# Finding the Best Name for a Set of Words Automatically

Pavel Rychlý

Natural Language Processing Centre
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
pary@fi.muni.cz

**Abstract.** Many natural language processing applicatons use clustering or other statistical methods to create sets of words. Such sets group together words with similar meaning and in many cases humans can find an appropriate term quickly. On the other hand computers represent such sets with a meaningless number or ID. This paper proposes an algorithm for automatic finding of names of word sets. It provides result examples as a simple evaluation of the method.

**Keywords:** names of word sets, naming clusters, distributional thesaurus

## 1 Introduction

There are many applications in natural language processing which process words or lemmas and create some sets of words. Usually it is done via some type of clustering but they could be done using many different statistical methods.

As an example of such applications see Figure 1, it presents an thesaurus for word *milk* in the Sketch Engine system [1]. Thesaurus is computed automatically using a distributional similarity method [2]. The individual words which are similar to the given word (*milk*) are clustered using a bottom up clustering. The front words of each cluster is the word with the highest similarity score in the cluster.

The Sketch Engine thesaurus is based on the Word Sketches. These are one page collocational behavior of a word, an exampel of a Word Skech for verb *break* is displayed in Figure 2. In is used mainly in lexicography and language learning. A Word Sketch provides lists of collocations devided into several grammatical relations. On the Figure 2, some collocations are clustered using the same technique as in the Thesarus.

The final example is from LDA-frames project [3], Figure 3. LDA-frames is an unsupervised approach to identifying semantic frames from semantically unlabelled text corpora. There are many frame formalisms but most of them suffer from the problem that all frames must be created manually and the set of semantic roles must be predefined. The LDA-Frames approach, based on the

Aleš Horák, Pavel Rychlý (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2014, pp. 77–81, 2014. 

**milk** *(noun)* **British National Corpus freq = 4692** (41.8 per million)

| Lemma | Score | Freq | Cluster |
|---|---|---|---|
| meat | 0.227 | 3690 | fruit [0.177, 4989] vegetable [0.164, 2714] potato [0.16, 2458] bean [0.134, 1744] rice [0.126, 1537] tomato [0.114, 1465] |
| coffee | 0.222 | 6372 | wine [0.221, 7123] tea [0.202, 8256] beer [0.199, 3629] drink [0.19, 6655] |
| juice | 0.207 | 1883 | salt [0.128, 3263] |
| cream | 0.201 | 3221 | bread [0.198, 3668] sugar [0.196, 3685] cheese [0.195, 2918] butter [0.19, 2062] chocolate [0.153, 2316] |
| egg | 0.191 | 6071 | |
| oil | 0.173 | 10126 | coal [0.108, 5302] gas [0.101, 8082] |
| food | 0.171 | 20774 | fish [0.134, 10322] goods [0.11, 10052] product [0.106, 21606] |
| soup | 0.17 | 1405 | sauce [0.137, 1597] salad [0.112, 1394] |
| water | 0.144 | 34246 | blood [0.133, 9780] |
| cake | 0.143 | 3666 | biscuit [0.13, 1567] sandwich [0.109, 1769] |
| stuff | 0.137 | 6629 | meal [0.114, 6532] |

Fig. 1: Thesaurus of *milk* in the Sketch Engine

Latent Dirichlet Allocation, avoids both these problems by employing statistics on a syntactically tagged corpus. The only information that should be given is a number of semantic frames and a number of semantic roles to be identified.

From all these examples we can see that many clusters clearly define one common meaning. A native speaker could easily choose a single word name for such cluster. This paper presents an algorithm to find such name automatically.

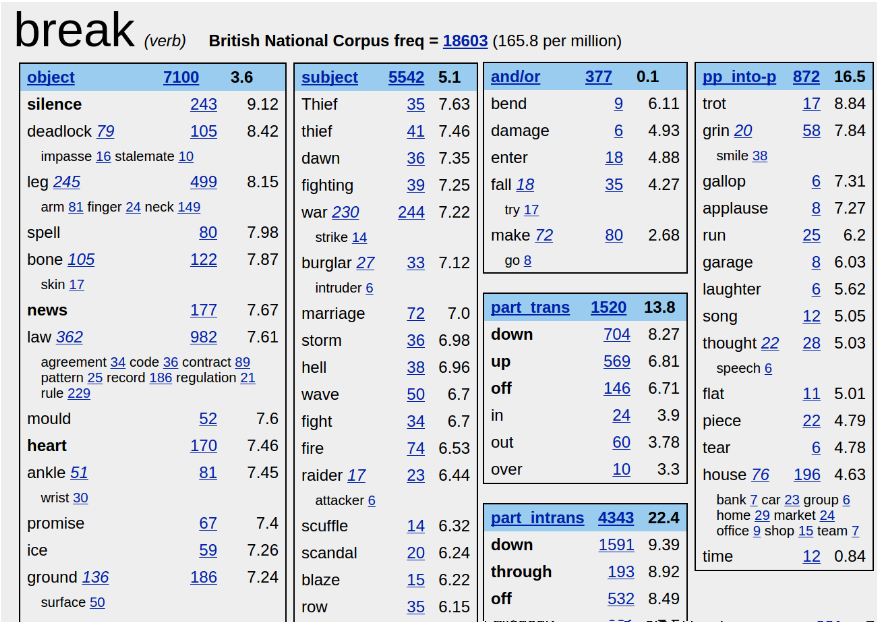
## 2 Proposed Method

The proposed method exploits the distributional thesaurus data which provide a list of similar words for a given word. The algorithm works as follows:

1. for each word in the given set find a list of top similar words in the thesaurus
2. sum the score for each of similar words across all given words
3. add 1 to the sums for each input words (the most similar word for any word is the word itself)
4. sort similar words according to the sums of scores
5. display the top items from the list

## 3 Evaluation

To our knowledge, there are no evaluation data available. We are going to prepare such gold data as a future work. As a simple form of evaluation we list results of the algorithm on our test data. They are presented in Table 1.


break (verb) British National Corpus freq = 18603 (165.8 per million)

| object | 7100 | 3.6 |
|---|---|---|
| **silence** | 243 | 9.12 |
| deadlock 79 | 105 | 8.42 |
| impasse 16 stalemate 10 | | |
| leg 245 | 499 | 8.15 |
| arm 81 finger 24 neck 149 | | |
| spell | 80 | 7.98 |
| bone 105 | 122 | 7.87 |
| skin 17 | | |
| **news** | 177 | 7.67 |
| law 362 | 982 | 7.61 |
| agreement 34 code 36 contract 89 pattern 25 record 186 regulation 21 rule 229 | | |
| mould | 52 | 7.6 |
| **heart** | 170 | 7.46 |
| ankle 51 | 81 | 7.45 |
| wrist 30 | | |
| promise | 67 | 7.4 |
| ice | 59 | 7.26 |
| ground 136 | 186 | 7.24 |
| surface 50 | | |

| subject | 5542 | 5.1 |
|---|---|---|
| Thief | 35 | 7.63 |
| thief | 41 | 7.46 |
| dawn | 36 | 7.35 |
| fighting | 39 | 7.25 |
| war 230 | 244 | 7.22 |
| strike 14 | | |
| burglar 27 | 33 | 7.12 |
| intruder 6 | | |
| marriage | 72 | 7.0 |
| storm | 36 | 6.98 |
| hell | 38 | 6.96 |
| wave | 50 | 6.7 |
| fight | 34 | 6.7 |
| fire | 74 | 6.53 |
| raider 17 | 23 | 6.44 |
| attacker 6 | | |
| scuffle | 14 | 6.32 |
| scandal | 20 | 6.24 |
| blaze | 15 | 6.22 |
| row | 35 | 6.15 |

| and/or | 377 | 0.1 |
|---|---|---|
| bend | 9 | 6.11 |
| damage | 6 | 4.93 |
| enter | 18 | 4.88 |
| fall 18 | 35 | 4.27 |
| try 17 | | |
| make 72 | 80 | 2.68 |
| go 8 | | |

| part_trans | 1520 | 13.8 |
|---|---|---|
| **down** | 704 | 8.27 |
| **up** | 569 | 6.81 |
| **off** | 146 | 6.71 |
| in | 24 | 3.9 |
| out | 60 | 3.78 |
| over | 10 | 3.3 |

| part_intrans | 4343 | 22.4 |
|---|---|---|
| **down** | 1591 | 9.39 |
| **through** | 193 | 8.92 |
| **off** | 532 | 8.49 |

| pp_into-p | 872 | 16.5 |
|---|---|---|
| trot | 17 | 8.84 |
| grin 20 | 58 | 7.84 |
| smile 38 | | |
| gallop | 6 | 7.31 |
| applause | 8 | 7.27 |
| run | 25 | 6.2 |
| garage | 8 | 6.03 |
| laughter | 6 | 5.62 |
| song | 12 | 5.05 |
| thought 22 | 28 | 5.03 |
| speech 6 | | |
| flat | 11 | 5.01 |
| piece | 22 | 4.79 |
| tear | 6 | 4.78 |
| house 76 | 196 | 4.63 |
| bank 7 car 23 group 6 home 29 market 24 office 9 shop 15 team 7 | | |
| time | 12 | 0.84 |

Fig. 2: Word sketch of verb *break* in the Sketch Engine

Table 1: Result of the algorithm on test data.

| input word set | output top names |
|---|---|
| oil coal gas | fuel-n 0.696<br>energy-n 0.536 |
| Britain Scotland Europe England | country-n 4.189<br>area-n 3.308 |
| apple pear orange | fruit-n 2.145<br>thing-n 1.441 |
| procedure study analysis method programme | system-n 5.367<br>work-n 4.959 |
| pint bottle litre gallon | glass-n 2.371<br>water-n 2.258 |
| meat fruit vegetable potato | food-n 3.291<br>fish-n 2.803 |
| village town | city-n 0.611<br>area-n 0.478 |

EAT

| | SUBJECT | | OBJECT | |
|---|---|---|---|---|
| | 222 | | 40 | |
| | 0.794216 | person | 0.085888 | food |
| | 0.010335 | people | 0.046396 | meal |
| | 0.007963 | one | 0.01947 | egg |
| | 0.005797 | man | 0.01947 | breakfast |
| 0.554086 frame 1166 | 0.004342 | who | 0.01726 | lunch |
| | 0.003409 | woman | 0.016846 | dinner |
| | 0.002687 | child | 0.015189 | fish |
| | 0.002519 | that | 0.013256 | meat |
| | 0.002307 | all | 0.012289 | potato |
| | 0.002215 | someone | 0.012151 | cake |
| | 152 | | 40 | |
| | 0.027104 | bird | 0.085888 | food |
| | 0.026926 | dog | 0.046396 | meal |
| | 0.023538 | animal | 0.01947 | egg |
| | 0.023181 | fish | 0.01947 | breakfast |
| 0.128011 frame 622 | 0.016049 | cat | 0.01726 | lunch |
| | 0.014979 | child | 0.016846 | dinner |
| | 0.013374 | people | 0.015189 | fish |
| | 0.01266 | prey | 0.013256 | meat |
| | 0.011947 | man | 0.012289 | potato |
| | 0.011769 | horse | 0.012151 | cake |

Fig. 3: Verb *eat* in LDA-frames

## 4 Interface

The algorithm is implemented as a command line script. It is written in Python and uses the Sketch Engine API to access the thesaurus data. We assume that after more finetuning the algorithm will be included into the Sketch Engine system. An example of a usage is at Figure 4.

```
$ clustname.py bnc2 bnc-hyper n Britain Scotland Europe England
country-n 4.18891489506
area-n 3.50870908797
year-n 3.5038651228
London-n 3.2635447681
world-n 3.13785666227
```

Fig. 4: An example of the clustname.py tool usage.

## 5 Conclusions

We have proposed an algorithm for finding names for a set of words. The implementation is mostly language and corpus independent and works quite well for many test data.

**Acknowledgements** This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013 and by the Czech-Norwegian Research Programme within the HaBiT Project 7F14047.

## References

1. Kilgarriff, A., Rychlý, P., Smrž, P., Tugwell, D.: The Sketch Engine. Proceedings of Euralex (2004) 105–116 `http://www.sketchengine.co.uk`.
2. Rychlý, P., Kilgarriff, A.: An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In: Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, Association for Computational Linguistics (2007) 41–44
3. Materna, J.: LDA-Frames: An Unsupervised Approach to Generating Semantic Frames. In Gelbukh, A., ed.: Proceedings of the 13th International Conference CICLing 2012, Part I. Volume 7181 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2012) 376–387