Proceedings

of the KI 2004 Workshop on

“Machine Learning and Interaction for Text-Based Information Retrieval”

TIR-04

Ulm, September 21, 2004

Edited by

Benno Stein
Sven Meyer zu Eißen
Andreas Nürnberger

Contents

Preface

Further Enhancement to the Porter's Stemming Algorithm 7
Fadi Yamout, Rana Demachkieh, Ghalia Hamdan, and Reem Sabra

Boosting for Text Classification with Semantic Features 25
Stephan Bloehdorn and Andreas Hotho

Learning Similarities for Collaborative Information Retrieval 43
Armin Hust

Experiments in Document Clustering using Cluster Specific Term Weights 55
Christian Borgelt and Andreas Nürnberger

Wrapper Generation with Patricia Trees 69
Sven Meyer zu Eissen and Benno Stein

Information Need Assessment in Information Retrieval—Beyond Lists and Queries 77
Frank Wissbrock

Preface

Being in the age of information—so to speak: information flooding—intelligent technologies for information mining and retrieval have become an important as well as exciting field of research. In this connection, methods of text-based information retrieval receive special attention, which results from the fundamental role of written text, but also because of the high availability of the Internet. E.g., information retrieval methods have the potential to improve the quality of the standard keyword search; moreover, they strike a path to the new developments from the field of the Semantic Web.

There are various techniques and methods being used for text-based information retrieval tasks, which stem from different research areas: machine learning algorithms, models from computer linguistics and psychology, paradigms from the field of user interaction and modeling, or algorithms for information visualization. The development of powerful retrieval tools requires the combination of these developments, and in this sense the workshop shall provide a platform that spans the different views and approaches.

The following list gives examples from classic and ongoing research topics from the field of text-based information retrieval: document models and similarity measures for special retrieval tasks, automatic category formation, topic identification and auto-abstracting, plagiarism analysis, ontologies and the Semantic Web, concepts and techniques for information visualization, user modeling and interaction for particular retrieval tasks, evaluation and construction of test collections.

Further Enhancement to the Porter's Stemming Algorithm

Fadi Yamout[1], Rana Demachkieh[1], Ghalia Hamdan[1], Reem Sabra[1]
[1] Faculty of Computer Sciences
C&E American University I., Beirut, Lebanon
Email: fyamout@inco.com.lb

Abstract. Stemming algorithms are used to transform the words in texts into their grammatical root form, and are mainly used to improve the Information Retrieval System's efficiency. Several algorithms exist with different techniques. The most widely used is the Porter Stemming algorithm. However, it still has several drawbacks, although many attempts were made to improve its structure. This paper reveals the inaccuracies encountered during the stemming process and proposes the corresponding solutions.

1. Introduction

Finding information is not the only activity that exists in an Information Retrieval (IR) system. Indexing, for instance, refers to how information and user's requests from the system are represented. We will refer to the information to be indexed as documents. Hence, documents are represented through a set of index terms or keywords. The terms are extracted from the text of the documents. This might be done automatically or generated by a specialist.

It was estimated in Kowalski [1] that for relatively short documents (e.g., 300-500 words) it normally takes a specialist at least five minutes per item to produce the terms, while it takes just a few seconds on a moderate computer. The extracted terms are mainly nouns since they describe better the semantic of the documents while adjectives, adverbs, and connectives (including transitions, conjunctions…) are less useful because they work mainly as complements.

These irrelevant terms are usually placed in a file called Stoplist. A Stoplist algorithm is applied to all the documents in the collection with an objective to eliminate the terms that have little value to the system. In addition, a word, which occurs in 80% of the documents in the collection, is useless [2]. An example of 425 stopwords is shown in a list in Frakes and Baeza-Yates [2]. The remaining terms are stemmed using Porter's algorithm [3], which brings down distinct words to their grammatical root and thus reduces further the number of unique terms.

Many attempts were made to improve the structure of the Porter algorithm [4], however, it still has several drawbacks. In this paper, further improvements are

introduced to overcome these problems in order to enhance the stemming process. We will refer to the existing Porter algorithm as Porter 2002 and the new as Porter 2004.

2. Porter's Algorithm

Porter Stemming Algorithm was developed by Martin Porter at the University of Cambridge in 1980 and was first published in Porter, M.F., [5] and reprinted in Sparck, Karen, and Peter [6]. As described in the publication, "*The Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems*". Since then it has been very widely used and coded in various programming languages. It is based mainly on stemming operations that remove suffixes from words, such as gerunds (motoring → motor), plurals (cats → cat), and replacing words ending with "ator" for example with "ate" (operator → oper), etc….

These operations are classified into rules where each of these rules deals with a specific suffix and having certain condition(s) to satisfy. A given word's suffix is checked against each rule in a sequential manner until it matches one, and consequently the conditions in the rule are tested on the stem that may result in a suffix removal or modification.

3. Drawbacks of the Porter Algorithm

Natural languages are not completely regular constructs, and therefore stemmers operating on natural words unavoidably make mistakes. For instance, words, which are distinct, may be wrongly conflated to give similar stems (ex: design → design; designate → design, etc...) and affect seriously the retrieval performance of an IR system since the semantic of the word is expressed differently; these are known as over-stemming errors. On the other hand, words which ought to be merged together may remain distinct after stemming (ex: characterizes → character; characteristic → characterist, etc…); these are known as under-stemming errors and do not affect the retrieval performance of an IR system [2]. In this paper we deal with over-stemming errors.

The modified Porter algorithm was tested on 23,531[1] words provided by Porter and compared to an already existing output provided from the same site. In addition, it was tested on 45,000 words extracted from the Oxford's dictionary, and the following over-stemming errors were observed:

[1] http://www.tartarus.org/~martin/PorterStemmer/index.html

Error #1:
The non-existence of "e" at the end of the words that have m=1 and begin with a consonant, and end with two consonants; for ex: paste, loathe…:

Paste → past
Past → past

Error #2:
The removal of "s" in step1 from words ending with "is" such as his and appendicitis:

Appendicitis → append
Append → append

Error #3:
Words ending with "yed" and "ying" and having different meanings may end up with the same stem:

Dying → dy (impregnate with dye)
Dyed → dy (passes away)

Error #4:
The removal of "ic" or "ical" from words having m=2 and ending with a series of consonant, vowel, consonant, vowel, such as generic, politic…:

Political → polit
Politic → polit
Polite → polit

Error #5:
The removal of the suffix "ative" from all words ending with it and having m=1 or m=2, the thing that leads to serious conflicts:

Combative → comb Generative → gener
Comb → comb General → gener

Error #6:
The removal of the suffix "ness" from all words where m=1 and end with consonant, vowel, consonant (cvc) such as witness:

Witness → wit
Wit → wit

Error #7:
The suffix "al" is removed from all words where m=2 e.g. admiral, animal…:

Admiral → admir
Admire → admir

Error #8:
The elimination of the suffix "eer" from words with m=2 such as engineer:

Engineer → engin
Engine → engin

Error #9:
The exclusion of the suffix "ible" from all words where m=2 starting by a consonant and not ending with a series of consonant, vowel, consonant, vowel, such as responsible:

Responsible → respons
Response → respons

Error #10:
The exclusion of the suffix "ance" from words with m=2 ending with a series of consonant, vowel, consonant, vowel:

Severance → sever
Several → sever

Error #11:
The removal of the suffix "ment" from all words even those ending with "iment" having m=2 and not ending with a series of consonant, vowel, consonant, vowel; e.g. experiment:

Experiment → experi
Experience → experi

Error #12:
The elimination of "ion" from all words where m=2 and not consonant, vowel, consonant, vowel, without replacement:

Secretion → secret
Secret → secret

Error #13:
The removal of the suffix "ate" or "nate" from all words where m=2 and ending with a series of consonant, vowel, consonant, vowel:

Designate → design
Design → design

Error #14:
The elimination of the suffix "ize" from all words having m=2 and starting by a consonant, and ending with a series of consonant, vowel, consonant, vowel:

Colonize → colon
Colon → colon

Error #15:
The exclusion of "itive" from words with m= 1 and starting by consonant, and ending with a series of consonant, vowel, consonant, vowel:

Positive → posit
Position → posit

Error #16:
The removal of "iti" from all words where m=2 starting by a vowel and ending with a series of consonant, vowel, consonant, vowel:

Ameniti → amen
Amen → amen

The removal of "iti" from all words where m=3 starting by a vowel and not ending with a series of consonant, vowel, consonant, vowel:

Universiti → univers
Universe → univers

4. Modifications

The following section describes the corresponding solutions for each of the errors revealed previously (Table 1 describes the symbols used).

k	: Pointer to the last letter in the word
m()	: Counts how many consecutive vowel, consonant exist in a word
cons()	: Checks whether the letter at a certain position is a consonant or not
ends()	: Determines if the word ends with the variable sent and consequently truncates this variable from the original word

Table1: Symbol's Intuitions

Solution #1:

To solve the problem ending with "e" a function is created to keep the "e" at the end of the word by returning false
If m=1:
Starts with a consonant and ends with two consonants
Paste, loathe, and bottle.

While adding this method, another problem arises for the words such as beaches, bushes…, so an additional statement is added to step1: If the word ends with "ches" or with "shes" the program will remove the "es" since in step6 the cvd method is used.

Beaches → beach
Bushes → bush

The cvd method is as follow:

```
function cvd (int d)
  if cons(d) then
    d := d-1;
    if !cons(d) and d!=0 then
       while !cons(d) and d>0 do
          d := d-1;
       if cons(d) then return true;
    return false;
  return true;
```

Step1:

```
  if ends("ches") or ends("shes") then   k := k-2;
```

Solution #2:

If the word ends with “is”, the “s” is not deleted

Appendicitis → appendicitis

The statement is:

```
if ends("is")
```

Solution #3:

To prevent words ending with “ying” and “yed”, and having different meanings, from producing the same stem, the “ying” will be set to “i” if it has m=0, starting with consonant and vowel.

Dying → di;

Dyed → dy;

The statements are:

```
if ends("ying) then
  if m()=0 and cons(0) and !cons(1) then setto("i");
```

Solution #4:

Usually the words that end by “ic” in step3 or “ical” in step4 must be removed but in other cases it must not. Therefore, if the word is of size m = 2 and consists of a series of consonant, vowel, consonant, vowel, it is replaced by “ica*” rather than being removed, then in step5 it is transformed to “ic”.

polite→ polit,

political→ politic,

political→ politic

The statements are:

Step3:

```
case 'i':  if ends("ic") and m()=2 then
             while k > 0 do
               if cons(k) and !cons(k-1) then k  :=  k – 2;
               else j  :=  j + 2; k  :=  j; break;
             if k <= 0 then r("ica*");
             break;
           else break;
```

Step4:

```
case 'l':  if ends("ical") then
             if m() = 2 then
               while k > 0 do
                 if cons(k) and !cons(k-1) then k  :=  k – 2;
                 else k  :=  j + 2; r("ic"); break;
             if  k <= 0 then r("ica*"); break ;
             else r("ic"); break;
```

Step5:

```
if ends ("ica*") then r ("ic"); j  :=  j + 2; break;
else j  :=  k; break;
```

Solution #5:

If the word ends by “ative” and m = 2, it is replaced by “ate”.

Generative→ generate

Or if it is m > 2 it is removed.

Authoritative→ authorit

Or if m = 1 it is replaced by “at”.

Combative→ combat

The statements are:

```
if ends("ative") then
  if  m() = 2 then r("ate");
  else if  m() =1 then r("at");
  else if  m() > 2  then r("");
```

Solution #6:

If the word ends with “ness”, m = 1, and ends with consonant, vowel, and a consonant, it is kept as it is.

Witness→witness

Else it will be removed.

The statements are:

```
case 's': if ends("ness") then
            if m() == 1 and cvc(k-4) then break;
            else r("");
            break;
          break;
```

Solution #7:

If it ends by "iral" and m = 2 it is left as it is.

Admiral→ admiral.

Or if it ends by "al", m = 2, and it consists of a series of consonant, vowel, consonant, vowel, it is removed

General→ gener

Admiral→ admiral

Else if m >1 it is removed

The statements are:

```
case 'a': if  ends("al") then
            if  m() = 2 then
              if ends ("iral") then j  :=  j + 4; break;
              p  :=  p – 2;
              while ( p > 0 )  do
                 if cons (p) and !cons(p-1) then p  :=  p – 2;
                 else k  :=  j; break;
              if  p <= 0 then j  :=  j + 2;
            else if  m() > 1  then k  :=  j; break;
```

Solution #8:

If it ends with "eer" and m = 2, then only the "r" is removed in step4 and consequently the last "e" is removed in step6

Engineer→ engine

The statements are:

```
case 'e': if ends("er") then
            if m()=2 and ends ("eer")  then j  :=  j + 2; break;
            else break;
          return
```

Solution #9:
If it ends with "ible", m = 2, and starts with a consonant and not ending with a series of consonant vowel consonant vowel, then it is kept as it is.

Responsible → responsible.
Reducible → reduc

Or if m> 1 it is removed.

Reprehensible → reprehens

The statements are:

```
if ends("ible") then
  if m()=2 and cons(0) then
    p := p – 4;
    while p > 0 do
      if cons (p) and !cons (p-1) then p := p – 2;
      else j := j + 3; break;
    if p <= 0 then k := j; break;
    else k := j;
  else if m() > 1 then k := j; break;
```

Solution #10:
If it ends with "ance", m = 2, and consist of a series of consonant, vowel, consonant, vowel, therefore, it is replaced by "e".

Severance→severe,

If not, it is removed.

Importance→import

The statements are:

```
case 'c': if ends ("ance") then
            if m() = 2 then
              p := p – 4;
              while p > 0 do
                if cons (p) and !cons(p-1) then p := p – 2;
                else k := j; break;
              if p <= 0 and cons(0) then b[j := j+1]='e'; k := j; break;
              else k := j; break;
            if m() > 1 then k := j; break;
```

Solution #11:
If it ends with "iment", m = 2, and not ending with a series of consonant vowel consonant vowel, therefore, it is left as it is.

Experiment→ experiment

Or if m> 1 it is removed.

Accompaniment→ accompani

The statements are:

```
if ends("iment") and m() = 2 then
   p  :=  p - 5
   while  p > 0 do
      if cons(p) and !cons(p-1) then p  :=  p – 2;
      else break;
   if p>0 then j  :=  j + 5; break;
if ends ("ement") then break;
if ends ("ment") then break;
```

Solution #12:

If it ends with “tion”, m = 2, and not ending with a series of consonant vowel consonant vowel…, it is replaced with an “e”.

Secretion→secrete

Sedition→sedit

Or if m> 1 it is removed. The statements are:

```
if ends("ion") and j >= 0 then
   if b[j] = 't' then
      if m()= 2 then
      p  :=  p – 3;
      while  p > 0 do
         if cons (p) and !cons (p-1) then  p  :=  p – 2;
         else b[j := j+1]  :=  'e'; k  :=  j; break;
      if p <= 0 then k  :=  j; break;
   else if  m() > 1 then k  :=  j; break;
```

Solution #13:

If it ends with “nate” or “ate”, m = 2, and ends with a series of consonant vowel consonant vowel…, it is not replaced.

Designate→designate

Or if m> 1, then it is removed.

Collaborate→ collabor

Or if m = 1, then the “at” is kept

Situate→ situat

Or if m = 0, then it is left as it is.

Ate→ate

The statements are:

```
case 't': if ends("nate") and m() = 2 then
            p := p - 4
            while p > 0 do
              if cons (p) and !cons (p-1) p := p – 2;
              else k := j + 1; break;
            if p <= 0 and cons(0) then j := j + 4; break;
          else if ends("ate") then
            if m() = 2 then
              p := p - 3
              while p > 0 do
                if cons (p) and !cons(p-1) then p := p – 2;
                else k := j; break;
              if p<=0 and cons(0) then j := j + 3; break;
              else   k := j; break;
            else if m() > 1 then k := j; break;
            else if m() = 1 then j := j + 2; k := j; break;
            else j := j + 3; break;
```

Solution #14:

If it ends with "ize", m = 2, and starts with a consonant, and ends with a series of consonant, vowel, consonant, vowel…, it is kept as it is:

Colonize→colonize

Or if m> 1 it is removed.

Aerosolize→ aerosol

The statements are:

```
case 'z': if ends("ize") then
            if m() = 2 then
              p := p – 3;
              while p > 0 do
                if cons(p) and !cons(p-1) then p := p – 2;
                else k := j; break;
              if p <= 0 and cons(0) then
                j := j + 3; break;
              else k := j; break;
            else if m() > 1 then k := j; break;
```

Solution #15:

If it ends with "itive", m = 1, starts with a consonant, and ends with a series of consonant, vowel, consonant, vowel…, it is kept as it is:

Positive→ positive

Or if m> 1 it is removed.

Acquisitive→ acquisit

The statements are:

```
case 'v': if ends("itive") and m() = 1 and cons(0) Then
              p  :=  p – 5;
              while p > 0 do
                 if cons(p) and !cons(p-1) then p  :=  p – 2;
                 else j  :=  j + 2; k  :=  j; break;
              if p <= 0 and cons(0) then j  :=  j + 5; break;
              else    k  :=  j; break;
           else if ends("ive") break;
return;
```

Solution #16:
If it ends with "iti", m = 2, starts with a vowel, and ends with a series of consonant, vowel, consonant, vowel…, it is kept as it is:

amenity→ ameniti

If it ends with "iti", m = 3, starts with a vowel, and ends with a series of consonant, vowel, consonant, vowel…, it is kept as it is:

Universiti→ universiti

Or if m> 1 it is removed

Minority→ minor

The statements are:

```
if ends ("iti")  then
   if  m() = 2 then
      p  :=  p – 3;
      while  p > 0 do
         if cons (p) and !cons(p-1) then p  :=  p – 2;
         else k  :=  j; break;
      if  p <= 0 and !cons(0)  then j  :=  j + 3; break;
      else k  :=  j; break;
   else if m() = 3 and !cons(0)  then
      p  :=  p - 3
      while  p > 0  do
         if  cons(p)  and !cons (p-1) then p  :=  p – 2;
         else j  :=  j + 3; break;
      if  p <= 0 and !cons(0) then k  :=  j; break;
      else k  :=  j; break;
   else if  m() > 1 then k  :=  j; break;
```

Exceptions:
Some of the words are considered as exception to the previously described rules, and therefore are treated separately. The following step contains the words that must keep their "e" while removing the "ing" or the "ed".

Loathing → loathe

Pasted → paste

```
function step0()
      String s1=new String(b);
      String s2=new String("rang secret loath past us butt");
      if s2.regionMatches(s2.indexOf(b[0]),s1,0,j+1) then
         return true;
      else return false;
```

5. Experiments

The previously described solutions produce different results than the existing Porter algorithm. Outputs from both Porter 2002 and 2004 are put alongside in Appendix to demonstrate the dissimilarities.

The two techniques were tested against CISI [7], which is a standards test collection that contains 1460 documents, in an attempt to move more relevant documents (the ones found in the queries' relevance judgments) further up the ranking. The result showed a slight improvement (1.5%) in precision and recall, however for some queries the improvement was 2.5%. The percentage is computed as an average for the precision and recall produced by the 30 queries that come with the collection. The results are illustrated in Figure 1 using the 11-point average curve.

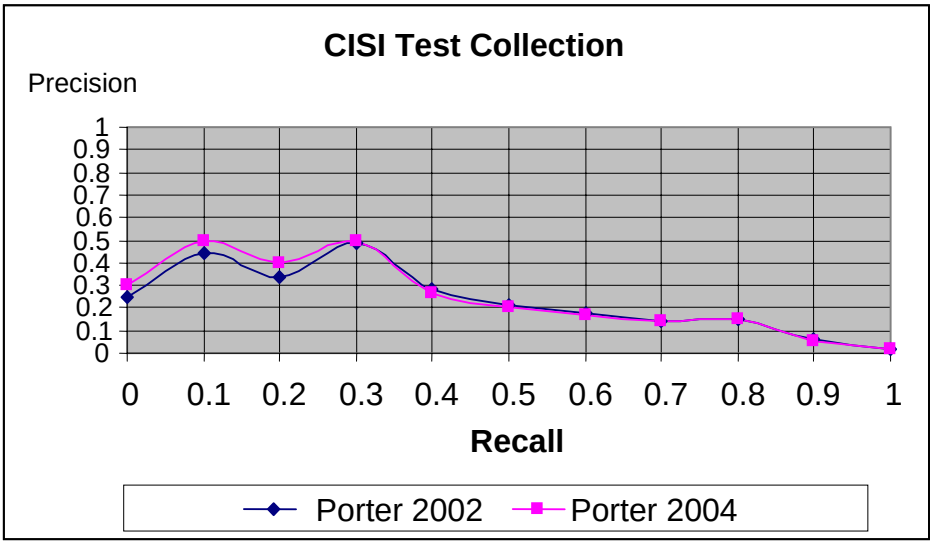

Figure 1: 11 point Average Curve

References

1. Kowalski G. (1997) “Information Retrieval Systems: Theory and Implementation”, Kluwer Academic Publisher, 1997. id387
2. Baeza-Yates R. and Ribeiro-Neto B. (1999) “Modern Information Retrieval”. New York: Addison Wesley
3. Frakes W. B. and Baeza-Yates R. (1992) “Information Retrieval: Data Structures and Algorithms”. Englewood Cliffs, NJ: Prentice-Hall. id175
4. Porter, M.F., (2002) “Developing the English Stemmer”, http://snowball.tartarus.org/.
5. Porter, M.F., (1980), “An Algorithm for Suffix Stripping”, Program, 14(3) :130-137.
6. Sparck Jones, Karen, and Peter Willet, (1997), “Readings in Information Retrieval”, San Francisco: Morgan Kaufmann, ISBN 1-55860-454-4.
7. CISI-collection. The CISI reference collection for information retrieval. 1460 documents and 30 queries, http://local.dcs.gla.ac.uk/idom/ir_resources/test-collections/cisi/, 1981

Appendix: Dissimilarities between existing and new algorithm

Word	Porter02	Porter04
abl	abl	abl
able	abl	able
ach	ach	ach
ached	ach	ach
aches	ach	ach
aching	ach	ach
ache	ach	ache
ad	ad	ad
add	add	add
added	ad	add
adding	ad	add
adds	add	add
abl	abl	abl
able	abl	able
ach	ach	ach
ached	ach	ach
aches	ach	ach
aching	ach	ach
ache	ach	ache
ad	ad	ad
add	add	add
added	ad	add
adding	ad	add
adds	add	add
admirable	admir	admir
admirably	admir	admir
admiration	admir	admir
admire	admir	admir
admired	admir	admir
admirer	admir	admir
admirers	admir	admir
admires	admir	admir
admiring	admir	admir
admiringly	admir	admir
admiral	admir	admiral
amen	amen	amen
amenable	amen	amen
amenities	(none)	amen
amenity	(none)	amen
and	and	and
ande	and	ande
andes	andes	ande
animate	anim	anim
animated	anim	anim
animates	anim	anim
animating	anim	anim
animation	anim	anim
animal	anim	animal
animalized	anim	animal
animals	anim	animal
Ann	ann	ann
anne	ann	anne

Word	Porter02	Porter04
bath	bath	bath
bathed	bath	bath
bathing	bath	bath
baths	bath	bath
bathe	bath	bathe
bathes	bath	bathe
bell	bell	bell
belled	bell	bell
belling	bell	bell
bells	bell	bell
belle	bell	belle
bonn	bonn	bonn
bonne	bonn	bonne
born	born	born
borne	born	borne
brown	brown	brown
browning	brown	brown
browns	brown	brown
browne	brown	browne
bush	bush	bush
bushes	bush	bush
bushe	bush	bushe
call	call	call
called	call	call
calling	call	call
calls	call	call
calle	call	calle
cloth	cloth	cloth
clothed	cloth	cloth
clothing	cloth	cloth
cloths	cloth	cloth
clothe	cloth	clothe
clothes	cloth	clothe
cross	cross	cross
crossed	cross	cross
crosses	cross	cross
crossing	cross	cross
crosse	cross	crosse
dank	dank	dank
danke	dank	danke
design	design	design
designed	design	design
designer	design	design
designing	design	design
designs	design	design
designates	design	designat
designation	design	designat
ear	ear	ear
eared	ear	ear
earings	ear	earing
ell	ell	ell
elle	ell	elle

Word	Porter02	Porter04
elles	ell	elle
engine	engin	engin
engines	engin	engin
engineer	engin	engine
engineering	engin	engine
even	even	even
evening	even	even
evenly	even	even
evenness	even	even
evenings	even	evening
fill	fill	fill
filled	fill	fill
filling	fill	fill
fills	fill	fill
fille	fill	fille
fort	fort	fort
forts	fort	fort
forte	fort	forte
forty	forti	forti
fortis	forti	fortis
front	front	front
fronted	front	front
fronting	front	front
fronts	front	front
fronte	front	fronte
funeral	funer	funeral
funerals	funer	funeral
funereal	funer	funere
futur	futur	futur
future	futur	future
futures	futur	future
gang	gang	gang
ganging	gang	gang
gangs	gang	gang
ganges	gang	gange
generous	generous	gener
generousness	(none)	gener
generously	generous	gener
general	general	general
generalities	general	general
generality	general	general
generalization	general	general
generally	general	general
generality	(none)	general
generalizations	(none)	general
generalize	(none)	general
generalized	(none)	general
generalizer	(none)	general
generalizers	(none)	general
generalizes	(none)	general
generalizing	(none)	general
generals	general	general

Word	Porter02	Porter04
generate	generat	generat
generated	generat	generat
generation	generat	generat
generates	(none)	generat
generating	(none)	generat
generative	(none)	generat
generator	(none)	generat
generators	(none)	generat
generations	generat	generat
generic	generic	generic
generically	(none)	generic
goeth	Goeth	goeth
goethe	goeth	goethe
grande	grand	grande
grandee	grande	grande
grandees	grande	grande
hand	hand	hand
handed	hand	hand
handful	hand	hand
handfuls	hand	hand
handing	hand	hand
hands	hand	hand
hande	hand	hande
hast	hast	hast
haste	hast	haste
her	her	her
hers	her	her
herrings	her	herring
hing	hing	hing
hinges	hing	hinge
host	host	host
hosts	host	host
hoste	host	hoste
however	howev	howev
howeve	howev	howeve
iron	iron	iron
ironed	iron	iron
ironing	iron	iron
irons	iron	iron
irony	ironi	ironi
ironical	iron	ironic
ironically	iron	ironic
later	later	later
lateral	later	lateral
laterally	later	lateral
loath	loath	loath
loathe	loath	loathe
loathed	loath	loathe
loathing	loath	loathe
lungs	lung	lung
lunge	lung	lunge
missy	missi	missi
missis	missi	missis
mond	mond	mond
monde	mond	monde

Word	Porter02	Porter04
mont	mont	mont
monte	mont	monte
montes	mont	monte
numerous	numer	numer
numerical	numer	numeric
of	of	of
off	off	off
offing	of	off
offe	off	offe
past	past	past
pasted	past	paste
moral	moral	moral
morality	moral	moral
moralities	moral	moral
morale	moral	morale
morally	moral	moral
morals	moral	moral
petulance	petul	petule
petulant	petul	petul
petulantly	petul	petul
petulance	petul	petule
petulant	petul	petul
petulantly	petul	petul
picture	pictur	picture
pictured	pictur	pictur
pictures	pictur	picture
picturing	pictur	pictur
pierce	pierc	pierce
pierced	pierc	pierc
pierces	pierc	pierce
piercing	pierc	pierc
piercingly	pierc	pierc
position	posit	posit
positions	posit	posit
positive	posit	positiv
positively	posit	positiv
positiveness	posit	positiv
private	privat	privat
privateer	privat	private
privately	privat	privat
privation	privat	privat
privations	privat	privat
proceed	proceed	proce
proceeds	proce	proceed
rang	rang	rang
range	rang	range
ranged	rang	range
rangees	range	range
ranges	rang	range
ranging	rang	range
regal	regal	regal
regale	regal	regale
regaled	regal	regal
regaling	regal	regal
response	respons	respons

Word	Porter02	Porter04
responsibilities	respons	responsibl
responsibility	respons	responsibl
responsible	respons	responsibl
responsive	respons	respons
roll	roll	roll
rolle	roll	rolle
rolled	roll	roll
rolling	roll	roll
rollings	roll	rolling
rolls	roll	roll
round	round	round
rounde	round	rounde
rounded	round	round
rounding	round	round
roundly	round	round
roundness	round	round
rounds	round	round
relax	relax	relax
relaxe	relax	relaxe
relaxes	relax	relaxe
remain	remain	remain
remaine	remain	remaine
singeing	sing	singe
singing	sing	sing
sings	sing	sing
scienc	scienc	scienc
science	scienc	science
sciences	scienc	science
secrete	secret	secrete
secreted	secret	secrete
secretes	secret	secrete
secreting	secret	secrete
secretion	secret	secrete
secretly	secret	secret
secrets	secret	secret
sever	sever	sever
severa	severa	severa
several	sever	several
severally	sever	several
severe	sever	severe
severed	sever	sever
severely	sever	severe
severer	sever	sever
severity	sever	sever
sooth	sooth	sooth
soothe	sooth	soothe
soothed	sooth	sooth
soothing	sooth	sooth
soothingly	sooth	sooth
start	start	start
starte	start	starte
started	start	start
starting	start	start
startings	start	starting
starts	start	start

Word	Porter02	Porter04
stern	stern	stern
sterne	stern	sterne
sternly	stern	stern

Word	Porter02	Porter04
sternness	stern	stern
wit	wit	wit
witness	wit	witness
witnessed	wit	witness

Word	Porter02	Porter04
witnesses	wit	witness
witnessing	wit	witness
wits	wit	wit
witted	wit	wit

Boosting for Text Classification with Semantic Features

Stephan Bloehdorn[1], Andreas Hotho[2]

[1] University of Karlsruhe, Institute AIFB, Knowledge Management Group, Germany
sbl@aifb.uni-karlsruhe.de
[2] University of Kassel, Knowledge and Data Engineering Group, Germany
hotho@cs.uni-kassel.de

Abstract. Current text classification systems typically use term stems for representing document content. Ontologies allow the usage of features on a higher semantic level than single words for text classification purposes. In this paper we propose such an enhancement of the classical document representation through concepts extracted from background knowledge. Boosting, a successful machine learning technique is used for classification. Comparative experimental evaluations in three different settings support our approach through consistent improvement of the results. An analysis of the results shows that this improvement is due to two separate effects.

1 Introduction

Most of the explicit knowledge assets of today's organizations consist of unstructured textual information in electronic form. Users are facing the challenge of organizing, analyzing and searching the ever growing amounts of documents. Systems that automatically classify text documents into predefined thematic classes and thereby contextualize information offer a promising approach to tackle this complexity. During the last decades, a large number of machine learning methods have been proposed for text classification tasks [16]. Recently, especially Support Vector Machines [9] and Boosting Algorithms [15] have produced promising results.

So far, however, existing text classification systems have typically used the *Bag-of-Words model* known from information retrieval, where single words or word stems are used as features for representing document content. By doing so, the chosen learning algorithms are restricted to detecting patterns in the used *terminology* only, while *conceptual* patterns remain ignored. Specifically, systems using only words as features exhibit a number of inherent deficiencies:

1. *Multi-Word Expressions* with an own meaning like *"European Union"* are chunked into pieces with possibly very different meanings when treated separately like – in this example – *"union"*.
2. *Synonymous Words* like *"tungsten"* and *"wolfram"* are mapped into different features.
3. *Polysemous Words* are treated as one single feature while they may actually have multiple distinct meanings.
4. *Lack of Generalization*: there is no way to generalize similar terms like "gold" and "silver" to their common hypernym "precious metal".

While items 1 – 3 directly address issues that arise on the lexical level, item 4 rather addresses an issue that is situated on a conceptual level. In this paper, we show how background knowledge in form of simple ontologies can improve text classification results by directly addressing these problems.

We propose a hybrid approach for document representation based on the common term stem representation which is enhanced with concepts extracted from the used ontologies. For actual classification we suggest to use the AdaBoost algorithm which has proven to produce accurate classification results in many experimental evaluations and seems to be well suited to integrate different types of features. Evaluation experiments on three text corpora, namely the Reuters-21578, OHSOMED and FAODOC collections show that our approach leads to consistent improvements of the results. We also show that in most cases the improvement can be traced to two distinct effects, one being situated mainly on the lexical level and the generalization on the conceptual level.

This paper is organized as follows. We introduce some preliminaries, namely the classical bag-of-words document representation and ontologies in section 2. A detailed process for compiling conceptual features into an enhanced document representation is presented in section 3. In section 4 we review the AdaBoost algorithm and its inner workings. Evaluation Measures for text classification are reviewed in section 5. In the following, experimental evaluation results of our approach are presented for the Reuters-21578, OHSOMED, and FAODOC corpora under varying parameter combinations. It turns out that combined feature representations perform consistently better than the pure term-based approach. We review related work in section 7 and conclude with a summary and outlook in section 8.

2 Preliminaries

The Bag-Of-Words Paradigm In the common term-based representation, documents are considered to be bags of terms, each term being an independent feature of its own. Let D be the set of documents and $T = \{t_1, \ldots, t_m\}$ the set of all different terms occurring in D. For each term $t \in T$ in document $d \in D$ one can define feature values functions like binary indicator variables, absolute frequencies or more elaborated measures like TFIDF [14].

Typically, whole words are not used as features. Instead, documents are first processed with stemming algorithms, e.g. the Porter stemmer for English [13]. In addition, *Stopwords*, i.e. words which are considered as non–descriptive within a bag–of–words approach, are typically removed.

Ontologies The background knowledge we have exploited is given through simple ontologies. We first describe the structure of these ontologies and then discuss their usage for the extraction of conceptual feature representations for text documents. The background knowledge we will exploit further on is encoded in a *core ontology*. For the purpose of this paper, we present only those parts of our more extensive ontology definition [2] that we need within this paper.

Definition 1 (Core Ontology). *A core ontology is a structure $\mathcal{O} := (C, <_C)$ consisting of a set C, whose elements are called concept identifiers, and a partial order $<_C$ on C, called concept hierarchy or taxonomy.*

Definition 2 (Subconcepts and Superconcepts). *If $c_1 <_C c_2$ for any $c_1, c_2 \in C$, then c_1 is a subconcept (specialization) of c_2 and c_2 is a superconcept (generalization) of c_1. If $c_1 <_C c_2$ and there exists no $c_3 \in C$ with $c_1 <_C c_3 <_C c_2$, then c_1 is a direct subconcept of c_2, and c_2 is a direct superconcept of c_1, denoted by $c_1 \prec c_2$.*

These specialization/generalization relationships correspond to what we know as is-a vs. is-a-special-kind-of, resulting in a hierarchical arrangement of concepts[3]. In ontologies that are more loosely defined, the hierarchy may, however, not be as explicit as is-a relationships but rather correspond to the notion of narrower-than vs. broader-than[4]

According to the international standard ISO 704, we provide names for the concepts (and relations). Instead of 'name', we here call them 'sign' or 'lexical entries' to better describe the functions for which they are used.

Definition 3 (Lexicon for an Ontology). *A lexicon for an ontology $\mathcal{O}$ is a tuple $Lex := (S_C, Ref_C)$ consisting of a set S_C, whose elements are called signs for concepts (symbols), and a relation $Ref_C \subseteq S_C \times C$ called lexical reference for concepts, where $(c, c) \in Ref_C$ holds for all $c \in C \cap S_C$. Based on Ref_C, for $s \in S_C$ we define $Ref_C(s) := \{c \in C | (s, c) \in Ref_C\}$. Analogously, for $c \in C$ it is $Ref_C^{-1}(c) := \{s \in S_C | (s, c) \in Ref_C\}$. An ontology with lexicon is a pair $(\mathcal{O}, Lex)$ where $\mathcal{O}$ is an ontology and Lex is a lexicon for $\mathcal{O}$.*

Ontologies for the experimental evaluation For the purpose of actual evaluation in the experiments, we have used three different resources, namely WordNet, the MeSH Tree Structures Ontology, and the AGROVOC ontology.

Although not explicitly designed as an ontology, *WordNet* [12] largely fits into the ontology definitions given above. The WordNet database organizes simple words and multi-word expressions of different syntactic categories into so called *synonym sets (synsets)*, each of which represents an underlying concept and links these through semantic relations. The current version 2.0 of WordNet comprises a total of 115,424 synsets and 144,309 lexical index terms. The noun category, which was the main focus of our attention[5], contains nearly 70 % of the total synsets, links from 114,648 index terms to 79,689 synsets in a total of 141,690 mappings. The collection of index terms in WordNet comprises base forms of terms and their exceptional derivations. The retrieval of base forms for inflected forms is guided by a set of category-specific morphological

[3] Note that this hierarchical structure is not necessarily a tree structure. It may also be a *directed acyclic graph* possibly linking concepts to multiple superconcepts at the same time.

[4] In many settings this view is considered as a very bad practice as it may lead to inconsistencies when reasoning with ontologies. However, this problem does not arise in the context of this work.

[5] Beside the noun category, we have also exploited verb synsets, however, without making use of any semantic links,

transformations, which ensure a high precision in the mapping of word forms to index words.

The MeSH Tree Structures Ontology is an ontology that has been compiled out of the Medical Subject Headings (MeSH) controlled vocabulary thesaurus of the United States National Library of Medicine (NLM). The ontology contains more than 22,000 concepts, each enriched with synonymous and quasi-synonymous language expressions. The underlying hierarchical structure is in large parts consistent with real hypernym relations but also comprises other forms of hierarchical arrangements. The ontology itself was ported into and accessed through the Karlsruhe Ontology and Semantic Web Infrastructure (KAON) infrastructure[6].

The third ontology that has been used is the AGROVOC Ontology, based on AGROVOC, a multilingual agricultural thesaurus[7] developed by the United Nations Food and Agricultural Organization (FAO). In total, the ontology comprises 17,506 concepts from the agricultural domain. The lexicon contains label and synonym entries for each concept in English and six additional languages. The concept hierarchy in the AGROVOC ontology is based on broader-term relationships thus not necessarily on strict superconcept relations in some cases.

3 Conceptual Document Representation

To extract concepts from texts, we have developed a detailed process, that can be used with any ontology with lexicon. The overall process comprises five processing steps that are described in this section.

Candidate Term Detection Due to the existence of multi-word expressions, the mapping of terms to concepts cannot be accomplished by querying the lexicon directly for the single words in the document.

We have addressed this issue by defining a candidate term detection strategy that builds on the basic assumption that finding the longest multi-word expressions that appear in the text and the lexicon will lead to a mapping to the most specific concepts. The candidate expression detection algorithm we have applied for this lookup procedure is given in algorithm 1[8].

The algorithm works by moving a window over the input text, analyze the window content and either decrease the window size if unsuccessful or move the window further. For English, a window size of 4 is sufficient to detect virtually all multi-word expressions.

Syntactical Patterns Querying the lexicon directly for any expression in the window will result in many unnecessary searches and thereby in high computational requirements. Luckily, unnecessary search queries can be identified and avoided through an analysis of the part-of-speech (POS) tags of the words contained in the current window. Concepts are typically symbolized in texts within *noun phrases*. By defining appropriate

[6] see http://kaon.semanticweb.org/

[7] see http://www.fao.org/agrovoc/

[8] The algorithm here is an improved version of one proposed in [17].

Algorithm 1 The candidate expression detection algorithm

Input: document $d = \{w_1, w_2, \ldots, w_n\}$,
$Lex = (S_C, Ref_C)$ and window size $k \geq 1$.
$i \leftarrow 1$
list L_s
index-term s
while $i \leq n$ **do**
  for $j = min(k, n - i + 1)$ to 1 **do**
    $s \leftarrow \{w_i \ldots w_{i+j-1}\})$
    if $s \in S_C$ **then**
      save s in L_s
      $i \leftarrow i + j$
      break
    else if $j = 1$ **then**
      $i \leftarrow i + j$
    end if
  end for
end while
return L_s

POS patterns and matching the window content against these, multi-word combinations that will surely not symbolize concepts can be excluded in the first hand and different syntactic categories can be disambiguated.

Morphological Transformations Typically the lexicon will not contain all inflected forms of its entries. If the lexicon interface or separate software modules are capable of performing base form reduction on the submitted query string, queries can be processed directly. For example, this is the case with WordNet. If the lexicon, as in most cases, does not contain such functionalities, a simple fallback strategy can be applied. Here, a separate index of stemmed forms is maintained. If a first query for the inflected forms on the original lexicon turned out unsuccessful, a second query for the stemmed expression is performed.

Word Sense Disambiguation Having detected a lexical entry for an expression, this does not necessarily imply a one-to-one mapping to a concept in the ontology. Although multi-word-expression support and pos pattern matching reduce ambiguity, there may arise the need to disambiguate an expression versus multiple possible concepts. The *word sense disambiguation (WSD)* task is a problem in its own right [8] and was not the focus of our work.

In our experiments, we have used three simple strategies proposed in [7] to process polysemous terms:

- The "all" strategy leaves actual disambiguation aside and uses all possible concepts.
- The "first" strategy exploits WordNet's capability to return synsets ordered with respect to usage frequency. This strategy chooses the most frequent concept in case of ambiguities.
- The "context" strategy performs disambiguation based on the degree of overlap of lexical entries for the semantic vicinity of candidate concepts and the document content as proposed in [7].

Generalization The last step in the process is about going from the specific concepts found in the text to more general concept representations. Its principal idea is that if a term like 'arrythmia' appears, one does not only represent the document by the concept corresponding to 'arrythmia', but also by the concepts corresponding to 'heart disease' and 'cardiovascular disease' etc. up to a certain level of generality. This is realized by compiling, for every concept, all superconcept up to a maximal distance h into the concept representation. Note that the parameter h needs to be chosen carefully as climbing up the taxonomy too far is likely to obfuscating the concept representation.

4 Boosting

Boosting is a relatively young, yet extremely powerful machine learning technique. The main idea behind boosting algorithms is to combine multiple *weak learners* – classification algorithms that perform only slightly better than random guessing – into a powerful composite classifier.

Although being refined subsequently, the main idea of all boosting algorithms can be traced to the first practical boosting algorithm, AdaBoost [4], which we will concentrate on in this paper. AdaBoost and related algorithms have proved to produce extremely competitive results in many settings, most notably for text classification [15]. At the beginning, the inner workings of boosting algorithms were not well understood. Subsequent research in boosting algorithms made them rest on a well developed theoretical framework and has recently provided interesting links to other successful learning algorithms, most notably to Support Vector Machines, and to linear optimization techniques [11].

AdaBoost The idea behind "boosting" weak learners stems from the observation that it is usually much easier to build many simple "rules of thumb" than a single highly complex decision rule. Very precise overall decisions can be achieved if these weak learners are appropriately combined.

This idea is reflected in the output of the boosting procedure: for AdaBoost the aggregate decisions are formed in an *additive model* of the form:

$$\hat{f}(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t \, h_t(x))$$

with $h_t : \mathbb{X} \rightarrow \{-1, 1\}$, $\alpha_t \in \mathbb{R}$, where α_t denotes the weight of the ensemble member h_t in the aggregate decision and where the output values $\hat{f}(x) \in \{1, -1\}$ denote positive and negative predictions respectively. In such a model, AdaBoost has to solve two questions: How should the set of base hypotheses h_t be determined ? How should the weights α_t determined, i.e. which base hypotheses should contribute more than others and how much ? The AdaBoost algorithm, described in algorithm 2 aims at coming up with an optimal parameter assignment for h_t and α_t.

AdaBoost maintains a set of weights D_t over the training instances $x_1 \ldots x_i \ldots x_n$. At each iteration step t, a base classifier is chosen that performs best on the *weighted* training instances. Based on the performance of this base classifier, the final weight

Algorithm 2 The AdaBoost algorithm.

Input: training sample $\mathcal{S}_{train} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
with $(x_i, y_i) \in \mathbb{X} \times \{-1, 1\}$ and $y_i = f(x_i)$,
number of iterations T.
Initialize: $D_1(i) = \frac{1}{n}$ for all $i = 1, \ldots, n$.
for $t = 1$ to T **do**
train base classifier h_t on weighted training set
calculate the weighted training error:

$$\epsilon_t \leftarrow \sum_{i=1}^{n} D_t(i)\, I_{y_i \neq h_t(x_i)} \tag{1}$$

compute the optimal update step as:

$$\alpha_t \leftarrow \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t} \tag{2}$$

update the distribution as:

$$D_{t+1}(i) \leftarrow \frac{D_t(i)\, e^{-\alpha_t\, y_i\, h_t(x_i)}}{Z_t} \tag{3}$$

where Z_t is a normalization factor to ensure that $\sum_{i=1}^{n} D_{t+1}(i) = 1$
if $\epsilon_t = 0$ or $\epsilon_t = \frac{1}{2}$ **then**
break
end if
end for
Result: composite classifier given by:

$$\hat{f}(x) = \text{sign}\left(\hat{f}_{soft}(x)\right) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{4}$$

parameter α_t is calculated in equation (2) and the distribution weights D_{t+1} for the next iteration are updated. The weight update in equation (3) assigns higher weights to training instances that have been misclassified, while correctly classified instances will receive smaller weights in the next iteration. Thereby, AdaBoost kind of "focusing in" on those examples that are more difficult while the weight each base classifier receives in the final additive model depends on its performance on the weighted training set at the respective iteration step.

Weak Lerners for AdaBoost In theory, AdaBoost can be used with *any* base learner capable of handling weighted training instances. Although the base classifiers are not restricted to belong to a certain classifier family, virtually all work with boosting algorithms has used the very simple class of *decision stumps* as base learners.

In this presentation, we focus on simple indicator function decision stumps of the form:

$$h(x) = \begin{cases} c \text{ if } x^j = 1 \\ -c \text{ else.} \end{cases}$$

with $c \in \{-1, 1\}$. A decision stump of this form takes binary features (e.g. word or concept occurrences) as inputs. The index j identifies a specific binary feature whose presence either supports a positive classification decision, i.e. $c = 1$ or a negative decision, i.e. $c = -1$.

5 Evaluation Metrics

A standard set of performance metrics is commonly used to assess classifier performance which we will review shortly in this section.

Classification Metrics Given a set of test documents $\mathcal{S} = \{x_1, \ldots, x_n\}$ with binary labels $\{y_1, \ldots, y_n\}$ where $y_i \in \{-1, 1\}$ codes the membership in a class in question. Given further a classifier $\hat{f}$ trained on an independent training set with $\hat{f}(x) \in \{-1, 1\}$ indicating the binary decisions of the classifier. Then the test sample can be partitioned into sets $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$, i.e. the set of positive and negative test documents. These partitions can be decomposed further into $\mathcal{S}^+ = TP \cup FN$ and $\mathcal{S}^- = FP \cup TN$ with: $TP = \{x_i \in \mathcal{S} | \hat{f}(x_i) = 1 \wedge y_i = 1\}$, $FP := \{x_i \in \mathcal{S} | \hat{f}(x_i) = 1 \wedge y_i = -1\}$, $TN := \{x_i \in \mathcal{S} | \hat{f}(x_i) = -1 \wedge y_i = -1\}$ and $FN := \{x_i \in \mathcal{S} | \hat{f}(x_i) = -1 \wedge y_i = 1\}$ called the sets of documents classified *true positive*, *false positive*, *true negative* and *false negative*, often referred to as the classification contingency table.

Based on these definitions, different evaluation measures have been defined [18]. Commonly used classification measures in text classification and information retrieval are the *classification error*, *precision*, *recall* and the F_β *measure*:

1. Classification Error

$$err(\hat{f}, \mathcal{S}) := \frac{|FP| + |FN|}{|TP| + |FP| + |TN| + |FN|} \,. \tag{5}$$

2. Precision

$$prec(\hat{f}, \mathcal{S}) := \frac{|TP|}{|TP| + |FP|} \,. \tag{6}$$

3. Recall

$$rec(\hat{f}, \mathcal{S}) := \frac{|TP|}{|TP| + |FN|} \,. \tag{7}$$

4. F_1 measure

$$F_1(\hat{f}, \mathcal{S}) := \frac{2\, prec(\hat{f}, \mathcal{S})\, rec(\hat{f}, \mathcal{S})}{prec(\hat{f}, \mathcal{S}) + rec(\hat{f}, \mathcal{S})} \,. \tag{8}$$

Ranking Metrics The ensemble classifiers produced by AdaBoost are capable of returning a real-valued output $\hat{f}_{soft}(x) \in [-1, 1]$. The magnitude $|\hat{f}_{soft}(x)|$ reflects the "confidence" of the classifier in a decision and allows to rank documents. Consequently, a parameterized classifier $\hat{f}_k$ can be defined that returns $\hat{f}_k(x) = 1$ if $\hat{f}_{soft}(x)$ ranks among the first k documents and $\hat{f}_k(x) = -1$ otherwise. On this basis, values for precision and recall can be calculated and tuned with respect to different values of k. When precision and recall coincide at some k, this value is called the break-even point (BEP). It can be shown that this is necessarily the case at $k = |\mathcal{S}^+|$[9].

Micro- and Macro Averaging To average evaluation results over binary classifications on the per-class level, two conventional methods exist. The *macro-averaged* figures are meant to be averages on the class level and are calculated as simple averages of the scores achieved for the different classes. In contrast, *micro-averaged* figures are computed by summing the cells of per-class contingency tables together and then computing performance scores based on these global figures. These can consequently be seen as averages on the document level.

Statistical Significance Tests Statistical significance tests are useful in order to verify to which extent the claim of an improvement can be backed by the observations on the test set. For the experiments we report in this paper, we focused on two statistical significance tests, a sign test ("S-test") and a paired t-test ("T-test") on an improvement of individual F_1 scores for the different classes that have been evaluated in each experiment described in detail in [19]. Following common statistical practice, we have required a significance level $\alpha = 0.05$ is required for claiming an improvement to be *significant*. The significance level of $\alpha = 0.01$ was used for the claim that an improvement was *very significant*.

6 Experiments

The focus of our evaluation experiments was directed towards comparing whether AdaBoost using the enhanced document representation would outperform the classical term representation.

6.1 Evaluation on the Reuters-21578 Corpus

A first set of evaluation experiments was conducted on the well-known Reuters-21578 collection. We used the "ModApte" split which divides the collection into 9,603 training documents, 3,299 test documents and 8,676 unused documents.

[9] This follows from the fact that if there are m negative documents among the first $|\mathcal{S}^+|$ documents in the ranked list, there must also be exactly m positive examples in the remainder of the list, thus: $FP_k = FN_k = m$, which guarantees precision and recall to be equal according to the formulas given above.

Experimental Setup In the first stage of the experiment, terms and concepts were extracted as features from the documents in the training and test corpus. For terms, the feature extraction stage consisted of the stages described in section 2, namely chunking, removal of the standard stopwords for English defined in the stopword list from the SMART system containing 571 words [10] and stemming using the porter stemming algorithm, resulting in a total number of 17,525 distinct term features. Conceptual features were then extracted for noun and verb phrases using WordNet as background ontology. Different sets of concept features were extracted based on varying parameters for disambiguation strategy and maximal hypernym distance ranging from 10,259 to 27,236 distinct concept features.

In the next stage of the experiment, classification was performed using AdaBoost. We performed binary classification on the top 50 categories containing the highest number of positive training documents. The number of boosting iterations for training was fixed at 200 rounds for all feature combinations.

Results As a general finding, the results obtained in the experiments suggest that AdaBoost typically achieves better classification for both macro- and micro-averaged results when used with a combination of term-based and concept-based features. Table 1 summarizes the results of the experiments for different feature types with the best values being highlighted. The relative gains on the F_1 value, which is influenced both by precision and recall, compared to the baseline show that in all but one cases the performance can be improved by including conceptual features, peaking at an relative improvement of 3.29 % for macro-averaged values and 2.00 % for micro-averaged values. Moderate improvements are achieved through simple concept integration, while larger improvements are achieved in most cases through additional integration of more general concepts.

The results of the significance tests allow us to conclude that these improvements are significant in at least half of the cases. In general, the improvements of macro-averaged F_1 are higher than with micro-averaging which seems to suggest that the additional concepts are particularly helpful for smaller classes.

6.2 Evaluation on the OHSOMED Corpus

A second series of experiments was conducted using the OHSOMED collection, initially compiled by Hersh et al. [6]. It consists of titles and abstracts from medical journals, each being indexed with multiple MeSH descriptors. We have used the 1987 portion of the collection containing a total of 54,708 entries. Two thirds of the entries were randomly selected as training documents while the remainder was used as test set, resulting in a training corpus containing 36,369 documents and a test corpus containing 18,341 documents.

Experimental Setup Term stems were extracted as with Reuters-21578 resulting in a total number of 38,047 distinct features. WordNet and the MeSH Tree Structures Ontology were used to extract conceptual features. For WordNet, noun and verb phrases

[10] see `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`

		macro-averaged (in percentages)			
Feature Type	**Error**	**Prec**	**Rec**	**F_1**	**BEP**
term	00.65	80.59	66.30	72.75	74.29
term & synset.first	00.64	80.66	67.39	73.43	75.08
term & synset.first.hyp5	00.60	80.67	**69.57**	74.71	74.84
term & synset.first.hyp10	00.62	80.43	68.40	73.93	**75.58**
term & synset.context	00.63	79.96	68.51	73.79	74.46
term & synset.context.hyp5	00.62	79.48	68.34	73.49	74.71
term & synset.all	00.64	80.02	66.44	72.60	73.62
term & synset.all.hyp5	**00.59**	**83.76**	68.12	**75.14**	75.55
		micro-averaged (in percentages)			
Feature Type	**Error**	**Prec**	**Rec**	**F_1**	**BEP**
term	00.65	89.12	79.82	84.21	85.77
term & synset.first	00.64	88.75	80.79	84.58	85.97
term & synset.first.hyp5	00.60	89.16	**82.46**	85.68	85.91
term & synset.first.hyp10	00.62	88.78	81.74	85.11	86.14
term & synset.context	00.63	88.86	81.46	85.00	85.91
term & synset.context.hyp5	00.62	89.09	81.40	85.07	85.97
term & synset.all	00.64	88.82	80.99	84.72	85.69
term & synset.all.hyp5	**00.59**	**89.92**	82.21	**85.89**	**86.44**

Table 1. Evaluation Results for Reuters-21578.

were considered while for the MeSH Tree Structures Ontology, only noun phrases were considered.

For WordNet, the same disambiguation strategies were used as in the Reuters-21578 experiments. For the MeSH Tree Structures Ontology, only the "all" strategy was used due to the observation that polysemy problems occur extremely rarely with this ontology as descriptor terms are most naturally unique. For both ontologies, different degrees of depth were used for hypernym or superconcept integration, resulting in a total of 16,442 to 34,529 synset features and 11,572 to 13,663 MeSH concept features.

On the documents of the OHSOMED dataset — as on Reuters-21578 — binary classification with AdaBoost was performed on the top 50 categories that contained the highest number of positive training documents. To cope with the on average larger number of features and the much higher number of documents compared to the Reuters-21578 corpus, the number of boosting iterations for all experiments with the OHSOMED collection was set to 1000 rounds.

Results Different runs of the classification stage were performed based on the different features, leading to often substantially different results. Again, the general finding is that complementing the term stem representation with conceptual features significantly improves classification performance.

Table 2 summarizes the macro- and micro-averaged results. The relative improvements for the F_1 scores compared to the term stem baseline are depicted in figure 6.2 for WordNet as background knowledge resource. These range from about 2% to a maxi-

Feature Type	Error	macro-averaged (in percentages) Prec	Rec	F_1	BEP
term	00.53	52.60	35.74	42.56	45.68
term & synset.first	00.52	53.08	36.98	43.59	46.46
term & synset.first.hyp5	00.52	53.82	38.66	45.00	48.01
term & synset.context	00.52	52.83	37.09	43.58	46.88
term & synset.context.hyp5	**00.51**	**54.55**	**39.06**	**45.53**	**48.10**
term & synset.all	00.52	52.89	37.09	43.60	46.82
term & synset.all.hyp5	00.52	53.33	38.24	44.42	46.73
term & mesh	00.52	53.65	37.56	44.19	47.31
term & mesh.sc1	00.52	52.91	37.59	43.95	46.93
term & mesh.sc3	00.52	52.77	38.06	44.22	46.90
term & mesh.sc5	00.52	52.72	37.57	43.87	47.16

Feature Type	Error	micro-averaged (in percentages) Prec	Rec	F_1	BEP
term	00.53	55.77	36.25	43.94	46.17
term & synset.first	00.52	56.07	37.30	44.80	47.01
term & synset.first.hyp5	00.52	56.84	38.76	46.09	48.31
term & synset.context	00.52	56.30	37.46	44.99	47.34
term & synset.context.hyp5	**00.51**	**58.10**	**39.18**	**46.81**	**48.45**
term & synset.all	00.52	56.19	37.44	44.94	47.32
term & synset.all.hyp5	00.52	56.29	38.24	45.54	46.73
term & mesh	00.52	56.81	37.84	45.43	47.78
term & mesh.sc1	00.52	56.00	37.90	45.20	47.49
term & mesh.sc3	00.52	55.87	38.26	45.42	47.45
term & mesh.sc5	00.52	55.94	37.94	45.21	47.63

Table 2. Evaluation Results for OHSOMED.

mum of about 7 %. The relative F_1 improvements when using the MeSH Tree Structure Ontology, were on the 3% to 5% level in all cases.

The statistical significance tests revealed that in virtually all cases, these improvements can be claimed to be significant and actually even very significant in most cases.

Again, the integration of conceptual features improved text classification results. The relative improvements achieved on OHSOMED are generally higher than those achieved on the Reuters-21578 corpus. This makes intuitively sense as the documents in the OHSOMED corpus are taken from the medical domain. Documents from this domain typically suffer heavily from the problems described in section 2, especially synonymous terms and multi-word expressions. But this is only a first effect. The even better results achieved through hypernym integration with WordNet indicate that also the highly specialized language is a problem that can be remedied through integration of more general concepts.

A comparison between WordNet and the MeSH Descriptor Ontology is hard. On the one hand, without generalization, the domain specific MeSH Tree Structures Ontology is able to achieve slightly better results. Taking into account that the extraction was here bases solely on noun phrases and that WordNet's coverage is much broader,

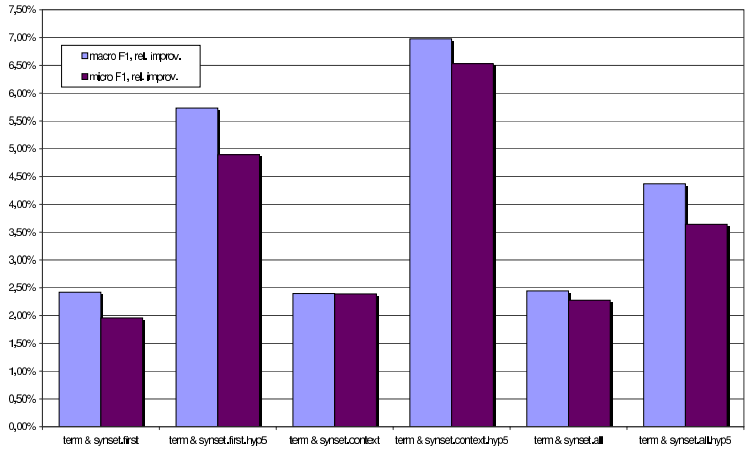

Fig. 1. Relative Improvements of F_1 Scores on OHSOMED for combined Term-Synset Features vs. Term Stems.

this is a positive surprise. On the other hand, WordNet achieves much better results when generalization comes into play. In contrast to WordNet, superconcept integration for MeSH does not really improve the results and varying levels of superconcept integration lead to similar or even worse results. Apparently, the broader-term relation of the MeSH thesaurus is indeed not well suited to improve the results. Also note that in contrast to the Reuters-21578 experiments, "context" word sense disambiguation strategy performs best in combination with hypernym integration. Apparently, it is easier to disambiguate polysemous words in the medical context.

6.3 Evaluation on the FAODOC Corpus

The third and last series of experiments uses a collection of documents from the FAO Document Online Catalogue (FAODOC)[11], managed by the United Nations Food and Agricultural Organization. The FAODOC database houses articles and other publications from the agricultural domain together with metadata information, including subject and category elements.

Experimental Setup The FAODOC collection contains English, French and Spanish HTML documents. All documents are indexed with one or multiple category codes, each of which refers to one of 115 FAODOC subject categories. In the experiments, only the subset of 1,501 English documents has been used where each of the categories has at least 50 positive documents, resulting in 21 distinct subject categories. From the total number of 1,501 documents, the first 1,000 documents were used for training while the remainder of 501 documents were held out as test set. The FAODOC dataset is very different from the other datasets encountered so far. Besides being taken from

[11] see `http://www4.fao.org/faobib/index.html`

		macro-averaged			
Feature Type	**Error**	**Prec**	**Rec**	**F_1**	**BEP**
term	06.87	45.47	27.11	33.97	36.93
term & agrovoc	**06.66**	**50.96**	28.63	36.66	39.84
term & agrovoc.sc1	06.76	49.26	27.48	35.28	39.40
term & agrovoc.sc3	06.79	49.08	**30.41**	**37.55**	**41.69**
		micro-averaged			
Feature Type	**Error**	**Prec**	**Rec**	**F_1**	**BEP**
term	06.87	50.44	31.22	38.57	44.29
term & agrovoc	**06.66**	**52.91**	32.46	**40.24**	**48.01**
term & agrovoc.sc1	06.76	51.75	**32.60**	40.00	46.77
term & agrovoc.sc3	06.79	51.47	31.36	38.97	47.73

Table 3. Results on FAODOC

a different domain, the total number of documents is much smaller. The documents in the FAODOC dataset are typically much larger in size, ranging from 1.5 kilobytes to over 600 kilobytes, which is also reflected in the resulting feature representations with 68,608 word stems. Besides the extraction of term stems as usual, conceptual features were extracted again, this time using the AGROVOC ontology as background knowledge resource. For both types of features, the documents were first converted from HTML to plain text, then proceeding in the same way as with the documents in the other corpora.

As in the other experiments, each of the 21 different labels resulted in a binary classification run of its own, each time using AdaBoost as learning algorithm with decision stump classifier based on the binary feature weights as base learners. The chosen number of 500 boosting iterations is based on a trade-off between the smaller number of training documents on the one hand and a typically larger size per document on the other. In all experiments, the results on the 21 individual labels were eventually macro- and micro-averaged.

Results Different runs of the classification stage were performed based on different features: term stems and again combinations of both types of features.

Table 3 summarizes the results of the experiments with the FAODOC for the different feature representations, evaluation metrics and averaging variants. For each performance metric, the best result is highlighted. Again, combinations of terms and concepts as features also achieve considerable improvements over the classic term stem representation in all scores, most notably with respect to precision.

Figure 2 undermines the good performance of the term and 'agrovoc' concept representation achieving an impressive relative improvement of 10.54 % on the macro-averaged F_1 value compared to the 'term' representation. The relative improvement on the micro-averaged F_1 lies at 4.33 %.

As with OHSOMED, one observes a heavy discrepancy between the macro- and micro-averaged scores. Again, macro-averaged performance gains are higher than those for micro-averaging, which makes sense taking into account the fairly unequal category sizes. In contrast to the other experiments, the amount of deviation however varies con-

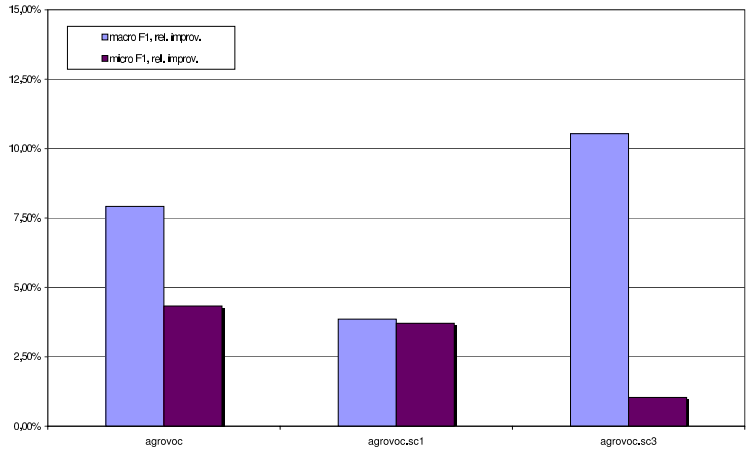

Fig. 2. Bar Chart Illustration of the Relative Improvements of F_1 Scores on all 21 FAODOC Categories for combined Term-Concept Representations vs. 'term'. All numbers are percentages.

siderably among the different feature representations. Furthermore, the question which superconcept integration depth leads to the best improvement cannot be answered easily because the effects vary between micro- and macro-averaging. We attribute the strong variation in the results to the fact that random effects are much likelier compared to the other experiments as the number of training and test documents was considerably smaller.

7 Related Work

Representing document content through metadata descriptions is a well-known task in the semantic web context, also known as annotation[5]. Typically, however, this is a semi-automatic task that aims at precise metadata descriptions and not at creating features for machine learning algorithms.

To date, the work on integrating semantic background knowledge into text classification or other related tasks is quite scattered. Much of the early work with semantic background knowledge in information retrieval was done in the context of *query expansion* techniques [1]. Feature representations based on concepts from ontological background knowledge were also used in text clustering settings [7] where it could be shown that conceptual representations can significantly improve text cluster purity and reduce the variance among the representations of related documents.

Recent experiments with conceptual feature representations for text classification are presented in [17]. These and other similar published results are, however, still too few to allow insights on whether positive effects can be achieved in general. In some cases, even negative results were reported. For example, a comprehensive comparison of approaches based on different word-sense document representations and different

learning algorithms reported in [10] ends with the conclusion of the authors that *"the use of word senses does not result in any significant categorization improvement"*.

Alternative approaches for conceptual representations of text documents that are not based on background knowledge compute kind of "concepts" statistically. Very good results with a probabilistic variant of LSA known as Probabilistic Latent Semantic Analysis (pLSA) were recently reported in [3]. The experiments reported therein are of particular interest as the classification was also based on boosting combined term-concept representation, the latter being however automatically extracted from the document corpus using pLSA.

8 Conclusions

In this paper, we have proposed an approach to incorporate concepts from background knowledge into document representations for text document classification. A very successful ensemble learning algorithm, AdaBoost, was proposed to perform the final classifications based on the classical word vector representations and the conceptual features. Boosting Algorithms, when used with binary feature representations, scale well to a large number of dimensions that typically occur when superconcepts are used as well. At the same time, AdaBoost is capable of integrating heterogenous features that are based on different paradigms without having to adjust any parameters in the feature space representation.

Experiments on three different datasets clearly showed that the integration of concepts into the feature representation clearly improves classification results. The absolute scores achieved on Reuters and OHSOMED are highly competitive with other published results and the reported relative improvements appear to be statistically significant in most cases. A comparative analysis of the improvements for different concept integration strategies revealed that two separate effects lead to these improvements. A first effect that can be mainly attributed to multi-word expression detection and synonym conflation is achieved through the basic concept integration. A second effect building on this initial improvement is attributed to the use of the ontology structures for generalization through hypernym retrieval and integration.

Outlook The experiments that have been conducted show that the presented approach appears to be promising in most settings. However it has also become obvious that the results depend on the specific constellation of parameters. These include — most importantly — the choice of the appropriate ontology. Further research and experiments should investigate how the specific choice and setup of the used ontologies can lead to even better results and wether other concept extraction strategies lead to a further improvement in classification performance.

Further attention should also be paid to the setup of the classification algorithm as the general nature of AdaBoost would allow to integrate more advanced weak learners. Such weak learners might also exploit background knowledge even more directly.

Acknowledgements This research was partially supported by the European Commission under contract FP6-001765 aceMedia. The expressed content is the view of the authors but not necessarily the view of the aceMedia project as a whole.

References

1. R. C. Bodner and F. Song. Knowledge-Based Approaches to Query Expansion in Information Retrieval. In *Advances in Artificial Intelligence*. Springer, New York, NY, USA, 1996.
2. E. Bozsak et al. KAON – Towards a Large Scale Semantic Web. In *Proc. of the 3rd International Conference on E-Commerce and Web Technologies (EC-Web 2002)*, pages 304–313, Aix-en-Provence, France, 2002. LNCS 2455 Springer.
3. L. Cai and T. Hofmann. Text Categorization by Boosting Automatically Extracted Concepts. In *Proc. of the 26th Annual Int. ACM SIGIR Conference on Research and Development in Informaion Retrieval*, Toronto, Canada, 2003. ACM Press.
4. Y. Freund and R. E. Schapire. A Decision Theoretic Generalization of On-Line Learning and an Application to Boosting. In *Second European Conference on Computational Learning Theory (EuroCOLT-95)*, pages 23–37, 1995.
5. S. Handschuh and S. Staab, editors. *Annotation for the Semantic Web*. IOS Press, 2003.
6. W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam. Ohsumed: An Interactive Retrieval Evealuation and new large Test Collection for Research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. ACM Press, 1994.
7. A. Hotho, S. Staab, and G. Stumme. Wordnet improves Text Document Clustering. In *Proc. of the Semantic Web Workshop of the 26th Annual International ACM SIGIR Conference*, Toronto, Canada, 2003.
8. N. Ide and J. Véronis. Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1):1–40, 1998.
9. T. Joachims. Text Categorization with Support Vector Machines: Learning With Many Relevant Features. In *Proceedings of ECML-98*, 1998.
10. A. Kehagias, V. Petridis, V. G. Kaburlasos, and P. Fragkou. A Comparison of Word- and Sense-Based Text Categorization Using Several Classification Algorithms. *Journal of Intelligent Information Systems*, 21(3):227–247, 2000.
11. R. Meir and G. Rätsch. An Introduction to Boosting and Leveraging. In *Advanced Lectures on Machine Learning*, LNCS. Springer, Heidelberg, DE, 2003.
12. G. A. Miller, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: an On-Line Lexical Database. *International Journal of Lexicography*, 3(4):235–244, 1990.
13. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
14. G. Salton. *Automatic Text Processing*. Addison-Wesley Publishing Inc, Boston, MA, USA, 1989.
15. R. E. Schapire and Y. Singer. BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168, 2000.
16. F. Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
17. B. B. Wang, R. I. Mckay, H. A. Abbass, and M. Barlow. A comparative study for domain ontology guided feature extraction. In *Proceedings of the 26th Australian Computer Science Conference (ACSC-2003)*, pages 69–78. Australian Computer Society, 2003.
18. Y. Yang. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, 1(1-2):69–90, 1999.
19. Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999.

Learning Similarities for Collaborative Information Retrieval

Armin Hust

armin.hust@onlinehome.de

Abstract. The accuracy of ad-hoc information retrieval (IR) systems has plateaued in the last few years. At DFKI, we are working on so-called collaborative information retrieval (CIR) systems which have the potential to overcome the current limits. We focus on a restricted setting in CIR in which only old queries and correct answer documents to these queries are available for improving a new query. For this restricted setting we propose new approaches for query expansion procedures. We show how collaboration of individual users can improve overall information retrieval performance.
In our first steps towards techniques, we proposed new algorithms for query expansion in CIR systems. Now in this paper we focus on learning similarity measures. We do not try to invent new similarity measures, but learn weighting schemes to be applied to the standard cosine similarity measure. After learning the new weightings we re-evaluate our previously proposed CIR algorithms on standard IR test collections. It turns out that retrieval performance of previously developed algorithms is improved after learning the weightings for the involved similarity measure.

1 Introduction

In this section we introduce the research area of Collaborative Information Retrieval (CIR). We motivate and characterize the primary goals of this paper, query expansion procedures for CIR and outline the structure and contents.

The ultimate goal in IR is finding the documents that are useful to the information need expressed as a query. Much work has been done on improving IR systems, in particular in the Text Retrieval Conference series (TREC). In 2000, it was decided at TREC-8 that this task should no longer be pursued within TREC, in particular because the accuracy has plateaued in the last few years [13]. We are working on new approaches which learn to improve retrieval effectiveness from the interaction of different users with the retrieval engine. Such systems may have the potential to overcome the current plateau in ad-hoc retrieval.

CIR is a methodology where an IR system makes full use of all the additional information available in the system, especially

- the information from previous queries

– the relevance information gathered during previous search processes, independent of the method used to obtain this relevance information, i.e., explicitly by user relevance feedback or implicitly by unobtrusively detected relevance information.

Collaboration here assumes that users can benefit from search processes carried out at former times by other users (although they may not know about the other users and their search processes) as long as the relevance information gathered from these previous users has some significant meaning.

Subject to these assumptions we expect that collaborative searches will improve overall retrieval quality for all users.

We are aware of the problems of "personalization" and "context", but in our first steps towards techniques we avoid further complexity of CIR by ignoring these challenges. "Personalization" means that different users may have different preferences on relevant documents, because of long-term interests; "context" means that different users may have different preferences on relevant documents, because of short-term interests.

This paper is organized as follows: Section 2 describes related work in the field of query expansion, section 3 introduces the vector space model and query expansion procedures that have been developed for use in the vector space model. Section 4 describes the method for learning similarity functions and describes one of the functions in detail. Then section 5 describes the document collections we have used for evaluating our new algorithms and describes the evaluation methodology, section 6 describes the results of the evaluation. Finally section 7 summarizes this paper, draws some conclusions, and shows the essential factors for improving retrieval performance in CIR.

2 Related Work

Usage of short queries in IR produces a shortcoming in the number of documents ranked according to their similarity to the query. Thus IR systems try to reformulate the queries in a semi-automatic or automatic way. Several methods, called query expansion methods (QE), have been proposed to cope with this problem [3], [10]. These methods fall into three categories: usage of feedback information from the user (e.g. interactive QE), usage of information derived locally from the set of initially retrieved documents, and usage of information derived globally from the document collection. The goal of all QE methods is to finally find the optimal query which selects all the relevant documents. A comprehensive overview of newer procedures is available from Efthimiadis in [6]. Another newer technique, called local context analysis (LCA), was introduced by Xu and Croft in [15].

Newest procedures in the field of query expansion are dealing with query bases, a set of persistent past optimal queries, for investigating similarity measures between queries (refer to Raghavan, Sever and Alsaffar et al. in [11], [12] [2]). Wen et al. [14] are using query clustering techniques for discovering frequently asked questions or most popular topics on a search engine. This query

clustering method makes use of user logs which allows to identify the documents the users have selected for a query. The similarity between two queries may be deduced from the common documents the users selected for them ([4]). Cui et al. [5] take into account the specific characteristics of web searching, where a large amount of user interaction information is recorded in the web query logs, which may be used for query expansion. Agichtein et al. [1] are learning search engine specific query transformations for question answering in the web.

3 Basics and Terminology

In this section we introduce the vector space model (VSM) which is employed in our work. We introduce the pseudo relevance feedback method for query expansion and two of our newly developed methods for CIR.

Vector Space Model Documents as well as queries are represented in a common way using a set of terms. Terms are determined from words of the documents, usually during preprocessing phases (e.g. stemming and stopword elimination). In the following a term is represented by $t_i,\ 1 \leq i \leq M$, where M is the number of terms in the document collection.

The vector space model assigns weights to terms in queries and in documents and represents them as M dimensional vectors

$$d_j = (w_{1j}, w_{2j}, ..., w_{Mj})^T, \quad 1 \leq j \leq N, \tag{1}$$

$$q_k = (w_{1k}, w_{2k}, ..., w_{Mk})^T, \quad 1 \leq k \leq L, \tag{2}$$

where T indicates the transpose of the vector, w_{ij} or w_{ik} is the weight of term t_i in document d_j or query q_k, N is the number of documents and L is the number of queries contained in the document collection.

The result of the execution of a query is a list of documents ranked according to their similarity to the given query. The similarity $sim(d_j, q_k)$ between a document d_j and a query q_k is measured by the cosine of the angle between these two M dimensional vectors:

$$\text{sim}(d_j, q_k) = \frac{d_j^T \cdot q_k}{||d_j|| \cdot ||q_k||}, \tag{3}$$

where $\| \cdot \|$ is the Euclidean norm of a vector. In the case that the vectors are already normalized (and hence have a unit length) the similarity is simply the dot product between the two vectors d_j and q_k.

Query Expansion by Pseudo Relevance Feedback (PRF) After retrieval of the list of documents (in a first stage) highly ranked documents are all assumed to be relevant [15] and their terms (all of them or some highly weighted terms) are used for expanding the original query. Then documents are ranked again according to their similarity to the expanded query.

In this work we employ a variant of pseudo relevance feedback described by Kise et al. [9]. In our comparison with the newly developed methods, we will use the PRF method.

Let E be the set of document vectors given by

$$E = \left\{ d_j \left| \frac{\mathrm{sim}(d_j, q_k)}{\max_{1 \le i \le N}\{\mathrm{sim}(d_i, q_k)\}} \ge \theta \right. \right\} \tag{4}$$

where q_k is the original query and θ is a threshold parameter of the similarity. Then the sum D_k of the document vectors in E, $D_k = \sum_{d_j \in E} d_j$ is used as expansion terms for the original query. The expanded query vector q'_k is obtained by

$$q'_k = q_k + \alpha \frac{D_k}{||D_k||} \tag{5}$$

where α is a parameter for weighting the expansion terms. Then the documents are ranked again according to their similarity $\mathrm{sim}(d_j, q'_k)$.

Parameters θ in Equation 4 and α in Equation 5 are tuning parameters. During evaluation best parameter value settings have been obtained by experiment and those which give the highest average precision were selected for comparison against other methods.

Query Expansion by Methods developed for CIR In our approaches we use global relevance feedback which has been learned from previous queries; this is in contrast to local relevance feedback which is produced during execution of an individual query. All our new query expansion procedures work as follows:

- for each new query to be issued compute the similarities between the new query and each of the existing old queries
- select the old queries having a similarity to the new query which is greater than or equal to a given threshold
- from these selected old queries get the sets of relevant documents from the ground truth data
- from this set of relevant documents compute some terms for expansion of the new query
- use this terms to expand the new query and issue the new expanded query

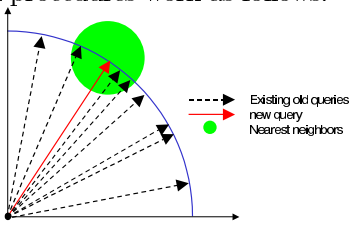

Fig. 1. Motivation for CIR methods: usage of the nearest neighbors

The algorithmic description is given here:

```
for each new query q do
    compute the set S = {q_k | sim(q_k, q) ≥ σ, 1 ≤ k ≤ L}
    compute the sets RD_k = {d_j | q_k ∈ S ∧ d_j is relevant to q_k}
    compute the expanded query q' = cirf(q, S, RD_k)
end
```

where S is the set of existing old queries q_k with a similarity of σ or higher to the new query q, RD_k are the sets of the documents being relevant to the queries q_k and $cirf$ is a function for query expansion.

The goal now is to find suitable functions $cirf$ which can be efficiently computed and which maximize the effectiveness of the new query q' in terms of recall

and precision. As is shown in figure 1 our approach is searching for neighbors of the new query. If suitable neighbors of a query q within a given distance are found, we try to derive information about the documents which are relevant to q from its nearest neighbors.

These functions introduce a new level of quality in the IR research area: while the term weighting functions such as tf-idf only work on documents and document collections, and relevance feedback works on a single query and uses information from their assumed relevant and non-relevant documents only, CIR now works on a single query, and uses the information of all other queries and their known relevant documents.

Methods Description. Due to lack of space we describe the methods informally very short. For detailed description and evaluation we point the reader to the referenced papers and articles.

***Q**uery **S**imilarity and Relevant **D**ocuments.* Method QSD ([7]) uses the relevant documents of the most similar queries for query expansion of a new query. The new query is rewritten as a sum of selected relevant documents of existing old queries, which have a high similarity to the new query, i.e.,

$$q' = q + \sum_{k=1}^{|S|} \sigma_k \frac{RD_k}{\| RD_k \|}, \tag{6}$$

where $|S|$ is the number of selected queries, σ_k are the similarities $\mathrm{sim}(q_k, q) \geq \sigma$ (σ is the threshold value) and RD_k are the sets of relevant documents.

***Q**uery **L**inear Combination and Relevant **D**ocuments.* Method QLD ([8]) uses the relevant documents of the most similar queries, which are used in re-writing the new query as a linear combination of the most similar queries. This query expansion method reconstructs the new query as a linear combination of existing old queries. Then the terms of the relevant documents of these existing old queries are used for query expansion, i.e.,

$$q' = q + \sum_{k=1}^{|S|} \tilde{\lambda_k} \frac{RD_k}{\| RD_k \|}, \tag{7}$$

where the $\tilde{\lambda_k}$ are parameter for weighting the expansion terms. The $\tilde{\lambda_k}$ are computed as follows: in most cases we cannot represent the new query q exactly as a linear combination of the old queries q_k, i.e., $q = \sum_{k=1}^{|S|} \lambda_k q_k$ will not have a solution for the coefficients λ_k. This equation is equivalent to a system of linear equations $Q\lambda = q$, where $Q = (q_1, q_2, \ldots, q_{|S|})$ is a matrix of M rows and $|S|$ columns and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{|S|})^T$ is a column vector consisting of $|S|$ elements. Because Q is normally singular ($M \gg |S|$) and there is no solution to the system, we find a vector $\tilde{\lambda}$ so that it provides a closest fit to the equation in some sense. Our approach is to minimize the Euclidean norm of the vector $Q\lambda - q$, i.e we solve

$$\tilde{\lambda} = argmin_\lambda \|Q\lambda - q\| \tag{8}$$

where $\tilde{\lambda} = (\tilde{\lambda_1}, \tilde{\lambda_2}, \ldots, \tilde{\lambda_{|S|}})^T$ is called the least squares solution for the system $Q\lambda = q$.

Limiting Factors in CIR Performance *Similarities of Queries to Documents.* One of the limiting factors for CIR retrieval performance is the similarity between the query and its non-relevant documents (as it is for non-CIR retrieval performance).

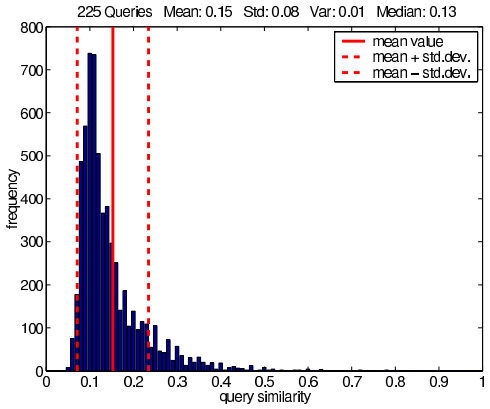

Fig. 2. CRAN: distribution of query similarities

Inter-Query Similarities. In our considerations for usage of similarities between different queries for retrieval performance improvements, we decided to analyze the inter-query similarities. We did not expect to have queries having highly correlated similarities as we would expect in real world applications. Indeed, the histograms show very low inter-query similarity for most of the text collections. Figure 2 displays the distribution of the inter-query similarity, excluding those similarities which are 0. Also the mean and the median value as well as the variance and the standard deviation are indicated in the graph. The vertical lines are: the mean similarity (solid line), and the values of the mean similarity $\pm$ the standard deviation (dotted lines).

Overlap of Relevant Documents. Another limiting factor for our CIR methods is some "overlapping" in relevant documents for different queries. We define the overlap of relevant documents as follows: Let $q_k, q_l \in Q,\ k \neq l$ be two different queries. Let RD_k and RD_l be the sets of documents which are relevant to queries q_k and q_l respectively. Then the overlap of relevant documents for these two queries is the number of documents in the set $O_{kl} = RD_k \cap RD_l = \{d_j |\ d_j \in RD_k \wedge d_j \in RD_l\}$. We expect retrieval performance improvements if the overlap of relevant documents is high.

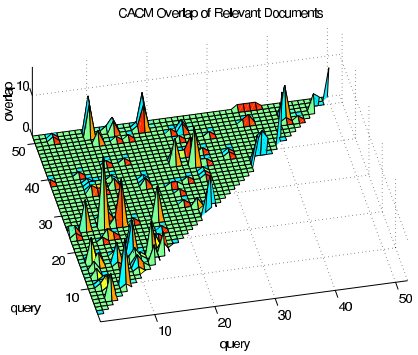

Fig. 3. CACM: overlap of relevant documents

4 Learning Similarity Functions

The motivation for learning similarity functions arises from the achieved performance improvements of our query expansion methods QSD and QLD.

Similarity between queries as it is used up to now is solely based on syntactical elements. Although we have used some normalization and cleaning operations

(stemming and stop-word-elimination) there is no further processing beyond the syntactical level. Similarity between two queries is high if they use the same words. Similarity is low if they use different words.

The same information need can be expressed in different queries, having a low similarity, although they are querying for the same facts and thus may have the same relevant documents. However, the methods developed up to now only use the inter-query similarity on the syntactical level, they do not consider the information need of the user. Figure 4 illustrates the proposed effect of learning, where the area of nearest neighborhood may change dramatically if the newly learned similarity functions are applied. In this way we can identify queries as nearest neighbors of a new query, even if they are far away (according to the standard cosine-similarity) from the new query.

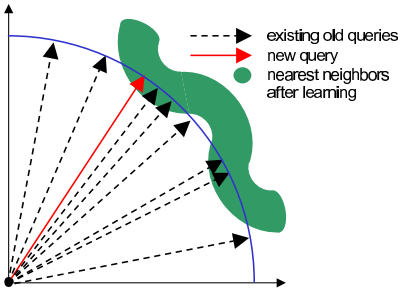

Fig. 4. Motivation for Learning Similarities: area of nearest neighbors changes dramatically

The Learning Problem We now formulate the learning of similarity functions as a minimization problem.

We measure the similarity of sets of relevant documents by

$$\mathrm{dsim}_{kl} := \mathrm{sim}(rd_k, rd_l), \tag{9}$$

where $\mathrm{sim}(\cdot,\cdot)$ is defined in Equation 3, and rd_i are the summarized and centered document vectors consisting of the relevant documents of query q_i, i.e.,

$$rd_i = \frac{1}{|RD_i|} \sum_{d_j \in RD_i} d_j \tag{10}$$

and the similarity between queries by

$$\mathrm{qsim}_{kl}(x) := \mathrm{sim}(g(x, q_k), h(x, q_l)) \tag{11}$$

where x is a vector of weights to be applied against the queries q_k and/or q_l with some functions g and h, each returning an M-dimensional vector which can be fed into the standard cosine similarity measure described in Equation 3.

The motivation for learning the weights for similarity functions is as follows: If the similarity between two different vectors rd_k, rd_l is high, then the similarity between the two queries q_k, q_l having these document vectors assigned as relevant documents should be high. If the similarity between vectors rd_k, rd_l is low, then the similarity between the corresponding two queries should be low. This directly leads to the functions f_{kl}, $1 \le k, l \le L$ to be minimized as

$$f_{kl}(x) = \mathrm{qsim}_{kl}(x) - \mathrm{dsim}_{kl} \tag{12}$$

and considers all pairs of queries. Let F be a vector-valued function consisting of L^2 functions, where each of these functions uses an M dimensional input vector x, i.e.,

$$F : \mathbb{R}^M \to \mathbb{R}^{L^2}$$
$$(x_1, x_2, \cdots, x_M)^T \mapsto (f_{11}(x), f_{12}(x), \cdots, f_{kl}(x), \cdots, f_{LL}(x))^T \quad (13)$$

Then we can state our learning problem as

$$\hat{x} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_M)^T = argmin_x ||F(x)||^2 = argmin_x \sum_{k,l=1}^{L} f_{kl}(x)^2 \quad (14)$$

i.e., we are searching for a vector $\hat{x}$ that minimizes the Euclidean norm of the function F.

The Similarity Functions The goal is to find reasonable functions $\mathrm{qsim}_{kl}(x)$ which give us significant performance improvements for IR whilst having a moderate computational complexity both in the learning process as well as during the application of the similarity measure in the query expansion methods QSD and QLD.

We have developed 9 reasonable functions. Due to lack of space we describe only one of them here.

Similarity Function F2 We first define the component-wise multiplication of the individual components of two vectors, denote it by $\dot{*}$ and use it in infix-notation:

$$\dot{*} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$
$$x \dot{*} y = (x_1, x_2, \cdots, x_n)^T \dot{*} (y_1, y_2, \cdots, y_n)^T = (x_1 y_1, x_2 y_2, \cdots, x_n y_n)^T$$

Then we define the new similarity function using the weights to be learned and denote it by a superscript

$$\mathrm{qsim}^2_{kl}(x) = \mathrm{sim}(q_k, x \dot{*} q_l) \quad (15)$$

leading to our minimization problem

$$\hat{x} = argmin_x \sum_{k,l=1}^{L} (\mathrm{qsim}^2_{kl}(x) - \mathrm{dsim}_{kl})^2 \quad (16)$$

5 Experimental Design

We use standard document collections and standard queries and questions provided by the SMART project and the TREC conferences. In addition we use some special collections that we have generated from the TREC collections to show special effects of our algorithms. In our experiments we used the following 10 collections:

- the SMART collections ADI, CACM, CISI, CRAN, MED and NPL.
- the TREC QA (question answering) collection prepared for the Question Answering track held at the TREC-9 conference, the QA-AP90 collection containing only those questions having a relevant answer document in the AP90 (Associated Press articles) document collection, the QA-AP90S collection (extracted from the QA-AP90 collection) having questions with similarity of 0.65 or above to any other question, and the QA-2001 collection prepared for the Question Answering track held at the TREC-10 conference.

On the one hand by utilizing these collections we take advantage of the ground truth data for performance evaluation. On the other hand we do not expect to have queries having highly correlated similarities as we would expect in a real world application. So it is a challenging task to show performance improvements for our methods.

Preparation of the Text Collections Terms used for document and query representation were obtained by stemming and eliminating stopwords. Then document and query vectors were created according to the so called tf-idf weighting scheme, where the document weights d_{ij} are computed as

$$d_{ij} = \sqrt{f_{ij}} \cdot idf_i \tag{17}$$

where f_{ij} is the raw frequency of term t_i, idf_i is the inverse document frequency $\log \frac{N}{n_i}$ of term t_i, and the query weights q_{ik} are computed as

$$q_{ik} = \sqrt{f_{ik}} \tag{18}$$

where f_{ik} is the raw frequency of term t_i in a query q_k.

Properties of the Text Collections Table 1 lists statistics about the collections after stemming and stopword elimination has been carried out; statistics about some of these collections before stemming and stopword elimination can be found in Baeza-Yates [3] and Kise et al. [9].

	ADI	CACM	CISI	CRAN	MED	NPL	QA	QA-AP90	QA-AP90S	QA-2001
size(MB)	0.1	1.2	1.4	1.4	1.1	3.8	28.2	3.7	3.7	20.1
number of documents	82	3204	1460	1400	1033	11429	6025	723	723	4274
number of terms	340	3029	5755	2882	4315	4415	48381	17502	17502	40626
mean number of terms per document	17.9 (short)	18.4 (short)	38.2 (med)	49.8 (med)	46.6 (med)	17.9 (short)	230.7 (long)	201.8 (long)	201.8 (long)	220.5 (long)
number of queries	35	52	112	225	30	93	693	353	161	500
mean number of terms per query	5.7 (med)	9.3 (med)	23.3 (long)	8.5 (med)	9.5 (med)	6.5 (med)	3.1 (short)	3.2 (short)	3.5 (short)	2.7 (short)
mean number of relev. documents per query	4.9 (low)	15.3 (med)	27.8 (high)	8.2 (med)	23.2 (high)	22.4 (high)	16.4 (med)	2.8 (low)	3.2 (low)	8.9 (med)

Table 1. Statistics about the test collections

Methodology of Evaluation The numerical methods used for function minimization do not guarantee that they will find a global minimum of the function. However they will find a local minimum in an area surrounding the initial start value. Thus we did the same experiment several times with different initial values.

The result of each experiment was the vector $\hat{x}$. We then fed these values into the QSD and QLD methods using the similarity measure $\text{qsim}_{kl}^2(x)$ as defined in Equation 15 for the query expansion methods.

The evaluation follows the "leave one out" technique used in several areas such as document classification, machine learning etc. From the set of L queries contained in each text collection we selected each query one after the other and treated it as a new query $q_l, 1 \leq l \leq L$. Then for each fixed query q_l we used the algorithm as described in section 3. Of course the now fixed query q_l itself does

not take part in the computation of the query expansion. We varied parameters of the algorithms according to suitable values, and selected those parameters where highest performance improvements (in terms of average precision over all queries) were achieved.

6 Results

The methods are denoted by adding the name of the learned similarity function to the basic name, i.e., QSDF2 denotes the QSD method after learning similarity function F2 using the similarity measure $\text{qsim}^2_{kl}(x)$ as defined in Equation 15.

Interpolated Average Precision Table 2 shows the interpolated average precision obtained by using the best parameter values for different methods. For each collection the best value of average precision is indicated by bold font, the second best value is indicated by italic font. In those cases, where our new methods outperform the PRF method, the value is underlined.

	ADI	CACM	CISI	CRAN	MED	NPL	QA	QA-AP90	QA-AP90S	QA-2001
VSM	0.375	0.130	0.120	0.384	0.525	0.185	0.645	0.745	0.643	0.603
PRF	0.390	0.199	0.129	0.435	**0.639**	**0.224**	0.685	0.757	0.661	**0.614**
QSD	0.374	0.237	0.142	0.428	0.503	0.184	0.727	*0.810*	0.786	0.603
QSDF2	*0.433*	**0.293**	**0.184**	*0.463*	*0.525*	*0.202*	*0.753*	**0.818**	*0.796*	*0.604*
QLD	0.369	0.227	0.171	0.436	0.507	0.185	0.734	0.812	0.789	0.603
QLDF2	**0.436**	*0.286*	*0.182*	**0.465**	*0.525*	0.196	**0.754**	**0.818**	**0.798**	*0.604*

Table 2. Interpolated average precision in CIR methods

Significance Testing Significance tests were applied to the results. Table 3 shows the results. Each row contains the results of two tests, i.e., test method X against method Y and vice versa.

- The indicator ++ (+) shows that method X is performing better than method Y at significance level $\alpha = 0.01$ ($\alpha = 0.05$).
- The indicator o shows that there is low probability that one of the methods is performing better than the other method.
- The indicator −− (−) shows that method Y is performing better than method X at significance level $\alpha = 0.01$ ($\alpha = 0.05$).

methods X	methods Y	ADI	CACM	CISI	CRAN	MED	NPL	QA	QA-AP90	QA-AP90S	QA-2001
PRF	VSM	+	++	++	++	++	++	++	+	o	++
QSD	PRF	o	o	o	o	−−	−−	++	++	++	−−
QSDF2	PRF	++	++	++	+	−−	−	++	++	++	−−
QSDF2	QSD	+	++	++	++	o	+	++	+	o	o
QLD	PRF	o	o	++	o	−−	−−	++	++	++	−−
QLDF2	PRF	++	+	++	+	−−	−−	++	++	++	−−
QLDF2	QLD	++	++	o	++	o	o	++	o	o	o

Table 3. Paired t-test results for significance levels $\alpha = 0.05$ and $\alpha = 0.01$

Relative Performance Improvements Table 4 shows the relative performance improvements for different methods. The ratio of improvement is computed as follows: let X be the average precision obtained by one of the methods and let Y be the average precision obtained by another method. Then the ratio is calculated by $ratio = \frac{X-Y}{Y}$. A positive value for the ratio indicates an improvement of method X over method Y, a negative value indicates a degradation in average precision from method X to Y.

methods X	Y	ADI	CACM	CISI	CRAN	MED	NPL	QA	QA-AP90	QA-AP90S	QA-2001
QSD	PRF	-4.0%	+18.8%	+9.8%	-1.7%	-21.3%	-17.8%	+6.1%	+6.9%	+18.8%	-1.8%
QSDF2	PRF	+10.9%	+47.0%	+42.2%	+6.3%	-17.9%	-9.7%	+10.0%	+8.0%	+20.4%	-1.7%
QSDF2	QSD	+15.5%	+23.7%	+29.6%	+8.2%	+4.3%	+9.9%	+3.6%	+1.0%	+1.3%	+0.1%
QLD	PRF	-5.5%	+14.1%	+32.3%	+0.1%	-20.7%	-17.2%	+7.2%	+7.2%	+19.3%	-1.8%
QLDF2	PRF	+11.9%	+43.5%	+40.8%	+6.8%	-17.8%	-12.5%	+10.1%	+7.9%	+20.6%	-1.6%
QLDF2	QLD	+18.3%	+25.7%	+6.5%	+6.7%	+3.6%	+5.7%	+2.7%	+0.7%	+1.1%	+0.2%

Table 4. Average precision improvement in different methods

Analysis of the Results From the average precision analysis we see that the QSDF2 method and the QLDF2 method perform best and second best in most cases. In all cases they also perform better than the basic method before learning.

For the MED and NPL text collections, the basic methods QSD and QLD do not perform better than the PRF method, nor do any of the methods after learning. We think that, in the case of the MED collection, this effect comes from the missing overlap of relevant documents, and in the case of the NPL collection from the high similarity of non-relevant documents to the queries.

In all but in one case we observe performance improvements after learning compared to the basic methods without learning. The highest performance improvement achieved is +40.3% (for the CACM collection). Only for the QA-2001 collection we observe a performance degradation for one method of -0.1% after learning; it should also be noted that for this collection the performance improvements achieved after learning are the lowest of all collections.

7 Conclusions

We have studied learning methods for improving retrieval performance in a restricted CIR environment where information about relevant documents from previous search processes carried out by several users is available for the current query.

Specifically, we developed, evaluated and analyzed new algorithms for query expansion, since query expansion methods are known to be successful in improving retrieval performance.

Results of the newly developed methods are encouraging. Retrieval performance improvements were achieved in most cases. For some text collections no significant retrieval performance improvements could be achieved, neither in the basic methods nor in applying the methods after learning similarity functions. We identified three essential factors for retrieval performance improvements:

- similarity between queries, also called inter-query similarity: we can not achieve performance improvements, if there are no pairs of queries with high similarities
- similarity of queries to their relevant documents and non-relevant documents: precision decreases, if non-relevant documents are ranked higher than relevant documents
- the overlap of relevant documents for pairs of queries: if there is no or low overlap in relevant documents, there are no document terms which are used for query expansion

We think that the first factor is the most important for our CIR methods. Best performance improvements have been achieved in text collections where the inter-query similarity is high, although the overlap in relevant documents is not high. Low or no retrieval performance improvements were achieved in those cases were the inter-query similarity is on average low.

For text collections, where similarity of queries to their non-relevant documents is high on average, we achieved low performance improvements.

For text collections, where the overlap of relevant documents is low or where no overlap in relevant documents exists, we did not achieve performance improvements, neither in the basic methods nor in the methods that have been applied after learning.

References

1. Eugene Agichtein, Steve Lawrence, and Luis Gravano. Learning search engine specific query transformations for question answering. *10th International WWW Conference*, pages 169–178, 2001.
2. Ali H. Alsaffar, Jitender S. Deogun, and Hayri Sever. Optimal queries in information filtering. *Foundations of Intelligent Systems, 12th International Symposium, Proceedings*, volume 1932 of *LNCS*, pages 435–443, 2000.
3. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, 1999.
4. Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Optimizing similarity using using multi-query relevance feedback. *JASIS*, 49(8):742–761, 1998.
5. Hang Cui, Ji-Rong Wen, Jian-Yun Nieand, and Wei-Ying Ma. Probabilistic query expansion using query logs. *11th International WWW Conference*, 2002.
6. Efthimis N. Efthimiadis. Query expansion. *Annual Review of Information Science and Technology*, 31:121–187, 1996.
7. Armin Hust, Stefan Klink, Markus Junker, and Andreas Dengel. Query Expansion for Web Information Retrieval. *WIR Workshop*, volume P-19 of *LNI*, pages 176–180, 2002.
8. Armin Hust, Stefan Klink, Markus Junker, and Andreas Dengel. Query Reformulation in Collaborative Information Retrieval. *IKS 2002*, pages 95–100, 2002.
9. Koichi Kise, Markus Junker, Andreas Dengel, and Keinosuke Matsumoto. Experimental evaluation of passage-based document retrieval. *ICDAR-01*, pages 592–596, 2001.
10. Christopher D. Manning and Hinrich Schütze. *Foundations of Natural Language Processing*. MIT Press, 1999.
11. Vijay V. Raghavan and Hayri Sever. On the reuse of past optimal queries. *18th ACM SIGIR*, pages 344–350, 1995.
12. Hayri Sever. *Knowledge Structuring for Database Mining and Text Retrieval Using Past Optimal Queries*. PhD thesis, University of Louisiana, Lafayette, 1995.
13. Ellen M. Voorhees and Donna K. Harman. Overview of the eighth text retrieval conference (TREC-8). *NIST Special Publication 500-246*, pages 1–23, 1999.
14. Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Clustering user queries of a search engine. *10th International WWW Conference*, pages 162–168, 2001.
15. Jinxi Xu and W. Bruce Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM TOIS*, 18(1):79–112, 2000.

Experiments in Document Clustering using Cluster Specific Term Weights

Christian Borgelt and Andreas Nürnberger

Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg, Germany
{borgelt,nuernb}@iws.cs.uni-magdeburg.de

Abstract. We study methods to initialize or bias different clustering methods using prior information about the "importance" of a keyword w.r.t. to the specific clusters. These studies give us hints on how to initialize clustering methods in order to improve the clustering performance if prior knowledge is available. This can be especially useful if a user-specific clustering of a document collection or web search result set is desired.

1 Introduction

The problem of finding descriptive weights for terms in document collections in order to improve retrieval performance has been studied extensively in the past (see, for instance, [12, 24, 23]). To achieve an improved classification or clustering performance for a given text collection, it is usually necessary to select a subset of all describing features (i.e. keywords) and/or to re-weight the features w.r.t. a specific classification or clustering goal. Consequently, several studies were conducted in this direction. For example, it was explored how to select keywords based on statistical and information theoretical measures [9, 21, 28] or how to combine clustering and keyword weighting techniques [10] in order to improve the clustering performance.

In prior work we studied different hard and fuzzy clustering methods with and without variances [5]. These experiments indicated that the use of variances—which can be considered as a method for cluster specific keyword weighting—can improve the clustering performance. Nevertheless, it is still unclear to what extent term re-weighting influences the clustering performance and whether initial—global or cluster specific—term re-weighting can be used to bias or improve the performance. Therefore, in the following, we compare clustering with and without term re-weighting techniques using different hard and fuzzy clustering methods.

This paper is organized as follows: In Section 2 we briefly review some basics of fuzzy clustering and a fuzzified version of learning vector quantization. In Section 3 we review pre-processing methods for documents and in particular the vector space model, which we use to represent documents. In Section 4 we

present our experimental results of clustering web page collections using different global and cluster-specific term weighting approaches and finally, in Section 5, we draw conclusions from our discussion.

2 Clustering

The best-known classical prototype based hard clustering methods are *c-means clustering* [7, 4] and *learning vector quantization* [17, 18]. In the following, we briefly describe their generalizations to fuzzy clustering and fuzzified learning vector quantization as we use it in our experiments. For a more detailed discussion and evaluation of these methods for document clustering see [5].

2.1 Fuzzy Clustering

While most classical clustering algorithms assign each datum to exactly one cluster, thus forming a crisp partition of the given data, fuzzy clustering allows for *degrees of membership*, to which a datum belongs to different clusters [1, 2, 14]. Most fuzzy clustering algorithms are objective function based: they determine an optimal (fuzzy) partition of a given data set $\mathbf{X} = \{\boldsymbol{x}_j \mid j = 1, \ldots, n\}$ into c clusters by minimizing an objective function

$$J(\mathbf{X}, \mathbf{U}, \mathbf{C}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} d_{ij}^{2}$$

subject to the constraints

$$\forall i; 1 \leq i \leq c: \quad \sum_{j=1}^{n} u_{ij} > 0, \qquad \text{and} \qquad \forall j; 1 \leq j \leq n: \quad \sum_{i=1}^{c} u_{ij} = 1,$$

where $u_{ij} \in [0, 1]$ is the membership degree of datum $\boldsymbol{x}_j$ to cluster i and d_{ij} is the distance between datum $\boldsymbol{x}_j$ and cluster i. The $c \times n$ matrix $\mathbf{U} = (u_{ij})$ is called the *fuzzy partition matrix* and $\mathbf{C}$ describes the set of clusters by stating location parameters (i.e. the cluster center) and maybe size and shape parameters for each cluster. The parameter w, $w > 1$, is called the *fuzzifier* or *weighting exponent*. It determines the "fuzziness" of the classification: with higher values for w the boundaries between the clusters become softer, with lower values they get harder. Usually $w = 2$ is chosen. Hard clustering results in the limit for $w \to 1$. However, a hard assignment may also be determined from a fuzzy result by assigning each data point to the cluster to which it has the highest degree of membership.

Since the objective function J cannot be minimized directly, an iterative algorithm is used, which alternately optimizes the membership degrees and the cluster parameters [1, 2, 14]. That is, first the membership degrees are optimized for fixed cluster parameters, then the cluster parameters are optimized for fixed membership degrees. The main advantage of this scheme is that in each of the two steps the optimum can be computed directly. By iterating the two steps

the joint optimum is approached (although, of course, it cannot be guaranteed that the global optimum will be reached—the algorithm may get stuck in a local minimum of the objective function J). The update formulae are derived by simply setting the derivative of the objective function J w.r.t. the parameters to optimize equal to zero (necessary condition for a minimum).

Depending on the distance measure used, several different fuzzy clustering algorithms can be distinguished. Classical fuzzy c-means clustering employs the *Euclidean distance*, while Gustafson-Kessel algorithm [13] uses the *Mahalanobis distance* and the fuzzy maximum likelihood estimation (FMLE) algorithm [11] is based on the assumption that the data was sampled from a mixture of c multivariate normal distributions as in the statistical approach of mixture models [8, 3]. It is worth noting that of both the Gustafson-Kessel as well as the FMLE algorithm there exist so-called *axes-parallel* versions, which restrict the covariance matrices to diagonal matrices and thus allow only axes-parallel ellipsoids [15]. These variants have certain advantages w.r.t. robustness and execution time.

2.2 Learning Vector Quantization

Learning vector quantization [17, 18], in its classical form, is a competitive learning algorithm that has been developed in the area of artificial neural networks and that can be applied to classified as well as unclassified data. Here we confine ourselves to unclassified data, where the algorithm consists in iteratively updating a set of c so-called *reference vectors* $\boldsymbol{\mu}_i$, $i = 1, \ldots, c$, each of which is represented by a neuron. For each data point $\boldsymbol{x}_j$, $j = 1, \ldots, n$, the closest reference vector (the so-called "winner neuron") is determined and then this reference vector (and only this vector) is updated according to

$$\boldsymbol{\mu}_i^{(\text{new})} = \boldsymbol{\mu}_i^{(\text{old})} + \eta_1 \left(\boldsymbol{x}_j - \boldsymbol{\mu}_i^{(\text{old})} \right), \tag{1}$$

where η_1 is a learning rate. This learning rate usually decreases with time in order to avoid oscillations and to enforce the convergence of the algorithm.

Membership degrees can be introduced into this basic algorithm in two different ways. In the first place, one may employ an activation function for the neurons, for which a radial function like the

$$\text{Cauchy function} \quad f(r) = \frac{1}{1 + r^2} \qquad \text{or the} \quad \text{Gaussian function} \quad f(r) = e^{-\frac{1}{2} r^2}$$

may be chosen, where r is the (radial) distance from the reference vector. In this case all reference vectors are updated for each data point, with the update being weighted with the value of the activation function. However, this scheme, which is closely related to *possibilistic fuzzy clustering* [19], usually leads to unsatisfactory results, since there is no dependence between the clusters, so that they tend to end up at the center of gravity of all data points. This corresponds to the fact that in possibilistic fuzzy clustering the objective function is truly minimized only if all cluster centers are identical [27]. Useful results are obtained only if the method gets stuck in a local minimum, which is an undesirable situation.

An alternative is to rely on a normalization scheme as in *probabilistic fuzzy clustering*, that is, to compute the weight for the update of a reference vector as the relative inverse (squared) distance from this vector, or as the *relative* activation of a neuron. This is the approach we employ here.

Furthermore we associate with each neuron not only a reference vector $\boldsymbol{\mu}_i$, but also a covariance matrix $\boldsymbol{\Sigma}_i$, which describes the shape and (if we do not require it to be normalized to determinant 1) the size of the represented cluster. A derivation of the update rule for this covariance matrix can be found in [5]. It should be noted that versions of this algorithm that require the covariance matrix to be normalized to determinant 1 or restrict the covariance matrix to a diagonal matrix may be considered, too. Such constraints can improve the robustness or the execution time of the algorithm. Finally it should be noted that the updates may be executed in batch mode, aggregating the changes resulting from the data points and actually updating the reference vectors and covariance matrices only at the end of an epoch.

3 Clustering Document Collections

To be able to cluster text document collections with the methods discussed above, we have to map the text files to numerical feature vectors. Therefore, we first applied standard preprocessing methods, i.e., stopword filtering and stemming (using the Porter Stemmer [22]), encoded each document using the vector space model [23] and finally selected a subset of terms as features for the clustering process as briefly described in the following.

3.1 The Vector Space Model

The vector space model represents text documents as vectors in an m-dimensional space, i.e., each document j is described by a numerical feature vector $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jm})$. Each element of the vector represents a word of the document collection, i.e., the size of the vector is defined by the number of words of the complete document collection.

For a given document j the so-called weight x_{jk} defines the importance of the word k in this document with respect to the given document collection C. Large weights are assigned to terms that are frequent in relevant documents but rare in the whole document collection [24]. Thus a weight x_{jk} for a term k in document j is computed as the term frequency tf_{jk} times the inverse document frequency idf_k, which describes the term specificity within the document collection.

In [25] a weighting scheme was proposed that has meanwhile proven its usability in practice. Besides term frequency and inverse document frequency (defined as $\mathrm{idf}_k = \log(n/n_k)$), a length normalization factor is used to ensure that all documents have equal chances of being retrieved independent of their lengths:

$$x_{jk} = \frac{\mathrm{tf}_{jk} \log \frac{n}{n_k}}{\sqrt{\sum_{l=1}^{m} \left(\mathrm{tf}_{jl} \log \frac{n}{n_l} \right)^2}}, \tag{2}$$

where n is the size of the document collection C, n_k the number of documents in C that contain term k, and m the number of terms that are considered.

Based on a weighting scheme a document j is described by an m-dimensional vector $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jm})$ of term weights and the similarity S of two documents (or the similarity of a document and a query vector) can be computed based on the inner product of the vectors (by which—if we assume normalized vectors—the cosine between the two document vectors is computed), i.e.

$$S(\boldsymbol{x}_j, \boldsymbol{x}_k) = \sum_{l=1}^{m} x_{jl} \cdot x_{kl}. \tag{3}$$

For a more detailed discussion of the vector space model and weighting schemes see, for instance, [12, 24, 23].

Note that for normalized vectors the scalar product is not much different in behavior from the Euclidean distance, since for two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ it is

$$\cos\varphi = \frac{\boldsymbol{xy}}{|\boldsymbol{x}| \cdot |\boldsymbol{y}|} = 1 - \frac{1}{2}\, d^2\!\left(\frac{\boldsymbol{x}}{|\boldsymbol{x}|}, \frac{\boldsymbol{y}}{|\boldsymbol{y}|}\right).$$

Although the scalar product is faster to compute, it enforces spherical clusters. Therefore we rely on the Mahalanobis distance in our approach.

3.2 Index Term Selection

To reduce the number of words in the vector description we applied a simple method for keyword selection by extracting keywords based on their entropy. In the approach discussed in [16], for each word k in the vocabulary the entropy as defined by [20] was computed:

$$W_k = 1 + \frac{1}{\log_2 n} \sum_{j=1}^{n} p_{jk} \log_2 p_{jk} \qquad \text{with} \quad p_{jk} = \frac{\mathrm{tf}_{jk}}{\sum_{l=1}^{n} \mathrm{tf}_{lk}}, \tag{4}$$

where tf_{jk} is the frequency of word k in document j, and n is the number of documents in the collection. Here the entropy gives a measure how well a word is suited to separate documents by keyword search. For instance, words that occur in many documents will have low entropy. The entropy can be seen as a measure of the importance of a word in the given domain context. As index words a number of words that have a high entropy relative to their overall frequency have been chosen, i.e. of words occurring equally often those with the higher entropy can be preferred. Empirically this procedure has been found to yield a set of relevant words that are suited to serve as index terms [16].

However, in order to obtain a fixed number of index terms that appropriately cover the documents, we applied a greedy strategy: from the first document in the collection select the term with the highest relative entropy as an index term. Then mark this document and all other documents containing this term. From the first of the remaining unmarked documents select again the term with the

Label	Dataset Category	Associated Theme
A	Commercial Banks	Banking & Finance
B	Building Societies	Banking & Finance
C	Insurance Agencies	Banking & Finance
D	Java	Programming Lang.
E	C / C++	Programming Lang.
F	Visual Basic	Programming Lang.
G	Astronomy	Science
H	Biology	Science
I	Soccer	Sport
J	Motor Racing	Sport
K	Sport	Sport

Table 1. Categories and Themes of the used benchmark data set of web pages.

highest relative entropy as an index term. Then mark again this document and all other documents containing this term. Repeat this process until all documents are marked, then unmark them all and start again. The process can be terminated when the desired number of index terms have been selected.

4 Experiments

For our experimental studies we chose the collection of web page documents used in [26].[1] The data set consists of 11,000 web pages classified into 11 equally-sized categories each containing 1,000 web documents. To each category one of four distinct themes, namely Banking and Finance, Programming Languages, Science, and Sport was assigned as shown in Table 1.

In the following we present results we obtained using the preprocessing strategies described above. After stemming and stop word filtering we obtained 163,860 words. This set was further reduced by removing terms that are shorter than 4 characters and that occur less then 15 times or more than $11,000/12 \approx 917$ times in the whole collection. In this way we made sure that no words that perfectly separate one class from another are used in the describing vectors. From the remaining 10626 words we selected 400 words by applying the greedy index-term selection approach described in Section 3.2. For our clustering experiments we selected finally subsets of the 50, 100, 150, ..., 350, 400 most frequent words in the subset to be clustered. Based on these words we determined vector space descriptions for each document (see Section 3.1, Equation (2)) that we used in our clustering experiments. All vectors were normalized to unit length (*after* the subset selection).

To assess the clustering performance using term re-weighting techniques, we computed the performance on the same data sets used in our previous experi-

[1] This collection is available for download at
http://www.pedal.rdg.ac.uk/banksearchdataset

ments [5], i.e., we clustered the union of the dissimilar data sets A and I, and the semantically more similar data sets B and C. In a third experiment we used all classes and tried to find clusters describing the four main themes, i.e., banking, programming languages, science, and sport.

For our experiments we used *c*-means, fuzzy clustering and learning vector quantization methods. The learning vector quantization algorithm updated the cluster parameters once for every 100 documents. [2]

A detailed discussion of the performance of these methods with and without cluster centers normalized to unit length, with and without variances (i.e., spherical clusters and axes-parallel ellipsoids—diagonal covariance matrices—of equal size), and with the inverse squared distance or the Gaussian function for the activation / membership degrees can be found in [5]. Here, however, we focus on term re-weighting aspects.

4.1 Clustering using Variances

Our prior experiments in document clustering [5] indicated that the use of variances—which can be seen as a method for cluster specific keyword weighting —can sometimes improve the clustering performance and stability. However, in our first studies we restricted ourselves to analyze the performance using mean performances and variances. As a consequence, the causes for the differences in the performance remained somewhat unclear. Therefore we repeated several of the experiments and present in Figures 2 to 3 the results obtained with cluster centers normalized to length 1 with and without variances for hard *c*-means, fuzzy *c*-means and (fuzzified) learning vector quantization. All results represent the values of ten runs, which differed in the initial cluster positions and the order in which documents were processed. For the experiments with variances we restricted the maximum ratio of the variances to $1.2^2 : 1 = 1.44 : 1$, which seemed to yield the best results over all three clustering experiments.

The dotted lines show the default accuracy (obtained if all documents are assigned to the majority class). The grey horizontal lines in each block, which are also marked by diamonds to make them more easily visible, show the average classification accuracy (computed from a confusion matrix by permuting the columns so that the minimum number of errors results) in percent (left axis), while the black crosses indicate the performance single experiments. The grey dots and lines close to the bottom show the average execution times in seconds (right axis), while the smaller grey dots and lines at the top of each diagram show the performance of a Naïve Bayes Classifier trained with the corresponding subset of words. The Naïve Bayes Classifier can be considered as an upper limit, while the default accuracy is a lower baseline.

For all data sets the clustering process for fuzzy *c*-means and (fuzzified) learning vector quantization is much more stable than *c*-means. However, all

[2] All experiments were carried out with a program written in C and compiled with gcc 3.3.3 on a Pentium 4C 2.6GHz system with 1GB of main memory running S.u.S.E. Linux 9.1. The program and its sources can be downloaded free of charge at http://fuzzy.cs.uni-magdeburg.de/~borgelt/cluster.html.

methods seem to switch between two strong local minima for the semantically similar data sets B and C.

The introduction of variances increases the performance of fuzzy c-means in all cases. However, the performance for c-means is only improved for the two class problem with data sets A and I and the four class problem, while the performance of (fuzzified) learning vector quantization is improved for the semantically more similar data sets B and C and the four class problem.

4.2 Keyword Weighting by Information Gain

Information gain (also known as *mutual (Shannon) information* or *(Shannon) cross entropy*), which is frequently used in decision tree learning, measures the average or expected entropy reduction resulting from finding out the value of a specific attribute. In text categorization information gain can be used to measure how well a term can be used to categorize a document, i.e., it measures the entropy reduction based on this specific term.

The information gain of a term t_k for a given set of r classes c_i is defined as:

$$\begin{aligned} I_{\text{gain}}(t_k) = & -\sum_{i=1}^{r} P(c_i) \log_2 P(c_i) \\ & + P(t_k) \sum_{i=1}^{r} P(c_i|t_k) \log_2 P(c_i|t_k) \\ & + P(\overline{t}_k) \sum_{i=1}^{r} P(c_i|\overline{t}_k) \log_2 P(c_i|\overline{t}_k) \end{aligned} \tag{5}$$

The information gain values are then either used to re-weight the terms of each document or to initialize the cluster-specific variances (see below).

4.3 Re-Scaling the Document Space

In order to study the effects of keyword weighting, we computed the "importance" of each keyword for the classification of a document based on the information gain (see above). These "importance" values are then used to re-weight the terms in each document by computing

$$x^*_{jk} = x_{jk} \cdot (I_{\text{gain}}(t_k) + o) \tag{6}$$

and then re-normalizing the document vectors to unit length, resulting in a re-scaling of the document space with respect to the importance of a keyword.

The offset o in the above formula was computed as

$$o = \frac{\max_{t_k \in T} I_{\text{gain}}(t_k) - r \cdot \min_{t_k \in T} I_{\text{gain}}(t_k)}{r - 1},$$

where r is a user-specified maximum ratio of the scaling factors for different terms and T is the current set of index terms. From several experiments we

conducted it seems that values of r must be small (close to 1) in order not to spoil the performance completely. Here we chose $r = 1.5$.

The results of these experiments are shown in the top rows of Figures 4 to 6. As can be seen, no gains result in any of the cases. The accuracy rather deteriorates slightly, an effect that gets stronger with higher values of r as we observed in other experiments. Hence we can conclude that re-scaling the document space in the way described does not lead to an improved performance.

4.4 Cluster Specific Keyword Weights

Instead of using the information gain to re-scale the document space one may also add shape parameters (i.e., (co)variances) to the cluster prototypes, which are initialized according to the "importance" of a term. This has the advantage that term weights can be cluster specific, since each cluster may use a different set of variances.

To evaluate this approach, we proceeded as follows: in a first step we clustered the documents with randomly chosen starting points and without variances. Afterwards, the best matching classes are automatically assigned by evaluating the confusion matrix of the classification result obtained with the learned clusters and the correct document classes.

Then the cluster prototypes were enhanced with cluster-specific variances computed as the product of the term frequency in the class and the information gain of the term w.r.t. a separation of the class assigned to the cluster from all other classes. In order to keep the cluster shapes close to spherical, we restricted the maximum ratio of the variances to $1.2^2 : 1 = 1.44 : 1$ (cf. Section 4.1. Other values for this maximum ratio (higher as well as lower) led to worse results. Especially larger values considerably worsened the performance.

Finally, in a second clustering run, these enhanced cluster prototypes were optimized without changing the variances (only the cluster centers were adapted).

The results of these experiments are shown in the bottom rows of Figures 4 to 6. As can be seen, the cluster-specific variances stabilize the results for the four cluster problem and—though only very slightly—improve the performance for the two cluster problems. Thus we can conclude that cluster-specific variance may provide some means for term weighting. However, the approach seems to be very sensitive to the parameter settings. Furthermore, the computational costs are very high.

4.5 Choosing Initial Cluster Centers

As we mentioned in Section 4.1 all clustering methods seem to switch between local minima depending on the initial cluster centers choosen—which is in fact a well known clustering problem, especially for the less robust c-means algorithm, which is prone to get stuck in local optima easily. Therefore we studied a quite simple initialization approach: for each class we sorted the index terms w.r.t. the product of the term frequency in the class and the information gain of the term

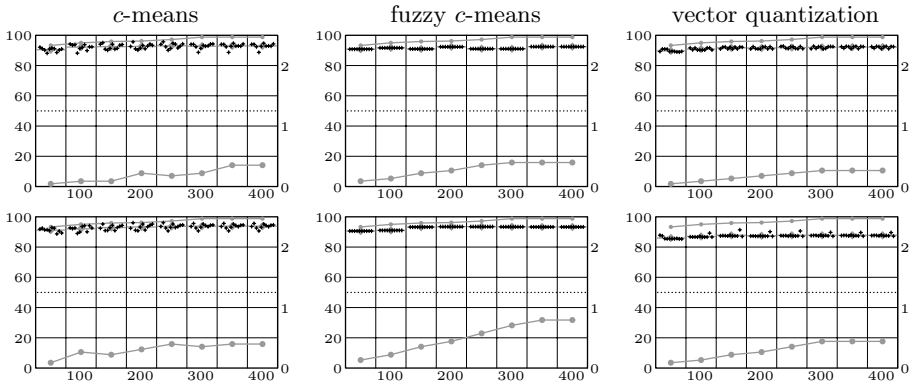

Fig. 1. Classification accuracy over number of keywords on commercial banks versus soccer (top row: standard, bottom row: with adaptable variances).

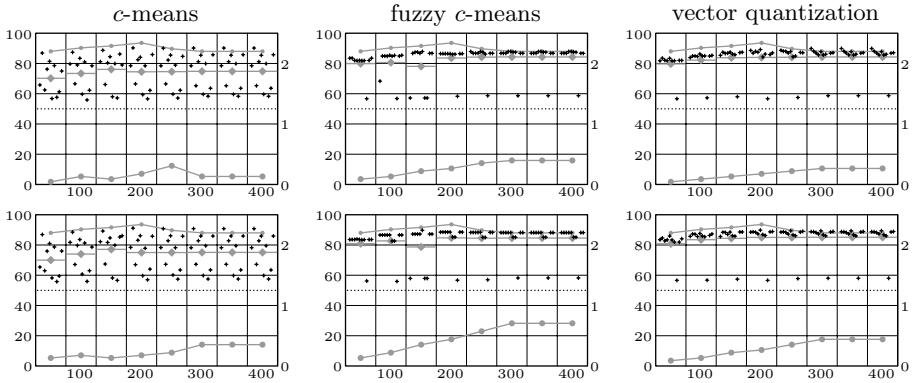

Fig. 2. Classification accuracy on building companies versus insurance agencies (top row: standard, bottom row: with adaptable variances).

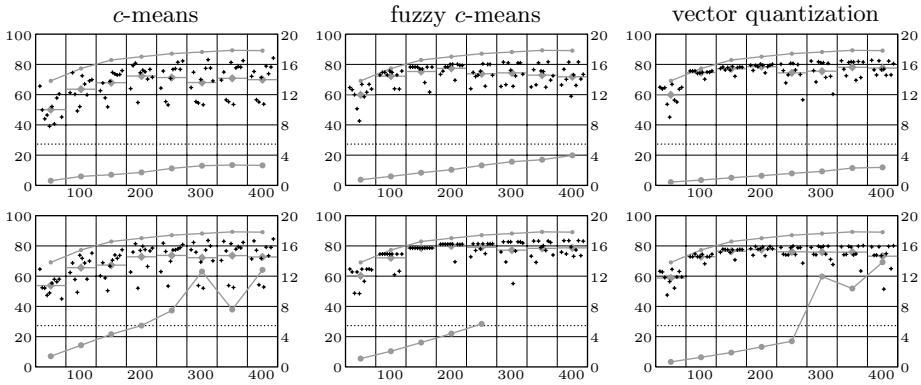

Fig. 3. Classification accuracy on major themes (four clusters; top row: standard, bottom row: with adaptable variances).

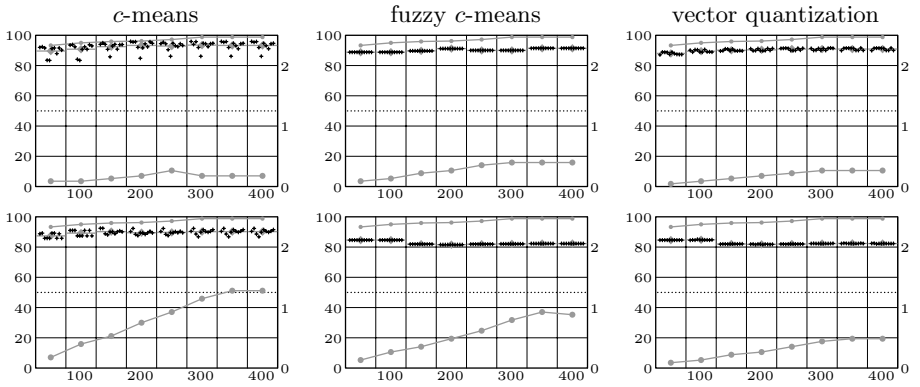

Fig. 4. Classification accuracy on commercial banks versus soccer (top row: document space re-scaled, bottom row: fixed cluster specific variances).

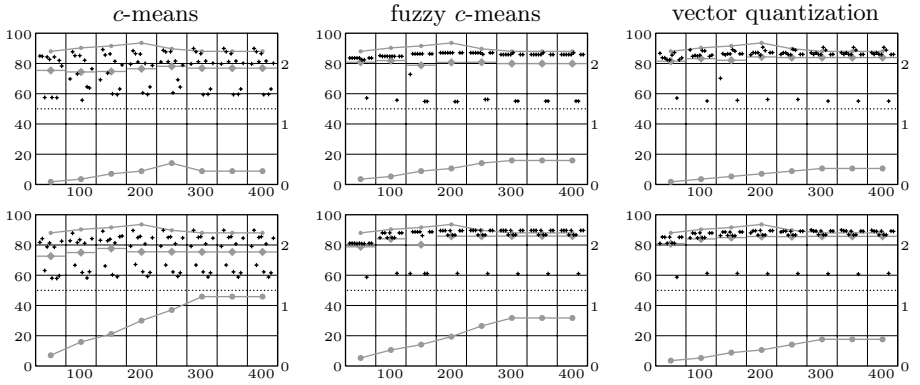

Fig. 5. Classification accuracy on building companies versus insurance agencies (top row: document space re-scaled, bottom row: fixed cluster specific variances).

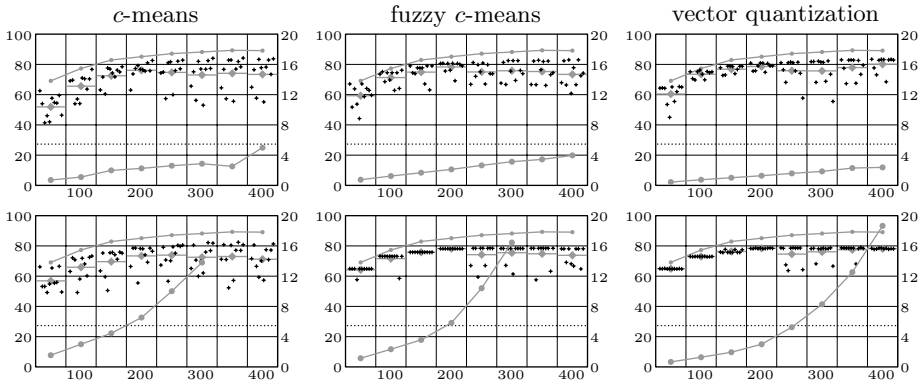

Fig. 6. Classification accuracy on major themes (top row: document space re-scaled, bottom row: fixed cluster specific variances).

w.r.t. a separation of the class from all other classes (cf. Section 4.4). Then we selected the first k words in these lists and initialized the cluster center using the same value for each selected word and zero for all others, finally normalizing the vector to unit length. Even for fairly small values of k (i.e. few selected words), this initialization results in a very stable clustering performance. Thus—similar to the idea of weight initialization in order to bias the clustering process—known describing keywords can be used in order to initialize the clustering process. In this way unwanted local minima may be avoided and the results may be stabilized.

5 Conclusions

Our experiments show that including prior information about the "importance" or "goodness" of a keyword for a desired class or cluster can, in principle, improve the clustering performance. However, it is fairly difficult to find a good way of scaling the documents or enhancing the cluster prototypes in an appropriate way. Scaling the document space does not yield any improvement at all. On the other hand, cluster-specific variances derived from the "importance" of index terms can slightly improve and stabilize the clustering performance. However, the gains are marginal and the approaches seem to be fairly sensitive to parameter settings.

References

1. J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, NY, USA 1981
2. J.C. Bezdek, J. Keller, R. Krishnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Kluwer, Dordrecht, Netherlands 1999
3. J. Bilmes. A Gentle Tutorial on the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. University of Berkeley, Tech. Rep. ICSI-TR-97-021, 1997
4. H.H. Bock. *Automatische Klassifikation.* Vandenhoeck & Ruprecht, Göttingen, Germany 1974
5. C. Borgelt and A. Nürnberger. Fast Fuzzy Clustering of Web Page Collections. *Proc. PKDD Workshop on Statistical Approaches for Web Mining (SAWM, Pisa, Italy).* 2004 (to appear).
6. A.P. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society (Series B)* 39:1–38. Blackwell, Oxford, United Kingdom 1977
7. R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* J. Wiley & Sons, New York, NY, USA 1973
8. B.S. Everitt and D.J. Hand. *Finite Mixture Distributions.* Chapman & Hall, London, UK 1981
9. G. Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research* 3:1289-1305, 2003
10. H. Frigui and O. Nasraoui. Simultaneaous Clustering and Dynamic Keyword Weighting for Text Douments. M.W. Berry, ed. *Survey of Text Mining*, 45–72. Springer, New York, NY USA 2003

11. I. Gath and A.B. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE Trans. Pattern Analysis & Machine Intelligence* 11:773–781. IEEE Press, Piscataway, NJ, USA, 1989
12. W.R. Greiff. A Theory of Term Weighting Based on Exploratory Data Analysis. *Proc. 21st Ann. Int. Conf. on Research and Development in Information Retrieval (Sydney, Australia)*, 17–19. ACM Press, New York, NY, USA 1998
13. E.E. Gustafson and W.C. Kessel. Fuzzy Clustering with a Fuzzy Covariance Matrix. *Proc. 18th IEEE Conference on Decision and Control (IEEE CDC, San Diego, CA)*, 761–766, IEEE Press, Piscataway, NJ, USA 1979
14. F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis.* J. Wiley & Sons, Chichester, England 1999
15. F. Klawonn and R. Kruse. Constructing a Fuzzy Controller from Data. *Fuzzy Sets and Systems* 85:177-193. North-Holland, Amsterdam, Netherlands 1997
16. A. Klose, A. Nürnberger, R. Kruse, G.K. Hartmann, and M. Richards. Interactive Text Retrieval Based on Document Similarities. *Physics and Chemistry of the Earth, Part A: Solid Earth and Geodesy* 25:649–654. Elsevier, Amsterdam, Netherlands 2000
17. T. Kohonen. *Learning Vector Quantization for Pattern Recognition.* Technical Report TKK-F-A601. Helsinki University of Technology, Finland 1986
18. T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Heidelberg, Germany 1995 (3rd ext. edition 2001)
19. R. Krishnapuram and J. Keller. A Possibilistic Approach to Clustering, *IEEE Transactions on Fuzzy Systems*, 1:98-110. IEEE Press, Piscataway, NJ, USA 1993
20. K.E. Lochbaum and L.A. Streeter. Combining and Comparing the Effectiveness of Latent Semantic Indexing and the Ordinary Vector Space Model for Information Retrieval. *Information Processing and Management* 25:665–676. Elsevier, Amsterdam, Netherlands 1989
21. D. Mladenic. Using Text Learning to help Web browsing. *Proc. 9th Int. Conf. on Human-Computer Interaction.* New Orleans, LA, USA 2001
22. M. Porter. An Algorithm for Suffix Stripping. *Program: Electronic Library & Information Systems* 14(3):130–137. Emerald, Bradford, United Kingdom 1980
23. G. Salton, A. Wong, and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM* 18:613–620 ACM Press, New York, NY, USA 1975
24. G. Salton and C. Buckley. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management* 24:513–523. Elsevier, Amsterdam, Netherlands 1988
25. G. Salton, J. Allan, and C. Buckley. Automatic Structuring and Retrieval of Large Text Files. *Communications of the ACM* 37:97–108. ACM Press, New York, NY, USA 1994
26. M.P. Sinka, and D.W. Corne. A Large Benchmark Dataset for Web Document Clustering. *A. Abraham, J. Ruiz-del-Solar, and M. Köppen (eds.), Soft Computing Systems: Design, Management and Applications*, 881–890. IOS Press, Amsterdam, The Netherlands 2002
27. H. Timm, C. Borgelt, and R. Kruse. A Modification to Improve Possibilistic Cluster Analysis. *Proc. IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2002, Honolulu, Hawaii).* IEEE Press, Piscataway, NJ, USA 2002
28. Y. Yang and J.O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. *Proc. 14th Int. Conf. on Machine Learning (ICML'97, Nashville, TN)*, 412–420. Morgan Kaufman, San Mateo, CA, USA 1997

Wrapper Generation with Patricia Trees

Sven Meyer zu Eißen smze@upb.de
Benno Stein stein@upb.de

Paderborn University
Department of Computer Science
D-33095 Paderborn, Germany

Abstract The automatic processing of search results that stem from Web-based search interfaces has come into focus, and it will remain important (as long as XML is not a universally applied technology). The reasons for this are twofold: (1) The need for value-added services such as filtering or graphical preparation of search results will increase. (2) The manual creation of tailored parsers for the information extraction from HTML pages cannot keep pace with the fast changing presentation of the search results in right these pages.

Automatic wrapper generation addresses this problem. It means the construction of a tailored parser for a certain type of HTML page with a minimum of manual intervention. This paper introduces the state of the art and presents an own development: A two-stage approach that combines highly efficient suffix matching based on a modified Patricia tree along with a knowledge-based analysis of candidate token sequences.

Key words: Information Extraction, Automatic Wrapper Generation, Wrapper Induction, Web Mining, Information Retrieval

1 Introduction

Web-based search interfaces are widely used as front-ends for information sources such as Web search engines, digital libraries, online shops, and other types of databases. Starting with a keyword search, they generate HTML result pages that contain a list of the found records. Such a semi-structured representation may be adequate for a human reader of this page; however, it is difficult to be further processed by applications that provide value-added services such as filtering, grouping, re-arranging, or graphical preparation: The generated record lists are not directly machine-readable and need to be "disrobed" of their wrapping code. Figure 1 shows an example.

Automatic wrapper generation deals with this problem; it aims at the automatic construction of a parser that extracts the interesting information by finding records and eliminating superfluous HTML code.[1] The following list mentions several challenges for automatic wrapper generation, and it also shows its importance for value-adding services.

[1] The term "automatic wrapper generation" may be misleading; perhaps a better description is "automatic parser generation for wrapping code".

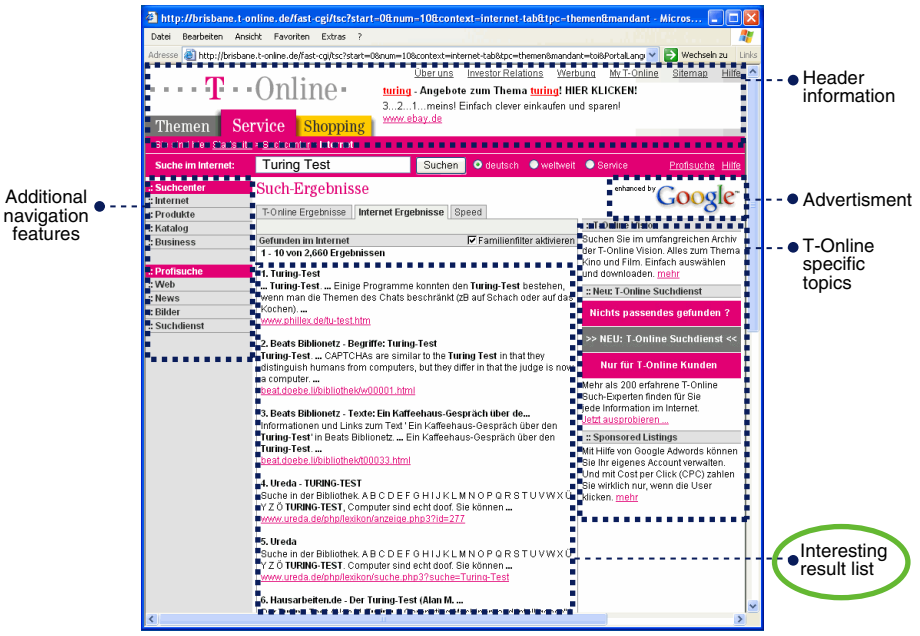

Figure 1. The snapshot shows a clipping of the T-Online search interface. For a subsequent processing of the search results a special block (bottom middle) has to be found as well as correctly parsed.

(1) A result list of records is embedded in header and footer information, which typically contains HTML code for navigation, logos, copyright notices, advertisements, etc.
(2) Each search interface brings along its own concept of wrapping its records, which includes particular font styles, enumeration styles, etc.
(3) The data structure within a record may vary with respect to presentation and data element constraints.
(4) Recurring navigational information like “search more of this” may be contained within a record.
(5) Structure and presentation of a generated list may change every now and then, when the provider modifies the design of the presentation.

The paper in hand is devoted to this problem; more specifically, it focuses on the extraction of records from Web search engines. This work is also related to our AIsearch project where search results from different information sources are grouped thematically within a categorization step [15, 18]. In this context, especially Point 5 is of a high importance, since we experience the generated HTML pages to change frequently. [2]

We present a two-stage approach that combines highly efficient suffix matching based on a modified Patricia tree with a knowledge-based analysis of candidate token

[2] This in turn means that human intervention becomes necessary to adapt in AIsearch the respective parser code for this information source.

sequences. The remainder of this paper is organized as follows. The next subsection gives an overview of existing approaches to automatic wrapper generation. Section 2 introduces our approach, and Section 3 presents some analysis results.

1.1 Classification of Existing Approaches

To hand-craft a tailored information extraction algorithm may be acceptable for a small number of search interfaces; however, in the long run it constitutes a considerable overhead: For each information source, a programmer must identify characteristic HTML patterns that wrap interesting data records, the so-called "extraction patterns", and use them to write a parser. This procedure is tedious and error-prone, and a small change in the design of a result page often renders hand-crafted parsers defective.

For this reason automatic information extraction algorithms have been developed in the last years. They can be divided into wrapper generation algorithms, wrapper verification algorithms, and wrapper re-induction algorithms [9]. Among these, wrapper generation algorithms are the best investigated ones. Their goal is to generate dedicated programs or program parameters like grammars or patterns, which can be used to identify record boundaries within result pages of a particular source.

Supervised wrapper generation algorithms[3] learn extraction patterns from a set of training documents wherein records or attribute boundaries have been labeled manually. The majority of these methods represent the patterns as a finite state machine, e. g. as a grammar, a regular expression, or in the form of a hidden Markov model [11, 1, 7, 3, 17, 5]. The underlying pattern identification algorithms include inductive and active learning strategies [10, 4].

Unsupervised techniques overcome the need to manually label training documents. The approach of Gao et al. uses a set of result pages from the same source and identifies a region that has a "tabular" structure [6]. Based on this table, candidate extraction patterns that will match the rows are inferred. Another system, called IEPAD, discovers repetitive patterns within a result page by means of a Patricia tree and proposes some of them as record candidates [2]. Liu et al. present an approach to extract data from HTML tables, which is based on the analysis of the parse tree of a Web site.

The task of wrapper verification algorithms is to check whether a generated wrapper still behaves as intended, or if design changes within its associated information source lead to a malfunctioning. Kushmerick et al. propose an algorithm that learns a probabilistic model of the extracted data during the training period, which captures data type characteristics like the fraction of numeric attributes within a record [8]. A significant change of the expected data type characteristics in a result page is interpreted as a design change, and intervention may become necessary. Given the case that a verification algorithm determines a malfunctioning of a wrapper, so-called wrapper re-induction algorithms come into play. Lerman and Minton propose a semi-automatic algorithm that also learns a probabilistic model of the data during a training period [12]. If the learned model does not fit the extracted data of a result page any longer, their algorithm maps expected attributes onto data fields within the records of the modified result page probabilistically.

[3] This class of algorithms is also known as semi-automatic wrapper generation algorithms.

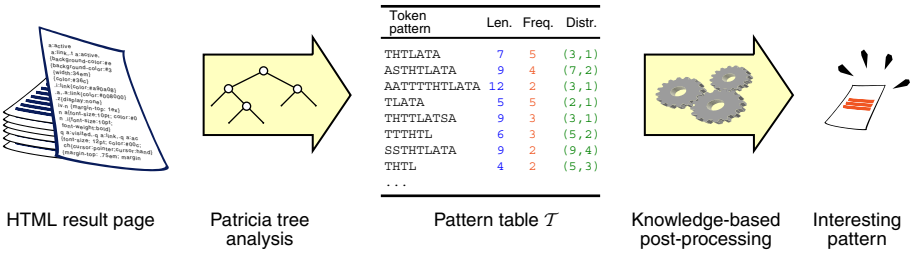

Figure 2. Two-stage approach to automatic wrapper generation: The identification of candidate patterns with a Patricia tree is coupled with a knowledge-based post-processing to find among the candidate patterns the most likely one(s).

In the literature on the subject the term "automatic" refers to the degree of automation in a wrapper generation algorithm for a given source at a given time. As pointed out above, the challenge in a meta-search situation is to construct a parser that is robust against changes of result pages in time. This is what we call adaptive.

2 Adaptive Wrapper Generation for Search Engines

Result pages of search engines contain several regular structures; one of these structures is the list with the snippets that characterize the matching documents for the query and which we would like to identify. A regular structure contains sequences that are tagged in a uniform way. For example at Lycos,[4] a snippet is wrapped in the following code:

```
<li><a>TEXT</a><font color="#808080"></font><br
/>TEXT<span>TEXT</span><a>TEXT</a></li>
```

If one considers the n suffix strings that can be formed from a given HTML page of length n, several among these suffixes start with the same prefix.[5] When inserting the suffixes in a Trie,[6] multiple occurrences of the prefixes can be efficiently detected. Since in an HTML result page several hundreds of such recurring patterns can be found, additional knowledge must be employed to detect those few patterns that actually wrap the interesting snippets. This observation suggests a two-stage approach for pattern identification (cf. Figure 2):

(1) Creation of a table $\mathcal{T}$ with candidate patterns.
(2) Knowledge-based post-processing of $\mathcal{T}$ to identify the interesting pattern(s).

However, a necessary prerequisite is the tokenization of an HTML page, which provides us with a string S of tokens. There is the question of how fine-grained the text

[4] `http://www.lycos.de`
[5] A suffix of a string S is a substring of S that starts at some position $i \leq |S|$ and ends at position $|S|$.
[6] The term "Trie" is derived from the terms "tree", "information", and "retrieval" and designates an index structure for efficient text search [19].

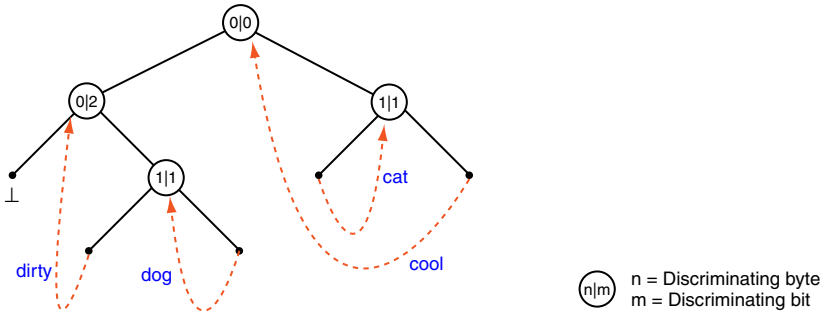

Figure 3. A Patricia tree that contains the words "cool", "cat", "dirty", and "dog". For performance reasons the discriminating position is encoded as a pair of byte and bit position.

elements in the HTML page shall be distinguished. Chang and Lui distinguish merely two tokens, namely "HTML tag" and "plain text" [2]; to leave more flexibility for the knowledge-based post-processing step we currently support about 130 different tokens.

2.1 Table Creation

Let S be a string (of tokens) of length n. As mentioned above, a Trie provides an efficient means to set up a table of candidate patterns. The theoretically optimum algorithm for constructing an index of all suffixes for S is the suffix tree [14]; its runtime is linear in the length of S. A naive algorithm would generate the n suffix strings of S and insert them in a standard Trie, which results in a runtime of $O(n^2)$.

Our approach is oriented at the naive algorithm: However, the n suffixes of S are not generated explicitly but "read off" by moving an index from 1 to n over S. Every suffix is identified by its starting position, and a Patricia tree (explained below) is used to identify all suffixes that start with the same token sequence. Though the theoretical runtime still is $O(n^2)$ this approach will behave even more efficient than a suffix tree implementation, except for a few pathological cases.[7]

A Patricia tree[8] is a particular digital search tree that has two salient properties: (1) Each inner node in the tree is used for differentiation purposes, say, each inner node has two successors. (2) The keys (strings) are not stored into leafs but into inner nodes, which saves half of the nodes. A Patricia tree has the characteristic of digital search trees in that its structure is independent of the insertion sequence. A digital search tree considers keys as bit sequences; an inner node defines the position of the key that shall be investigated for discrimination purposes. Figure 3 gives an example.

From a Patricia tree all repeating sequences in the token string S can be collected in $O(n)$ runtime and put into a table $\mathcal{T}$.

[7] Rationale for this behavior is that the length of the longest common subsequence in a tokenized HTML page can be assessed by a constant.

[8] The term "Patricia" is an acronym for "practical algorithm to retrieve information coded in alphanumeric". The data structure was proposed by Morrison [16].

2.2 Table Post-Processing

Typically the table $\mathcal{T}$ of candidate patterns contains more than hundred entries. For reliable identification of the interesting element, all patterns (token sequences) are characterized by several features. The most important ones are: pattern length, pattern frequency, pattern distribution, average pattern distance.

This information is used within heuristic rules that assign positive and negative evidence values to the patterns—example:

```
IF length(p)*frequency(p)/n > 0.2 THEN raise_evidence(p, 2)
```

Here $p \in \mathcal{T}$ designates a pattern; the rule assesses the portion by which p covers the entire token string S. In our current implementation, which focuses on HTML result pages of search engines, the evidence values can be estimated; nevertheless, it is planned to acquire them by a machine learning approach soon.

3 Quantitative Analysis

Adaptive wrappers are generated on the fly, in extreme cases each time a search result is delivered from a search engine. I. e., performance analyses are not only interesting with respect to extraction quality, but also with respect to wrapper generation time. We did some analyses in this connection, based on a test corpus with about 100 generated result pages for several popular search engines. Our wrapper implementation is done in Java, and the experiments were conducted on a Pentium IV 1.2 GHz system running RedHat Linux.

Amount of sample pages	Tokenization	Patricia tree generation	Pattern extraction	Pattern analysis	Total
28KB	121 ms	23 ms	32ms	19ms	195ms

Table 1. Average runtime of different steps in the course of adaptive wrapper generation.

Table 1 shows the averaged runtime for wrapper generation for one result page. The tokenization of the result pages took over 60% of the total runtime, since we used a generic HTML parser that was not optimized for our tokenization step. The heuristic ranking of relevant patterns within the created table $\mathcal{T}$ was always very good, and thus explains the low pattern analysis time. Due to the fact that a wrapper verification algorithm also has to parse and analyze a result page, we do not expect substantial runtime differences between both approaches. Note that it is conceivable to generate an adaptive wrapper on the client machine of a user, in a stand-alone meta search application.

AltaVista	Lycos	Netscape
$\approx$ 80%	$\approx$ 90%	$\approx$ 90%

Table 2. Portion of correctly identified result lists. Basis were about 100 different result pages for each of the mentioned search engines.

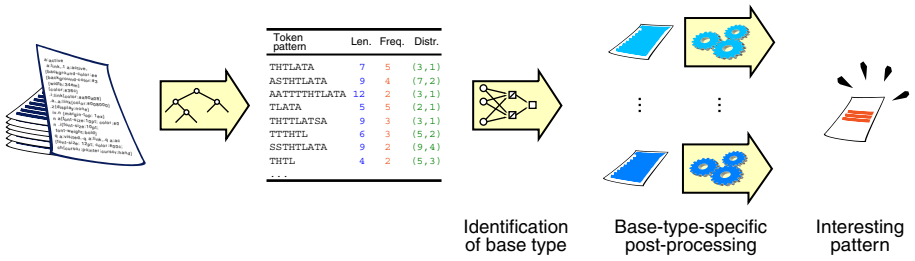

Figure 4. If the process of automatic wrapper generation is organized as a three-stage approach, the knowledge-based post-processing step gains twice: It becomes more effective and easier to be implemented.

Table 2 contains some classification results. The post-processing was able to identify most of the records. However, we employed the knowledge that a record at least contains a URL and a headline.

4 Current Work

The two-stage approach to wrapper generation presented in this paper provides a high degree of flexibility. Nevertheless, the knowledge-based post-processing step becomes more and more intricate with the number of different information sources that shall be handled.

It would be useful in this connection, if a certain “result page base type” is recognized in advance, such as “Shop” or “Link List”, and a dedicated set of rules is chosen and applied in the knowledge-based post-processing step (see Figure 4). In our current work we investigate how the necessary recognition step can be realized by learning a fingerprint from the pattern table.

Moreover, we are developing measures of robustness and flexibility for a generated wrapper in order to prognose both (1) its reliability when parsing HTML pages from information sources the parser was not designed for, and (2) the expected malfunctioning rate depending of extent of modifications of the HTML page.

References

[1] Naveen Ashish and Craig Knoblock. Wrapper Generation for Semi-Structured Internet sources. *SIGMOD Rec.*, 26(4):8–15, 1997. ISSN 0163-5808.

[2] Chia-Hui Chang and Shao-Chen Lui. IEPAD: Information Extraction Based on Pattern Discovery. In *Proceedings of the Tenth International Conference on World Wide Web*, pages 681–688. ACM Press, 2001. ISBN 1-58113-348-0.

[3] B. Chidlovskii, J. Ragetli, and M. de Rijke. Automatic Wrapper Generation for Web Search Engines. In *Proceedings WAIM’00*, LNCS. Springer, 2000.

[4] A. Finn and N. Kushmerick. Active Learning Selection Strategies for Information Extraction. In *ECML-2003 Workshop on Adaptive Text Extraction & Mining*, 2003.

[5] Dayne Freitag and Nicholas Kushmerick. Boosted Wrapper Induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 577–583. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6.

[6] X. Gao, M. Zhang, and P. Andreae. Learning Information Extraction Patterns from Tabular Web Pages without Manual Labeling. Technical report, Victoria University of Wellington, 2003.

[7] C. N. Hsu and C. C. Chang. Finite-State Transducers for Semi-Structured Text Mining. In *Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*. Pergamon Press, 1999.

[8] N. Kushmerick. Wrapper Verification. *World Wide Web Journal*, 3(2):79–94, 2000.

[9] N. Kushmerick and B. Thomas. Adaptive Information Extraction: Core Technologies for Information Agents. In *Intelligent Information Agents R&D in Europe: An AgentLink perspective*, 2002.

[10] N. Kushmerick, D. Weld, and B. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of IJCAI-97*, 1997.

[11] Nicholas Kushmerick and Daniel S. Weld. *Wrapper Induction for Information Extraction*. PhD thesis, Department of Computer Science & Engineering, University of Washington, 1997.

[12] Kristina Lerman and Steven Minton. Learning the Common Structure of Data. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 609–614. AAAI Press / The MIT Press, 2000. ISBN 0-262-51112-6.

[13] Bing Liu, Robert Grossman, and Yanhong Zhai. Mining Data Records in Web Pages. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–606. ACM Press, 2003. ISBN 1-58113-737-0.

[14] E. McCreight. A Space-Economical Suffix Tree Construction Algorithm. *Journal of the ACM*, 23(2):262–272, 1976.

[15] Sven Meyer zu Eißen and Benno Stein. The AIsearch Meta Search Engine Prototype. In Amit Basu and Soumitra Dutta, editors, *Proceedings of the 12th Workshop on Information Technology and Systems (WITS 02), Barcelona Spain*. Technical University of Barcelona, December 2002.

[16] D. R. Morrison. PATRICIA—Practical Algorithm to Retrieve Information Coded in Alphanumeric. *Journal of the ACM*, 15(4):514–534, October 1968.

[17] Stephen Soderland. Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1-3):233–272, 1999. ISSN 0885-6125.

[18] Benno Stein and Sven Meyer zu Eißen. AIsearch Homepage. `http://www.aisearch.de`, 2003-2004.

[19] G. Stephen. *String Searching Algorithms*. World Scientific, 1994.

Information Need Assessment in Information Retrieval

Beyond Lists and Queries

Frank Wissbrock

Department of Computer Science
Paderborn University, Germany
frankw@upb.de

Abstract. The goal of every information retrieval (IR) system is to deliver relevant documents to an user's information need (IN). Therefore an accurate IN assessment is essential to the quality of the system's search results. However, many IR systems ask the users to assess their information needs and communicate them to the system, usually in form of queries. The systems assume the queries to be a perfect assessment of the information needs and deliver relevant information, ending the interaction. However, experiences showed that in many cases the information need cannot be specified in a single query.
This paper addresses the problems of simple IN assessment and proposes a multi-interface IR system to overcome the problems. Such a system supports the user with several search interfaces for different search contexts. Exemplarily the document retrieval engine AiSearch from the Knowledge-based Systems Group at Paderborn University is reviewed to demonstrate some interfaces. This includes a cluster-based interface, a concept taxonomy interface, and a chronological document relations interface.

1 Introduction

Information need (IN) is one of the most important concepts in information retrieval (IR) theory. It is the main input parameter for most IR operations as well as the main evaluation criteria for the quality of the delivered information. But even though the concept of information need is central to the success of any IR system, most IR models treat the concept as intuitively clear and informal. From this viewpoint the importance of information need assessment is often underestimated. Indeed in most IR systems information need assessment is user business. Take for example common internet search engines. They require the users to formulate their information needs in form of a query, assuming that the query is an accurate definition of the information need. However, it was shown that this assumption does not hold for many IR transactions [1] [2].

Starting from the viewpoint that common search engine interfaces do not support an accurate information need assessment this paper proposes an IR sytem with multiple user interfaces, where each of the interfaces fits a certain

search context of the user. Based on a theoretical and historical discussion of IN assessment in section 2-4 the multi-interface model is presented in section 4. Section 5 describes AiSearch, a search engine project of the Knowledge-based Systems Group at Paderborn University, to demonstrate how parts of the model were implemented and how they look like. [3].

2 Historical Developments in Information Need Assessment

Before a formal definition of information need and informantion need assessment is given some approaches to information need assessment are briefly reviewed in their historical context. The intention is to build a foundation for the definitions given in the next section.

2.1 Query approach

The query approach was the first IN assessment method and is still widely used. It was developed in the late 1950s and early 1960s in the context of text properties research and the formulation of the standard IR model [4] [5]. The basic idea of the approach is to let the user assess his information need. Therefore the user enters a query, which usually consists of one or more natural language terms. In turn the system presents all documents from its database that match the query. In 1965 Roccio added an additional step to the query approach: the relevance feedback [6]. With relevance feedback the user judges the result in light of its relevance to his or her information need. Therefore he classifies the returned documents into two classes, the relevant documents and the non-relevant documents. After that the system uses the classification to adjust the initial query and the retrieval process starts again with the adjusted query. The new result is, if necessary, classified again by the user. The assessment is repeated until the query is a perfect representation of the user's information need.

2.2 Dialog approach

The query approach bases on the assumption that the user knows what his information need is and that he can adequately communicate it to the system. Relevance feedback takes care of an accurate IN assessment. However, relevance feedback implicitly assumes that the information need itself stays constant over time, even when the user has gained new knowledge during the search process. Recognizing that this assumptions did not hold always, Oddy proposed a dialog interface in 1977 [1]. The basic idea is that a user's understanding of his information need underlies a continuing evolution while new information is retrieved. The dialog interface allows the user to reformulate his previous query to broaden or narrow the retrieved information or to shift the search goal. The interaction is continued until the needed information is found. The difference to the query

approach is that Oddy embedds the user into the IR system. The user is no longer only an input giver but a part of the retrieval process.

Some years later Belkin shifted the focus even farther to the user and his information need [2]. He asked why most users are not able to specify their information needs in an appropriate way. The answer was given by a new element in the user model: the "anomalous state of knowledge" (ASK) of the user [2]. Therefore every user who faces a problem or situation has a feeling about a gap in his knowledge, the anomaly. In how far the anomaly is understood by the user depends on his cognition of the particular situation. Belkin introduced two levels of specificability: the cognitive level and the linguistic level. The cognitive level refers to what degree the user is able to specify (understand) his current situation. The linguistic level refers to the degree the user is able to specify his information need in linguistic terms. Belkin states that if a user is not able to understand his current situation at the cognitive level well enough, then he will hardly be able to express his information need at the linguistic level. He suggests a system design that is built around the user and his ASKs. He refers to Oddy's dialog approach as a good example for such a system design [7] [8].

2.3 Berrypicking approach

In 1989 Bates discovered that the relevant documents are not only the documents which are retrieved at the end of the search, but also some of the documents encountered during the search [9]. He proposed a new approach, which accounts for the changing information need during the search. In every step of the search the user may reformulate his information request based on the knowledge gathered in previous steps. The user is also allowed to keep some of the retrieved documents as relevant. His approach is an evolving search like Oddy's, but differs in that the relevant documents are collected step by step like berries are picked in the forest. Therefore the approach is named berrypicking. In addition he observed that users tend to change their search strategy depending on their rational information need.

2.4 Clustering approach

The above approaches assume some kind of interaction between system and user. In contrast clustering infers from the structure of the document collection on the information needs that could be satisfied with the document collection. Document clustering was subject to research since the 1960s [10] [11] [12]. In 1979 van Rijsbergen formally connected clustering and information need by formulating the cluster hypothesis, which states that closely associated documents are relevant to the same information request [11]. Therefore clustering algorithms highlight patterns in a document collection and allow the users to browse for the needed information. The explosion of digital stored information during the 1990s made this approach very attractive. However, many design questions are still open, most namely the evaluation of document cluster quality [13] [14].

3 Essentials of Information Need Assessment

Based on the historic review in the previous section the following definitions intend to clarify the concept of information need.

Definition 1 (Information Need). *Information need refers to the amount of all absence information, which is necessary for a user to reach his or her goals in a particular situation. The following assumptions hold:*

1. *The user may not know what exactly his information need is.*
2. *The user may not be able to formulate his information need.*
3. *The information need of a particular user may shift during a search session.*

Definition 2 (Rational Information Need and Radical Information Need). *Let $I(U, S)$ be the information need of user U in situation S. The part of the information need the user is aware of is referred to as rational information need I_{Rt}. The part of the information need the user is not aware of is referred to as radical information need I_{Rd}. Rational and Radical information need are disjunct:*

1. $I_{Rt}(U, S) \cup I_{Rd}(U, S) = I(U, S)$.
2. $I_{Rt}(U, S) \cap I_{Rd}(U, S) = \emptyset$.

Definition 3 (Information Need Assessment). *Information need assessment refers to the process of increasing the degree of rational information need of a user during a search session.*

4 IR Assessment Model

The IN Assessment approaches are not competing with each other for which one is the best. Instead each approach fits a certain search context better than the others. IR system interfaces should account for this and dynamically adapt to the user's search context. The model in Figure 1 shows the IR Multi-Interface Model, which incorporates different IN assessment approaches.

The model consists of three layers built around the user. The inner layer represents the interfaces. Every interface gives the user another view on the data. The middle layer represents the engines, which are necessary to realize the interfaces. The outer layer represents the coordination system. The coordination system decides what interface is presented to the user in a particular situation.

For the coordination system to work the classification framework in figure 2 is applied. The framework classifies IN assessment methods along two dimensions: the assessment time and the assessment style.

The assessment time refers to the timeframe in which information is gathered about the user. In the case that the system encounters an unknown user, who demands just in time information, the assessment time is short-term. This situation is common for mass-user internet search engines. In the case that the system continuously collects data about the information need of its users, the

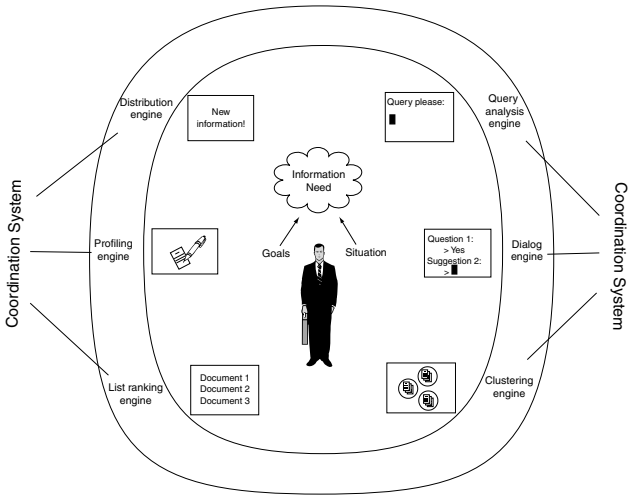

Fig. 1. Multi-Interface Model: The IR system is build around the user. It offers different interfaces for searching in the system's database.

assessment time is long-term. The advantage of long-term IN assessment is that the system can distribute new relevant information to its users when it enters the system. However, for this setting the users should have, at least to some degree, a constant information need over the time.

The assessment style refers to the degree of human/computer involvement in the IN assessment process. If the user formulates his information need by himself, then the assessment style is supervised. This style is very useful when the user knows what source he is looking for. If the system assesses the information need of the user, then the assessment style is unsupervised. This situation is very common when a user acquaint himself with some new topic and does not know the important keywords. But also in the case that an overwhelming amount of relevant information exists unsupervised methods are useful to discover some structure in the information. If both, the user and the system, are involved in the IN assessment, then the assessment style is semi-supervised.

The assessment style is closely tied to the degree of rational IN/radical IN. The higher the degree of rational information need in relation to radical information need the more likely a supervised method will support the user and vice versa. Therefore a search usually starts with an unsupervised or semi-supervised IN assessment method and moves during the search session torwards a supervised method.

5 AiSearch

AiSearch is a Web document retrieval engine developed by the Knowledge-based Systems Group at Paderborn University [3]. The engine is used for research in information retrieval. For the purpose of information need assessment the engine

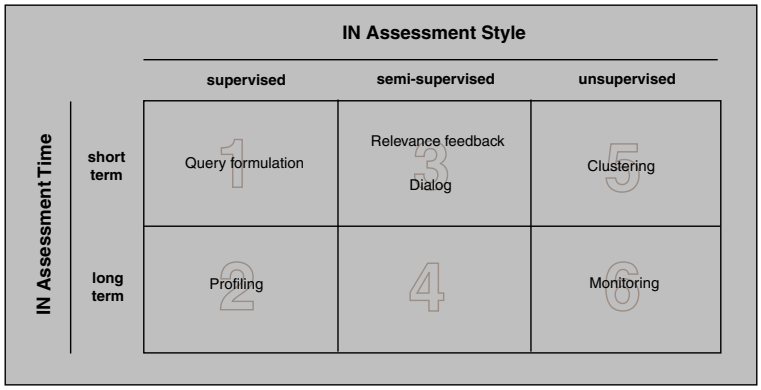

Fig. 2. IN Assessment classification: The IN assessment approaches are classified along the two dimensions assessment style and assessment time. The transparent numbers indicate the degree of IR system involvement in the IN assessment. They range from one (low IR system involvement) to six (high IR system involvement).

incorporates different user interfaces. Up to now two clustering based interfaces are implemented and a third, which highlights chronological relations between documents, is subject to research.

5.1 Implemented Interfaces

Figure 3 shows a clustering based IR interface. In this view the retrieved documents are clustered into conceptionally similar groups. The groups are represented by rectangulars and their content is described by terms on the corresponding rectangular. The content of the selected cluster, which is always centered, is shown in the window on the right side of the screen. The conceptional distance between two clusters are indicated by the distribution of the rectangulars on the screen. Therefore the closer a cluster is located to the center the more closely it is related to the selected cluster. In addition a numbered line between two clusters indicate their closeness in quantitative terms.

In contrast the screenshot in Figure 4 shows a taxanomic view of document clusters. In this view only a small number of all clusters are displayed. The clusters are represented by a term, which describes the content. When a user clicks on one of the terms the corresponding cluster is extended and the view displays its subtopics. The view is very useful when the information need is highly unspecific and the IR system returns a large number of different clusters. In this case a presentation of all clusters at the same time would confuse the user.

5.2 Future work

An interface that highlights chronological relations between documents is subject to current reseach. The basic idea is that knowledge about the development of a

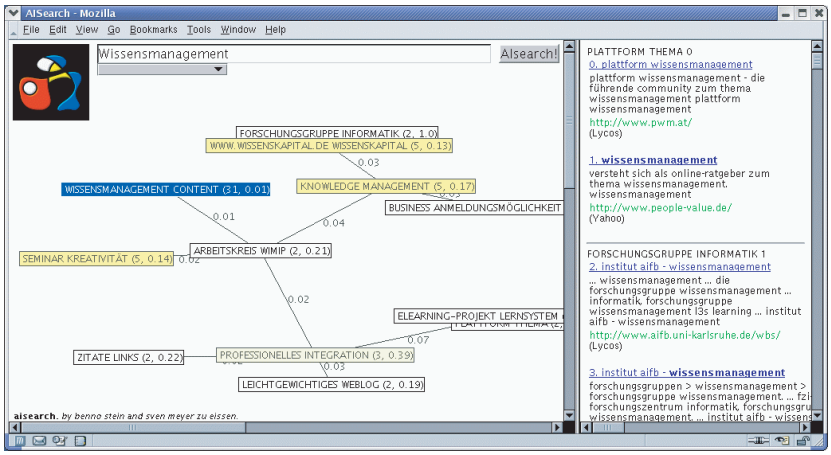

Fig. 3. Cluster-based view: The documents are clustered in conceptionally similar groups. The rectangulars represent the clusters, the terms on every rectangular describe the content, and the line between two connected rectangulars indicate their closeness. On the right side of the sceen the content of the selected cluster is displayed in ranked order.

certain topic over time is useful in some situations. Figure 5 shows schematically two views on chronological structured documents. The view on the left side shows a visualization for clusterd results. The vertical axis represents the clusters and the horizontal axis the timeline. Circles in the coordination system represent documents. The bigger a circle the more documents of the corresponding topic refer to events at that time. The view on the rights side shows the "chronological environment" of the current document.

The realization of the engine for the chronological analysis demanded the construction of a knowledge base. At the core of the knowledge base is a set of manual tagged text documents. The tag structure is used to extract time/event entities. A time/event entity is for example the sentence "He plans to change to another club in 2005.". It is called time/event entity because the sentence describes an event that takes place at a certain time. Every single time/event entity is used as an example in the knowledge base database. Figure 6 shows a screenshot of the engines rule manager and a set of examples. The structure of every example is finegrained with additional tags like ¡Year¿ and ¡/Year¿ or ¡Number¿ and ¡/Number¿. Based on the examples and a set of principles the system automatically indentifies time/event entities in texts.

At the moment the engine is still a prototype and its result quality subject to current research. A more detailed description of the system and its performance in practical settings will occur in follow-up publications during this and next year. In addition the content of the texts is restricted to sports topics. However an extension to political and business topics is planned.

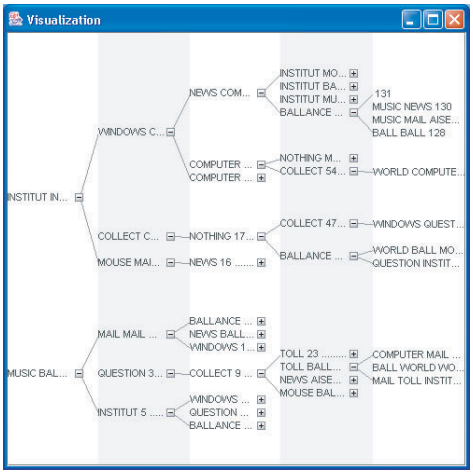

Fig. 4. Taxonomic view: A small number of all cluster is displayed at the beginning. Every cluster is represented by a term, which describes its content. The user can extend the clusters to display subtopics.

6 Summary and Outlook

The purpose of this paper was to shift the eye of the reader to the importance of information need assessment. Therefore the text started by criticising the shortcomings of current IN assessment practices, namely the query input/list output IR systems. A historical survey showed that a user is embedded in different search contexts, which determine how much the user knows about his current information need. The IR Multi-Interface Model was presented to address the existence of several search contexts and it was stated that an IR system should offer different user interfaces and views on the data. Finally the search engine AiSearch was surveyed to demonstrate the functioning of different interfaces in practice.

For the furture the Knowledge-based Systems Group at Paderborn University plans to introduce more interfaces for AISearch. In the short run the view on chronological structured documents will be added to the system and performance statistics will be published in follow-up papers.

References

1. Oddy, R.: Information retrieval through man-machine dialogue. Journal of Documentation **33** (1977) 1–14
2. Belkin, N.: Anomalous states of knowledge as a basis for information retrieval. Canadian Journal of Information Science **5** (1980) 133–143
3. Stein, B., zu Eißen, S.M.: Aisearch: Category formation of web search results. Technical report, Paderborn University (2003)
4. Luhn, H.: The automatic creation of literature abstracts. IBM Journal of Research and Development **2** (1958) 159–165

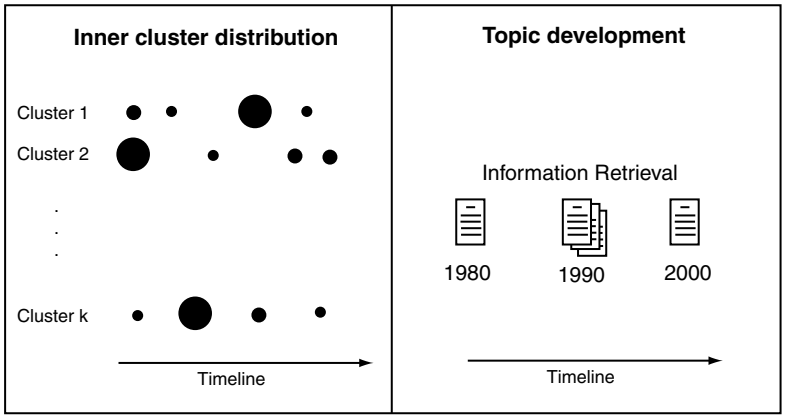

Fig. 5. Views on chronological structured documents.

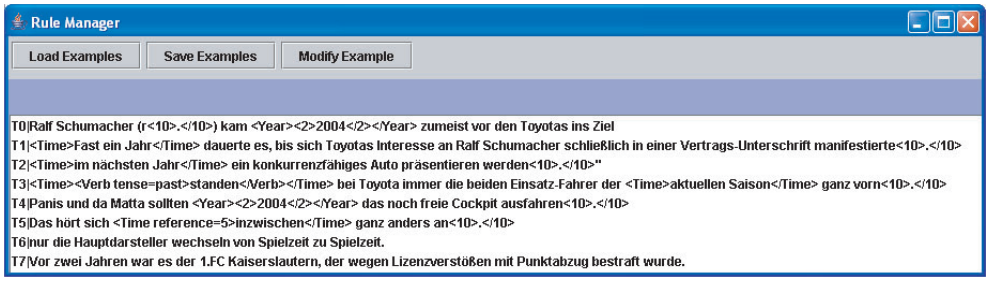

Fig. 6. Screenshot of the rule manager of the chronological analysis engine.

5. Salton, G., Lesk, M.: The smart automatic document retrieval system - an illustration. Communications of the ACM **8** (1965) 391–398
6. Rocchio, J., Salton, G.: Information optimization and interactive retrieval techniques. In: Proceedings of the AFIPS-Fall Joint Computer Conference, Part I. Volume 27. (1965) 293–305
7. Belkin, N., Oddy, R., Brooks, H.: Ask for information retrieval: Part i. Journal of Documentation **38** (1982) 61–71
8. Belkin, N., Oddy, R., Brooks, H.: Ask for information retrieval: Part ii. Journal of Documentation **38** (1982) 145–164
9. Bates, M.: The design of browsing and berrypicking techniques for the online search interface. Online Review **13** (1989) 407–424
10. Doyle, L.: Semantic road maps for literature searchers. Journal of the ACM **8** (1961) 553–578
11. Rijsbergen, C.: Information Retrieval. Buttersworth, London (1979)
12. Salton, G.: Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer. Addison-Wesley (1988)
13. Stein, B., Meyer zu Eißen, S., Wißbrock, F.: On Cluster Validity and the Information Need of Users. In Hanza, M., ed.: Proceedings of the 3rd IASTED

International Conference on Artificial Intelligence and Applications (AIA 03), Benalmdena, Spain, Anaheim, Calgary, Zurich, ACTA Press (2003) 216–221
14. Stein, B., Meyer zu Eißen, S.: Automatic Document Categorization: Interpreting the Perfomance of Clustering Algorithms. In Gnter, A., Kruse, R., Neumann, B., eds.: KI 2003: Advances in Artificial Intelligence. Volume 2821 LNAI of Lecture Notes in Artificial Intelligence., Springer (2003) 254–266