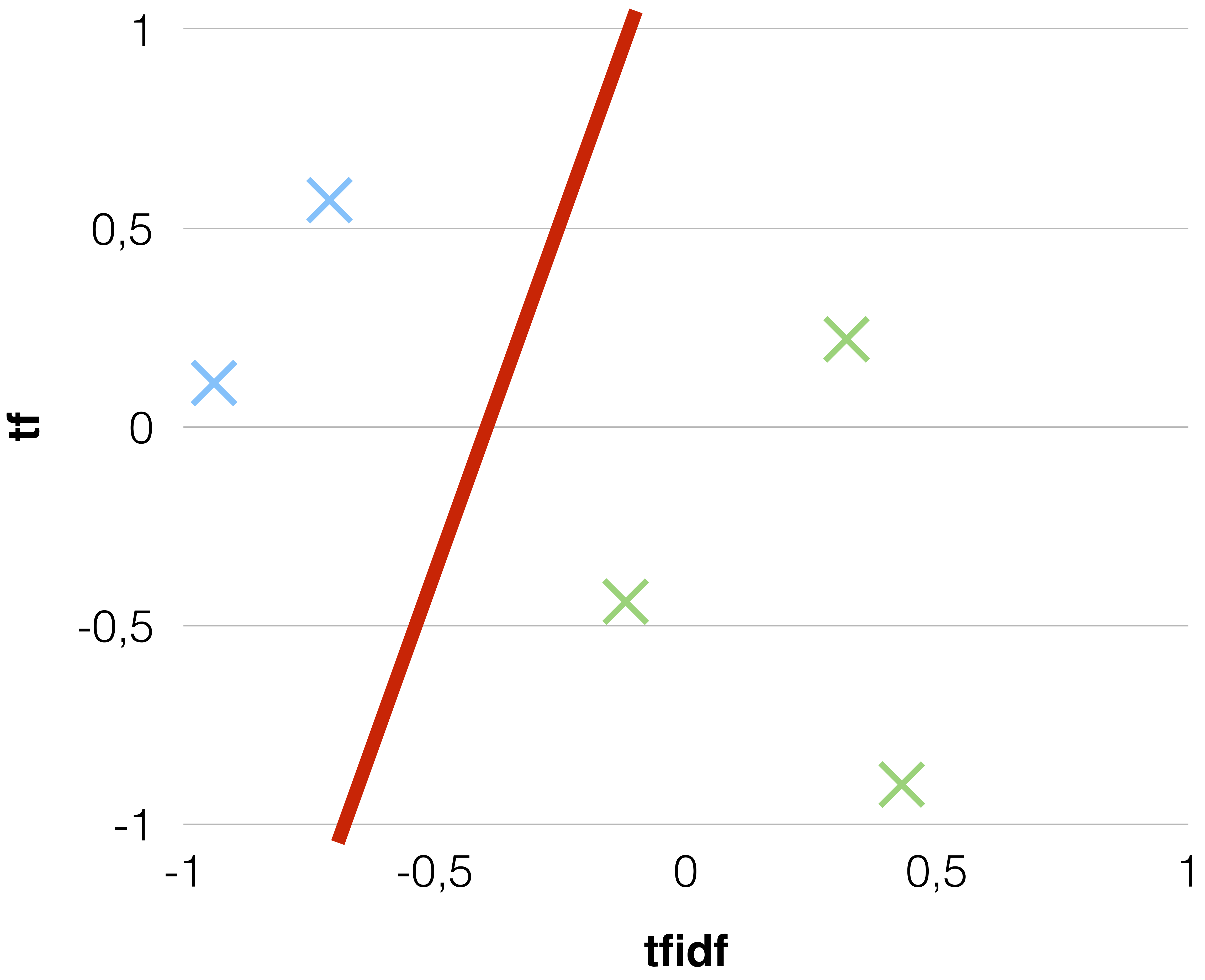
Keyword classification

	tf	tfidf
quark	0.22	0.32
neutrino/tau	0.57	-0.71
Higgs: coupling	-0.44	-0.12
elastic scattering	-0.90	0.43
Sigma0: mass	0.11	-0.94



Keyword classification

	tf	tfidf
quark	0.22	0.32
neutrino/tau	0.57	-0.71
Higgs: coupling	-0.44	-0.12
elastic scattering	-0.90	0.43
Sigma0: mass	0.11	-0.94



Ranking approach

- keywords should not be classified in isolation
- keyword relevance is not binary
- keyword extraction is a **ranking** problem!
- model should produce a ranking of the vocabulary for every abstract
- model learns to order all the terms by relevance to the input text
- we can represent a ranking problem as a binary classification problem

Pairwise transform

	a	b	c	result
w1	a1	b1	c1	✓
w2	a2	b2	c2	✗
w3	a3	b3	c3	✓
w4	a4	b4	c4	✗

	a	b	c	result
w1 - w2	a1 - a2	b1 - b2	c1 - c2	↑
w1 - w3	a1 - a3	b1 - b3	c1 - c3	↑
w1 - w4	a1 - a4	b1 - b4	c1 - c4	↓
w2 - w3	a2 - a3	b2 - b3	c2 - c3	↑
w2 - w4	a2 - a4	b2 - b4	c2 - c4	↓
w3 - w4	a3 - a4	b3 - b4	c3 - c4	↑

RankSVM result

	a	b	c	result
w1 - w2	a1 - a2	b1 - b2	c1 - c2	↑
w1 - w3	a1 - a3	b1 - b3	c1 - c3	↑
w1 - w4	a1 - a4	b1 - b4	c1 - c4	↓
w2 - w3	a2 - a3	b2 - b3	c2 - c3	↑
w2 - w4	a2 - a4	b2 - b4	c2 - c4	↓
w3 - w4	a3 - a4	b3 - b4	c3 - c4	↑

1. black hole: information theory
2. equivalence principle
3. Einstein
4. black hole: horizon
5. fluctuation: quantum
6. radiation: Hawking
7. density matrix

Mean Average Precision

- metric to evaluate rankings
- gives a single number
- can be used to compare different rankings of the same vocabulary
- **average precision values at ranks of relevant keywords**
- mean of those averages across different queries

Mean Average Precision

1. black hole: information theory
2. equivalence principle
3. Einstein
4. black hole: horizon
5. fluctuation: quantum
6. radiation: Hawking

Mean Average Precision

1. black hole: information theory — Precision = 1/1 = 1
2. equivalence principle — ~~Precision = 1/2 = 0.5~~
3. Einstein — Precision = 2/3 = 0.66
4. black hole: horizon — Precision = 3/4 = 0.75
5. fluctuation: quantum — ~~Precision = 3/5 = 0.6~~
6. radiation: Hawking — Precision = 4/6 = 0.66

AveragePrecision = (1 + 0.66 + 0.75 + 0.66) / 4 ≈ 0.77

Traditional ML approach aftermath

- Mean Average Precision (MAP) of RankSVM ≈ **0.30**
- MAP of random ranking of 100 keywords with 5 hits ≈ 0.09
- need something better
- candidate generation is difficult, features are not meaningful
- is it possible to skip those steps?

Deep learning approach

1	This
2	is
3	the
4	beginning
5	of
6	the
7	abstract
8	and

1	-0.2	0.9	0.6	0.2	-0.3	-0.4
2	0.3	-0.5	-0.8	0.3	0.6	0.1
3	0.7	-0.8	-0.1	0.2	-0.9	-0.6
4	0.6	-0.5	-0.8	0.3	0.6	0.4
5	-0.9	0.2	0.4	0.7	-0.3	-0.3
6	0.3	0.7	0.6	-0.5	-0.9	-0.1
7	0.2	-0.9	0.4	-0.8	-0.4	-0.5
8	-0.8	-0.4	-0.3	0.7	-0.1	0.6

NN

0.91	black hole
0.34	Einstein
0.06	leptoquark
0.21	neutrino/tau
0.01	CERN
0.29	Sigma0
0.48	p: decay
0.12	Yann-Mills

Word vectors

- strings for computers are meaningless tokens
- *"cat"* is as similar to *"dog"* as it is to "*skyscraper*"
- in vector space terms, words are vectors with one 1 and a lot of 0

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

- it's major problem is:

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Word vectors

- we need to represent the **meaning** of the words
- we want to perform arithmetics e.g. vec[*"hotel"*] - vec[*"motel"*] ≈ 0
- we want them to be low-dimensional
- we want them to preserve relations
  e.g. vec[*"Paris"*] - vec[*"France"*] ≈ vec[*"Berlin"*] - vec[*"Germany"*]
- vec[*"king"*] - vec[*"man"*] + vec[*"woman"*] ≈ vec[*"queen"*]

word2vec

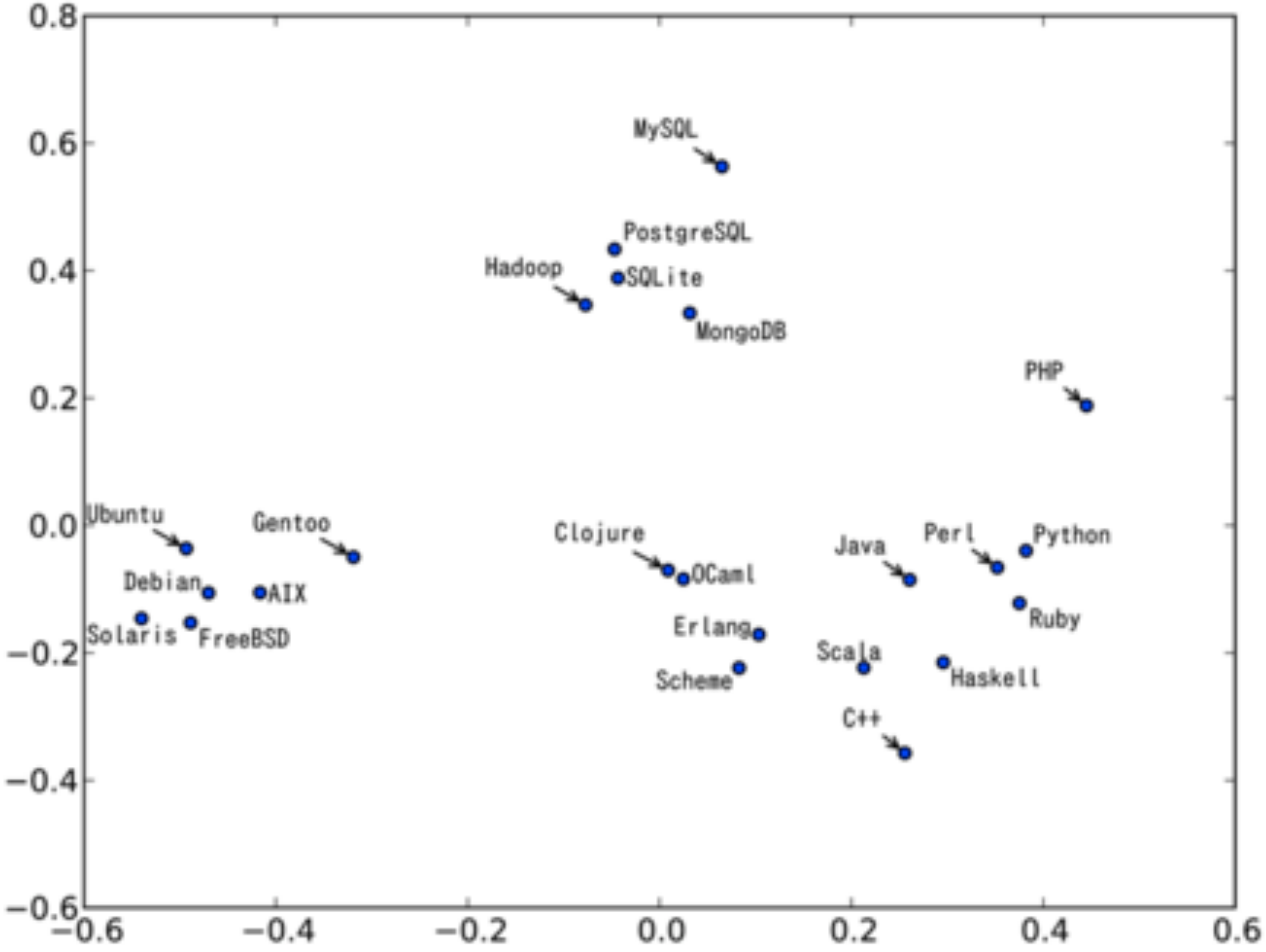
- proposed by Mikolov et al. in 2013
- learn the model on a large raw (not preprocessed) text corpus
- trains a model by predicting a target word by its neighbours
- “Ioannis is a _____ Greek man” or “Eamonn ____ skiing” or “Ilias’ _____ is really nice”
- use a context window and walk it through the whole corpus iteratively updating the vector representations

word2vec

- cost function: $$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t)$$

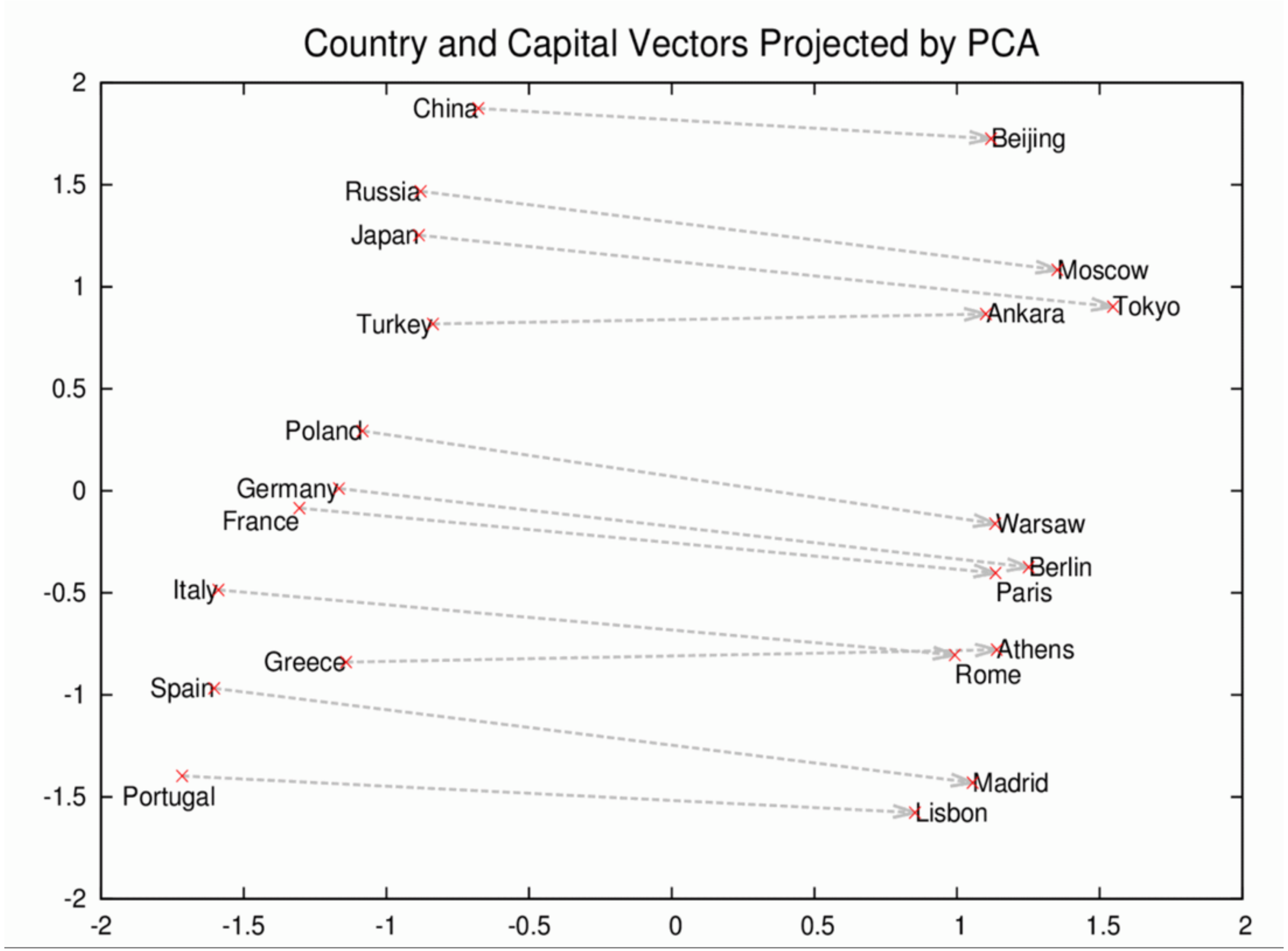
- where the probabilities: $$p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

word2vec



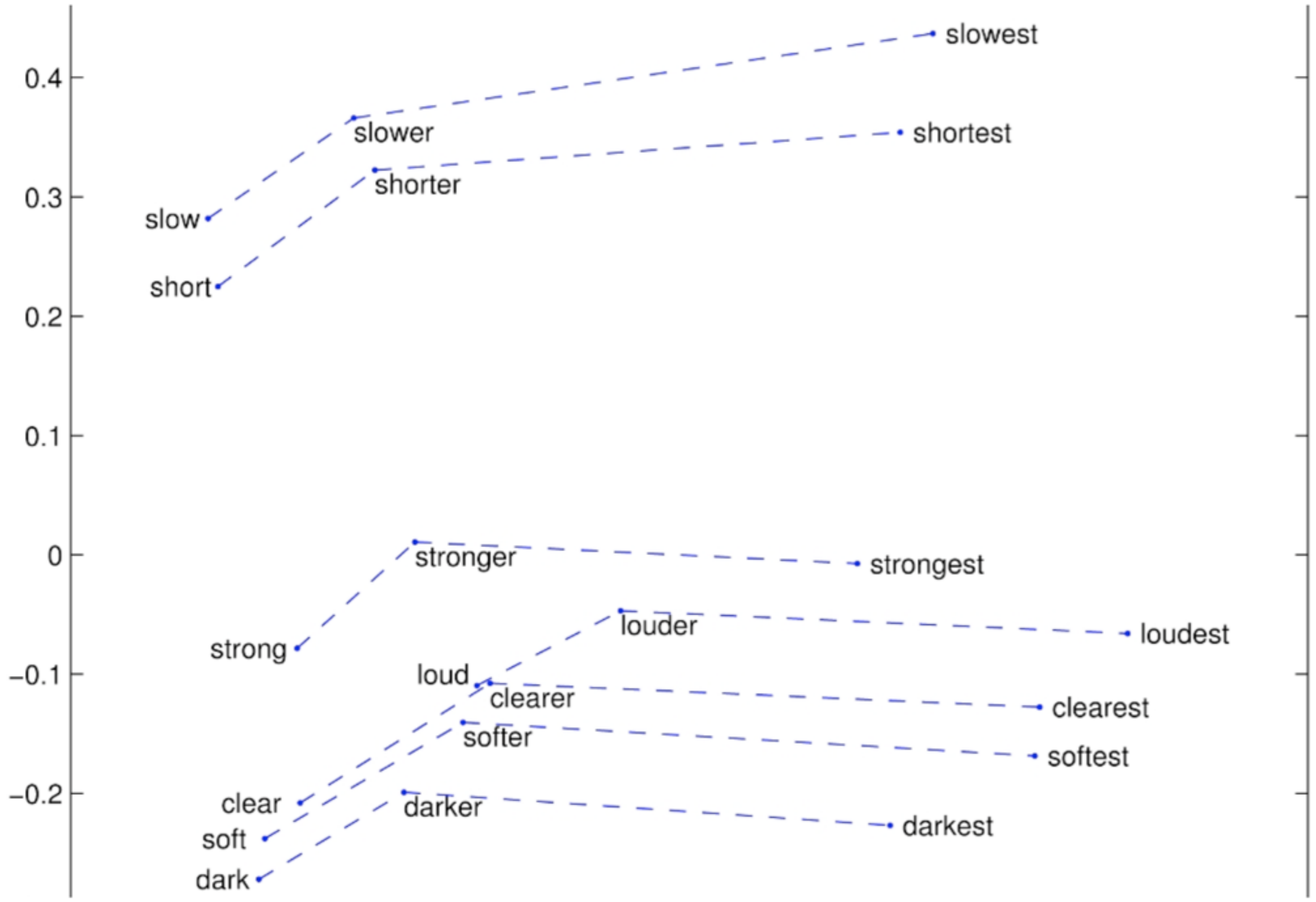
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
MySQL
PostgreSQL
Hadoop
SQLite
MongoDB
PHP
Ubuntu
Gentoo
Clojure
Java
Perl
Python
Debian
AIX
OCaml
Ruby
Solaris
FreeBSD
Erlang
Scala
Scheme
Haskell
C++

word2vec



Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Spain
Athens
Rome
Portugal
Madrid
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

GloVe



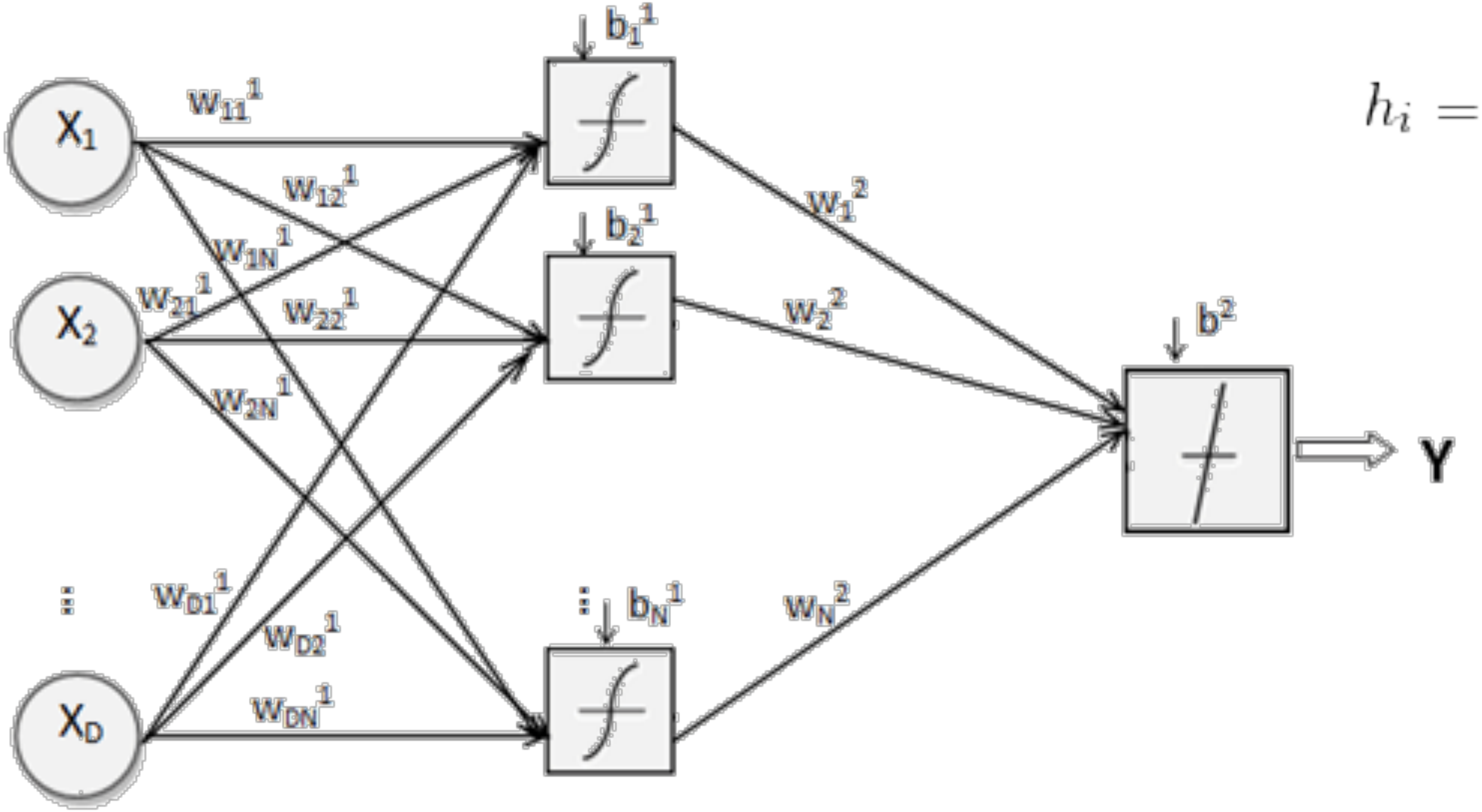
0.4
0.3
0.2
0.1
0
−0.1
−0.2
slowest
slower
shortest
shorter
slow
short
stronger
strongest
strong
louder
loudest
loud
clearer
clearest
softer
softest
clear
soft
darker
darkest
dark

Demo

Classic Neural Networks

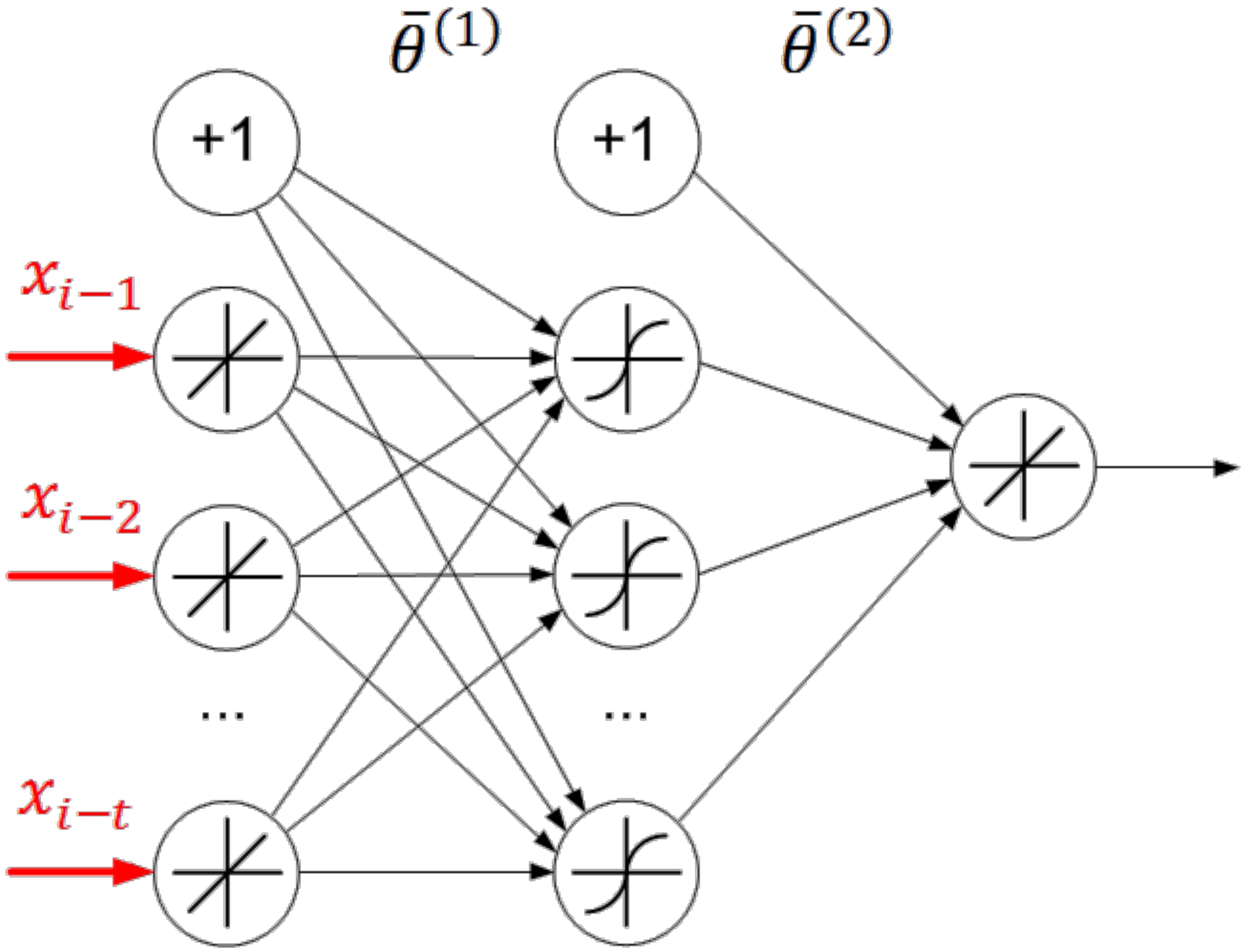
- just a directed graph with weighted edges
- supposed to simulate our brain architecture
- nodes are called neurons and divided into *layers*
- usually at least three layers - input, hidden (one or more) and output
- feed the input into the input layer, propagate the values along the edges until the output layer

Forward propagation in NN



$$h_i = sig(\sum_j w_{ij} x_j)$$

Backpropagation in NN

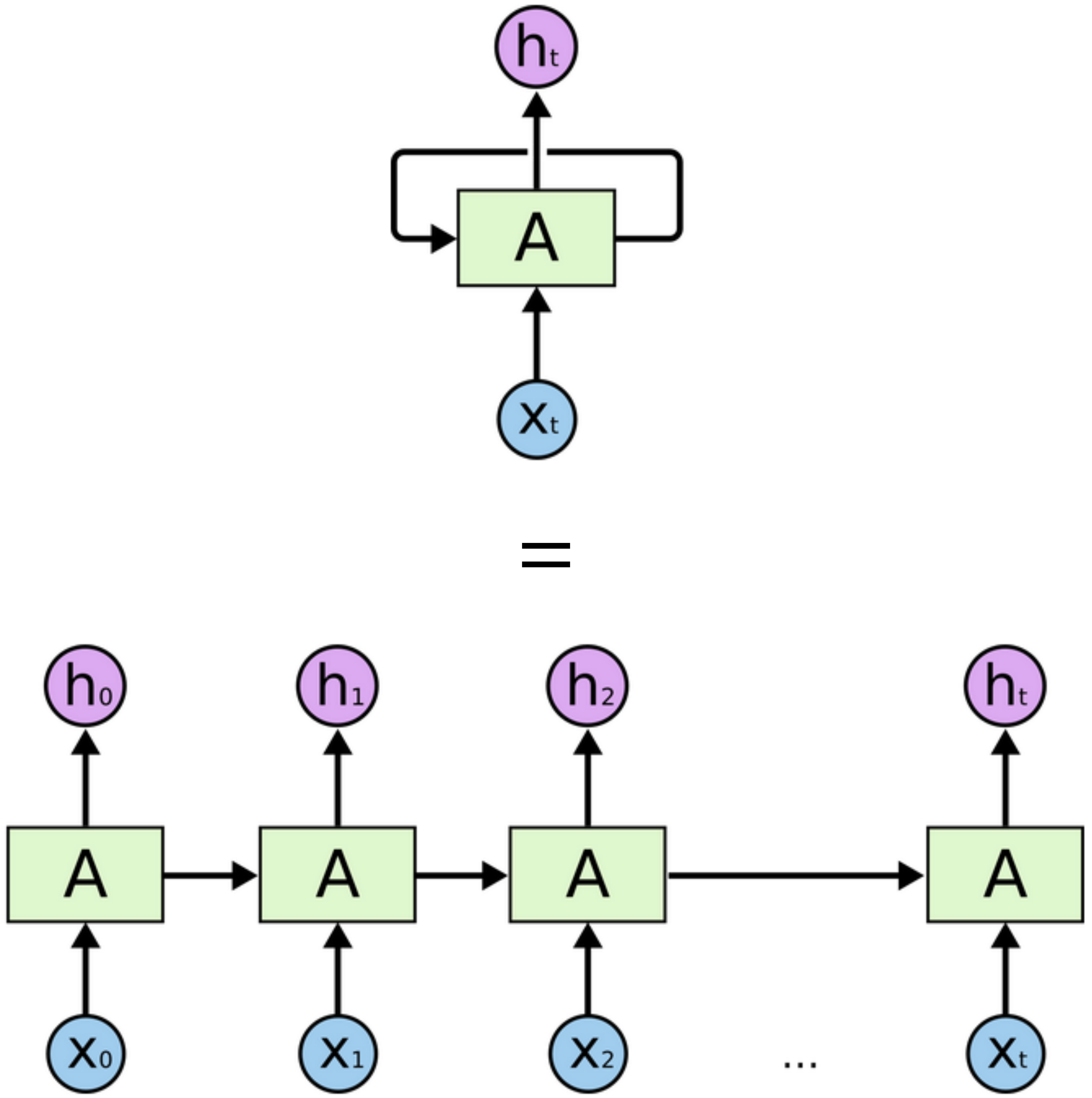


Neural Networks

- just adjust parameters to minimise the errors and conform to the training data
- in theory able to approximate any function
- take a long time to train
- come in different variations e.g. recurrent neural networks and convolutional neural networks

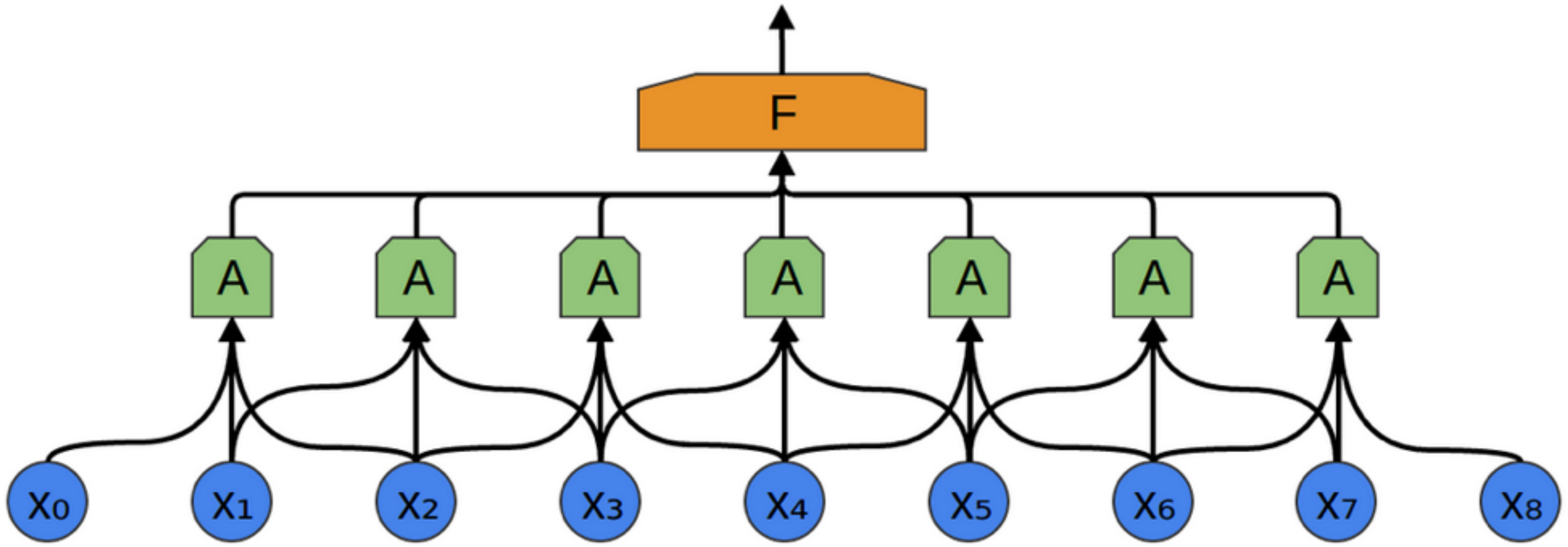
Recurrent Neural Networks

- classic NN have no state/memory
- RNNs try to go about this by adding an additional matrix in every node
- computing the state of a neuron depends on the previous layer and on the current state (inner matrix)
- used for learning sequences
- come in different kinds e.g. LSTM or GRU



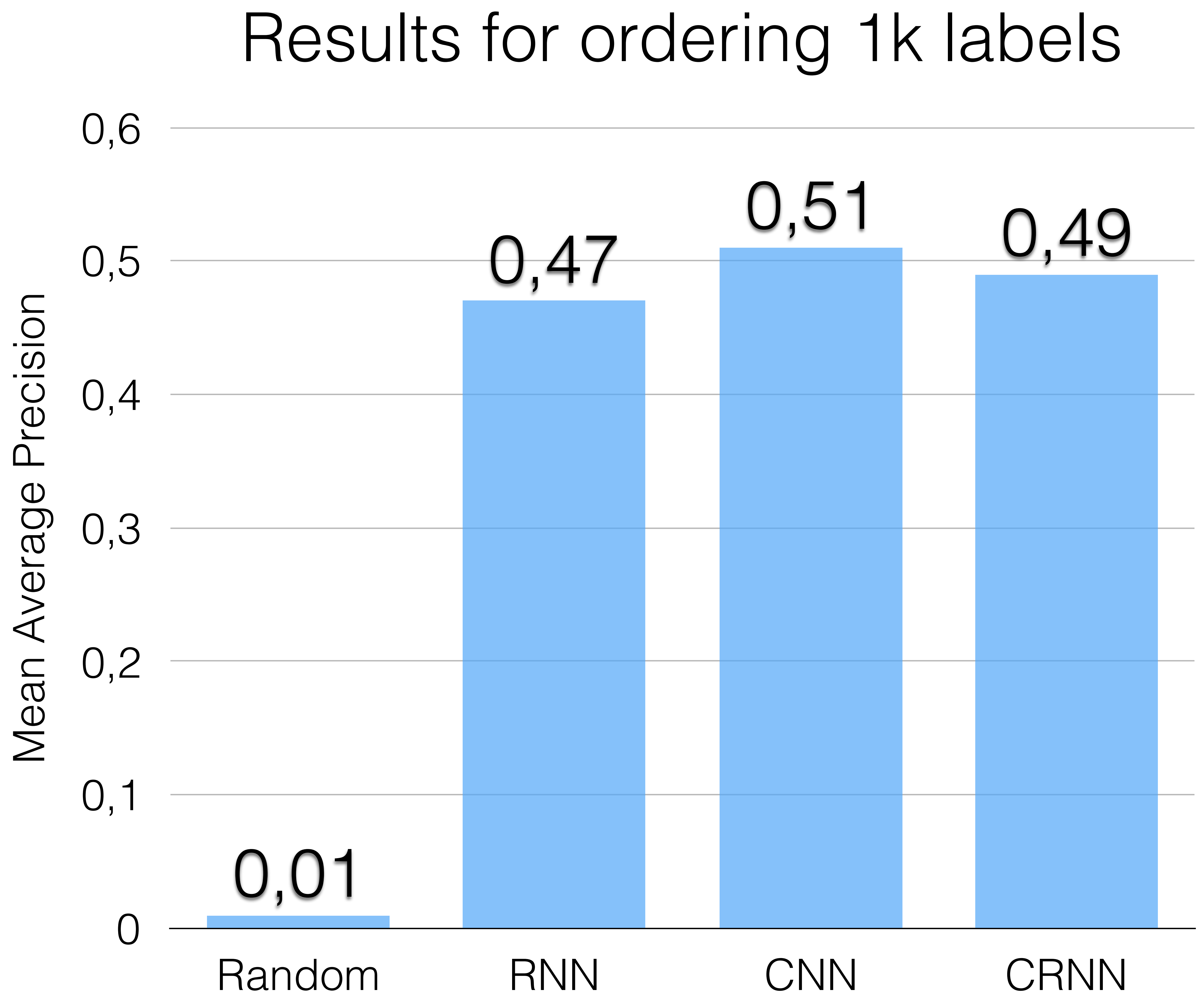
Convolutional Neural Networks

- inspired by convolutions in image and audio processing
- you learn a set of neurons once and reuse them to compute values from the whole input data
- similar to convolutional filters
- very successful in image and audio classification



NN approach

- we tested CNN, RNN and a combination of both - CRNN
- trained on half of the full corpus
- the output layer was a vector of N neurons where N ∈ {1k, 2k, 5k, 10k} corresponding to N most popular keywords in the corpus
- NNs learned to predict 0 or 1 for each keyword (relevant or not), however we used the confidence values for each label to produce a ranking

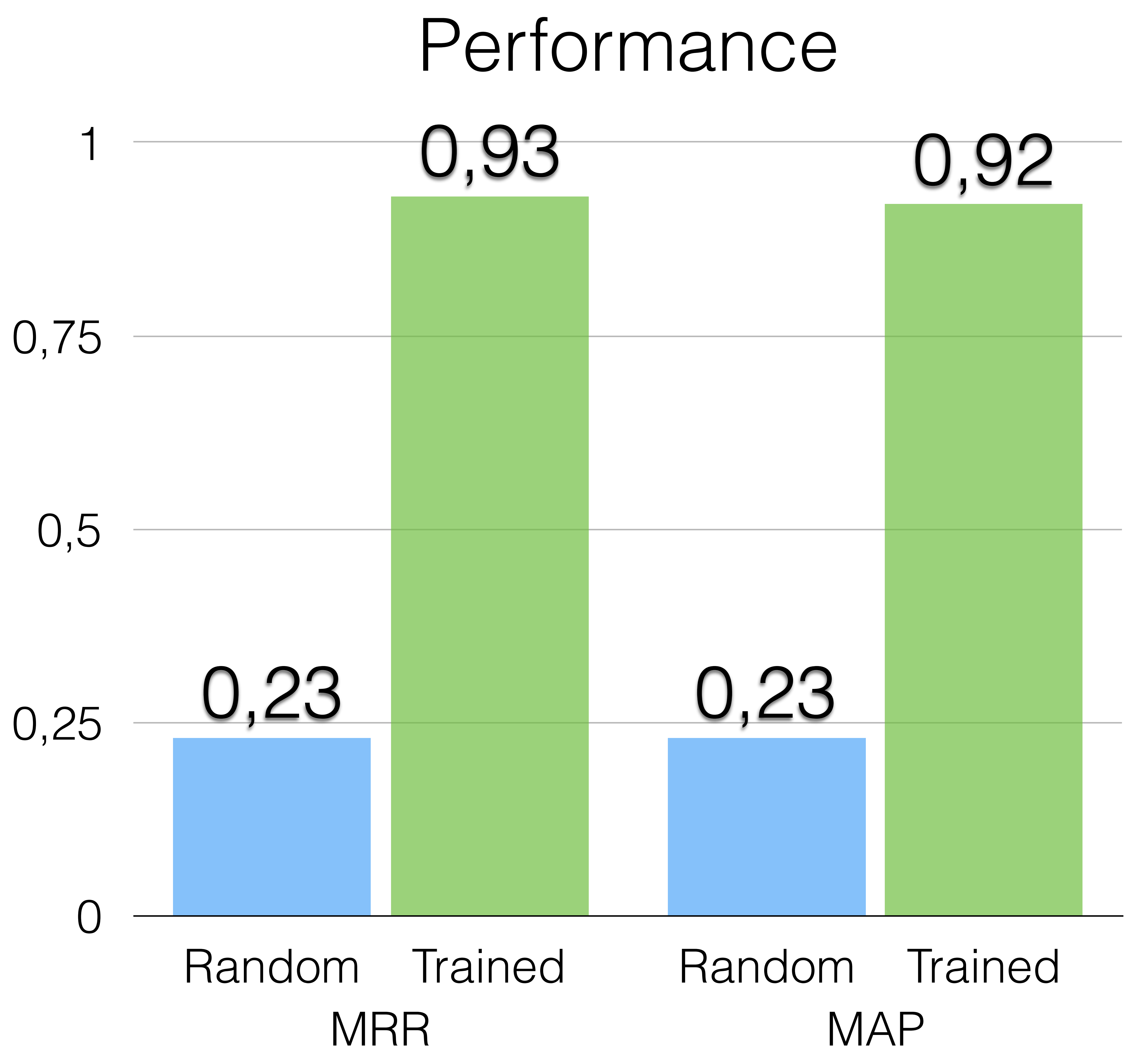


Generalisation

- keyword extraction is just a special case
- what we were actually doing was **multi-label text classification** i.e. learning to assign many arbitrary labels to text
- the models can be used to do **any text classification** - the only requirement is a predefined vocabulary and a large training set

Predicting subject categories

- we used the same CNN model to assign subject categories to abstracts
- 14 subject categories in total (more than one may be relevant)
- a small output space makes the problem much easier
- Mean Reciprocal Rank (MRR) is just the inversion of the rank of the first relevant label (1, ½, ⅓, ¼, ⅕ …)



Feedback

- the model should be able to learn continuously on incoming data
- learning on your own predictions only enforces the mistakes
- there should be a possibility to provide the network more ground truth (human curated) answers that would improve its performance
- workflow: model automatically suggests the keywords, cataloguer makes corrections and confirms, model learns on this new data
- in that way the neural network should improve over time

Demo

But what about invenio-classifier?

- difficult to compare accuracy - one produces a ranking, the other set of keywords
- data that magpie is trained on is naturally biased towards invenio-classifier
- best to evaluate manually

magpie

- requires training
- better handles short text
- doesn’t require explicit mentioning
- understands synonyms and handles fuzzy matching
- works only on top N keywords
- improves over time

invenio-classifier

- works “out of the box”
- needs a fairly long text
- needs keywords to be explicitly mentioned in a certain form
- works on the whole ontology

Links

https://github.com/jstypka/magpie

http://inspire.jacenkow.com:5050/

http://cs224d.stanford.edu/syllabus.html

http://bdewilde.github.io/blog/2014/09/23/intro-to-automatic-keyphrase-extraction/

http://colah.github.io/

http://fa.bianp.net/blog/2012/learning-to-rank-with-scikit-learn-the-pairwise-transform/

Thanks!