Word Sense Induction Using Word Sketches

Ondřej Herman(✉), Miloš Jakubíček, Pavel Rychlý, and Vojtěch Kovář

Faculty of Informatics, Masaryk University,
Botanická 68a, 602 00 Brno, Czech Republic
{xherman1,jak,pary,xkovar3}@fi.muni.cz

Abstract. We present three methods for word sense induction based on Word Sketches. The methods are being developed a part of an semiautomatic dictionary creation system, providing annotators with the summarized semantic behavior of a word. Two of the methods are based on the assumption of a word having a single sense per collocation. We cluster the Word Sketch based collocations by their co-occurrence behavior in the first method. The second method clusters the collocations using word embedding model. The last method is based on clustering of Word Sketch thesauri. We evaluate the methods and demonstrate their behavior on representative words.

Keywords: Word sense induction · Word sketch · Collocations · Word embeddings

1 Word Sense Induction

The task of word sense induction (WSI) aims to identify the different senses of polysemous words from bulk text in an unsupervised setting. The problem has a long history, but none of the current solutions yield satisfactory results.

The closely related task of word sense discrimination assigns a specific occurrence of a word within its context to a predefined sense inventory.

Based on the Harris' distributional hypothesis [5], words with similar meanings appear in similar contexts, and therefore different meanings of the same word tend to be present in differing contexts. Insight into the senses of the word can be gained by investigating the contexts the word appears in.

The methods we are looking for are to be used to assist an annotator to properly describe the different senses a word can take on, therefore we would like the method to be transparent and give understandable sense clusters. For this application it is not an issue if more than one cluster consist of the same word sense, as long as a single cluster does not contain a mixture of different senses. For a speaker of the language, joining clusters is easy, while separating them is laborious. Therefore, in the following exposition we specify a higher number of word sense clusters than we expect to occur in the examples, even though the actual amount of investigated senses is likely smaller.

C. Martín-Vide et al. (Eds.): SLSP 2019, LNAI 11816, pp. 83–91, 2019.
https://doi.org/10.1007/978-3-030-31372-2_7

2 Spectral Clustering

The methods described below employ spectral clustering as an important building block. Spectral clustering [9] is a family of techniques which operate on the pairwise similarities between the clustered objects, that is, on the similarity graph. Spectral clustering is based on the eigendecomposition of the graph Laplacian L:

$$L = D - A \tag{1}$$

where D is a diagonal matrix where $d_{ii} \in D$ is the sum of weights of edges coincident with the i-th vertex and A is a non-negative symmetric matrix, where $a_{ij} = a_{ji} \in A$ is the similarity between the i-th and j-th vertex. The number of clusters n is then chosen and the first n smallest eigenvalues and their corresponding eigenvectors are used to project L into a well-behaved space with reduced dimension, in which clusters are easier to find. The usual choice for clustering the reduced space is k-means.

The technique does not depend on clusters of specific shape and does not require the similarity function to satisfy the properties of a metric. The technique is based on standard linear algebra methods, which have been studied deeply and can be implemented in an efficient way.

The usual formulation of spectral clustering requires the number of clusters to be specified beforehand. One commonly used heuristic is based on the eigengap heuristic [9], which selects the position of the first large difference between the eigenvalues of the Laplacian ordered by magnitude as the number of clusters. This method, while simple to implement, has no theoretical basis [9,10]. A more robust (and complex) heuristic is described in [10].

To calculate the clustering, we use the implementation provided by the venerable scikit-learn [8] library, with modifications which allow us to examine the eigenvalues used during the computation.

3 Clustering of Word Sketch Co-occurrences

Word Sketch, as implemented in the Sketch Engine [6] is a summary of the contexts a specific word appears in, collated by different grammatical relations. The Word Sketches are extracted from text employing a collection of regular rules, each of which describe a collocation and the grammatical relation the collocation appears in. The result is a collection of triples of the form (*headword, relation, collocate*). The rules aim to trade precision for recall, so that the resulting relations of a word describe the contexts in which the word appears as completely as possible. Triples which do not satisfy a criterion specified by a co-occurrence metric are discarded, so that only salient triples remain. The Fig. 1 shows the word sketch for the word *palm* calculated from the BNC corpus [3] (Fig. 2).

palm *(noun)* Alternative PoS: verb (freq: 93)
British National Corpus (BNC) v2.2 freq = 1,759 (15.66 per million)

modifiers of "palm"		19.27
coconut	27	10.75
coconut palms		
potted	13	9.70
potted palms		
sweaty	7	8.96
sago	3	8.13
cupped	3	8
cabbage	3	7.74
banana	3	7.31
hand	13	6.75
date	12	6.37
date palms		
open	21	6.19
open palm		
pink	4	6.19
warm	8	6.07
left	5	5.75

nouns and verbs modified by "palm"		30.13
springs	24	10.18
in palm springs		
tree +	173	9.72
palm trees		
frond	13	9.44
palm fronds		
grove	20	9.16
palm groves		
sunday	27	9.02
on palm sunday		
beach	40	9.01
palm beach		
sander	9	8.91
thatch	4	7.78
leaf	6	7.50
oil	22	7.46

verbs with "palm" as object		17.96
grease	8	9.31
slap	12	9.04
sway	8	8.91
outstretch	7	8.64
sweat	4	8.09
rub	10	7.88
kiss	8	7.65
upturn	3	7.65
press	17	7.16
flatten	3	7.16
rest	4	6.88
wipe	5	6.83
wave	5	6.73
place	11	5.37
lift	5	5.27
spread	3	5.18
push	5	5.15

verbs with "palm" as subject		12.05
sweat	8	9.68
damp	4	9
fringe	4	8.93
cup	3	8.46
face	40	7.97
with palms facing upwards		
leave	5	3.83
lie	3	3.58
be	48	0.19
palms were		

"palm" and/or ...		13.08
banana	4	8.11
finger	10	8.08
thigh	3	7.45

prepositional phrases		
"palm" of ...	237	13.47
... in "palm"	110	6.25
... of "palm"	92	5.23
... into "palm"	83	4.72
... with "palm"	71	4.04
... on "palm"	31	1.76
... against "palm"	23	1.31
"palm" in ...	22	1.25
"palm" against ...	17	0.97

Fig. 1. Word sketch for the noun *palm*

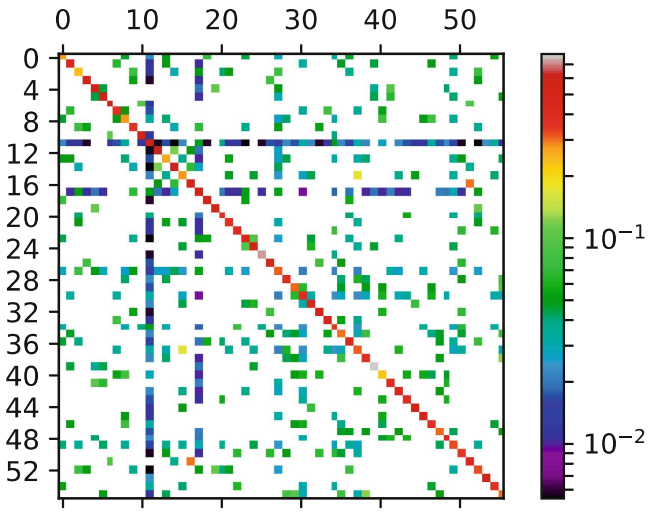

Fig. 2. Document co-occurrences of the Word Sketch collocations of the noun *palm*

The WSI method described in this section is based on the following assumptions. For a specific word,

1. each (*relation, collocate*) pair has a single sense
2. two (*relation, collocate*) pairs co-occurring in a document belong to the same sense

To identify the word senses, a $n \times n$ matrix C is constructed, where $n = |(relation,\ collocate)|$ and c_{ij} is the number of documents in which the i-th and

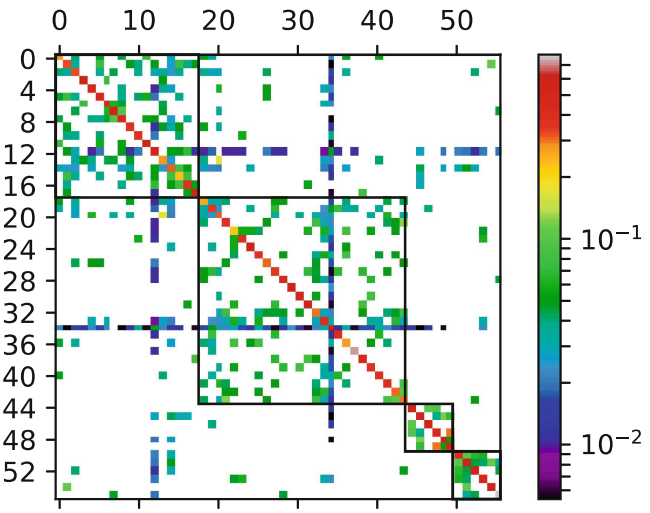

Fig. 3. Clustered document co-occurrences of the Word Sketch collocations of the noun *palm*

j-th (*relation, collocate*) pair appeared together. Only pairs with positive rank are kept and the pairs which do not occur in at least four different documents are discarded.

While C can be clustered directly, better result can be obtained by normalizing it. Using raw counts has tendency to create singleton clusters for collocates which appear in many different contexts, such as prepositions. To reduce the influence, we calculate the Dice-normalized C' as

$$C' = 2C \oslash (C + C^T) \tag{2}$$

where $\oslash$ represents element-wise matrix division. The result obtained by clustering C' using the spectral clustering algorithm with the desired number of clusters set to 4, contain the following (*relation, collocate*) pairs:

Cluster 1 (18 pairs)	**Cluster 2** (26 pairs)
nouns and verbs modified by "palm-n" leaf-n	verbs with "palm-n" as object outstretch-v
verbs with "palm-n" as object sway-v	verbs with "palm-n" as object raise-v
"palm-n" and/or ... flower-n	verbs with "palm-n" as object open-v
nouns and verbs modified by "palm-n" grove-n	"palm-n" of ... hand-n
Cluster 3 (6 pairs)	**Cluster 4** (6 pairs)
modifiers of "palm-n" sweaty-j	nouns and verbs modified by "palm-n" oil-n
verbs with "alm-n" as object wipe-v	nouns and verbs modified by "palm-n" court-n
nouns and verbs modified by "palm-n" springs-n	modifiers of "palm-n" oil-n
nouns and verbs modified by "palm-n" beach-n	nouns and verbs modified by "palm-n" sunday-n

The pairs are shown in the order they appear within the clusters in the Fig. 3. The clusters obtained are mostly pure, with some exceptions, such as *palm-n springs-n* and *palm-n beach-n* appearing in the third cluster. This can be likely alleviated by modifying the construction of the similarity matrix.

The method yields reasonable results for many words and has the ability to provide extract the specific occurrences of the induced senses.

3.1 Clustering of Word Sketch Thesaurus

Thesaurus is a list of words similar a given word. As similar words appear in similar context, their word sketches will be similar, so the similarity of two words can be obtained by calculating the intersection of the word sketches of the two words.

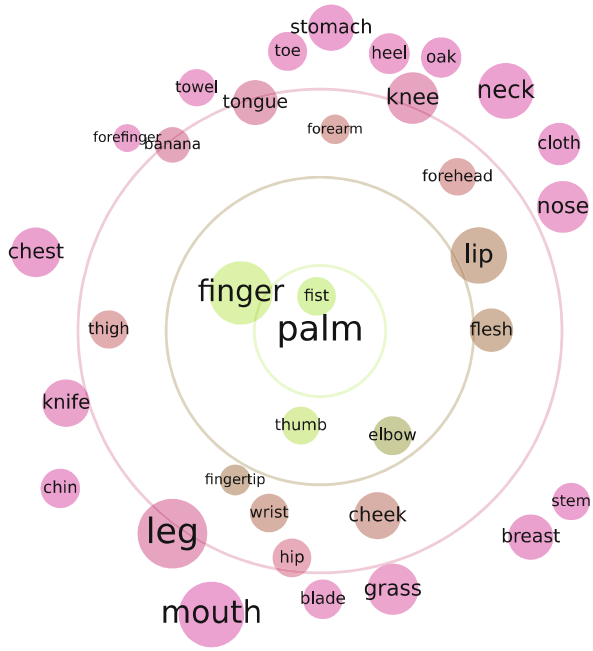

Fig. 4. Thesaurus for the noun *palm*

In Sketch Engine, the thesaurus is calculated by comparing all pairs of word sketches. For illustration, the thesaurus of the noun *palm* is visualized in the Fig. 4.

Based on the assumption that the thesaurus for a polysemous word will contain words similar to the different senses, clustering the words contained in the thesaurus based on their pairwise similarities can give insight into the senses the word can take on.

In this method, the matrix T to be clustered consists of the elements t_{ij}, which give the similarity of the i-th and j-th entries in the thesaurus of the word we are identifying the senses for, in the respective thesauri corresponding to the i-th and j-th entries. For example, the Fig. 5 shows the pairwise similarities of the elements of the thesaurus for the noun *palm*. Extracting 4 clusters yields the result shown in the Fig. 6. The most similar words grouped by cluster are:

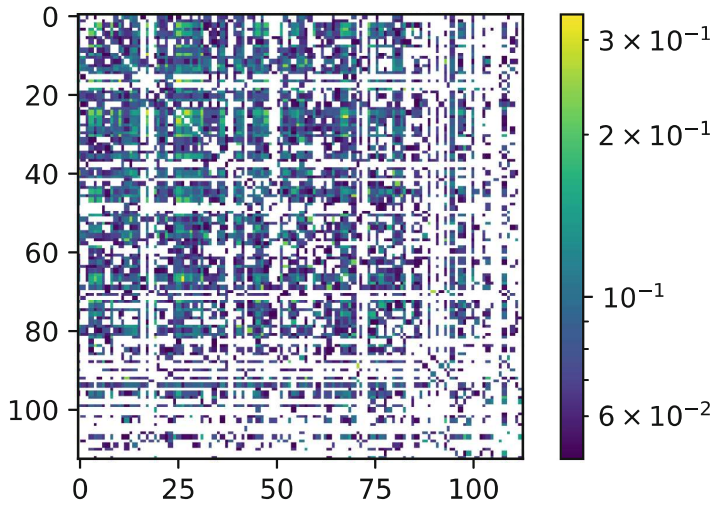

Fig. 5. Pairwise similarities of thesaurus entries for the noun *palm*

max cluster similarity	**cluster representatives**
.081	banana-n oak-n meadow-n fern-n vine-n
.127	fist-n finger-n thumb-n elbow-n flesh-n
.096	fingertip-n forefinger-n outward-n knuckle-n
.063	pine-n willow-n bamboo-n fig-n

This method gives easily interpretable results and seems to be more stable than clustering the word sketch co-occurrences directly and fewer occurrences of the investigated word are necessary to provide a satisfactory result. The drawback is that the information about the senses of specific occurrences in the corpus is not retained. Another issue is that the minimal similarity, for which the word sketch thesaurus items are indexed is .05, which is close to the maximum similarity in some of the obtained clusters, so the more distant or less frequent senses might be hidden below this threshold.

3.2 Clustering Context Word Embeddings

Another approach we investigated is based on clustering the contexts according to the embedding vectors. We calculated skip-gram embeddings of dimension 100 using the fastText package [4] and for every occurrence of the examined word, we calculated the average of the left and right collocate embeddings and used the HDBSCAN [7] algorithm to cluster these vectors. The length of the context to be examined turns out to be a crucial parameter. When the context is too narrow, the clusters are strongly influenced by noise. On the other hand, context which is too wide will not contain enough discriminating information for the clustering algorithm to exploit. For this experiment, we use 10 tokens to the left and 10 tokens to the right of the word.

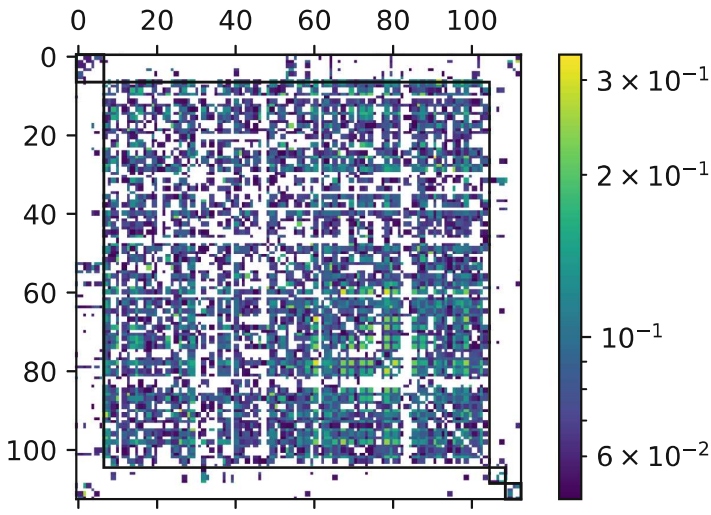

Fig. 6. Clustered pairwise similarities of thesaurus entries for the noun *palm*

HDBSCAN can determine the number of clusters automatically. For the word *palm*, 6 clusters have been found and 93 % of the total vectors had been left unclassified. Three of the clusters are similar, containing mostly repeated text:

`BUST UPLIFT 6. With hands clasped behind, and palms facing inwards, raise the arms 30 times`
`BUST UPLIFT 9. With hands clasped behind and palms facing inwards, raise the arms 30 times`
`BUST UPLIFT 9. With hands clasped behind and palms facing inwards, raise the arms 35 times`

The remaining three clusters consist of three salient senses: palm oil, palm trees, and palm as a body part. While the actual obtained senses are well separated, the result is very sparse and the model hard to inspect, modify and understand. Ensuring that this method works reliably on arbitrary words seems to be difficult.

3.3 Clustering Word Sketches by Word Embeddings

The previous method can be enriched using the information contained within word sketches. Instead of considering each context of the word as a candidate entering the clustering algorithm, we create a single vector for each (*relation, collocate*) pair by averaging the context vectors obtained from the obtained in the same way as in the previous method and then cluster these vectors using HDBSCAN. For the noun *palm*, when using the default configuration of HDBSCAN, all word sketch pairs are discarded by the algorithm. Changing the *alpha* (robust single linkage distance scaling) parameter is reduced to 0.5, discards 55 % of the pairs and the rest is split into two senses, of which the most salient ones according to word sketch rank are:

Cluster 1 (77 pairs)	**Cluster 2** (16 pairs)
"palm-n" of ... hand -n	modifiers of "palm-n" coconut-j
... into "palm-n" nail-n	nouns and verbs modified by "palm-n" tree-n
... into "palm-n" fist-n	modifiers of "palm-n" potted-j
... with "palm-n" forehead-n	"palm-n" from ... stone-n
nouns and verbs modified by "palm-n" springs-n	nouns and verbs modified by "palm-n" frond-n
"palm-n" on ... apron-n	nouns and verbs modified by "palm-n" grove-n
... into "palm-n" bit-n	verbs with "palm-n" as subject fringe-v
verbs with "palm-n" as subject sweat-v	verbs with "palm-n" as object sway-v
... into "palm-n" kiss-n	nouns and verbs modified by "palm-n" thatch-n
... into "palm-n" dig-v	modifiers of "palm-n" cabbage-n

While the senses are well separated, the sparsity could be an issue, as an important sense may have been missed. Additionally, this method employs not only one, but two black boxes: word embeddings and HDBSCAN.

4 Conclusion

While all of the methods give interesting results, we have not evaluated them thoroughly and have not explored the parameter space well at this time.

In addition to the described methods, we implemented the very elegant method based on sparse dictionary learning as described in [1], which aims to decompose the word vector into a weighted sum of *discourse atoms*, but on our data, it failed to yield any interesting result.

We also investigated the Adaptive Skip-gram [2], which trains a word embedding model for multiple senses for each word in a single step. The senses we obtained are reasonable, but we found the model to be too opaque and the information about the sense at a specific corpus position can be reconstructed only inexactly. We devised a method based on clustering word sketch co-occurrences, which is efficient and provides reasonable senses along with the specific corpus positions the senses for a given word appear at.

A different method based on word sketch thesaurus provides better word sense clustering, but has the drawback of not providing the specific positions of the senses.

The two word embedding based methods have shown the ability to produce great results, however they are finicky and difficult to tune and interpret.

Acknowledgments. This work has been partly supported by the Grant Agency of CR within the project 18-23891S and the Ministry of Education of CR within the OP VVV project CZ.02.1.01/0.0/0.0/16_013/0001781 and LINDAT-Clarin infrastructure LM2015071.

References

1. Arora, S., Li, Y., Liang, Y., Ma, T., Risteski, A.: Linear algebraic structure of word senses, with applications to polysemy. Trans. Assoc. Comput. Linguist. **6**, 483–495 (2018)
2. Bartunov, S., Kondrashkin, D., Osokin, A., Vetrov, D.: Breaking sticks and ambiguities with adaptive skip-gram. arXiv preprint arXiv:1502.07257 (2015)

3. Consortium, B., et al.: The British national corpus, version 2 (BNC world). Distributed by Oxford University Computing Services (2001)
4. Grave, E., Mikolov, T., Joulin, A., Bojanowski, P.: Bag of tricks for efficient text classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL, pp. 3–7 (2017)
5. Harris, Z.S.: Distributional structure. Word **10**(2–3), 146–162 (1954)
6. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: Itri-04-08 the sketch engine. Inf. Technol. **105**, 113 (2004)
7. McInnes, L., Healy, J., Astels, S.: hdbscan: hierarchical density based clustering. J. Open Source Softw. **2**(11), 205 (2017)
8. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
9. Von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
10. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: Advances in Neural Information Processing Systems, pp. 1601–1608 (2005)