

Query Expansion

CS5154/6054

Yizong Cheng

9/20/2022

9 *Relevance feedback and query expansion*

SYNONYMY

In most collections, the same concept may be referred to using different words. This issue, known as *synonymy*, has an impact on the recall of most information retrieval systems. For example, you would want a search for **aircraft** to match **plane** (but only for references to an *airplane*, not a woodwork-ing plane), and for a search on **thermodynamics** to match references to **heat** in appropriate discussions. Users often attempt to address this problem themselves by manually refining a query, as was discussed in Section 1.4; in this chapter we discuss ways in which a system can help with query refinement, either fully automatically or with the user in the loop.

Thesaurus, WordNet, Spelling Correction

The methods for tackling this problem split into two major classes: global methods and local methods. Global methods are techniques for expanding or reformulating query terms independent of the query and results returned from it, so that changes in the query wording will cause the new query to match other semantically similar terms. Global methods include:

- Query expansion/reformulation with a thesaurus or WordNet (Section 9.2.2)
- Query expansion via automatic thesaurus generation (Section 9.2.3)
- Techniques like spelling correction (discussed in Chapter 3)

Relevance Feedback (RF)

Local methods adjust a query relative to the documents that initially appear to match the query. The basic methods here are:

- Relevance feedback (Section 9.1)
- Pseudo relevance feedback, also known as Blind relevance feedback (Section 9.1.6)
- (Global) indirect relevance feedback (Section 9.1.7)

9.2 Global methods for query reformulation

In this section we more briefly discuss three global methods for expanding a query: by simply aiding the user in doing so, by using a manual thesaurus, and through building a thesaurus automatically.

9.2.1 Vocabulary tools for query reformulation

Various user supports in the search process can help the user see how their searches are or are not working. This includes information about words that were omitted from the query because they were on stop lists, what words were stemmed to, the number of hits on each term or phrase, and whether words were dynamically turned into phrases. The IR system might also suggest search terms by means of a thesaurus or a controlled vocabulary. A user can also be allowed to browse lists of the terms that are in the inverted index, and thus find good terms that appear in the collection.

9.2.2 Query expansion

[Yahoo!](#) [My Yahoo!](#) [Mail](#) Welcome, **Guest** [\[Sign In\]](#) [Help](#)

[Web](#) | [Images](#) | [Video](#) | [Local](#) | [Shopping](#) | [more](#)

[Options](#)

1 - 10 of about 534,000,000 for **palm** ([About this page](#)) - 0.11 sec.

Also try: [palm trees](#), [palm springs](#), [palm centro](#), [palm treo](#), [More...](#)

SPONSOR RESULTS

[Palm - AT&T](#)
[att.com/wireless](#) - Go mobile effortlessly with the **PALM** Treo from AT&T (Cingular).

[Palm Handhelds](#)
[Palm.com](#) - Organizer, Planner, WiFi, Music Bluetooth, Games, Photos & Video.

[Palm, Inc.](#)
Maker of handheld PDA devices that allow mobile users to manage schedules, contacts, and other personal and business information.
[www.palm.com](#) - [Cached](#)

[Palm, Inc. - Treo and Centro smartphones, handhelds, and accessories](#)
Palm, Inc., innovator of easy-to-use mobile products including **Palm®** Treo_ and Centro_ smartphones, **Palm** handhelds, services, and accessories.
[www.palm.com/us](#) - [Cached](#)

SPONSOR RESULTS

[Handhelds at Dell](#)
Stay Connected with Handheld PCs & PDAs. Shop at Dell™ Official Site.
[www.Dell.com](#)

[Buy Palm Centro Cases](#)
Ultimate selection of cases and accessories for business devices.
[www.Cases.com](#)

[Free Plam Treo](#)
Get A Free **Palm** Treo 700W Phone. Participate Today.
[EvaluationNation.com/treo](#)

► **Figure 9.6** An example of query expansion in the interface of the Yahoo! web search engine in 2006. The expanded query suggestions appear just below the “Search Results” bar.

Thesaurus for Query Expansion

Methods for building a thesaurus for query expansion include:

- Use of a controlled vocabulary that is maintained by human editors. Here, there is a canonical term for each concept. The subject headings of traditional library subject indexes, such as the Library of Congress Subject Headings, or the Dewey Decimal system are examples of a controlled vocabulary. Use of a controlled vocabulary is quite common for well-resourced domains. A well-known example is the Unified Medical Language System (UMLS) used with MedLine for querying the biomedical research literature. For example, in Figure 9.7, **neoplasms** was added to a search for **cancer**. This Medline query expansion also contrasts with the Yahoo! example. The Yahoo! interface is a case of interactive query expansion, whereas PubMed does automatic query expansion. Unless the user chooses to examine the submitted query, they may not even realize that query expansion has occurred.

- User query: cancer
- PubMed query: ("neoplasms"[TIAB] NOT Medline[SB]) OR "neoplasms"[MeSH Terms] OR cancer[Text Word]
- User query: skin itch
- PubMed query: ("skin"[MeSH Terms] OR ("integumentary system"[TIAB] NOT Medline[SB]) OR "integumentary system"[MeSH Terms] OR skin[Text Word]) AND ((("pruritus"[TIAB] NOT Medline[SB]) OR "pruritus"[MeSH Terms] OR itch[Text Word]))

► **Figure 9.7** Examples of query expansion via the PubMed thesaurus. When a user issues a query on the PubMed interface to Medline at <http://www.ncbi.nlm.nih.gov/entrez/>, their query is mapped on to the Medline vocabulary as shown.

- A manual thesaurus. Here, human editors have built up sets of synonymous names for concepts, without designating a canonical term. The UMLS metathesaurus is one example of a thesaurus. Statistics Canada maintains a thesaurus of preferred terms, synonyms, broader terms, and narrower terms for matters on which the government collects statistics, such as goods and services. This thesaurus is also bilingual English and French.
- An automatically derived thesaurus. Here, word co-occurrence statistics over a collection of documents in a domain are used to automatically induce a thesaurus; see Section 9.2.3.
- Query reformulations based on query log mining. Here, we exploit the manual query reformulations of other users to make suggestions to a new user. This requires a huge query volume, and is thus particularly appropriate to web search.

Traditionally Roget's thesaurus has been the best known English language thesaurus (Roget 1946). In recent computational work, people almost always use WordNet (Fellbaum 1998), not only because it is free, but also because of its rich link structure. It is available at: <http://wordnet.princeton.edu>.



WordNet

A Lexical Database for English

What is WordNet

People

News

Use Wordnet Online

Download

Citing WordNet

License and Commercial Use

Related Projects

What is WordNet?

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the creators of WordNet and do not necessarily reflect the views of any funding agency or Princeton University.

When writing a paper or producing a software application, tool, or interface based on WordNet, it is necessary to properly **cite the source**. Citation figures are critical to WordNet funding.

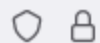
About WordNet

WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the **browser**. WordNet is also freely and publicly available for **download**. WordNet's structure makes it a useful tool for computational

Note

Due to funding and staffing issues, we are no longer able to accept comment and suggestions.

We get numerous questions regarding topics that are addressed on our **FAQ** page. If you have a problem or question regarding something you downloaded from the "**Related projects**" page, you must contact the



NLTK

Search

[NLTK Documentation](#)[API Reference](#)[Example Usage](#)[Module Index](#)[Wiki](#)

Documentation

Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to **over 50 corpora and lexical resources** such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active **discussion forum**.

9.2.3 Automatic thesaurus generation

As an alternative to the cost of a manual thesaurus, we could attempt to generate a thesaurus automatically by analyzing a collection of documents. There are two main approaches. One is simply to exploit word cooccurrence. We say that words co-occurring in a document or paragraph are likely to be in some sense similar or related in meaning, and simply count text statistics to find the most similar words. The other approach is to use a shallow grammatical analysis of the text and to exploit grammatical relations or grammatical dependencies. For example, we say that entities that are grown, cooked, eaten, and digested, are more likely to be food items. Simply using word cooccurrence is more robust (it cannot be misled by parser errors), but using grammatical relations is more accurate.

The simplest way to compute a co-occurrence thesaurus is based on term-term similarities. We begin with a term-document matrix A , where each cell $A_{t,d}$ is a weighted count $w_{t,d}$ for term t and document d , with weighting so A has length-normalized rows. If we then calculate $C = AA^T$, then $C_{u,v}$ is a similarity score between terms u and v , with a larger number being better. Figure 9.8 shows an example of a thesaurus derived in basically this manner,


Word	Nearest neighbors
absolutely	absurd, whatsoever, totally, exactly, nothing
bottomed	dip, copper, drops, topped, slide, trimmed
captivating	shimmer, stunningly, superbly, plucky, witty
doghouse	dog, porch, crawling, beside, downstairs
makeup	repellent, lotion, glossy, sunscreen, skin, gel
mediating	reconciliation, negotiate, case, conciliation
keeping	hoping, bring, wiping, could, some, would
lithographs	drawings, Picasso, Dali, sculptures, Gauguin
pathogens	toxins, bacteria, organisms, bacterial, parasite
senses	grasp, psyche, truly, clumsy, naive, innate

► **Figure 9.8** An example of an automatically generated thesaurus. This example is based on the work in [Schütze \(1998\)](#), which employs latent semantic indexing (see [Chapter 18](#)).

FileEditViewHistoryBookmarksToolsHelp

6.8. Pairwise metrics, Affinities a X

←→↺🏠🔒https://scikit-learn.org/stable/modules/metrics.html📄150%★📧🔔👤☰



6.8. Pairwise metrics, Affinities and Kernels

The `sklearn.metrics.pairwise` submodule implements utilities to evaluate pairwise distances or affinity of sets of samples.

This module contains both distance metrics and kernels. A brief summary is given on the two here.


Distance metrics are functions `d(a, b)` such that `d(a, b) < d(a, c)` if objects `a` and `b` “more similar” than objects `a` and `c`. Two objects exactly alike would have a `d(a, b) = 0`. One of the most popular examples is Euclidean distance. To be a ‘true’

Toggle Menu

File Edit View History Bookmarks Tools Help

6.8. Pairwise metrics, Affinities

← → ↻ 🏠 🔒 https://scikit-learn.org/stable/modules/metrics.html#cosine-similarity 150% ☆ 📧 📧 📧 ☰



6.8.1. Cosine similarity

`cosine_similarity` computes the L2-normalized dot product of vectors. That is, if x and y are row vectors, their cosine similarity k is defined as:

$$k(x, y) = \frac{xy^T}{\|x\| \|y\|}$$

This is called cosine similarity, because Euclidean (L2) normalization projects the vectors onto the unit sphere, and their dot product is then the cosine of the angle between the

Toggle Menu

6.8.2. Linear kernel

The function `linear_kernel` computes the linear kernel, that is, a special case of `polynomial_kernel` with `degree=1` and `coef0=0` (homogeneous). If x and y are column vectors, their linear kernel is:

$$k(x, y) = x^T y$$

6.8.3. Polynomial kernel

The function `polynomial_kernel` computes the degree- d polynomial kernel between two vectors. The polynomial kernel represents the similarity between two vectors. Conceptually, the polynomial kernels considers not only the similarity between vectors under the same dimension, but also across dimensions. When used in machine learning algorithms, this allows to account for feature interaction.

The polynomial kernel is defined as:

$$k(x, y) = (\gamma x^T y + c_0)^d$$

linear_kernel(C, C) is np.matmul(C, C.T)

- Linear_kernel is more efficient than np.matmul in calculating the co-occurrence matrix CC^T , given C the term-document matrix.

```
f = open("bible.txt", "r")  
docs = f.readlines()  
f.close()
```

```
cv = CountVectorizer(binary=True, max_df=0.04, min_df=8)  
C = cv.fit_transform(docs).toarray().T  
r = linear_kernel(C, C)  
np.fill_diagonal(r, 0)
```

O'REILLY®

Blueprints for Text Analytics Using Python

Machine Learning-Based Solutions for
Common Real World (NLP) Applications



Jens Albrecht,
Sidharth Ramachandran
& Christian Winkler

Chapter 5. Feature Engineering and Syntactic Similarity

Finding Related Words

```
tfidf_word = TfidfVectorizer(stop_words=stopwords, min_df=1000)
dt_word = tfidf_word.fit_transform(headlines["headline_text"])
r = cosine_similarity(dt_word.T, dt_word.T)
np.fill_diagonal(r, 0)
voc = tfidf_word.get_feature_names()
size = r.shape[0] # quadratic
for index in np.argsort(r.flatten())[::-1][0:40]:
    a = int(index/size)
    b = index%size
    if a > b: # avoid repetitions
        print('"{}" related to "{}"'.format(voc[a], voc[b]))
```

```
"sri" related to "lanka"
"hour" related to "country"
"seekers" related to "asylum"
"springs" related to "alice"
"pleads" related to "guilty"
"hill" related to "broken"
"trump" related to "donald"
"violence" related to "domestic"
"climate" related to "change"
"driving" related to "drink"
"care" related to "aged"
"gold" related to "coast"
"royal" related to "commission"
"mental" related to "health"
"wind" related to "farm"
"flu" related to "bird"
"murray" related to "darling"
"world" related to "cup"
"hour" related to "2014"
"north" related to "korea"
```

Cosine_similarity Normalizes Co-occurrence

- Instead of simply using CC^T , given C the term-document matrix, or equivalently, performing `linear_kernel(C, C)`, to find pairs of terms with high co-occurrence count, `cosine_similarity` provides a normalized co-occurrence matrix.
- `Cosine_similarity` picks term pairs that are more interesting.

```
# IR9A.py CS5154/6054 cheng 2022
# modified from chapter 5 of Blueprints for text analytics using Python
# CountVectorizer (binary=True) is used to make the term-document matrix A
# cosine similarity is used to find normalized co-occurrence of pairs of terms
# argsort is used to find the pairs with the top scores
# documents where they co-occur are printed out
# Usage: python IR9A.py
```

```
cv = CountVectorizer(binary=True, max_df=0.04, min_df=8)
A = cv.fit_transform(docs).toarray().T
voc = cv.get_feature_names()
voclen, doclen = A.shape
r = cosine_similarity(A, A)
np.fill_diagonal(r, 0)
```

```
for index in np.argsort(r.flatten())[::-1][0:10]:
    a = int(index / voclen)
    b = index % voclen
    if a > b:
        print(r[a][b], voc[a], voc[b])
        for i in range(doclen):
            if A[a][i] > 0 and A[b][i] > 0:
                print(i, docs[i])
```

Co-Occurrence Is Not Synonym

- Synonyms are terms that may never co-occur.
- But their co-occurrence vectors are very similar.
 - They may occur in the same environment but not co-occur.
 - Think about automobile and car.
- One way to find them is first finding the co-occurrence matrix, and then find terms with similar co-occurrence vectors.
- This requires applying linear_kernel (or better, cosine_similarity) twice, second time on its own resulting matrix.
- Assignment 9: find pairs of terms with high co-occurrence similarity but not co-occurring.

```
# IR9B.py CS5154/6054 cheng 2022
# CountVectorizer (binary=True) is used to make the term-document matrix A
# cosine similarity is used to find normalized co-occurrence of pairs of terms
# another cosine similarity is used on the first cosine similarity matrix
# Usage: python IR9B.py
```

```
cv = CountVectorizer(binary=True, max_df=0.04, min_df=8)
A = cv.fit_transform(docs).toarray().T
voc = cv.get_feature_names()
voclen, doclen = A.shape
r = cosine_similarity(A, A)
np.fill_diagonal(r, 0)

s = cosine_similarity(r, r) # second application of cosine_similarity
np.fill_diagonal(s, 0)

for index in np.argsort(s.flatten())[::-1]:
    a = int(index / voclen)
    b = index % voclen
    if a > b: # add a condition so that a and b do not co-occur
        print(s[a][b], voc[a], voc[b])
        if s[a][b] < 0.4:
            break
```