

A Neural Embedding Compressor for Scalable Document Search

1. Information Retrieval and Document Similarity

What we are trying to accomplish

2. The Embedding Compressor

Training and evaluating the neural auto-encoder model

3. The Relaxed Hamming Word Movers Distance (RHWMD)

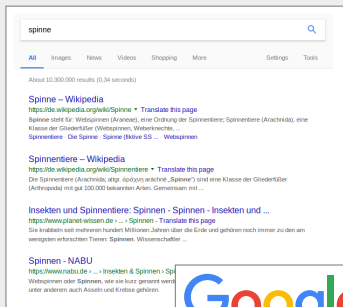
Determining document similarity with binary hash codes

4. The Ungol Index

An RHWMD implementation for information retrieval

IR and Document Similarity

- ❖ Here: Searching for text documents given a text query
- ❖ Good IR systems:
 - ❖ Sort results by relevance effectively
 - ❖ Handle large amounts of data
 - ❖ Are fast



spinne

All Images News Videos Shopping More Settings Tools


About 10,300,000 results (0.34 seconds)

Spinne – Wikipedia
<https://de.wikipedia.org/wiki/Spinne> • Translate this page
Spinne steht für: Webspinnen (Araneae), eine Ordnung der Spinnentiere; Spinnentiere (Arachnida), eine Klasse der Gliederfüßer (Webspinnen, Webkrebse, ...)
Spinnentiere · Die Spinne · Spinne (Kfz) · Webspinnen

Spinnentiere – Wikipedia
<https://de.wikipedia.org/wiki/Spinnentiere> • Translate this page
Die Spinnentiere (Arachnida, abgr. *spidyov* arachne „Spinne“) sind eine Klasse der Gliederfüßer (Arthropoda) mit gut 100.000 bekannten Arten. Gemeinsam mit ...

Insekten und Spinnentiere: Spinnen - Spinnen - Insekten und ...
<https://www.planet-wissen.de> > ... > Spinnen - Translate this page
Sie krabbeln seit mehreren hundert Millionen Jahren über die Erde und gehören noch immer zu den am weitesten erkrankten Tieren: Spinnen. Wissenschaftler ...

Spinnen - NABU
<https://www.nabu.de> > ... > Insekten & Spinnen - Sp
Webspinnen oder Spinnen, wie sie kurz genannt werden, unter anderem auch Asseln und Krebse gehören.



455 hits New Save Open Share Reporting Auto-refresh

spinne Options Q

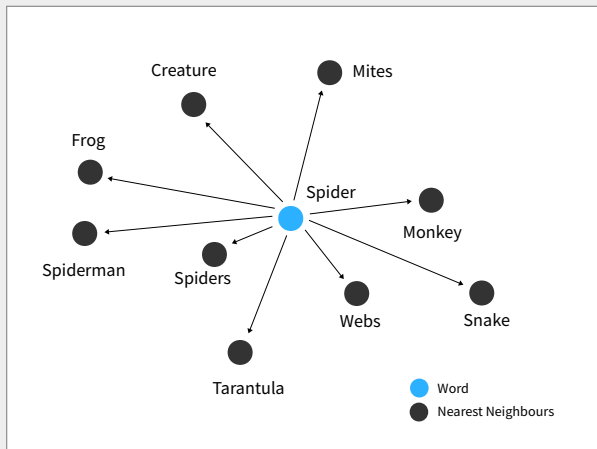
Add a filter +

_source

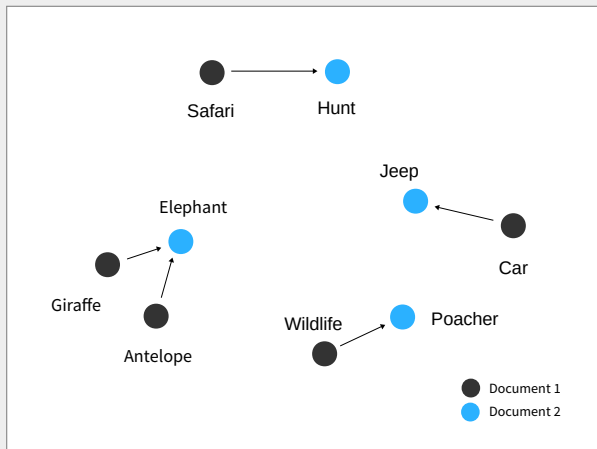
- content: Eine Amseljungfer ist noch lange keine Libelle. Nicht jede **Spinne** ist eine Kreuzspinne. Die Schwarze Mathilde ist eine Frau aus der bayerischen CSU. Wo Heischen grillen, moht das Glück. Dicke **Spinnen** sind deshalb dick, weil sie viel Ungeziefer ausgesaugt haben. Dünne **Spinnen** sind noch nicht ausgewachsen. Wer **Spinnen** ausrottet, sollte zum Psychiater gehen. Kleine **Spinnen** **spinnen** kleine Netze, aber **spinnen** tun sie al
- content: Die Kreuzspinne hat ihren Faden verloren. Bensebel von Marihuana, hört sie einfach auf zu **spinnen**. Sie löst das halbfertige Radnetz in Wind flattern und fängt an zu dösen. Da freuen sich die Fliegen. Die Disco-Droge Ecstasy hingegen, ein Aufputschmittel, macht die mäde **Spinne** wieder munter: Viel schneller als sonst spult sie ihren klebrigen Zaum ab. Im Übereifer kommt den Krabbeltier nur leider die Übersicht abhande
- title: "Rote **Spinne**" löst "Bärner Rose" ab Neu als eigenes Informationsmittel der kantonalen SP content: Bern, 21. Sept. (sda) "Aufgrund veränderter Informationsbedürfnisse selbständige Informationsorgan "Rote **Spinne**" gesc "Bärner Tagwacht" monatlich publizierten zwei SP ab. "Die Roten **Spinnen** nicht" stellt der Leiter

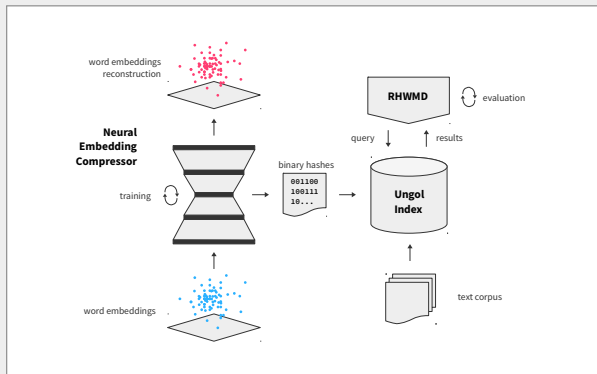


Word Embeddings: Relatedness by distance



(Relaxed) Word Movers Distance (R)WMD: Idea





The Embedding Compressor

What we have:

- ❖ Real-valued word embeddings in Euclidean space
- ❖ Spatial word similarity structure

What we do with it:

- ❖ Training a neural auto-encoder model to reconstruct word embeddings
- ❖ Use the encoder to create binary hash codes
- ❖ Examine whether the spatial properties are transferred into Hamming space

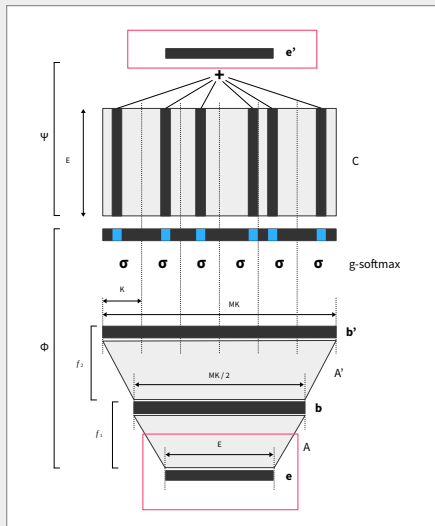
The Neural Embedding Compressor

- Training objective: reduce reconstruction loss

$$L(\mathbf{e}_t, \mathbf{e}'_t) := \frac{1}{2} \|\mathbf{e}_t - \mathbf{e}'_t\|_2^2$$

$$(\psi_\theta \circ \phi_\theta)(\mathbf{e}_t) := \mathbf{e}'_t$$

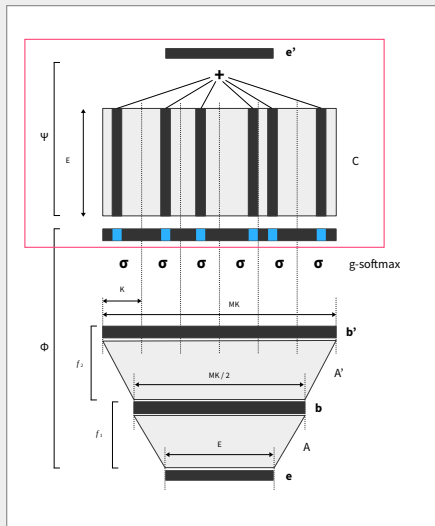
$$\theta := \{A, \mathbf{b}, A', \mathbf{b}', C\}$$



The Neural Embedding Compressor: Decoder

- Given MK basis vectors
 $C \in \mathbb{R}^{E \times MK}$
- Accepts vector
 $\mathbf{x} \in \{0, 1\}^{MK}$ of
concatenated one-hot
encodings
- Combination by additive
vector quantization

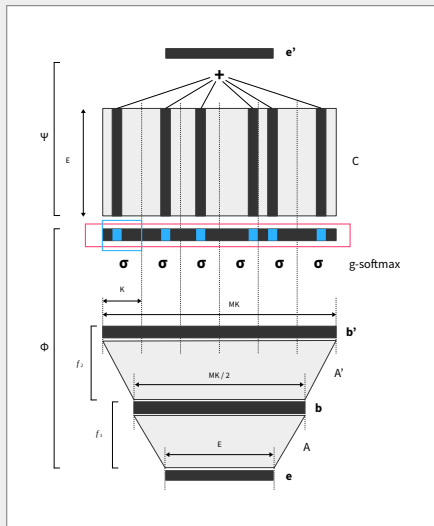
$$\psi_{\theta}(\mathbf{x}) := C\mathbf{x} = \mathbf{e}'$$



The Neural Embedding Compressor: One-Hot Encoding

- Encoder output are one-hot encodings
- Here: $M = 6, K = 5$
- Code: $(1, 3, 2, 4, 1, 3)$

$$X = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix}$$



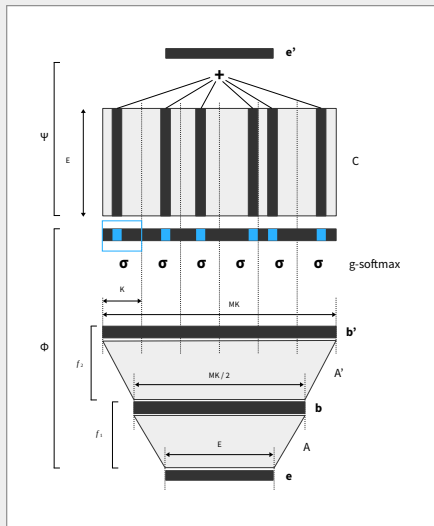
The Neural Embedding Compressor: Gumbel Softmax I

- ❖ Alter softmax function
- ❖ Add Gumbel noise
- ❖ Forces the model to approach categorical distribution

$$\text{g-softmax}(\mathbf{x}_i) := \frac{\exp(f(\mathbf{x}_i))}{\sum_{k=1}^K \exp(f(\mathbf{x}_k))}$$

$$f_{\tau}(\mathbf{x}_k) = \frac{1}{\tau} \cdot (\mathbf{x}_k + \mathbf{g}_k)$$

$$\mathbf{g} = (g_1, \dots, g_K); g_i \sim G$$



The Neural Embedding Compressor: Gumbel Softmax II

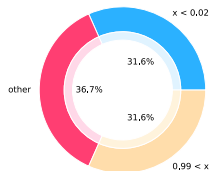
After a few epochs:

training	0.505	0.001	0.003	0.186	0.305
validation	1	0	0	0	0

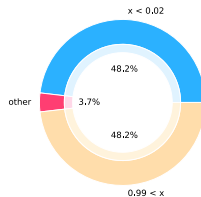
End of training:

training	0.993	0.001	0.003	0.002	0.001
validation	1	0	0	0	0

Encoder Activations (Epoch 500)
(binary/fasttext.de-256x2)



Encoder Activations (Epoch 2000)
(binary/fasttext.de-256x2)



The Neural Embedding Compressor: Training Results (Binary)

- Training excerpt for $K = 2$
- Good loss and entropy for $M = 256$

M	Θ	GloVe		fasttext.de	
		Loss	Entropy	Loss	Entropy
32	19200	16.82	0.972	8.037	0.960
64	38400	14.99	0.981	6.809	0.932
128	76800	11.72	0.985	5.319	0.909
256	153600	6.986	0.961	3.682	0.873
512	307200	6.921	0.970	3.762	0.849

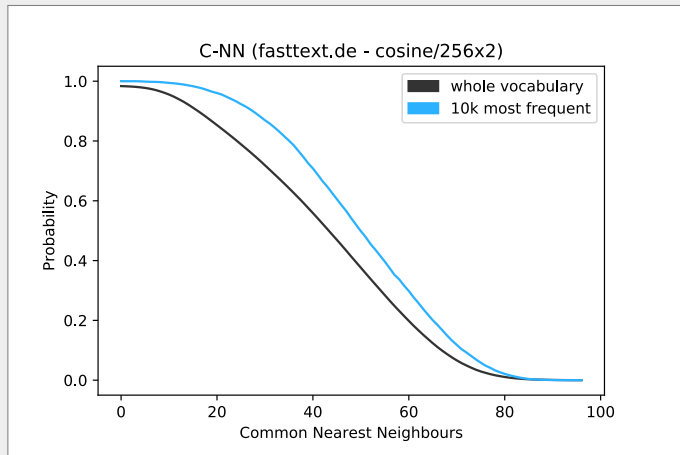
$$\text{entropy}(\mathbf{x}) := \frac{1}{\log_2 K} \sum_{i=1}^K \begin{cases} 0 & \text{if } p(\mathbf{x}_i) = 0 \\ -p(\mathbf{x}_i) \cdot \log_2 p(\mathbf{x}_i) & \text{else} \end{cases}$$

Nearest Neighbours of **Compressor**

Cosine - C					Hamming - H			
idx	word	dist _C	dist _H	idx _H	word	dist _C	dist _H	idx _C
1	compressors	0.694	60	2	turbine	0.693	60	2
2	turbine	0.693	60	1	compressors	0.694	60	1
3	supercharger	0.541	78	11	valve	0.440	72	25
4	high-pressure	0.541	75	8	engine	0.415	73	40
5	nozzle	0.524	82	19	combustion	0.503	74	8
6	turbocharger	0.513	80	16	cylinder	0.483	75	14
7	turbines	0.504	75	7	turbines	0.504	75	7
8	combustion	0.503	74	5	high-pressure	0.541	75	4
9	conditioner	0.501	89	54	engines	0.384	77	56
10	piston	0.495	85	23	cylinders	0.438	78	27

dist_H is the count of unequal bits of two hash codes

The Embedding Compressor: Nearest Neighbours (quantitatively)



The Relaxed Hamming WMD

What we have:

- ❏ Binary hash codes of words
- ❏ Hamming distance as similarity measure

What we do with it:

- ❏ Measure similarity between documents
- ❏ Following the RWMD: Select nearest neighbour pairs
- ❏ Quantify similarity of documents by using the distance of all nearest neighbour pairs

- Calculate a score s for a document pair:

$$s(d, d') := \sum_{t \in d} \text{n-idf}(t) \cdot (1 - \eta(h_t, h_{t'})) \quad (1)$$

$$\text{n-idf}(t) := \frac{\text{idf}(t)}{\sum_{t' \in d} \text{idf}(t')} \quad \text{and} \quad \eta(h_t, h_{t'}) := \frac{1}{M} \left| \left\{ i \mid h_{t_i} \neq h_{t'_i} \right\} \right| \quad (2)$$

- $t' \in d'$ is the nearest neighbour token of $t \in d$
- h_{t_i} is the i -th bit of hash code h_t with M bits
- Heuristic normalisation with **n-idf**
- This score is not symmetric:

$$r_{\text{rhwmd}}(d, d') := s(d, d') + s(d', d)$$

- ❏ Document 1:
Auf der Mauer auf der Lauer sitzt ne kleine Wanze.
- ❏ Document 2:
Ein Marienkäfer schläft auf dem Zaun.
- ❏ Score: **0.711667**

token	nn	sim	tf	idf	n-idf	weight
wanze	marienkäfer	0.742	0.100	9.335	0.269	0.200
lauer	schläft	0.668	0.100	7.134	0.206	0.137
ne	auf	0.664	0.100	5.561	0.160	0.106
mauer	zaun	0.758	0.100	4.674	0.135	0.102
sitzt	schläft	0.727	0.100	4.361	0.126	0.091
kleine	auf	0.688	0.100	3.290	0.095	0.065
auf	auf	1.000	0.200	0.305	0.009	0.009
der	dem	0.785	0.200	0.033	0.001	0.001

The Ungol Index

What we have:

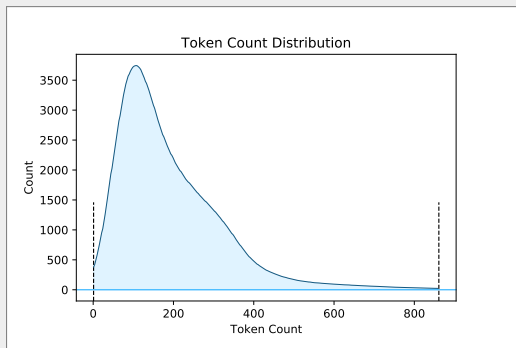
- ❖ Binary codes for words
- ❖ RHWMD for measuring similarity between documents

What we do with it:

- ❖ Evaluation of the RHWMD for Information Retrieval (IR)
- ❖ Calculate the RHWMD for query/candidate-documents
- ❖ Measure the execution speed of the implementation
- ❖ Compare the ranking quality to TF-IDF, BM25 and WMD

The Evaluation: Overview

- ❖ Around 300k news articles (Spiegel/SDA/Fr. Rundschau)
- ❖ CLEF 2003 Ad-Hoc Monolingual: IR Task
- ❖ 56 Topics: 1-226 positives, queries are very short



❖ An example query:

- ❖ *Finde Berichte über den Kruzifixstreit in bayerischen Schulen.*

München, 22. Sept. (sda/Reuter) Erstmals seit dem Kruzifix-Urteil des deutschen Bundesverfassungsgerichts sind in zwei bayerischen Schulen Kreuze abgehängt worden. Dies erklärte am Freitag in München der Sprecher des bayerischen Kultusministeriums. (...)

❖ Preprocessing includes:

- ❖ tokenizing
- ❖ lower casing
- ❖ compound splitting
- ❖ sanitizing

180921_ungol.db.pickle

Documents	294659
Vocabulary	400000
Code Size	256
Unique Tokens	258391
Avg. Doc. Length	197
Skipped	8
Memory Footprint	~ 3 GB

- ❖ How to calculate the RHMWD fast with Python?
- ❖ Problem: $|d| \cdot |d'|$ many token combinations

Approach:

- ❖ Create lookup table with each $x_i =$ number of ones for i_2

$$\Xi := (x_0, \dots, x_{255}) = (0, 1, 1, 2, 1, \dots)$$

- Given two documents (d, d') :

- Construct two matrices $H, H' \in \{0, \dots, 255\}^{|d| \times |d'| \times B}$

$$H := \begin{bmatrix} h_1 & \dots & h_{|d|} \\ \vdots & \ddots & \vdots \\ h_{|d|} & \dots & h_{|d|} \end{bmatrix} \quad H' := \begin{bmatrix} h'_1 & \dots & h'_{|d'|} \\ \vdots & \ddots & \vdots \\ h'_1 & \dots & h'_{|d'|} \end{bmatrix}$$

- Use numpy's vectorized XOR implementation $Z := H \oplus^v H'$
- Select from lookup table and collapse last dimension:

$$T_{ij} = \frac{1}{M} \sum_{k=1}^B \Xi_{Z_{ijk}}$$

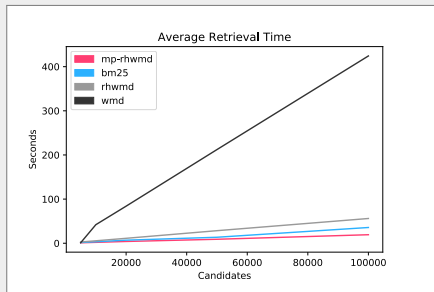
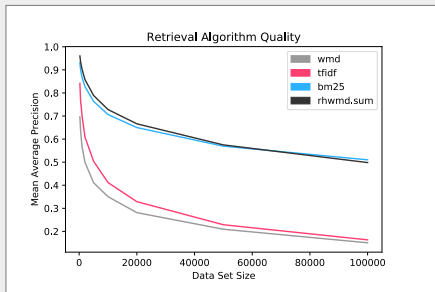
- Create two vectors with nearest neighbours along each axis:

$$\mathbf{v}_d := (v_1, \dots, v_{|d|}) \quad \text{with} \quad v_i := \min_j T_{ij}$$

$$\mathbf{v}_{d'} := (v'_1, \dots, v'_{|d'|}) \quad \text{with} \quad v'_i := \min_j T_{ij}$$

Experiment I: Baseline Experiment

- ❏ Use different corpus subsets (1k, 5k, 10k, ... documents)
- ❏ Compare **rhwmd**, **bm25**, **tf-idf** and **wmd**
- ❏ Measure execution speed in seconds for a **linear search**
- ❏ Measure performance with **Mean Average Precision μ AP**

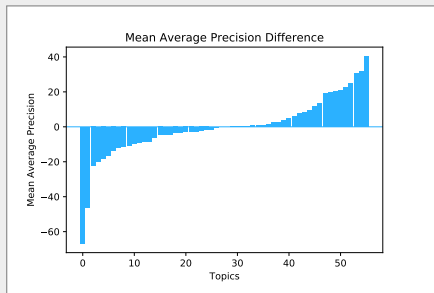


Experiment II: Re-Ranking

- ❖ Use an inverted index for candidate pre-selection
- ❖ Elasticsearch is used for pre-selection
- ❖ Apply the RWMD to the candidate set for re-ranking
- ❖ Evaluate whether re-ranking improves the μAP

Algorithm	μAP
rhwmd.sum	38.30
bm25 (Elasticsearch)	38.40
bm25 (Ungol)	38.65

- ❖ BM25 and RHWMD rank differently \rightarrow room for improvement



token	nn	sim	idf	weight
zöliakie	erkrankungen	0.738	11.900	0.187
erkrankten	erkrankten	1.000	6.363	0.135
diskutieren	diskutieren	1.000	4.760	0.101
finde	habe	0.758	5.330	0.086
ernährungs	gesundheit	0.754	5.999	0.096
bzw	und	0.770	5.027	0.082
berichte	berichtete	0.809	4.304	0.074
probleme	probleme	1.000	3.020	0.064
von	von	1.000	0.221	0.005
die	die	1.000	0.050	0.001

❖ Approach works correctly but ranks worse

token	nn	sim	idf	weight
irisches	irland	0.742	9.226	0.022
dublin	dublin	1.000	6.225	0.020
freundschaftsspiel	fußball	0.738	7.884	0.019
ausschreitungen	ausschreitungen	1.000	5.782	0.019
krawallen	ausschreitungen	0.781	7.295	0.018
row	finde	0.688	8.211	0.018
randalierer	ausschreitungen	0.777	7.247	0.018
rowdies	ausschreitungen	0.660	8.450	0.018
landfriedensbruchs	ausschreitungen	0.699	7.757	0.018
Birmingham	england	0.738	6.924	0.017

- ❖ Ten highest contributors for a document ranked worse with BM25

Summary

- ❖ Producing binary hash codes using the compressor model works nicely
- ❖ Fast similarity computation: **much** faster than the WMD
- ❖ Competitive ranking algorithm even in this state

Ad-Hoc Improvements

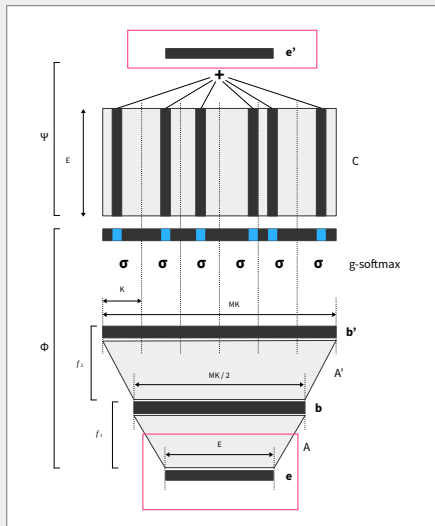
- ❖ Faster native implementation for query → batch of candidate documents
- ❖ Use the rank of the nearest neighbour
- ❖ IDF combination (use IDF of NN)

Thank you! Questions?

The Neural Embedding Compressor: Loss

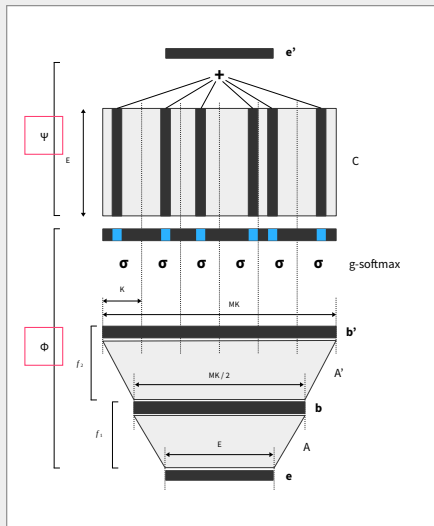
- ❖ Training objective: reduce reconstruction loss
- ❖ Loss: Mean squared error of the Euclidean distance
- ❖ Input: embedding vector \mathbf{e}_t of a token t
- ❖ Output: embedding vector \mathbf{e}'_t given \mathbf{e}_t

$$L(\mathbf{e}_t, \mathbf{e}'_t) := \frac{1}{2} \|\mathbf{e}_t - \mathbf{e}'_t\|_2^2$$



The Neural Embedding Compressor: Encoder/Decoder

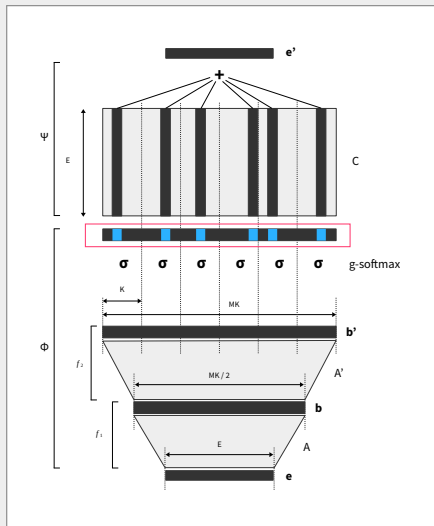
- ❖ M : Code components
- ❖ K : Code domain
- ❖ E : Embedding dims.
- ❖ **Encoder:**
 $\phi : \mathbb{R}^E \mapsto \{0, 1\}^{MK}$
- ❖ **Decoder:**
 $\psi : \{0, 1\}^{MK} \mapsto \mathbb{R}^E$
- ❖ **Model:**
 $(\psi_\theta \circ \phi_\theta)(\mathbf{e}_t) = \mathbf{e}'_t$



The Neural Embedding Compressor: Codes

- Encoder output are one-hot encodings
- Here: $M = 6, K = 5$
- Code: (1, 3, 2, 4, 1, 3)

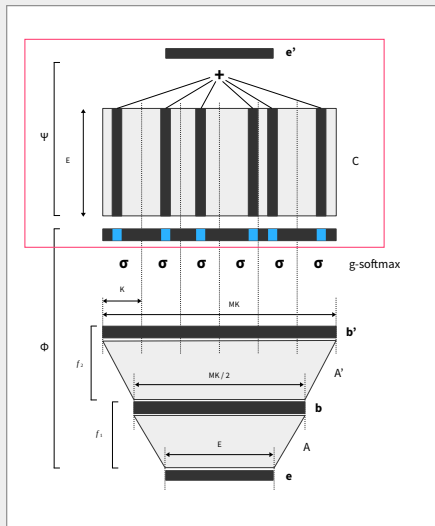
$$X = \begin{matrix} & 0 & 1 & 2 & 3 & 4 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix}$$



The Neural Embedding Compressor: Decoder

- Given MK basis vectors
 $C \in \mathbb{R}^{E \times MK}$
- Accepts vector
 $\mathbf{x} \in \{0, 1\}^{MK}$ of
concatenated one-hot
encodings
- Combination by additive
vector quantization

$$\psi_{\theta}(\mathbf{x}) := C\mathbf{x} = \mathbf{e}'$$



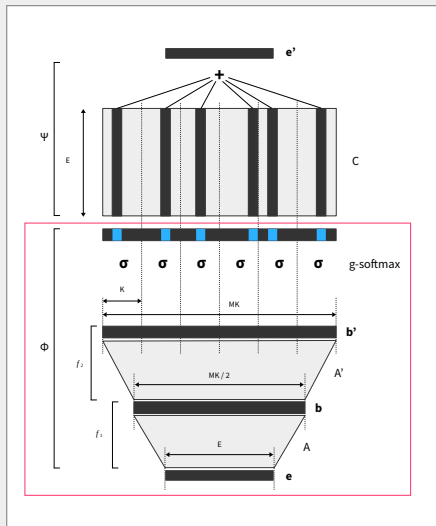
The Neural Embedding Compressor: Encoder (MLP)

$$\phi_{\theta}(\mathbf{e}_t) := \sigma(f_2 \circ f_1)(\mathbf{e}_t)$$

$$f_1(\mathbf{x}) := \tanh(A \cdot \mathbf{x} + \mathbf{b})$$

$$f_2(\mathbf{x}) := \text{softplus}(A' \cdot \mathbf{x} + \mathbf{b}')$$

- Encoder ϕ_{θ} : Two fully connected layers
- Select the arg max per codebook
- Problem: gradient!



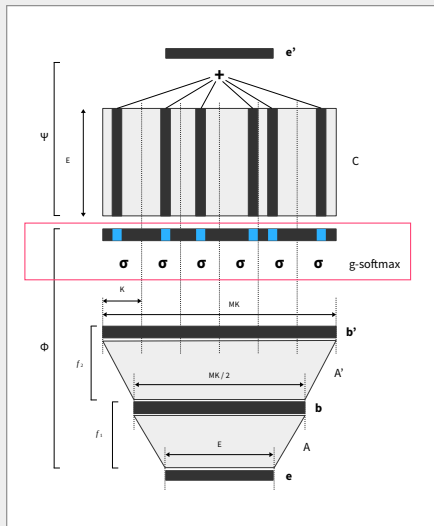
The Neural Embedding Compressor: Gumbel Softmax I

- ❖ Alter softmax function
- ❖ Add Gumbel noise
- ❖ Forces the model to approach categorical distribution

$$\text{g-softmax}(\mathbf{x}_i) := \frac{\exp(f(\mathbf{x}_i))}{\sum_{k=1}^K \exp(f(\mathbf{x}_k))}$$

$$f_{\tau}(\mathbf{x}_k) = \frac{1}{\tau} \cdot (\mathbf{x}_k + \mathbf{g}_k)$$

$$\mathbf{g} = (g_1, \dots, g_K); g_i \sim G$$



- ❖ No redundant information
- ❖ Allow for multiprocessing

Index used for evaluation:

180921_ungol.db.pickle	
Documents	294659
Vocabulary	400000
Code Size	256
Unique Tokens	258391
Avg. Doc. Length	197
Skipped	8
Memory Footprint	~ 3 GB

