Topic-based Vector Space Model

Jörg Becker
Dept. of Information Systems
Leonardo-Campus 3
48149 Münster; Germany
becker@wi.uni-muenster.de

Dominik Kuropka
Dept. of Information Systems
Leonardo-Campus 3
48149 Münster; Germany
isdoku@wi.uni-muenster.de

Abstract

This paper motivates and presents the Topic-based Vector Space Model (TVSM), a new vector-based approach for document comparison. The approach does not assume independence between terms and it is flexible regarding the specification of term-similarities. Stop-word-list, stemming and thesaurus can be fully integrated into the model. This paper shows further how the TVSM can be fully implemented within the context of relational databases. This facilitates the use of this approach by generic applications. At the end short comparisons with other vector-based approaches namely the Vector Space Model (VSM) and the Generalized Vector Space Model (GVSM) are presented.

1. Motivation

Information filtering (IF) systems are designed for permanently scanning document streams (e. g. news-ticker or Usenet). They identify potentially important or interesting documents for users by classifying them. For designers of IF systems the conceptualization of user profiles is a great challenge. For each user a profile has to be defined. This profile determines criteria which are utilized for user-specific classification of documents.

One approach (the *explication approach*) is the definition of a formalized language which is utilized by the user to describe his profile. This approach has been implemented in the IF prototypes Rama [Binkley 1991], Borges V2 [Smeaton 1996] and Sift [Yan 2000]. The main problem with this approach is that users are often not capable of specifying their information demand properly. There are two main reasons for this: Firstly, it is difficult for a user to explicate required criteria. Secondly, the formalized language has to be powerful enough to deal with the challenges of natural language processing like flexions of words[1], synonyms[2] and polysems[3]. Additionally, it should be powerful enough to allow complex expressions (e. g. using Boolean operators like "or", "and", "not", etc.). On the one hand this leads to a huge amount of time the user needs to master the language in case it is very powerful. On the other hand the filtering results of the IF system will be deficient if the language is easy to use but not powerful enough.

One solution of this dilemma is the use of an *adaptive approach*. The idea is to present some evaluated documents to the IF system and to let it generate the user profile on its own. As a side-effect the system can improve the user profile continuously if the user himself gives a feedback on misclassified documents. This approach has already been implemented in NewsSIEVE [Haneke 2001] and PI-Agent [Kuropka 2001] systems. NewsSIEVE adapts the user profile by using evolutionary algorithms while the PI-Agent uses neuronal networks. Both approaches have in common that the initial information about user profiles (= training set) are transformed into an internal representation (e. g. neuron weights in case of a neuronal network) which makes the profile representation difficult to understand for users. So the system is not able to explicate its classification rules in a user-friendly way. This leads to the following problems: Firstly, the user has to rely on the classification given by the IF system without knowing how the classification is done in detail. Secondly, in case the user's information demand shifts from one day to another or the system is unable to adapt his information demand, it is impossible for him to make reasonable corrections on his profile. Consequently, the user has to wait until the system has corrected his profile automatically. Meanwhile, a lot of documents may be misclassified.

Our intention is the use of a *case-based approach* for defining user profiles. This means, the user defines his profile by presenting some evaluated documents to the system, like in the adaptive approach. In contrast to the

[1] Words may appear in different forms like singular or plural depending on their use within a text (e. g. "house" and "houses").

[2] Synonyms are different words (e. g. "automobile" and "car") with the same meaning.

[3] Depending of the context a word could have different meanings (e. g. "mouse" can stand for a small rodent or for a computer input device).

adaptive approach this initial profile information is *not* transformed into some kind of internal representation. New documents are classified by adopting the classification of the most similar documents from the user profile. A simple adoption of user profiles can be implemented this way: Every time the system has misclassified a document, user corrected evaluations of documents are added to the profile. The benefit of this approach is the possibility to understand the profile, because it is just an set of evaluated documents. By removing evaluated documents or by inserting new evaluated documents reasonable corrections on the profile are possible. The following aspects should be considered when implementing case-based approaches:

a) *Document similarity*: A similarity function is necessary to determine the similarities between old and new cases. The "cases" here are documents. Therefore, some kind of document similarity function is needed. This function will be derivated from a theoretical foundation which is presented in section 2 in detail.
b) *Document classification*: For document classification we propose the use of *k-Nearest Neighbour* classification (kNN). This is a simple method. If you insert a test document into the system, the system finds the *k* nearest neighbours among the profile documents. It uses the categories of the *k* neighbours to weight the category of the test document. Referring to the examination of YANG and LIU, the kNN (using the cosine similarity on document vectors) is one of the best methods for text categorization [Yang 1999]. This paper describes our work which is still in progress. At present, we do not have enough qualified data to run optimizations on parameters of the kNN. Thus, the kNN is future work and will not be discussed in this paper in more detail.
c) *Efficient processing of huge amounts of persistent data*: The IF system should work reliable for several news sources as well as for a large amount of users. Further requirements are: Every user should have an own profile which supports several classification schemes (e. g. classification by importance or by topic). Reliability determines persistency of data. The other requirements entail the possibility of storing and processing a huge amount of data (documents, user profiles, natural language depending data and administrative data). Databases are optimized for processing huge amounts of data and they can be used effectively if the main part of data is processed within the database. Section 3 will show how the document similarity function can be fully implemented within the context of a relational database using the *Structured Query Language* (SQL) [ISO 1992].
d) *Optimized adoption of user profiles*: The simple adoption method described above adds all documents (and their affiliation to a class) to the user profile, which have been misclassified by the system. On the long run this leads to two problems: Firstly, the user profile extends without any limit. This reduces the classification performance. Secondly, the system is not able to "forget" anything adopted. This makes an adoption of changes of user's information demand impossible. Therefore, it is necessary to develop some kind of reorganisation and garbage collection methods. Qualified data is advantageous for the development of an optimized adoption method. This point belongs to future work and will not be discussed in this paper in more detail.

2. Development of the model

This section introduces the *Topic-based Vector Space Model* (TVSM). Relations to other vector-based models, in particular the Vector Space Model (VSM) [Salton 1968; Baezea-Yates 1999, pp. 27-30] and the Generalized Vector Space Model (GVSM) [Wong 1987; Baezea-Yates 1999, pp. 41-44], are discussed in section 4.

2.1 Topic-based Vector Space

The basic premise of the TVSM is the existence of a *d* dimensional space *R* which only has positive axis-intercepts:

$$R \in \mathbf{R}^{d}_{\geq 0} \text{ with } d \in \mathbf{N}_{>0} \qquad (1)$$

Interpretation of *R*: Each dimension of *R* represents a *fundamental topic*. It is assumed that all fundamental topics are independent from each other (= orthogonal). The precise number of fundamental topics *d* has no further relevance for the TVSM. Therefore, there is no need to specify this number in more detail.

A Term $i \in \{1,\ldots,n\}$ is represented by a *term-vector* $\vec{t}_i$ and has a *term-weight* between one and zero. The term-weight is defined as the algebraic length of the term-vector $|\vec{t}_i|$:

$$\vec{t}_i = (t_{i1}, t_{i2}, \ldots, t_{id}) \in R \qquad (2)$$
$$\text{with } |\vec{t}_i| = \sqrt{t_{i1}^2 + t_{i2}^2 + \ldots + t_{id}^2} \in [0;1]$$

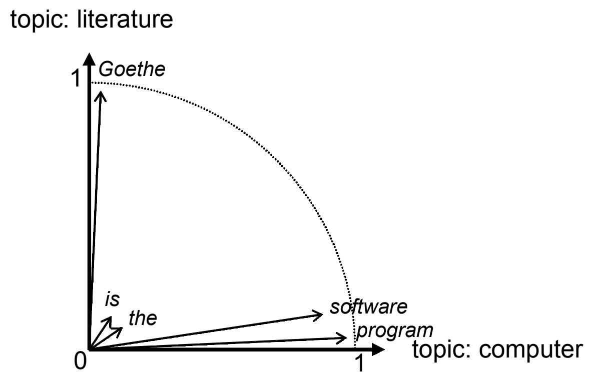

Figure 1. Interpretation of term-vectors

Figure 1 illustrates the interpretation of term-vectors: Topic specific terms (e. g. "program", "software" and "Goethe") point into the same direction as their fundamental topic which they are related to. In case a term combines two or more fundamental topics (due to visualisation limits this case is not shown in figure 1) it points into a combined direction of the two or more fundamental topics. Terms which are not topic specific at all (e. g. "is" or "the") have a term-vector which has an angle near 45° towards all fundamental topics of the space *R*. The term-weight represents the relevance of a term concerning its applicability to derive information about its relation to a topic in general. Term-weights are near to value one for topic specific terms and near to value zero for unspecific terms.

We premise that the following data is known for all terms $i, j \in \{1, \ldots, n\}$:

- Term-weight for each term *i*: $|\vec{t}_i| \in [0;1]$
- Term-angle between each term *i* and *j*: $\omega_{ij} \in [0°;90°]$

Using the data the scalar product $\vec{t}_i \vec{t}_j$ can be deduced as:

$$\vec{t}_i \vec{t}_j = |\vec{t}_i| \cdot |\vec{t}_j| \cdot \cos(\omega_{ij}) \qquad (3)$$

2.2 Document definition

In the TVSM a document $k \in \{1, \ldots, m\}$ is represented by a document-vector $\vec{d}_k \in R$ which vector-length is standardized to the length of value one for better comparison:

$$\vec{d}_k = \frac{1}{|\vec{\delta}_k|} \vec{\delta}_k \quad \Rightarrow \quad |\vec{d}_k| = 1 \qquad (4)$$

with $\vec{\delta}_k = \sum_{i=1}^{n} e_{ki} \vec{t}_i$

while e_{ki} = number of occurrences of term *i* in document *k*.

2.3 Similarity of documents

The similarity between two documents *k* and *l* is defined as the scalar product of the document-vectors, which is equal to the cosine of the angles between the document-vectors:

$$\vec{d}_k \vec{d}_l = |\vec{d}_k| \cdot |\vec{d}_l| \cdot \cos(\omega_{kl}) = \cos(\omega_{kl}) \qquad (5)$$

$$\text{because } |\vec{d}_k| = |\vec{d}_l| = 1$$

This similarity measure has the following properties:

$$\vec{d}_k \vec{d}_l \in [0;1] \quad \text{and} \quad \vec{d}_k \vec{d}_l = \vec{d}_l \vec{d}_k \quad \text{and} \quad \vec{d}_k \vec{d}_k = 1 \qquad (6)$$

With the knowledge of term-weights and term-angles, the similarity between two documents can be computed as follows:

$$\vec{d}_k \vec{d}_l = \frac{1}{|\vec{\delta}_k|} \vec{\delta}_k \frac{1}{|\vec{\delta}_l|} \vec{\delta}_l \qquad (7)$$

$$= \frac{1}{|\vec{\delta}_k| \cdot |\vec{\delta}_l|} \sum_{i=1}^{n} \sum_{j=1}^{n} e_{ki} e_{lj} \vec{t}_i \vec{t}_j$$

The length of the unnormed document-vectors $\vec{\delta}_k$ and $\vec{\delta}_l$ can be computed as follows (here only shown for $\vec{\delta}_k$):

$$|\vec{\delta}_k| = \left| \sum_{i=1}^{n} e_{ki} \vec{t}_i \right| \qquad (8)$$

$$= \sqrt{\left(\sum_{i=1}^{n} e_{ki} \vec{t}_i\right)^2}$$

$$= \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} e_{ki} e_{kj} \vec{t}_i \vec{t}_j}$$

Calculation complexity for term similarity is low enough to enable an implementation in SQL. This facilitates a database-optimized computation of document similarities within a relational database as shown in section 3.

2.4 Ideal term-weights and -angles

Term-weights $|\vec{t}_i|$ should have the following characteristics in order to ensure good behaviour of the similarity function:

- Terms which can be used as credible topic indicators for documents (e. g. technical terms) should have a term-weight near value one.
- Terms which do not have a relation to a topic (e. g. the so-called *stopwords* "is", "a", "the", etc.) should have a term-weight near or equal value zero. Weighting stopwords by value zero allows the integration of a classic stopword-list into the TVSM.

The angle ω_{ij} between two terms should follow these rules:

- $\omega_{ii} = 0$
- $\omega_{ij} = \omega_{ji}$

- The angle between two terms which have the same stem (e. g. “house” and “houses”) should have zero degrees. This integrates the idea of stemming into the TVSM.
- The angle between two synonyms or between two words from a similar topic should have a value near zero degrees. This integrates the concept of a thesaurus into the TVSM.
- The angle between two terms from different topics should have a value near 90 degrees.
- The angle between terms which are credible topic indicators and terms without relation to a topic (stopwords) should have a value near 45 degrees.
- The angle between two terms without relation to a topic (stopwords) should have a value near zero.

2.5 Estimation of term-weights and -angles

A manual setting of all term-weights and term-angles is not efficient because of the high amount of terms and the even higher amount of angles. We used the following estimation for our first experiment:

Assumption: All terms which occur in more than 50% of the documents are not useful for any kind of classification. Thus, term-weights for these terms are set to value zero which implies that the scalar product of these terms in relation to other terms is also zero.

Assumption: Terms which occur in less than 1% of the documents do not provide enough data to calculate a reliable estimation on term relations. Therefore, these terms are treated as orthogonal to all other terms. Consequently, $\omega_{ij} = 90°$ for all $i \neq j$. The term-weights for these terms are set to value one.

The term-weights for all other remaining terms are set to value one while the angles are set to the following values:

$$\omega_{ij} = \begin{cases} 90° - 90° \cdot \mathrm{Corr}(T_i', T_j') & \text{if } \mathrm{Corr}(T_i', T_j') \geq 0 \\ 90° & \text{else} \end{cases}$$

$\mathrm{Corr}(T_i', T_j')$ is the empirical correlation of terms *i* and *j* within a document base.

The estimation of term-weights and -angles can be improved by using explicit information about the natural language of the documents and by using explicit information about semantic coherence of words. The following improvements should be investigated in the future regarding their influence on classification quality:

- A stopword-list should be used to assign a null value as weight to all stopwords.
- A stemming-list or a stemming-algorithm should be used to assign a null-value to all angles between two words with the same stem.
- Term-angles reflecting semantic coherence of words could be derived from a thesaurus, a semantic web or from a formalized ontology.

3. Database model

3.1 Schema definition

The TVSM has been tested by implementing it in a relational database using the following statements (we assume that an index on all unique or primary key attributes is automatically created by the database):

```
CREATE TABLE document (
       id      INTEGER UNIQUE NOT NULL,
       length  DOUBLE PRECISION DEFAULT NULL;
       text    TEXT NOT NULL,
       PRIMARY KEY(id));

CREATE TABLE term (
       id      INTEGER UNIQUE NOT NULL,
       text    TEXT UNIQUE NOT NULL,
       weight  DOUBLE PRECISION DEFAULT NULL,
       PRIMARY KEY(id));

CREATE TABLE doc_term_ass (
       document INTEGER NOT NULL
                  REFERENCES document(id),
       term     INTEGER NOT NULL
                  REFERENCES term(id),
       quantity INTEGER NOT NULL,
       PRIMARY KEY(document, term));
CREATE INDEX doc_term_ass_term_document_idx
       ON doc_term_ass (term, document);

CREATE TABLE skalarproduct (
       term1    INTEGER NOT NULL
                  REFERENCES term(id),
       term2    INTEGER NOT NULL
                  REFERENCES term(id),
       value    DOUBLE PRECISION NOT NULL,
       PRIMARY KEY(term1, term2));
```

Table “document” stores text as well as unnormed document-vector length $|\vec{\delta}_k|$ of the documents. Table “term” stores all terms and their term-weights $|\vec{t}_i|$. Documents and terms are connected by “doc_term_ass” which also stores the number of occurrences of a term within a document. Finally, “skalarproduct” stores the scalar-product $\vec{t}_i \vec{t}_j$ between two terms. Each time a document is inserted, the “doc_term_ass” has to be updated. After this the unnormed document-vector length has to be calculated once and stored for performance reasons in table “document” using the following view as data source:

```
CREATE VIEW doc_length AS
    SELECT dta1.document,
           sqrt(sum(dta1.quantity *
                   dta2.quantity * value))
           AS length
    FROM   doc_term_ass dta1,
           doc_term_ass dta2, skalarproduct s
    WHERE  dta1.document = dta2.document
       AND dta1.term = s.term1
       AND dta2.term = s.term2
    GROUP BY dta1.document;
```

Two[4] views should be created in order to calculate the similarity between documents:

```
CREATE VIEW unnorm_doc_sim AS
    SELECT dta1.document AS document1,
           dta2.document AS document2,
              sum(value * dta1.quantity *
                  dta2.quantity)
           AS unnorm_sim
    FROM   doc_term_ass dta1,
           doc_term_ass dta2, skalarproduct s
    WHERE  s.term1 = dta1.term
       AND s.term2 = dta2.term
    GROUP BY dta1.document, dta2.document;

CREATE VIEW doc_sim AS
    SELECT uds.document1, uds.document2,
           uds.unnorm_sim / n1.length /
             n2.length AS sim
    FROM   unnorm_doc_sim uds,
           document n1, document n2
    WHERE  uds.document1 = n1.id
       AND uds.document2 = n2.id;
```

The first view calculates the unnormed similarity between two documents:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ki}e_{lj}\vec{t}_i\vec{t}_j$$

The second view calculates the norming factor using the previously stored unnormed document-vector lengths. This results in a virtual table which "holds" the similarity for all combinations of documents:

$$\vec{d}_k\vec{d}_l = \frac{1}{\left|\vec{\delta}_k\right| \cdot \left|\vec{\delta}_l\right|}\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ki}e_{lj}\vec{t}_i\vec{t}_j$$

The following statement has to be executed in order to gain the similarity between document 5 and document 7:

```
SELECT * FROM doc_sim
WHERE document1=5 AND document2=7;
```

The following statement has to be executed in order to compare document 5 with all other documents (including reverse ordering by similarity):

```
SELECT * FROM dok_sim
WHERE nachricht1=5
ORDER BY sim DESC;
```

3.2 Performance

We implemented the TVSM on a PostgreSQL[5] version 7.2 relational database. For a better performance, only entries with a scalar-product larger than the *scalar-threshold* 0.5 are stored in the "scalarproduct" table (this is equivalent to set all scalar-products lesser than the scalar-threshold to value zero). For our tests, we used 7184 news documents from the German Heise-Ticker[6] Website. 96887 terms have been extracted from these documents and have been stored in the "term" table. From this data-basis term-weights and -angles have been derived as already described in section 2.5 (with the restriction of the scalar-threshold). Table "scalarproduct" contained 97509 entries. The calculation of the similarity between a general document (having 164 different terms) and all 7184 documents (including reverse ordering by similarity) needed approximately five seconds on our generic PC (Athlon XP 1600+ processor with 768 MByte Ram and FreeBSD operating system). First performance tests showed that the calculation speed highly depends on the number of entries in table "scalarpoduct" and that it only depends very low on the number of terms or documents. This means the scalar-threshold is a good variable to adjust the calculation speed versus the quality of similarity-calculation.

[4] We are using two views in order to avoid nested select statements.

[5] http://www.postgresql.org

[6] http://www.heise.de/newsticker

4. Comparison with other vector-based approaches

Both, the Vector Space Model (VSM) [Salton 1968; Baezea-Yates 1999, pp. 27-30] and the TVSM assign a document-vector to each document. In contrast to the TVSM the VSM assumes that all terms are independent (orthogonal) to each other. This leads to a relatively high performance. The assumption of orthogonal terms is incorrect regarding natural languages which causes problems with synonyms or strong related terms. In order to reduce these problems messages are usually passed through a stopword-list, stemming- and thesaurus-algorithms before they are forwarded to the VSM. This abrogates the assumption of term independence only in parts, because two terms can simply be treated as equivalent or as not equivalent. Similarity levels between these two extremes are not possible. From the theoretical point of view the TVSM has the advantage of not assuming independence for terms which allows a full integration of stopword-list, stemming and thesaurus into the model. Similarity between terms can be gradually defined from "not equivalent" (term-angle: 90°) to "equivalent" (term-angle: 0°).

The Generalized Vector Space Model (GVSM) [Wong 1987; Beaza-Yates 1999, pp. 41-44] assigns a document-vector to each document without the assumption of orthogonal terms. In contrast to the TVSM the GVSM allows no flexibility regarding the computation of term-angles: in the GVSM term-angles are based on the computation of co-occurrence of terms. Because of this limitation messages have to be pre-processed in a similar way like for the VSM: Messages are passed through a stopword-list and stemming-algorithms before they are forwarded to the GVSM. In contrast to the GVSM the TVSM specifies only ideal properties of term-angles (refer section 2.4). Therefore the TVSM allows more flexibility regarding the calculation of term-angles. Term-angles can be computed using different statistical methods like co-occurrence or correlation (refer section 2.5). Further the TVSM allows the deduction of term-angles from explicit information about the semantic coherence of words (e. g. from semantic networks or ontology).

5. Conclusion

This paper presents a new approach (TVSM) to compare documents regarding their content. This approach has the following advantages: From the theoretical point of view, the TVSM is an open approach which enables the integration of several natural language processing algorithms as stopword-list, stemming and thesaurus into one model. This facilitates the possibility of exploration of dependencies between these algorithms and provides a potential to optimize natural language processing models in general.

From the practical point of view, the TVSM enables complete calculation of document-similarities within a relational-database by using plain SQL. Therefore reliable processing of huge amounts of data is supported by using database-integrated optimization algorithms for accessing and processing the data.

6. Bibliography

[Beaza-Yates 1999] R. Beaza-Yates, B. Ribeiro-Neto: *Modern Information Retrival.* ACM Press, New York, 1999.

[Binkley 1991] J. Binkley, L. Young: *Rama: An Architecture for Internet Information Filtering.* ftp://ftp.cs.pdx.edu/pub/faculty/jrb/rama/rama_kl.ps, download: 2002-06-10.

[Haneke 2001] E. Haneke: *NewsSIEVE Ein selbstadaptiver Filter für textuelle Informationen.* 2001, ftp://ftp3.informatik.uni-bonn.de/pub/paper/tr/IAI-TR-2001-1.pdf.gz, download: 2002-06-11.

[ISO 1992] International Standards Organization: *ISO/IEC 9075.* 1992.

[Kuropka 2001] D. Kuropka, T. Serries: "Personal Information Agent", *Informatik 2001 Conference Proceedings*, books@ocg.at, 2001, pp. 940-946.

[Salton 1968] G. Salton, M. Lesk: "Computer evaluation of indexing and text processing", *Journal of the ACM*, 15(1), pp. 8-36, 1968.

[Smeaton 1996] A. Smeaton: "Lessons learned from developing and delivering the BORGES information filtering tool", *Ariadne*, 1996, http://www.ariadne.ac.uk/issue6/borges, download: 2002-06-14.

[Wong 1987] S. Wong, W. Ziarko, V. Raghavan, P. Wong: "On Modeling of Information Retrival Concepts in Vector Spaces", *ACM Transactions on Database Systems (TODS)*, Volume 12, Issue 2, June 1987, pp. 299-321.

[Yan 2000] T. Yan, H. Garcia-Molina: "The SIFT Information Dissemination System", *ACM TODS*, 2000.

[Yang 1999] Y. Yang, X. Liu: "A re-examination of text categorization methods", *Proceedings of the 22nd Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkley, 1999, pp. 42-49.