Beyond Counting Words: Working with Word Embeddings

Damian Trilling

d.c.trilling@uva.nl
@damian0604
www.damiantrilling.net

12–13 April 2021

Afdeling Communicatiewetenschap
Universiteit van Amsterdam

This part: Embeddings for downstream tasks

In document comparison

In SML

In document comparison

An example (Trilling & van Hoof, 2020)

Let's say we have a large corpus of news articles and what to find those that are about the same events.

Data

- 45K articles
- 6 months
- volkskrant.nl, ad.nl, nu.nl

Step 1: Get candidate articles

Comparing everything with everything is

- computationally infeasible
- theoretical nonsensical

Our solution

- Three-day moving window (but "chaining" possible)
- Saturday/Sunday merged into one day

Step 2: Get similarity scores

How to determine similarity between articles?

Our solution

Compare combinations of

- different measures (in particular, $tf \cdot idf$ cosine similarity vs. softcosine similarity
- different thresholds (to get rid of the overwhelming majority of close-to-zero edges)

Step 3: Network clustering

How to determine events?

After experimenting *a lot*:

Our final solution

- One network for all (instead of one per window)
- Articles are nodes, similarity scores = edge weights
- all edges with weight $<$ threshold removed
- Leiden algorithm (Traag et al., 2019) with Surprise method (Traag et al., 2015) (very suitable for smaller, but more clusters)

Number of articles per event

Table 1: Descriptives for different threshold/similarity combinations

	cosine					softcosine				
	0.2	0.3	0.4	0.5	0.6	0.2	0.3	0.4	0.5	0.6
mean	2.03	1.58	1.35	1.21	1.12	6.78	2.89	1.88	1.51	1.27
std	3.48	2.00	1.22	0.71	0.45	30.41	10.04	4.27	2.27	1.07
max	88	53	41	21	15	551	367	161	70	30
single-art. events	15626	21854	27135	32232	36348	4262	11043	18305	24337	30700
multi-art. events	6685	6777	6241	5165	3899	2460	4736	5961	5940	5257

What does that mean?

- Use a high threshold!
- Soft-cosine finds some more events, leaves less articles unassigned (good), but that comes at the expense of slightly lower precision
- Example from our data: Because soft-cosine "understands" that Nike and Puma are both sports brands, it incorrectly assigned economic coverage about the two to one event.

How correct are the events?

We manually checked 6×100 events, qualitatively (not shown) and quantitatively:

Similarity	Threshold	Prec. 1 (%)	Prec. 2 (%)	TP/max. TP
cosine	0.4	74	88.52	223/268
cosine	0.5	78	89.02	217/253
cosine	0.6	89	94.39	204/225
softcosine	0.4	56	76.20	234/521
softcosine	0.5	65	81.77	236/379
softcosine	0.6	75	86.92	222/289

Note. *Precision 1:* The percentage of news events that are entirely clustered correctly. *Precision 2:* The percentage of news articles that are correctly clustered. *max. TP* is the number of articles that are assigned to an event in the sample; hence, the maximum number of true positives that can be achieved.

Cosine vs Softcosine

Also a matter of computational costs

- the document needs to be converted into embeddings
- but once that is done, our document vectors only have 300 instead of thousands of dimensions!

[let's try this out in a notebook]

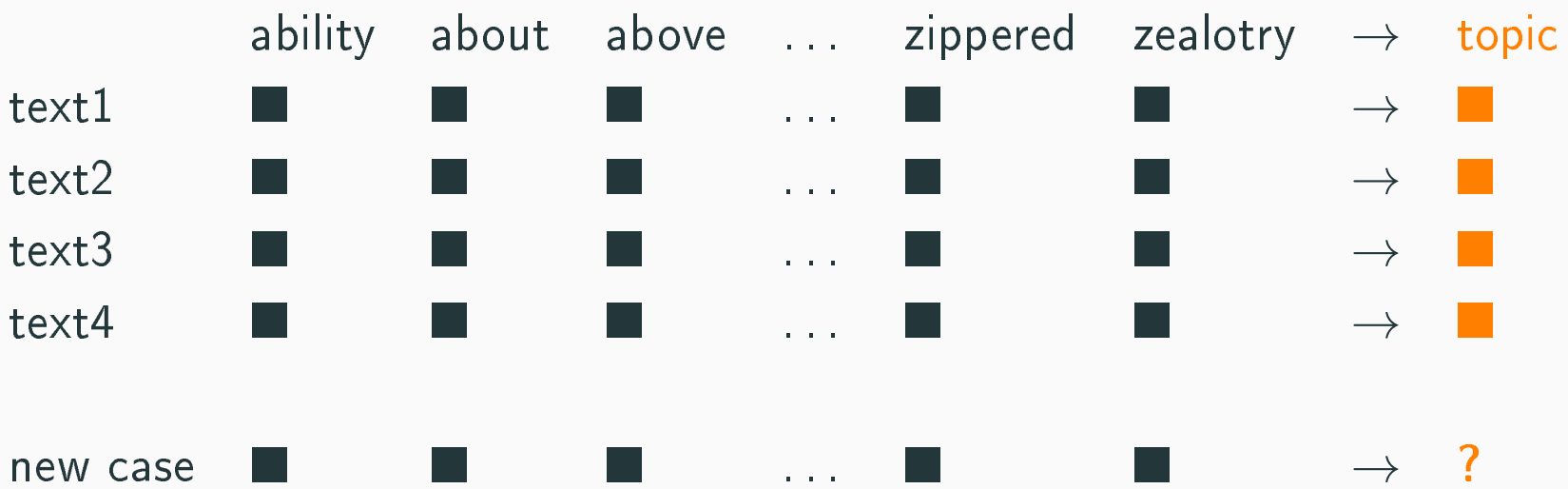
In SML

In classical SML:

- we represent each document by a vector of word frequencies (or tf· idf scores)
- use these vectors to predict the label

In classical SML:

- we represent each document by a vector of word frequencies (or tf· idf scores)
- use these vectors to predict the label



For instance, topic can be [“sports”, “economy”, “politics”] and the other entries are word frequencies

Our BOW approach until now

Representing a document by word frequency counts

Result of preprocessing and vectorizing:

0. `He took the dog for a walk to the dog playground`

⇒ `took dog walk dog playground`

⇒ `'took':1, 'dog': 2, walk: 1, playground: 1`

Consider these other sentences

1. He took the doberman for a walk to the dog playground
2. He took the cat for a walk to the dog playground
3. He killed the dog on his walk to the dog playground

The vectorized representations of these sentences have a "distance" (dissimilarity) of 1 each, but arguably, sentences 0 and 1 should be "closer" than others

The idea

We modify our vectorizer such that

- for each word in the document, we look up its embedding
- we then aggregate these embeddings (e.g., mean, max, or sum)
- For each document, we now have a 300-dimensional instead of a 10,000-dimensional vector[1]

[1] in the case of a 300-dimensional embedding model and a vocabulary size of 10,000 of the traditional CountVectorizer)

The idea

We modify our vectorizer such that

- for each word in the document, we look up its embedding
- we then aggregate these embeddings (e.g., mean, max, or sum)
- For each document, we now have a 300-dimensional instead of a 10,000-dimensional vector[1]

[1] in the case of a 300-dimensional embedding model and a vocabulary size of 10,000 of the traditional CountVectorizer)

The idea

We modify our vectorizer such that

- for each word in the document, we look up its embedding
- we then aggregate these embeddings (e.g., mean, max, or sum)
- For each document, we now have a 300-dimensional instead of a 10,000-dimensional vector[1]

[1] in the case of a 300-dimensional embedding model and a vocabulary size of 10,000 of the traditional CountVectorizer)

What does that mean?

A couple of things:

- Our model is smaller
- We can use words in the prediction dataset *even if it's not in the training dataset*[2]
- We can learn from similar training samples even if they do not use the same words
- But we also may loose some nuance

[2] as long as it's in the embedding model, of course

What does that mean?

A couple of things:

- Our model is smaller
- We can use words in the prediction dataset *even if it's not in the training dataset*[2]
- We can learn from similar training samples even if they do not use the same words
- But we also may loose some nuance

[2] as long as it's in the embedding model, of course

What does that mean?

A couple of things:

- Our model is smaller
- We can use words in the prediction dataset *even if it's not in the training dataset*[2]
- We can learn from similar training samples even if they do not use the same words
- But we also may loose some nuance

[2] as long as it's in the embedding model, of course

What does that mean?

A couple of things:

- Our model is smaller
- We can use words in the prediction dataset *even if it's not in the training dataset*[2]
- We can learn from similar training samples even if they do not use the same words
- But we also may loose some nuance

[2] as long as it's in the embedding model, of course

Let's look at an example

As we see, not *all* embedding models are improving downstream tasks – but good ones can:
https://github.com/annekroon/amsterdam-embedding-model

[explain results in README.md]

[Let's do this ourselves in a notebook]

References

Traag, V. A., Aldecoa, R., & Delvenne, J.-C. (2015). Detecting communities using asymptotical surprise. *Physical Review E*, *92*(2). https://doi.org/10.1103/physreve.92.022816

Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, *9*(1), 1–12. https://doi.org/10.1038/s41598-019-41695-z

Trilling, D., & van Hoof, M. (2020). Between article and topic: News events as level of analysis and their computational identification. *Digital Journalism*, *8*, 1317–1337. https://doi.org/10.1080/21670811.2020.1839352