Salvatore Romeo

Multi-topic and Multilingual Document Clustering via Tensor Modeling

Ph.D. Thesis

December 5, 2014

Springer
Berlin Heidelberg New York
Hong Kong London
Milan Paris Tokyo

Contents

1

Introduction

The proliferation of documents, on both the Web and in private systems, makes knowledge discovery in document collections arduous. As opposed to traditional mining in numeric relational data, document mining faces several additional issues. Documents are indeed characterized by a high-dimensional feature space, often incurring in the *curse of dimensionality* problem. Other related issues, which directly impact on the choice of document representation model, are the high sparsity and the varying document length. Yet, as pieces of natural language information, documents normally suffer from lexical ambiguity issues, which can further be exacerbated by linguistic peculiarities depending on the particular language used in the writing of the documents in a collection.

Document clustering has been long recognized as a useful analysis and exploratory tool for the task. It groups like-items together, maximizing intra-cluster similarity and inter-cluster distance. Document clustering can provide insight into the make-up of a document collection and is often used as the initial step in data analysis. Document clustering research was initially focused on the development of general purpose strategies for grouping unstructured text data. Recent studies have started developing new methodologies and algorithms that take into account both linguistic and topical characteristics, where the former include the size of the text and the type of language used to express ideas, and the latter focus on the communicative function and targets of the documents. In particular, the *length* of documents, their *languages* and their *topical variety* are usually strongly interrelated.

Real-life collections are often comprised of very short or long documents. While short documents do not contain enough text and they can be very noisy, long documents often span multiple topics and this is an additional challenge to general purpose document clustering algorithms that assume every document is an indivisible unit for the text representation and for the similarity computation, and tend to associate every document with a single topic. A key idea to solving this problem is to consider the document as being made up of smaller topically cohesive text blocks, named segments. When used as a base

step in long document clustering, segmentation has indeed shown significant performance improvements (e.g., [190]).

Documents often can be subject to different interpretations, as they reflect multiple *aspects*, or *views*, that may represent side information or other subjects of interest related to the topics discussed in the documents. Even assuming to deal with this problem by first extracting a representation contextually to each view and then obtaining a unique, consensus representation through "fusion" or aggregation of the view-specific ones, the resulting representation might cause loss of information such as correlation between the various views. Limitations due to this kind of flat representation could be overcome by using a multidimensional representation model that is able to explicitly capture the characteristics and relations between the different facets of the documents in a collection. *Data tensor models*, or simply tensors, lend themselves to be particularly appropriate to provide an effective representation model for multi-view and high-dimensional data. The applicability of tensor models has in fact attracted increasing attention in pattern recognition, information retrieval, and data mining in the last two decades, due to the advanced expressiveness power in representing the data through this multidimensional modeling paradigm. Moreover, techniques for tensor decompositions, which can be viewed as a generalization of multi-linear matrix factorizations, allow us to extract a low-dimensional yet meaningful representation which incorporates, in addition to the content information, the information related to the views' interactions.

Another challenge in document clustering research arises from the *multilingualism* of many document sources, which has growth in the last few years due to the increased popularity of collaborative editing platforms, such as Wikipedia, which involve contributors across the world. Detecting the structure of clusters in a multilingual document collection based on the different languages can in principle aid a variety of applications in cross-lingual information retrieval, including statistical machine translation and corpora alignment.

Summary of contributions

The research underlying this thesis is concerned with the definition of document clustering frameworks in which the essential core of document modeling is based on tensor models and decomposition techniques, which are originally exploited to suitably deal with multi-topic, multi-view and multilingual documents. The key assumption is that many text repositories over the Web and other online information sources often contain documents that discuss multiple thematic subjects (i.e., *multi-topic*), cover different aspects of some of the topics or other external information (i.e., *multi-view*), and can be written in different languages (i.e., *multilingual*). More precisely, the contributions of this thesis address problems that fall into two main research lines:

- *Line 1 – Tensor modeling for multi-topic, multi-view documents.* The multi-faceted nature of documents is considered under two interpretations: presence of multiple topics and views (i) across the documents in a collection, and (ii) within individual documents in a collection.
  Concerning the first interpretation, we have developed a tensor-based framework for document clustering in the context of availability of multiple document organizations of the same document collection. Multiple document organizations are assumed to be available and the key idea is to represent the document collection in a multidimensional way by taking into account content information as well as information about how the documents tend to group across the different organizations. In order to extract this additional information, the documents are considered to be items and the document clusters as transactions. A frequent pattern mining algorithm is also developed in order to extract the frequent document aggregations. Starting from the content information and that provided by the frequent aggregations, a tensor model is defined considering three dimensions: frequent aggregations, terms and documents. Experimental evaluation has been performed on Reuters Corpus Volume 1 considering different scenarios that correspond to varying characteristics of the available set of document classifications. This study, which has been originally presented in [162], is the focus of Chapter 3.
  With regards to the second interpretation, we have developed a document clustering framework in which the documents are assumed to discuss multiple topics under different views that can be inferred from an explicit or implicit segmentation provided over every document. Since document segments represent topically cohesive text passages, they are treated as mini-documents and subject to a clustering stage, thus obtaining groups of document portions that discuss a single topic. These segment clusters are interpreted as views and the documents in the collection are modeled contextually to each of the views. A tensor model is built around the dimensions documents, terms, and segment clusters. In experimental evaluation, three different types of document have been considered: news articles (with paragraph boundary based segmentation), hotel reviews (with rating aspect based segmentation) and user's tweets (with hashtag based segmentation). This study, which has been originally presented in [163], is the focus of Chapter 4.
  Line 2 – Tensor modeling for multilingual documents. The key aspect in this line of research is the definition of a representation model of multilingual documents over a unified conceptual space. This is generated through a large-scale multilingual knowledge base, namely BabelNet. Thematic alignment across documents is accomplished by obtaining semantically coherent cross-lingual topics, so to enable language-independent preserving of the content semantics. The multilingual documents are also decomposed into topically-coherent segments in order to enable the grouping of linguistically different portions of documents by content. Moreover, these two

key aspects are integrated into a multidimensional data structure in order to better capture the multi-topic multilingual nature of the documents. For the experimental evaluation, we have used two different multilingual datasets, Wikipedia and Reuters Corpus Volume 2. For both datasets, we have considered three languages, namely, English, French and Italian. This study, which has been originally presented in [164], is the focus of Chapter 5.

2

Background

In this chapter we provide basic notions that are functional to the understanding of the subsequent chapters. More precisely, in Section 2.1 we supply basic notions about document representation models for the document clustering, providing details about the feature definition, identification, selection and weighting steps. Subsequently, Section 2.2 provides a very brief overview of the task of document clustering and related challenges, also detailing differences between the clustering of long documents e short documents. In Section 2.3 basic notions about the lexical knowledge bases are given, focusing on WordNet, Wikipedia and BabelNet. Finally, in Section 2.4 the tensors and their decompositions are described giving also details about the related algorithms.

2.1 Document modeling

Any data model for text representation should exhibit a number of characteristics that allow for effective and efficient analysis of the text itself. Some of these characteristics make sense solely by referring to the original text, some others are related to the characteristics of representations of other texts in a collection. The following sketches a summary of the desiderata for text representations [129].

- Topic identification – A key aspect of text representations is the ability in characterizing the content of a target text, that is identifying its concepts, or topics, and the underlying meanings. Moreover, topic identification should be user-oriented to allow a fine-grained representation.
- Text content reduction – Providing a condensed characterization, or prototype, of the text content is fundamental, especially when dealing with large document collections. The prototype is substantially a compact description obtained by abstracting, indexing, or selecting the original text.
- Text content discrimination – A good text representation should allow for discriminating the associated content from other text representations'

content. However, the more a text representation is reduced, the more its ability in discriminating contents tends to get worse.

- Text content normalization – In natural language texts a concept, or semantically related concepts, can be expressed using different syntactic structures. Such structures are often characterized by lexical and morphological variations, which should be reduced to a "standard" form to attenuate *semantic ambiguity*.

The above gives evidence that creating an "optimal" text representation is quite hard, since some characteristics represent conflicting requirements. Therefore, a valid text representation should be designed depending on the specific *function* to be accomplished.

2.1.1 Models

Classic text representation models have been developed in the Information Retrieval (IR) context. A basic assumption is that a set of terms is identified to represent the text content. Such representative terms are text content descriptors as they exhibit semantics that allow for capturing the main topics of a text. Besides reflecting content, these terms are used as access points or identifiers of the text, by which the text can be located and retrieved in a text collection, thus they are usually called *index terms* [129, 16]. The problem of how to define index terms is postponed until next section. We now provide an outline of the two major IR models for text representation, namely the Boolean and the Vector space models.

Boolean model

The most simple representation model is based on the set theory and the Boolean algebra. A text representation is seen as a binary vector defined over the set of index terms: value 1 denotes the presence of a term, and 0 its absence, in the text. The representation of a query is instead a Boolean expression in disjunctive normal form, that is a disjunction of conjunctive vectors, each of which is a binary vector defined over the set of index terms appearing in the query [167, 16].

The Boolean model has been adopted by many commercial systems, thanks to its inherent simplicity in terms of development and implementation. However, the model suffers from several drawbacks. Firstly, a Boolean query is based on the assumption that a user knows exactly what he/she is getting and is able to translate the information need into a Boolean expression. Secondly, and most seriously, the Boolean model is very rigid, as the binary decision criterion imposes that a document is relevant to a query only if their representations match exactly: the query satisfaction corresponds to an overlap measure, and no ranking of documents according to relevance is provided. As a consequence, it is very difficult to control the number of documents retrieved, since all matched documents are returned in response to a query.

Vector space model

The overlap criterion used in the Boolean model does not consider two key elements to prevent better retrieval performance: *i)* the degree of relevance of any index term with respect to a given text, and *ii)* the text length.

The Vector space model [170] takes the above aspects into account to allow partial matching between text representations. Given a collection $\mathcal{D} = \{d_1, \ldots, d_n\}$ of documents and a vocabulary $\mathcal{V} = \{w_1, \ldots, w_m\}$, any document $d_j \in \mathcal{D}$ is represented by a m-dimensional vector $\mathbf{d}_j = [d_{j1}, \ldots, d_{jm}]$, such that each component d_{ji} measures the importance of index term w_i as a valid descriptor of the content of document d_j. As we shall discuss in Sect. 2.1.2, the "importance" can be suitably quantified on the basis of a combination of the frequency of occurrence of the term with respect to the document (local term frequency) and to the whole collection (global term frequency).

The relevance of a document to a query is evaluated as a measure of proximity between their respective vectors. A measure of proximity must satisfy at least three main properties: if document d_1 is near to document d_2, then d_2 is near to d_1 (symmetry); if d_1 is near to d_2, and d_2 is near to d_3, then d_1 is not far from d_3 (transitivity); no document is closer to d than d itself (reflexivity). A natural way to quantify the proximity between generic objects could be to resort to a distance function defined on a geometrical space, so that the distance between d_1's vector and d_2's vector would be the length of the difference vector. Actually, this idea turns out to be inappropriate for documents: indeed, the direct use of metrics does not address the issue of text length normalization, so that long documents would be more similar to each other by virtue of length, not topic.

The Vector space model proposes to quantify the proximity between documents by computing a function of *similarity*, typically expressed as the cosine of the angle between their respective vectors. Equipped with the cosine measure, the Vector space model allows for effectively ranking documents according to their degree of similarity to the query, which is represented as a vector in the same space as the documents. In practice, this model allows simple and efficient implementation for large document collections, and tends to work quite well despite the following simplifying assumptions [156]:

- the model is based on a "bag-of-words", that is terms are considered to be independent each other;
- document vectors are pair-wise orthogonal;
- both syntactic information (e.g., phrase structure, word order, proximity information) and semantic information (e.g., word sense) is missing;
- the model flexibility might be a shortcoming. For example, given a query with two terms a and b, the model may lead to preferring a document that contains a frequently but not b, over a document that contains both a and b, but both less frequently.

2.1.2 Feature identification

Our attention has been so far paid to functions and models for text representation. We have assumed the availability of a set of index terms representing the content of a text, that is a set of terms whose lexical meanings recall the main topics in the text. For this reason, such terms are used as attributes, concepts, or *features*, for text representation.

In this section, we focus on properties that are useful for evaluating what terms are worthy of being indexed. For instance, some terms (e.g., nouns) are natural candidates since they have meaning by themselves that allows for capturing text semantics, but the potential of meaning might be lowered depending on the term popularity, or rarity, in the text.

Feature identification involves the following main activities: *feature definition*, which investigates how the syntactic structure of the text is to be broken to recognize distinct concepts, and *feature selection* or *feature extraction*, which are devoted to reduce the concept space dimensionality. We examine each of the above points in turn.

Feature definition

The basic unit of linguistic structure is a *word*. Words differently contribute to the syntax and semantics of a text, depending on their syntactic class or *part-of-speech*. According to its class,[1] a word may refer to a precise kind of subject: for instance, nouns describe classes of concepts or objects, adjectives their properties, and verbs their relationships.

However, only some word classes are useful to reflect the text content, therefore words are usually divided in *content words* and *function words* [74]. The former identify concepts, objects, events, relationships in the discourse domain, thus they effectively serve as indicators of the text content. Nouns, adjectives, verbs, and some adverbs belong to this category. By contrast, function words are rather lexical elements that play a role in the syntactic structure, by serving grammatical purposes, and are not able to identify any concept. Example of function words are articles, pronouns, prepositions, connectives.

Besides syntactic category, any word is originally associated with a sense or *lexical meaning*, which determines the word semantics. Actually, this association is not always clear-cut, because of the following: a word may have multiple meanings, which may be related (*polysemy*) or unrelated (*homonymy*), and different words may have the same meaning (*synonymy*) or opposite meanings (*antonymy*). Moreover, there are words with a meaning that broadens/narrows the meaning of other words—*hypernymy*/*hyponymy*, for nouns; *hypernymy*/*troponymy*, for verbs—and words with a meaning that is-part-of/contains the meaning of other words—*meronymy*/*holonymy*, for nouns;

[1] A word belongs to more parts-of-speech when its morphological form refers to distinct concepts (e.g., noun 'bear' vs. verb 'bear', noun 'play' vs. verb 'play', noun 'major' vs. adjective 'major' etc.).

entailment/cause, for verbs. For example, 'tree' is hyponym of 'plant' (vice versa, 'plant' is hypernym of 'tree'), 'ship' is meronym of 'fleet' (vice versa, 'fleet' is holonym of 'ship'), 'walk' is troponym of 'move', 'drive' entails 'ride'.

Words form the components of which *phrases* are built. More precisely, any phrase consists of a head word, which is typically a content word indicating the described subject, and optional remaining words (e.g., modifiers and complements) that create the surrounding context of the head word. Depending on the part-of-speech of the head word, phrases can be of the following types: noun, adjective, verb, adverbial, and prepositional phrases [7].

It is commonly agreed that phrases can be better indicators of text content than individual words. Indeed, a phrase is semantically less ambiguous than the single constituent words, and may improve the specificity of the concepts expressed in a text. However, treating phrases is non-trivial due to their inherent complexity. For instance, phrases may contain *anaphora* (respectively, *cataphora*), that is lexical elements naturally used to avoid excessive repetition of terms by referring to other elements that appear earlier (respectively, later) in the text. Such referred elements identify the same concept of anaphora/cataphora, but with more descriptive phrasing. Moreover, a phrase can be characterized by the presence of *ellipses*. An ellipsis is a figure of speech with a deliberate omission of one or more words that are, however, understood in light of the grammatical context.

Feature selection

Feature selection is a complex activity that can be divided mainly into five phases, each of which corresponds to one or more *text operations* [16, 167]:

1. lexical analysis
2. removal of stopwords
3. word stemming
4. part-of-speech tagging
5. recognition of phrases

Before discussing each of these phases, it is worthy noticing that the above list does not represent a perfect outline of the feature selection process, because of a number of reasons. Firstly, a different ordering is possible, for instance recognition of phrases can be accomplished before removal of stopwords. Secondly, some application contexts require only some phases, while some others are optional or never performed. Thirdly, some phases can be seen as extraction of new features, rather than selection of existing ones: for instance, stemming conflates a number of words to one single term that does not necessarily appear in the text, while phrase formation groups individual words to obtain a new, complex feature. Feature extraction arises hard problems, which will be briefly discussed later in this chapter.

At the time of *lexical analysis* text is regarded as a stream of characters to be converted into a stream of tokens. Tokens are then transformed into words,

which will be further processed and eventually selected as index terms. Token transformation typically involves digits, hyphens, punctuation marks, and case letters processing. Since digits are inherently vague if not contextualized, numerical strings should be removed from text unless specified otherwise, for example by means of regular expressions. Hyphens and punctuation marks are usually neglected. The case of letters is also irrelevant for feature selection, thus characters are often converted to either upper or lower case.

For each of the above, there are language-dependent exceptions that should be carefully examined, by exploiting domain knowledge. For instance, proper names, company suffixes and dates cannot be recognized if case of letters is not preserved and all digits are neglected, respectively. Abbreviations and acronyms may lose any sense if cleaned from punctuation, instead they could be expanded using a machine-readable dictionary. Processing hyphenated tokens (i.e., compound words that include hyphens) is also more critical: breaking up these terms does not always improve the consistency of words, especially in case of proper names or words originally formed with hyphenation.

Consistency of words is also a major goal of *spelling correction*. This allows for recovering the actual form of a target "mispelled" word by looking for all words that most closely match it, within a predefined maximum number of errors. Edit distance is typically exploited to evaluate proximity between words,[2] therefore the maximum number of allowed errors for a word to match the target is measured as the maximum allowed edit distance. For example, 'text minino' is correctly replaced with 'text mining' (edit distance equals 1), and 'eterogeneiti' with 'heterogeneity' (edit distance equals 2).

Removal of stopwords is devoted to check whether a candidate index term is found in a *stoplist*, which is a negative list of words chosen among function words and terms that are highly frequent in the text or insufficiently specific to bear upon the text content. Such words, referred to as *stopwords*, are filtered out as irrelevant terms. Removal of stopwords acts as a pruning step in the feature selection, thus improving the efficiency in further processing of potential index terms and compressing the adopted storage structure considerably [61, 167].

Stemming, or conflating words, consists in reducing morphological variants of words to a standard form, called root or stem. The basic assumption is that words sharing a stem are semantically related or equivalent to the understanding of the text user. From a text representation perspective, stemming aims at improving the match between a candidate index term and its morphological variants, thus contributing to enhance the retrieval recall and to reduce the size of the text representation [105].

Four main approaches to stemming have been developed [61]: *table lookup*, *letter successor variety*, *n-grams*, and *affix removal*. The table lookup strategy consists in looking for word stems stored in a machine-readable dictio-

[2] Edit distance between two strings is computed as the minimum number of insertions, deletions, and replacements of characters needed to align the two strings.

nary. Despite its simplicity and high accuracy, the method is dependent on the specific domain language, and has to cope with problems of storage space and efficient search. Letter successor variety stemming [73] exploits linguistic knowledge and frequency of letter sequences to find morpheme boundaries as a basis for detecting word stems. The method does not distinguish inflectional from derivational affixes. The n-gram-based stemming acts on the number of n-grams shared by words to be conflated [204, 3]. n-grams enable stemmers to be language-independent and tolerate noise and misspelled words. However, n-gram-based stemmers are not able to distinguish inflectional from derivational affixes; moreover, too many n-grams lead to dramatically increase the size of feature set. The affix removal strategy is to trim affixes—prefixes and suffixes (e.g., plurals, tenses, gerund forms)—from terms, by mainly exploiting linguistic knowledge to infer the morphological form of stems. Many stemming algorithms have been developed, especially for the English language [61], and the leading ones are by Porter [154], Lovins [119] and Paice/Husk [143]. Despite a good trade-off between efficiency, accuracy and simplicity of development, affix removal stemmers may behave unnaturally as they often generate truncated pseudo-roots, instead of valid words, as stems.

The heterogeneity of existing stemming approaches has led to development of a methodology of evaluation of the performance of a stemmer when it is applied to test samples of words which have been previously organized into classes of equivalence or conflation groups. More precisely, a stemmer can be evaluated by counting two kinds of error that may occur, namely *under-stemming* errors and *over-stemming* errors [144]: the former refer to words that should be merged to the same stem (e.g., 'adhere' and 'adhesion') but remain distinct after stemming, while the latter refer to words that are really distinct (e.g., 'experiment' and 'experience') but are wrongly conflated. Under-stemming causes a single concept to be spread over different stems, thus negatively affecting retrieval recall, while over-stemming tends to dilute the meanings of a stem, which leads to a decrease of retrieval precision [85]. The error-counting method is effective, although the manual construction of the test sample of words is time-consuming and error-prone. Moreover, it is sometimes unclear whether two words should be reduced to the same stem, therefore the conflation groups should be sensitive to the specific topics of the text.

The absolute improvement of retrieval effectiveness due to stemming is still a matter of debate, although in most cases there has been overwhelming evidence in favor of stemming. Intuitively, stemming accuracy might be increased by combining affix removal stemmers and spelling correction algorithms, with the support of a dictionary composed of actual words. Additional benefit might be possible by including domain-specific dictionaries in the stem classes.

Part-of-speech (POS) tagging refers to labeling each word in a sentence with its proper syntactic class. POS tagging is useful to create linguistically motivated index terms, but also makes sense as intermediate task for detecting

slot filler candidates in IE. Common approaches range from (hidden) Markov models [41, 50] to transformation-based learning [24] and decision trees [123]; however, POS tagging performance is directly affected by the granularity and complexity of the adopted tag set (e.g., Brown tag set, Penn Treebank tag set) [122].

Selecting phrases as index terms entails two essential issues: the identification of meaningful phrases that possibly denote concepts relevant to particular subject domains, and the normalization of recognized phrases to a standard form. *Phrase recognition* can be accomplished according to two main approaches, namely *statistical phrase recognition* and *syntactic phrase recognition*. The former exploits statistical associations and constraints among words, such as co-occurrence of words and proximity of phrase components in the text [169, 38, 171]. The latter is based on the assumption that a syntactic relationship between phrase components implies also a semantic relationship [174, 38], and mainly exploits machine-readable dictionaries [55], stochastic POS taggers [49], syntactic templates [51], and context-free grammars [174].

Syntactic recognition works better than statistical recognition [56]; however, the high demand of index storage space and computational resources, which is only in part compensated by the slight improvement in text representation, seems to discourage the use of phrases as index terms. This tendency is mainly due to a number of problems related to phrase normalization. Indeed, a set of selected phrases often shows a marked lack of coherence with, most glaringly, the presence of dangling anaphora/cataphora: a phrase containing an anaphor but no antecedent (i.e., the term referred by the anaphor) may be jarring, or even unintelligible. Therefore, phrase normalization needs to handle ticklish cases, such as deciding whether a potential anaphor is actually being used in an anaphoric sense, or determining whether an anaphor has an antecedent within the same phrase or elsewhere.

Feature weighting

The above text operations produce features for text representation, that is a set of index terms that are assumed to reflect at best the text content. However, features have different discriminating power, as they contribute differently in representing the semantics of a text. Therefore, it is necessary to associate a *relevance indicator*, or weight, with each index term.

Many factors play a crucial role in weighting index terms, such as: statistics on the text (e.g., size, number of different terms appearing in), relations between an index term and the document containing it (e.g., location, number of occurrences), and relations between an index term and the overall document collection (e.g., number of occurrences). Term weights are commonly computed on the basis of the distribution patterns of terms in the text, and only rarely weighting relies upon expert knowledge on term relevance [182].

Looking at the distribution patterns of terms in the text to recognize terms that are good content bearing was suggested by Luhn [120]. He discovered that both extremely frequent and extremely unfrequent words[3] are not very useful for representing the text content, while the most discriminative terms have low-to-medium frequency. This claim is strictly related to the earlier well-known constant rank-frequency law of Zipf [214].

The Zipf's law models the frequency of occurrence of distinct objects in particular sorts of collections. Applied to the the term weighting context, the law states that the term with rank r (i.e., the r-th most frequent term) with respect to a set $\mathcal{V}$ of terms will appear with a frequency

$$f(r) = \frac{|\mathcal{V}|}{r^{\theta} \cdot H_{\theta}(\mathcal{V})},$$

where $H_{\theta}(\mathcal{V})$ is the harmonic number of order θ of $\mathcal{V}$. The value of θ typically ranges between 1 and 2, and is between 1.5 and 2 for the English text case. Roughly speaking, the Zipf's law says that the logarithm of the frequency of each term and its rank is approximately constant, that is the r-th most common word in a human language text occurs with a frequency inversely proportional to r.

The implication of the Zipf's law is that there is always a set of words in a text that dominate most of the other words from a frequency of occurrence viewpoint. This is true both of words in general, and of words that are specific to a particular subject. Moreover, there is a smooth continuum of dominance from most frequent to least, that is the smooth nature of the frequency curves implies that specific frequency thresholds are not necessary.

Starting from the Zipf-Luhn findings, most term weighting functions have been defined. A commonly used weighting function is based on the assumption that a term frequently occurring in a text represents better the content of that text than a rare term. Given a term w and a text d, the *term frequency* (tf) function computes the number of times term w occurs in d [168].

The term frequency makes sense as a weighting function provided that the text is not too short. In this case, information on term frequency may be negligible. Moreover, function tf does not consider term scarcity in the collection, while weighting should rely on the term distribution over all documents. Common terms tend to appear in many documents of the collection, so a measure of informativeness of an index term should include its rarity across the whole collection. Indeed, it has been observed that the higher the number of documents in which a term appears, the lower its discriminating power [182, 167]. The *inverse document frequency* (idf) function takes into account the above observation. Given a term w and a text d in a collection $\mathcal{D}$, idf can be just defined as the inverse of the fraction of documents in $\mathcal{D}$ that contain term w, but by far the most commonly used version includes the logarithm to decrease the effect of inverse document frequency [182, 168].

[3] Most words in a corpus are *hapax legomena*, that is words appearing only once.

Two statistical criteria have hence to be considered to evaluate a term as a valuable index term for a given document: high term density in the document, and high term rarity in the collection. It follows that the effect of function tf, which determines the term significance locally to a document, and the effect of function idf, which indicates the discriminating power of a term with respect to a document collection, have to be balanced out. This leads to define a combination of the above weighting functions into a unique function able to judge the best index terms as those appearing frequently within a text, but rarely in the remaining texts of the collection. Formally, given a document collection $\mathcal{D} = \{d_1, \ldots, d_n\}$, the $tfidf$ function computes the weight of any term w with respect to any document $d \in \mathcal{D}$ as:

$$tfidf(w, d) = tf(w, d) \times idf(w) = freq(w, d) \times \log\left(\frac{|\mathcal{D}|}{|D(w)|}\right),$$

where $freq(w, d)$ denotes the number of occurrences of w in d, and $D(w) = \{d \in \mathcal{D} \mid freq(w, d) \geq 1\}$. It is worthy noticing that the function increases with the number of occurrences within a document, as well as with the rarity of the term across the reference collection.

Another important aspect is the *normalization* of document lengths. Indeed, a collection may contain documents with different sizes: if the tf weights remain high for long texts and low for short ones, the real term specificity will be blurred. The importance of the raw term frequency should be hence reduced, so that the rarity of a term will have more negative impact in a long text rather than a short text. Common methods of length normalization act on the tf weight, for instance, by applying to it the logarithmic function, or dividing it by the maximum tf weight computed for the same document, or also dividing it by the squared root of the sum of the squared tf weights in the document [182, 168]. More frequently, length normalization is accomplished by applying the cosine normalization to the whole $tfidf$ weight:

$$\frac{tfidf(w, d)}{\sqrt{\sum_{u \in \mathcal{V}} tfidf(u, d)^2}}.$$

Many feature weighting approaches other than $tfidf$ have been developed, such as the probabilistic term relevance weight functions [160, 167], or the connectivity-based term discrimination model [171, 167]. Interesting weighting methods when features are not single words but phrases can be found in [113, 63, 38]. Nevertheless, phrase weighting has to cope with new issues, including the anaphora resolution and the uncertainty whether considering a phrase as a distinct concept (i.e., phrase weight is computed independently of the composing word weights) or as a compound of words (i.e., phrase weight is computed as the average weight of the composing word weights) [38].

Semantic analysis for feature extraction

As stated at the beginning of this section, one of the major requirements for text representation is content normalization, which aims at resolving semantic ambiguity in the text. This is substantially accomplished by exploiting knowledge on the usage of terms in certain subject domains to discover the meanings contextually associated to terms and the semantic relationships among terms. Viewed in this respect, a basic role is played by the construction and use of subject *thesauri* for transforming semantically related terms into more uniform and general concepts. A thesaurus has the form of a machine-readable dictionary, which is built on a vocabulary of important words in a given domain of knowledge and provides each such words with a semantic class [167, 16, 129]. Any semantic class consists of a set of words having a related meaning, and is labeled by a representative term. Most of thesaurus relationships refer to synonymies and near-synonymies, and classes are often hierarchically organized according to relationships of type *is-a* (hypernymy/hyponymy) and *part-of* (meronymy/holonymy).

A typical function of a thesaurus is organizing synonyms, and near-synonyms, into equivalence classes, so that any synonym in the text can be replaced with the representative of its membership class (*term clustering*). Besides synonyms, a thesaurus can be effectively used to control the problem of polysemous and homonymous words, whose meaning has to be "disambiguated" with respect to a certain context of use of the words.

Word sense disambiguation (WSD) is frequently mentioned as one of the most important problems in NLP research, and has been even described AI-complete, that is a problem which can be solved only by first resolving all the difficult problems in artificial intelligence (AI), such as the representation of common sense and knowledge. WSD is not usually seen as an independent task, but rather is essential at intermediate level for most language understanding applications, as well as for tasks and applications whose aim is not language understanding, such as: machine translation, information retrieval and document browsing (e.g., removing occurrences of words that are used in an inappropriate sense), content and thematic analysis (e.g., including only the instances of a word in its proper sense), POS tagging for grammatical analysis, correct phonetization of words in speech processing, spelling correction and case folding in text processing [86].

WSD aims at associating a given word in a text or discourse with a semantic definition, or sense, which is distinguishable from other potential senses of that word. This can be accomplished substantially involving two steps: *i)* identifying all the senses for each content word, and *ii)* assigning each instance of a word to the (most) appropriate sense. While the definition of a word sense is a matter of debate since the origins and usage of a word in certain contexts determine its lexical meaning, there is no doubt that the assignment of word to senses relies strictly on two major information sources: the context (i.e.,

content of the text in which the word appears, and extra-linguistic information about the text) and external knowledge bases (e.g., lexicons).

Many thesauri have been originally hand-built, mainly for convenience of human readers. However, manually building up a thesaurus is time-consuming and often is constrained to restricted subject domains. Nowadays, thesauri are machine-readable versions of semantically coded dictionaries.

Thesaurus classes are functionally close to *ontologies* used in NLP. Ontology is the branch of science concerned with the nature of being and relationships among things that exist [26, 27]. In computer science, term ontology refers to some conceptualization, that is primarily a formal specification of a certain knowledge domain, thus needs some knowledge representation [181]. While this is close to the definition of *conceptual model*, ontologies differ from conceptual models by *i)* focusing on abstract models based on a set of domain entities and a set of relationships among entities, and *ii)* having the explicit goal of sharing knowledge by defining a common theoretical framework and vocabulary so that interested agents can make, and share, a particular *ontological commitment* [70].

Ontologies provide a means for encoding complex information on words meaning and context [59]. Typical kinds of ontologies are taxonomies and inference rules. A taxonomy defines classes of entities and relationships among them. Classes are assigned with properties and subclasses are allowed to inherit such properties. Inference rules in ontologies use information about relationships among concepts underlying entities and their properties to conduct automated reasoning.

In the effort of providing explicit specification of conceptualizations, ontology representation formalisms (e.g., first-order logic formalisms [36]) have proved their potential in many application domains, including database integration, agent-based systems and distributed and federated data processing. Such formalisms stems from the requirement that a conceptualization has to be shared, i.e., using the same ontology implies the same view of the world (or, at least, of the concepts therein); however, their associated research issues are usually connected with the trade-off between the expressiveness and robustness of the formalism and the computational tractability of the inference mechanisms.

Semantic issues such as synonymy, polysemy and term dependence, are also addressed by *Latent Semantic Indexing* (LSI) [64, 48, 19, 145], which is frequently used to reduce the dimensionality of the set of features for text representation. The motivating reason is the indirect evidence that semantic connections among documents may exist even if they do not share terms. For example, terms 'car' and 'auto' cooccurring in a document may lead us to believe they are related. If 'car' and 'auto' are related, then they not only are expected to occur in similar sets of documents but also make other co-occurring terms indirectly related, e.g., if a document contains 'car' and 'engine' and another document contains 'auto' and 'motor', then 'engine' and 'motor' have some relatedness. Extending this to a general case and representing, as usual,

documents by the Vector space model, some rows and/or columns of the term-by-document incidence matrix may be somewhat "redundant", thus the matrix may have a rank far lower than its dimensions.

LSI formalizes the above intuition, using notions from linear algebra, to compress document vectors into a new, lower dimensional space. The new dimensions, or features, of documents are obtained as linear combinations of the original dimensions by looking at their patterns of co-occurrence. Co-occurring terms are projected onto the same dimensions by means of some similarity metric.

To perform dimensionality reduction, LSI typically applies the *Singular Value Decomposition* (SVD) [67] to the incidence matrix $\mathbf{A}$ formed by the original document vectors, and projects this matrix to a new matrix $\mathbf{A}_k$ in a lower k-dimensional space, such that the distance $\|\mathbf{A} - \mathbf{A}_k\|_2$ is minimized. Formally, a term-by-document matrix $\mathbf{A}_{t\times d}$ is factorized into three matrices $\mathbf{T}_{t\times n}\mathbf{S}_{n\times n}(\mathbf{D}_{d\times n})^{\mathrm{T}}$, where $\mathbf{T}$ is the column-orthogonal term matrix in the new space, $\mathbf{S}$ is the diagonal matrix of the *singular values* of $\mathbf{A}$ (i.e., the eigenvalues of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$) in descending order, and $\mathbf{D}$ is the column-orthogonal document matrix in the new space. LSI retains only some k singular values, where $k \ll n = \min(t, d)$, together with the corresponding rows of $\mathbf{T}$ and $\mathbf{D}$, which induce an approximation to $\mathbf{A}$, denoted as $\mathbf{A}_k = \mathbf{T}_k\mathbf{S}_k(\mathbf{D}_k)^{\mathrm{T}}$. Each row of $\mathbf{T}_k$ represents a term as a k-dimensional vector, similarly each row of $\mathbf{D}_k$ represents a document as a k-dimensional vector.

Major drawbacks in LSI are that newly obtained dimensions are not readily understandable (a solution to this issue is proposed in [84]), and the discriminating power of some original term may be lost in the new vector space. Moreover, most LSI methods adopt the conventional $tfidf$ weighting function (cf. Sect. 2.1.2) to produce the feature vectors. However, this approach does not seem to be optimal as it has been observed that low-frequency terms are underestimated, whereas high-frequency terms are overestimated. An implementation of LSI without SVD has been presented in [202], and several methods alternative to LSI have been proposed, such as probabilistic LSI [83], Linear Least Squares Fit [205], and Iterative Residual Rescaling [10]. The application of spectral analysis techniques to a variety of text mining tasks is presented in [13].

2.2 Document clustering

Compared to the traditional clustering task on relational data, document clustering faces several additional challenges. Corpora are high-dimensional with respect to words, yet documents are sparse, are of varying length, and can contain correlated terms [5]. Finding a document model, a set of features that can be used to *discriminate* between documents, is key to the clustering task. The clustering algorithm and the measure used to compute similarity between documents is highly dependent on the chosen document model.

Traditionally, documents are grouped based on how similar they are to other documents. Similarity-based algorithms define a function for computing document similarity and use it as the basis for assigning documents to clusters. Each group (cluster) should have documents that are similar to each other and dissimilar to documents in other clusters.

Clustering algorithms fall into different categories based on the underlying methodology of the clustering algorithm (e.g., *agglomerative* or *partitional*), the structure of the final solution (e.g., *flat* or *hierarchical*), or the multiplicity of cluster membership (*hard* or *soft*, *overlapping*, *fuzzy*). For instance, agglomerative algorithms find the clusters by initially assigning each object to its own cluster and then repeatedly merging pairs of clusters until a certain stopping criterion is met. A number of different methods have been proposed for determining the next pair of clusters to be merged, such as group average (UPGMA) [11], single-link [180], complete link [100], CURE [71], ROCK [72], and Chameleon [92]. Hierarchical algorithms produce a clustering that forms a *dendrogram*, with a single all-inclusive cluster at the top and single-point clusters at the leaves. On the other hand, partitional algorithms, such as k-Means [121], k-Medoids [11, 94], graph partitioning based [11, 184, 208], and spectral partitioning based clustering [21, 52], find the clusters by partitioning the entire dataset into either a predetermined or an automatically derived number of clusters. Depending on the particular algorithm, a k-way clustering solution can be obtained either directly or via a sequence of repeated bisections.

The Spherical k-Means algorithm (Sk-Means) [11] is used extensively for document clustering due to its low computational and memory requirements and its ability to find high-quality solutions. A spherical variant of the "fuzzy" version of k-Means, called Fuzzy Spherical k-Means (FSk-Means) [212, 109], produces an overlapping clustering by using a matrix of degrees of membership of objects with respect to clusters and a real value $f > 1$. The latter is usually called the "fuzzyfier", or fuzzyness coefficient, and controls the "softness" of the clustering solution. Higher f values lead to harder clustering solutions.

In recent years, various researchers have recognized that partitional clustering algorithms are well suited for clustering large document datasets due to their relatively low computational requirements [4, 93]. A key characteristic of many partitional clustering algorithms is that they use a global criterion function whose optimization drives the entire clustering process[4]. The criterion function is implicit for some of these algorithms (e.g., PDDP [21]), whereas for others (e.g., k-Means) the criterion function is explicit and can be easily stated. This later class of algorithms can be thought of as consisting of two key components. The first is the criterion function that needs to be optimized by the clustering solution, and the second is the actual al-

[4] Global clustering criterion functions are an inherent feature of partitional clustering algorithms, but they can also be used in the context of agglomerative algorithms.

gorithm that achieves this optimization. These two components are largely independent of each other. Various clustering algorithms and criterion functions are part of the CLUTO [91] clustering toolkit, which is available online at `http://www.cs.umn.edu/~cluto`.

Not that there is a broad corpus of research in document clustering. However, a detailed discussion on the various existing methods for document clustering is beyond the goal of this chapter. We refer the interested reader to [8] for a recent overview of document modeling and clustering methods and related challenges.

2.2.1 Clustering long documents

Long documents often discuss multiple subjects. This presents added challenge to general purpose document clustering algorithms that tend to associate a document with a single topic. The key idea to solving this problem is to consider the document as being made up of smaller topically cohesive text blocks, named *segments*. Segments can be identified independent of or concurrent to the clustering procedure.

Document segmentation

Text segmentation is concerned with the fragmentation of input text into smaller units (e.g., paragraphs) each possibly discussing a single main topic. Regardless of the presence of logical structure clues in the document, linguistic criteria and statistical similarity measures have been mainly used to identify thematically coherent, contiguous text blocks in unstructured documents [79, 18, 33].

The *TextTiling* algorithm [79] is the exemplary similarity block-based method for text segmentation. TextTiling is able to subdivide a text into multiparagraph, contiguous and disjoint blocks that represent passages, or subtopics. More precisely, TextTiling detects subtopic boundaries by analyzing patterns of lexical co-occurrence and distribution in the text. Terms that discuss a subtopic tend to co-occur locally. A switch to a new subtopic is detected when the co-occurrence of a given set of terms ends and the co-occurrence of another set of terms starts. All pairs of adjacent text blocks are compared using the cosine similarity measure and the resulting sequence of similarity values is examined in order to detect the boundaries between coherent segments.

Recent segmentation techniques have taken advantage of advances in generative topic modeling algorithms, which were specifically designed to identify topics within text. Brants et al. [23] use PLSA to compute wordtopic distributions, fold in those distributions at the block level (in their case blocks are sentences), and then select segmentation points based on the similarity values of adjacent block pairs. Sun et al. [188] use LDA on a corpus of segments, compute intrasegment similarities via a Fisher kernel, and optimize segmentation

via dynamic programming. Misra et al. [127] use a document-level LDA model, treat segments as new documents and predict their LDA models, and then perform segmentation via dynamic programming with probabilistic scores.

Clustering segmented documents

As previously discussed, a multi-topic document can be decomposed into segments that correspond to thematically coherent contiguous text passages in the original document. Segmentation can be used as a base step in long document clustering.

Tagarelli and Karypis [190] propose a framework for clustering of multi-topic documents that leverages the natural composition of documents into text segments in a "divide-et-impera" fashion. First, the documents are segmented using an existing document segmentation technique (e.g., TextTiling). Then, the segments in each document are clustered (potentially in an overlapping fashion) into groups, each referred to as a *segment-set*. Each segment-set contains the thematically coherent segments that may exist at different parts of the document. Thinking of them as mini-documents, the segment-sets across the different documents are clustered together into nonoverlapping thematically coherent groups. Finally, the segment-set clustering is used to derive a clustering solution of the original documents. The key assumption underlying this *segment-based document clustering* framework is that multi-topic documents can be decomposed into smaller single-topic text units (segment-sets) and that the clustering of these segment-sets can lead to an overlapping clustering solution of the original documents that accurately reflects the multiplicity of the topics that they contain.

2.2.2 Clustering short documents

Clustering short documents faces additional challenges above those of general purpose document clustering. Short documents normally address a single topic, yet they may do so with completely orthogonal vocabulary. Noise, contracted forms of words, and slang are prevalent in short texts.

There has been a relatively large corpus of study on alternative approaches to the clustering of short texts. Wang et al. [200] propose a frequent-term-based parallel clustering algorithm specifically designed to handle large collections of short texts. The algorithm involves an information-inference mechanism to build a semantic text feature graph which is used by a k-NN-like classification method to control the degree of cluster overlapping. Pinto et al. [150] resort to the information-theory field and define a symmetric KL divergence to compare short documents for clustering purposes. Since the KL distance computation relies on the estimation of probabilities using term occurrence frequencies, a special type of back-off scheme is introduced to avoid the issue of zero probability due to the sparsity of text. Carullo et al. [29] describe an incremental online clustering algorithm that utilizes a generalized

Dice coefficient as a document similarity measure. The algorithm requires two thresholds as input, one to control the minimum accepted similarity that any document must have to be assigned to a cluster, and the other to define the maximum similarity of a document that can still contribute to the definition of a cluster.

Particle-swarm optimization techniques and bio-inspired clustering algorithms have also been proposed for short text data. Ingaramo et al. [88] develop a partitional clustering algorithm to handle short texts of arbitrary size. The key aspect of that study is the adaptation of the AntTree algorithm [75], which integrates the "attraction of a cluster" and the Silhouette Coefficient concepts, to detecting clusters. Each ant represents a single data object as it moves in the clustering structure according to its similarity to other ants already connected to the tree under construction. Starting from an artificial support, all the ants are incrementally connected, either to that support or to other already connected ants. This process continues until all ants are connected to the structure, i.e., all objects are clustered.

2.2.3 Evaluation criteria

Evaluation of the effectiveness of a clustering task is usually accomplished by adopting either *external criteria* or *internal criteria*. External criteria aim to assess how well a clustering fits a predefined scheme of known classes (natural clusters). By contrast, internal criteria are defined over quantities that involve the data representations themselves, without any reference to external knowledge. This type of evaluation is useful when ideal classification of documents is not available. In this thesis we have considered a set of external criteria, since for each of the used datasets a ground-truth was available.

F-measure [178] is the most commonly used external criterion. Given a collection $\mathcal{D}$ of documents, let $\Gamma = \{\Gamma_1, \ldots, \Gamma_h\}$ be the desired classification of the documents in $\mathcal{D}$, and $\mathcal{C} = \{C_1, \ldots, C_k\}$ be the output partition yielded by a clustering algorithm. *Precision* of cluster C_j with respect to class Γ_i is the fraction of the documents in C_j that has been correctly classified:

$$P_{ij} = \frac{|C_j \cap \Gamma_i|}{|C_j|},$$

while *recall* of cluster C_j with respect to class Γ_i is the fraction of the documents in Γ_j that has been correctly classified:

$$R_{ij} = \frac{|C_j \cap \Gamma_i|}{|\Gamma_i|}.$$

For each pair (C_j, Γ_i), the F-measure FM_{ij} is defined as the harmonic mean of precision and recall [178]. In case of hierarchical clustering, the cluster structure can be flattened to also include the documents from all sub-clusters, that is all the documents in the subtree of cluster C_j are considered as the

documents in C_j [183]. The overall F-measure can be defined as the weighted sum of the maximum F-measures for all the natural clusters:

$$FM(\mathcal{C}, \Gamma) = \sum_{i=1}^{h} \frac{|\Gamma_i|}{|\mathcal{D}|} \max_{j=1..k} \{FM_{ij}\}.$$

Rand Index (RI) [159] measures the percentage of decisions that are correct, penalizing false positive and false negative decisions during clustering. It takes into account the following quantities: "true positives" (TP), i.e., the number of pairs of documents that are in the same cluster in $\mathcal{C}$ and in the same class in Γ; "true negatives" (TN), i.e., the number of pairs of documents that are in different clusters in $\mathcal{C}$ and in different classes in Γ; "false negatives" (FN), i.e., the number of pairs of documents that are in different clusters in $\mathcal{C}$ and in the same class in Γ; and "false positives" (FP), i.e., the number of pairs of documents that are in the same cluster in $\mathcal{C}$ and in different classes in Γ. Rand Index is hence defined as:

$$RI(\mathcal{C}, \Gamma) = \frac{TP + TN}{TP + TN + FP + FN}$$

Another external clustering validity criterion is based on *entropy* [176]. For each cluster C_j, the class distribution of data is computed as the probability $\Pr(\Gamma_i|C_j)$ that a document in C_j belongs to class Γ_i. Using this class distribution, the entropy of C_j is computed as

$$E_j = -\sum_{i=1}^{h} [\Pr(\Gamma_i|C_j) \times \log\big(\Pr(\Gamma_i|C_j)\big)].$$

where $\Pr(\Gamma_i|C_j)$ is estimated as P_{ij} The overall entropy is defined as the sum of the individual cluster entropies weighted by the size of each cluster:

$$E(\mathcal{C}, \Gamma) = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{k} (|C_j| \times E_j).$$

To compute *purity*, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by total number of documents [47]. Formally:

$$Pty(\mathcal{C}, \Gamma) = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{k} \max_{i=1..h} |C_j \cap \Gamma_i|$$

It has been shown that the mutual information $I(\mathcal{C}, \Gamma)$ between a clustering $\mathcal{C}$ and a reference classification Γ is a superior measure to purity or entropy [53, 185]. Moreover, by normalizing this measure to lie in the range

$[0, 1]$, it becomes relatively impartial to the number of clusters. *Normalized mutual information* (NMI) [213] can be information-theoretically interpreted and is defined as

$$NMI(\mathcal{C}, \Gamma) = \frac{\sum_{i=1}^{h} \sum_{j=1}^{k} |\Gamma_i \cap C_j| \log \left(\frac{|\mathcal{D}||\Gamma_i \cap C_j|}{|\Gamma_i||C_j|} \right)}{\sqrt{\left(\sum_{j=1}^{k} |C_j| \log \frac{|C_j|}{|\mathcal{D}|} \right) \left(\sum_{i=1}^{h} |\Gamma_i| \log \frac{|\Gamma_i|}{|\mathcal{D}|} \right)}}$$

2.3 Lexical knowledge bases

Lexical knowledge represents an essential component of language-oriented automatic tasks, including: question answering [76, 116], Word Sense Disambiguation (WSD) [39, 40, 136, 138], named entity disambiguation [25], text summarization [134], text categorization [199, 65, 137], coreference resolution [153, 157], sentiment analysis [189, 198] and plagiarism detection [17]. Many forms of lexical knowledge are available, such as unstructured terminologies, glossaries [57], thesauri [161], machine-readable dictionaries [155] and full-fledged computational lexicons and ontologies. The latter includes Cyc [111] and, more importantly, WordNet [60, 126].

2.3.1 WordNet

WordNet is a valuable resource for identifying taxonomic and networked relationships among concepts due to the following characteristics. Related concepts are grouped into equivalence classes, called *synsets* (sets of synonyms). Each synset represents one underlying lexical concept and is described by a short textual description (*gloss*). For example, {'car', 'auto', 'automobile', 'machine', 'motorcar'} is a synset representing the sense defined by the gloss "4-wheeled motor vehicle, usually propelled by an internal combustion engine". Synsets are explicitly connected to each other through existing relations (e.g., synonymy, antonymy, is-a, part-of), which connect senses of words that are used in the same part-of-speech.[5] Lexical inheritance, which underlies the is-a relationships between noun concepts[6] is perhaps the distinguishing characteristic of WordNet. Bipolar oppositions (antonymies) are used mainly to organize adjectives, while the different relationships that link verb concepts can be cast in terms of lexical entailment [125, 69, 58].

[5] The second release of WordNet (summer of 2003) allows now for links between derived forms of noun and verb concepts.

[6] Is-a hierarchy also exists for verbs, although it is referred to as is-way-of-doing, also known as troponymy, and is much shallower.

2.3.2 Wikipedia

Recently, research has been attracted by the increasing availability of online collaborative resources which contains semi-structured information in textual and/or hyperlinked form. A significant example of this type of resource is Wikipedia[7], the largest and most popular collaborative and multilingual resource of world and linguistic knowledge.

Wikipedia is a multilingual Web-based encyclopedia. It is a collaborative resource edited by volunteers from all over the world, providing a very large wide-coverage repository of encyclopedic knowledge. Each page in Wikipedia represents an article describing a particular concept or a named entity. The title of each page is composed in such a way it contains the lemma related to the concept plus an optional label (in parenthesis) in order to specify its meaning in case the lemma is polysemic.

Various relations between the pages are available:

- *Redirect pages*: this kind of pages is used to forward to the page containing the actual information about a concept of interest. This is the case of alternative expressions for the same concept, and thus it models synonymy.
- *Disambiguation pages*: These pages model the homonymy and the polysemy. They report links for all possible concepts expressed by means of an arbitrary expression.
- *Internal links*: The pages in Wikipedia typically contain hypertext linked to other pages (concepts).
- *Inter-language links*: Each page in Wikipedia provides links to its counterparts in other languages (i.e., corresponding concept described in other languages).
- *Categories*: A Wikipedia page can be assigned to one or more categories.

Many works in literature have used this resource in order to extraction structured information, including lexical and semantic relations between concepts [166, 186], factual information [203], and transforming the Web encyclopedia into a full-fledged semantic network [135, 46, 131]. However, despite its richness of explicit and implicit semantic knowledge, the encyclopedic nature of Wikipedia represents a major limit, lacking full coverage for the lexicographic senses a particular lemma. Such lack of lexical coverage can be provided by a highly-structured computational lexicon such as WordNet.

2.3.3 BabelNet

In the last two decades, the growing amount of text data that are written in different languages, also due to the increased popularity of a number of tools for collaboratively editing through contributors across the world, has exacerbated the need of cross-lingual and multilingual tasks, fostering the

[7] http://www.wikipedia.org.

development of multilingual lexical knowledge bases. Manual efforts of multilingual knowledge bases are EuroWordNet [197], MultiWordNet [149] and BalkaNet [195], but building them is an onerous task and requires many years for each new language. Furthermore, an additional, but not less important, cost is that required for interlinking the resources across different languages. More recently defined multilingual knowledge base is BabelNet [139], characterized by a wide-coverage and the automatic linking of two knowledge bases. Despite the availability of these monolingual/multilingual lexical knowledge bases, resources for non-English languages often suffer from a lack of coverage.

BabelNet [139] is a multilingual semantic network built with the goal to supply a highly-structured computational lexicon. This knowledge resource is obtained by linking Wikipedia with WordNet, that is, the largest multilingual Web encyclopedia and the most popular computational lexicon. The linking of the two knowledge bases was performed through an automatic mapping of WordNet synsets and Wikipages, harvesting multilingual lexicalization of the available concepts through human-generated translations provided by the Wikipedia inter-language links or through machine translation techniques. The result is an encyclopedic dictionary containing concepts and named entities lexicalized in 50 different languages.

The multilingual semantic network has been generated in three main steps: (i) combining WordNet and Wikipedia concepts by automatically acquiring WordNet senses and wikipages, (ii) harvesting multilingual lexicalization of the available concepts through human-generated translations provided by the Wikipedia *inter-language* links or through machine translation, and (iii) establishing relations between concepts or named entities exploiting the relations provided by WordNet as well as Wikipedia. The interested reader is referred to [139] for the description of the detailed procedure.

Multilingual knowledge in BabelNet is represented as a labeled directed graph in which nodes are concepts or named entities and edges connect pairs of nodes through a semantic relation. Each edge is labeled with a relation type (is-a, part-of, etc.), while each node corresponds to a *BabelNet synset*, i.e., a set of lexicalizations of a concept in different languages.

Starting from all available WordNet word senses along with lexical or semantic relations and from all Wikipedia concept along with relations provided by hyperlinks, the intersection (in terms of concepts) of these two knowledges was merged and

BabelNet can be accessed and easily integrated into applications by means of a Java API provided by the toolkit described in [140]. The information can be accessed at four main levels: lexicographic, encyclopaedic, conceptual, and multilingual. That toolkit, besides information access, provides functionalities for *graph-based WSD in a multilingual context*. Given an input set of words, a semantic graph is built by looking for related synset paths and by merging all them in a unique graph. Once the semantic graph is built, the graph nodes can be scored with a variety of algorithms. Finally, this graph with scored nodes is used to rank the input word senses by a graph-based approach.

2.4 Tensor models and decompositions

In this section we provide basic definitions about tensors, operations on tensors, and tensor decomposition methods which will be used throughout this thesis. We will mainly refer to notations usually adopted in classic references on tensor analysis, such as, e.g., [103, 35], to which the interested reader can refer for further details.

2.4.1 Basic notions

Tensors are the natural generalization of the matrices. A tensor is a multidimensional array. The number of dimensions (*ways* or *modes*) is called *order* of the tensor, so that a tensor with order N is also said a Nth-order, N-way or N-mode tensor and formally it is an element of the tensor product of N vector spaces, each of them with its own coordinate system. A vector is a first-order tensor, a matrix is a second-order tensor and a tensor with three or more modes is a higher-order tensor.

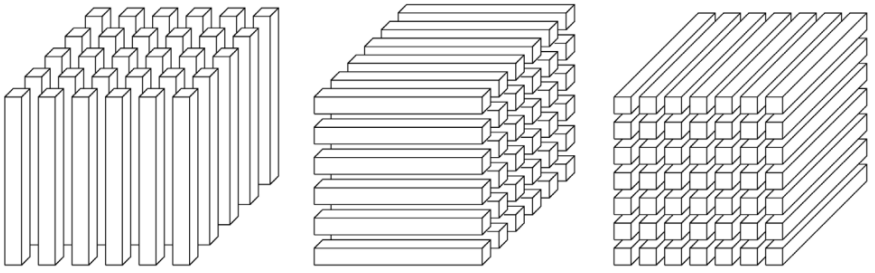
A *higher-order* tensor is denoted by boldface calligraphic letters, e.g., $\boldsymbol{\mathcal{X}}$; a matrix is denoted by boldface capital letters, e.g., $\mathbf{A}$; a vector is denoted by boldface lowercase letters, e.g., $\mathbf{a}$; a scalar is denoted by lowercase letters, e.g., a. The i-th entry of a vector $\mathbf{a}$, the element (i, j) of a matrix $\mathbf{A}$, and the element (i, j, k) of a third-order tensor $\boldsymbol{\mathcal{X}}$ are denoted by a_i, a_{ij}, and x_{ijk}, respectively. Hereinafter, we also use the symbols i_n (with $n = 1, \ldots, N$) and their capital version to denote the index along the specific n-th mode (e.g., $i_n = 1, \ldots, I_n$). Moreover, for the special case of third-order tensor, we will also use symbols i, j, k to denote the indices along the first, the second and the third mode. A superscript in parenthesis denotes an element in a sequence, e.g., $\mathbf{A}^{(n)}$ denotes the nth matrix of a sequence of matrices.

Tensor fibers

A one-dimensional fragment of tensor defined by varying one index and keeping the others fixed is a 1-way tensor called *fiber*. Fibers are the higher-order analogue of rows and columns of a matrix. A third-order tensor has column, row and tube fibers (Figure 2.1), denoted by $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$ and $\mathbf{x}_{ij:}$, respectively. A fiber in which the n-th mode is varying, is also called a mode-n fiber (e.g., a column fiber is said a mode-1 fiber).

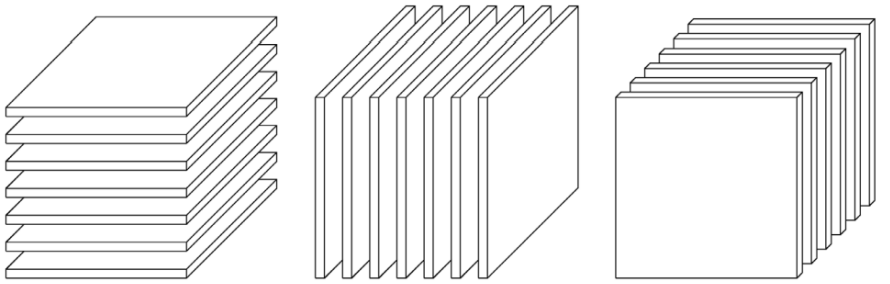
Tensor slices

Analogously, a two-dimensional fragment of tensor, defined by varying two indices and keeping the rest fixed, is a 2-way tensor called *slice*. A third-order tensor has horizontal, lateral and frontal slices (Figure 2.2), denoted by $\mathbf{X}_{i::}$, $\mathbf{X}_{:j:}$ and $\mathbf{X}_{::k}$, respectively.



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Fig. 2.1: Third-order tensor fibers [103]



(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

Fig. 2.2: Third-order tensor slices [103]

Rank-one tensor

An Nth-order tensor is *rank one* if it can be expressed as the outer product (denoted by "$\circ$") of N vectors. Formally, an Nth-order rank-one tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ can be expressed as:

$$\boldsymbol{\mathcal{X}} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)} \tag{2.1}$$

where $\mathbf{a}^{(n)} \in \mathbb{R}^{I_n}$, for $n = 1, \ldots, N$, and $x_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \ldots a^{(N)}_{i_N}$, for all $1 \leq i_n \leq I_n$; Figure 2.3 report a graphical representation of a third-order rank-one tensor.

Symmetric tensors

A *cubical* tensor (a tensor having all modes of the same size) is *supersymmetric* if its elements remain constant with any permutation of its indices. Formally for a 3-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, with $I = I_1 = I_2 = I_3$

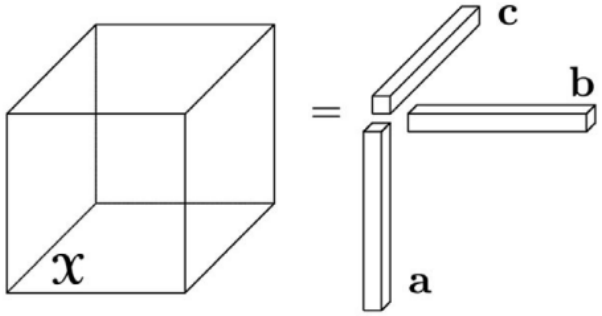


Fig. 2.3: Third-order rank-one tensor [103]

$$x_{i_1 i_2 i_3} = x_{i_1 i_3 i_2} = x_{i_2 i_1 i_3} = x_{i_2 i_3 i_1} = x_{i_3 i_1 i_2} = x_{i_3 i_2 i_1} \quad \forall i_1, i_2, i_3 = 1, \ldots, I \tag{2.2}$$

A tensor can be also symmetric in two or more modes. For instance, a 3-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, with $I_1 = I_2$, is symmetric in modes one and two if all its frontal slices are symmetric

$$\mathbf{X}_{i_3} = \mathbf{X}_{i_3}^T \quad \forall i_3 = 1, \ldots, I_3 \tag{2.3}$$

Diagonal tensor

A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is called *diagonal* if $x_{i_1 i_2 \ldots i_N} \neq 0$ only if $i_1 = i_2 = \cdots = i_N$ (Figure 2.4).

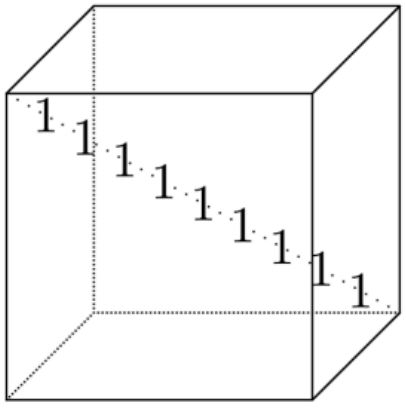


Fig. 2.4: Third-order digonal tensor [103]

Tensor matricization

The *matricization* (or *unfolding*) is the process of reordering the elements of an Nth-order tensor into a matrix. There are different definitions of the tensor matricization [101]. In this section we present only a particular matricization, which is relevant to our purposes. The mode-n matricization of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, denoted as $\mathbf{X}_{(n)}$, is obtained by arranging the mode-n fibers as columns of the resulting matrix. Formally, $(i_1, i_2, \ldots, i_N)$ is mapped to the element (i_n, j), where

$$j = 1 + \sum_{k=1, k \neq n}^{N} (i_k - 1) J_k \quad \text{with} \quad J_k = \prod_{m=1, m \neq n}^{k-1} I_m \tag{2.4}$$

The following is the example reported in [103], useful for an easier understanding. Let the frontal slices of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{3 \times 4 \times 2}$ be

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \qquad \mathbf{X}_2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

Then the three mode-n unfoldings are

$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & \ldots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & \ldots & 21 & 22 & 23 & 24 \end{bmatrix}$$

2.4.2 Math operators for tensors

Norm of a tensor

The *norm* of a tensor is the analogous of the matrix Forbenius norm (i.e., the square root of the sum of squares of all its elements) and, for a Nth-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ it is defined as:

$$\|\boldsymbol{\mathcal{X}}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N}^2} \tag{2.5}$$

Inner product

The *inner product* of two tensors of the same size is defined as sum of the products of their entries. Formally, for two 3-way tensors $\boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is defined as

$$\langle \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{X}} \rangle \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} y_{i_1 i_2 i_3} x_{i_1 i_2 i_3} \tag{2.6}$$

It follows that $\langle \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{Y}} \rangle = \|\boldsymbol{\mathcal{Y}}\|^2$

Multiplying a tensor by a matrix

A tensor can be multiplied by a matrix in mode n. The *n-mode product*, denoted by $\boldsymbol{\mathcal{X}} \times_n \mathbf{A}$, of an N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ is a tensor of size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_n$, whose generic entry is defined as

$$(\boldsymbol{\mathcal{X}} \times_n \mathbf{A})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} a_{j i_n} \tag{2.7}$$

The matrix $\mathbf{A}$ multiplies each mode-n fiber, hence

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_n \mathbf{A} \quad \Leftrightarrow \quad \mathbf{Y}_{(n)} = \mathbf{A}\mathbf{X}_{(n)} \tag{2.8}$$

In case of different modes in a series of n-mode products, the order of multiplication is irrelevant

$$\boldsymbol{\mathcal{X}} \times_n \mathbf{A_1} \times_m \mathbf{A_2} = \boldsymbol{\mathcal{X}} \times_m \mathbf{A_2} \times_n \mathbf{A_1} \tag{2.9}$$

A full treatment of tensor multiplication is reported in [15].

Matrix products

The *Kronecker product* of two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, denoted by $\mathbf{A} \otimes \mathbf{B}$, is a matrix of size $(IK) \times (JL)$ defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{2.10}$$

The *Khatri-Rao product* of two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, denoted by $\mathbf{A} \odot \mathbf{B}$, is a matrix of size $(IJ) \times K$ defined as

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \quad \mathbf{a}_2 \otimes \mathbf{b}_2 \quad \cdots \quad \mathbf{a}_K \otimes \mathbf{b}_K] \tag{2.11}$$

The *Hadamard product* is the element-wise matrix product. Given two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{I \times J}$, it is defined as

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I1} & \cdots & a_{IJ}b_{I1} \end{bmatrix} \tag{2.12}$$

The above matrix products have the following properties:

$$\begin{aligned} (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) &= \mathbf{AC} \otimes \mathbf{BD} \\ (\mathbf{A} \otimes \mathbf{B})^\dagger &= \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger \\ \mathbf{A} \odot \mathbf{B} \odot \mathbf{C} &= (\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C} = \mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}) \\ (\mathbf{A} \odot \mathbf{B})^{\mathrm{T}}(\mathbf{A} \odot \mathbf{B}) &= \mathbf{A}^{\mathrm{T}}\mathbf{A} * \mathbf{B}^{\mathrm{T}}\mathbf{B} \\ (\mathbf{A} \odot \mathbf{B})^\dagger &= ((\mathbf{A}^{\mathrm{T}}\mathbf{A}) * (\mathbf{B}^{\mathrm{T}}\mathbf{B}))^\dagger(\mathbf{A} \odot \mathbf{B})^{\mathrm{T}} \end{aligned} \tag{2.13}$$

where $\mathbf{A}^\dagger$ denotes the Moore-Penrose pseudo-inverse of $\mathbf{A}$ [68].

2.4.3 Tensor decompositions

To analyze tensors, several tensor decomposition algorithms have been proposed in the literature [103], the most popular of which are based on the more restricted CANDECOMP/PARAFAC [95] model or on the Tucker model [194].

CANDECOMP/PARAFAC decomposition

In the CANDECOMP/PARAFAC (CP) decomposition, a tensor is expressed in the polyadic form, i.e., as sum of a finite number of rank-one tensors (cf. Section 2.4.1). The idea of expressing a tensor in the polyadic form was first proposed by Hitchcock in the 1927 [81, 82]. Cattell proposed the idea of "parallel proportional analysis" in the 1944 [30] and the idea of "multiple axes for analysis" [31]. In 1970, this kind of decomposition was introduced in the psychometrics community by Carroll and Chang in the form of *Canonical Decomposition* (CANDECOMP) [28] and by Harshman in the form of *parallel factors* (PARAFAC) [77]. Finally, Kiers proposed a standardized notation and terminology for multiway analysis [95].

In the CP decomposition, the input tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is factorized as

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{2.14}$$

where R is the number of rank-one tensors in which $\boldsymbol{\mathcal{X}}$ is decomposed and $\mathbf{a}_r \in \mathbb{R}^{I_1}$, $\mathbf{b}_r \in \mathbb{R}^{I_2}$ and $\mathbf{c}_r \in \mathbb{R}^{I_3}$ for $r = 1, \ldots, R$. The element-wise expression of the generic $\boldsymbol{\mathcal{X}}$'s entry is

$$x_{i_1 i_2 i_3} \approx \sum_{r=1}^{R} a_{i_1 r} b_{i_2 r} c_{i_3 r} \tag{2.15}$$

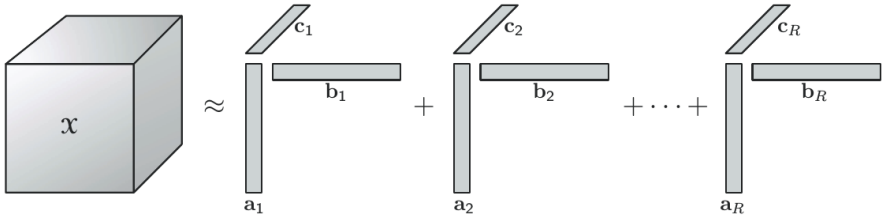


Fig. 2.5: CP decomposition [103]

The CP decomposition is illustrated in Figure 2.5.

In this kind of decomposition, the *factors matrices* refer to the combination of the vector forming the rank-one components (tensors), i.e., $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_R]$ and similarly for $\mathbf{B}$ and $\mathbf{C}$.

Often, it useful to assume that the columns of the factor matrices are normalized to length one with the weights absorbed into the vector $\lambda \in \mathbb{R}^R$

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{2.16}$$

Rank of tensor

The *rank* of a tensor $\boldsymbol{\mathcal{X}}$, denoted with rank($\boldsymbol{\mathcal{X}}$), is the smallest number of rank-one tensors that generate $\boldsymbol{\mathcal{X}}$ as their sum [81, 106]. By this definition it follows that setting $R = \text{rank}(\boldsymbol{\mathcal{X}})$ for the CP decomposition leads to an "exact" decomposition, hence there is equality in Equation 2.14. An exact CP decomposition with $R = \text{rank}(\boldsymbol{\mathcal{X}})$ components is called the *rank decomposition.*

It can be observed an analogy of the tensor rank with the matrix rank. As reported in [103], one difference between matrix rank and tensor rank is that the rank of a real-valued tensor may be different over $\mathbb{R}$ and $\mathbb{C}$. Furthermore, there is no straightforward algorithm to determine the rank of a generic given tensor, since the problem is NP-hard [78].

Computing CP decomposition

To compute the CP decomposition, the first issue that arises is how to choose the number of rank-one component tensors. Most procedures compute multiple decompositions using different numbers of components until a "good" one is reached, but, as argued in [103], by using this strategy many problems can be encountered. On the other hand, assuming the number of components R fixed, many algorithms for the computation of the CP decomposition are available. The most popular one is the alternating least squares (ALS) algorithm proposed by Carroll and Chang [28] and Harshman [77]. Given a third

order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, computing CP decomposition with R components aims to find the set of R rank-one tensor that best approximate $\boldsymbol{\mathcal{X}}$. Formally

$$\min_{\hat{\boldsymbol{\mathcal{X}}}} \|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\| \quad \text{with} \quad \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{2.17}$$

CP decomposition is easily generalizable to the case of Nth-order tensors. Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be a Nth-order tensor. The CP decomposition of $\boldsymbol{\mathcal{X}}$ is defined as:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)} \tag{2.18}$$

The ALS approach for an N-order is shown in Algorithm 1. As it can be seen, at each iteration, each of the factor matrices is solved keeping fixed the other ones. The algorithm iterates until a convergence criterion is satisfied.

Algorithm 1 CP-ALS

Require: A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and the number of components R
Ensure: The normalized factor matrices $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$, and related norms λ
1: initialize $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$
2: **repeat**
3: **for** $n = 1, 2, \ldots, N$ **do**
4: $\mathbf{V} \leftarrow \mathbf{A}^{(1)}\mathbf{A}^{(1)\mathrm{T}} * \cdots * \mathbf{A}^{(\mathrm{n}-1)}\mathbf{A}^{(\mathrm{n}-1)\mathrm{T}}\mathbf{A}^{(\mathrm{n}+1)}\mathbf{A}^{(\mathrm{n}+1)\mathrm{T}} * \cdots * \mathbf{A}^{(\mathrm{N})}\mathbf{A}^{(\mathrm{N})\mathrm{T}}$
5: $\mathbf{A}^{(\mathrm{n})} \leftarrow \mathbf{X}_{(\mathrm{n})}(\mathbf{A}^{(\mathrm{N})} \odot \cdots \odot \mathbf{A}^{(\mathrm{n}+1)} \odot \mathbf{A}^{(\mathrm{n}-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
6: **end for**
7: **until** convergence or maximum number of iterations is reached
8: **return** λ, $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$

CP decomposition applications

CP decomposition was applied for first time in psychometrics. Carroll and Chang [28] proposed CANDECOMP in for the analysis of multiple similarity or dissimilarity matrices from a variety of subjects with the idea that simply averaging the data for all the subjects leads to the loss of information about the different points of view. Harshman [77] proposed the PARAFAC in phonetics with the aim of eliminating the ambiguity of a two-dimensional PCA, providing better uniqueness properties. Appellof and Davidson [12] used the CP decomposition in chemometrics and Andersson and Bro [9] provided a survey about the use of CP decomposition in this context. Sidiropoulos et al. [179] proposed the use of CP decomposition in the context of sensor array processing. Mocks [128] independently discover the CP decomposition in brain imaging.

The CP decomposition has been applied also in data mining tasks. The first was by Acar et al.[1, 2] in the context of discussion detanglement in online chat rooms. Other applications were by Bader et a. [14] in text analysis, Chew et al.[32] in cross-lingual information retrieval and Shashua and Levin [177] in image compression and classification. Further details about the applications of CP decomposition can be found in [103].

Tucker decomposition

The Tucker decomposition was proposed for the first time by Tucker in 1963 [192] and it was refined in subsequent works by Tucker [193, 194] and Levin [112]. Tuckers work [194] is the most comprehensive of the early literature and is generally the most cited.

The Tucker decomposition is a class of decomposition and approximates a tensor into a smaller *core tensor* and a factor matrix along each mode. It can be considered a form of higher-order principal component analysis. For a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the Tucker decomposition is:

$$\begin{aligned} \boldsymbol{\mathcal{X}} &\approx \boldsymbol{\mathcal{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \\ &= \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} \mathbf{a}_{r_1} \circ \mathbf{b}_{r_2} \circ \mathbf{c}_{r_3} \end{aligned} \tag{2.19}$$

where $\mathbf{A} \in \mathbb{R}^{I_1 \times R_1}$, $\mathbf{B} \in \mathbb{R}^{I_2 \times R_2}$, and $\mathbf{C} \in \mathbb{R}^{I_3 \times R_3}$ are the *factor matrices* and can be seen as the principal components in each mode. The tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ is called *core tensor* and its entries intuitively express correlations among the different principal components in each mode. Meaningful information can be extracted from the factor matrices, although the choice of the dimensions of the core tensor might be critical as it impacts on the identification of the mode principal components. Figure 2.6 illustrates the Tucker decomposition.
The element-wise form of Equation 2.19 is

$$x_{i_1 i_2 i_3} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_1 r_2 r_3} a_{i_1 r_1} b_{i_2 r_2} c_{i_3 r_3} \tag{2.20}$$

As we shall describe in the following, most fitting algorithms for the Tucker decomposition make the assumption that the columns of the factor matrices are pair-wise orthogonal, but the Tucker model in principle does not make this assumption. In fact, the CP decomposition can be seen as a particular case of the Tucker one, in which the factor matrices have the same number of components (i.e., $R_1 = R_2 = R_3$) and the core tensor is super-diagonal.

The Tucker decomposition in the matricized forms (one for each mode) is as follows [101]:

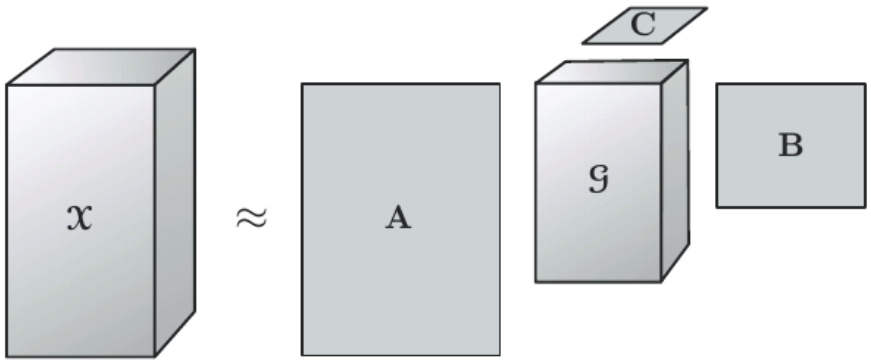


Fig. 2.6: Tucker decomposition [103]

$$\begin{aligned}\mathbf{X}_{(1)} &\approx \mathbf{A}\mathbf{G}_{(1)}\left(\mathbf{C}\otimes\mathbf{B}\right)^{\mathrm{T}}\\ \mathbf{X}_{(2)} &\approx \mathbf{B}\mathbf{G}_{(2)}\left(\mathbf{C}\otimes\mathbf{A}\right)^{\mathrm{T}}\\ \mathbf{X}_{(3)} &\approx \mathbf{C}\mathbf{G}_{(3)}\left(\mathbf{B}\otimes\mathbf{A}\right)^{\mathrm{T}}\end{aligned} \tag{2.21}$$

Although the Tucker decomposition was introduced for the three-way case, it can be easily generalized to the N-way case [90]. Formally, a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ is decomposed as

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} \tag{2.22}$$

where $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n\times R_n}$ for $n = 1\ldots N$ and $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1\times R_2\times\cdots\times R_N}$. The element-wise form of Equation 2.22 is

$$x_{i_1 i_2 \ldots i_N} = \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\cdots\sum_{r_N=1}^{R_N} g_{r_1 r_2 \ldots r_N} a^{(1)}_{i_1 r_1} a^{(2)}_{i_2 r_2} \ldots a^{(N)}_{i_N r_N} \quad \text{for } i_n = 1,\ldots,I_n \text{ and } n = 1,\ldots,N \tag{2.23}$$

while the matricized form is

$$\mathbf{X}_{(n)} = \mathbf{A}^{(n)}\mathbf{G}_{(n)}\left(\mathbf{A}^{(N)}\otimes\cdots\otimes\mathbf{A}^{(n+1)}\otimes\mathbf{A}^{(n-1)}\otimes\cdots\otimes\mathbf{A}^{(1)}\right)^{\mathrm{T}} \tag{2.24}$$

Two variants of the decomposition are commonly used. In first one, called *Tucker2* decomposition, one the factor matrices is set to be an identity matrix. In the second variant, called *Tucker1* decomposition, two of the factor

matrices are set to be identity matrices. Formally, for a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the *Tucker2* decomposition is defined as

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \tag{2.25}$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times R_1}$, $\mathbf{A}^{(2)} \in \mathbb{R}^{I_2 \times R_2}$ and $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times I_3}$; Equation 2.25 is the same as Equation 2.19 but with $R_3 = I_3$ and $\mathbf{A}^{(3)} = \mathbf{I}_{I_3}$ (i.e., the $I_3 \times I_3$ identity matrix). Similarly, the *Tucker1* decomposition is defined as

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \tag{2.26}$$

where $\mathbf{A}^{(1)} \in \mathbb{R}^{I_1 \times R_1}$ and $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times I_2 \times I_3}$; Equation 2.26 is the same as Equation 2.19 but with $R_2 = I_2$, $R_3 = I_3$, $\mathbf{A}^{(2)} = \mathbf{I}_{I_2}$ (i.e., the $I_2 \times I_2$ identity matrix) and $\mathbf{A}^{(3)} = \mathbf{I}_{I_3}$, (i.e., the $I_3 \times I_3$ identity matrix). Intuitively, in the *Tucker2* and *Tucker1* models, the factor matrices are incorporated in the core tensor.

The n-rank

The n-rank of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, denoted $\text{rank}_n(\boldsymbol{\mathcal{X}})$, is the column rank of the mode-n matricization of $\boldsymbol{\mathcal{X}}$. In other words, the rank-n of $\boldsymbol{\mathcal{X}}$ is the dimension of the vector space spanned by the columns of $\mathbf{X}_{(n)}$ (i.e., the mode-n fibers). A rank-$(R_1, R_2, \ldots, R_N)$ tensor is a tensor for which $R_n = \text{rank}_n(\boldsymbol{\mathcal{X}})$ for $n = 1, \ldots, N$.

Computing Tucker decomposition

There are three main approaches to the computation of the Tucker decomposition [194]. In the first one, the basic idea is to extract those components that best capture the variation in mode n, independently of the other modes. This method is today known as *Higher-Order Singular Value Decomposition* (HOSVD) from a work of De Lathauwer, De Moor, and Vandewalle [43], in which they showed as the HOSVD is a generalization of the matrix Singular Value Decomposition. In the HOSVD, the factor matrices $\mathbf{A}^{(n)}$ $(n = 1, \ldots, N)$ are orthogonal matrices and the core tensor $\boldsymbol{\mathcal{G}}$ is an all-orthogonal and ordered tensor of the same dimension as the data tensor $\boldsymbol{\mathcal{X}}$. It should be noted that, in this definition of HOSVD, the number of components for a particular mode is equal to the dimension of the data tensor in that mode, that is $I_n = R_n$ for $n = 1, \ldots, N$.

The result of the HOSVD is an ordered orthogonal basis for multidimensional representation of input data. The dimensionality reduction in each space is achieved by projecting the data sample onto subspace defined by the principal axis and keeping only the components related to the leading (largest) singular values in that subspace. This leads to the concept of best rank-$R_1, R_2, ..., R_N$ approximation [44], formulated as follows: "Given a real

Nth-order tensor $\boldsymbol{\mathcal{X}} \in R^{I_1 \times I_2 \ldots \times I_N}$ find a lower rank tensor $\hat{\boldsymbol{\mathcal{X}}}$ of the same dimension which minimizes the FIT". In this kind of decomposition, called *Truncated HOSVD* (T-HOSVD), R_n is much smaller than I_n, for $n = 1, \ldots, N$. T-HOSVD is then computed in two steps:

1. For each mode, compute the tensor matricization $\mathbf{X}_{(n)}$ from $\boldsymbol{\mathcal{X}}$ and their standard SVD: $\mathbf{X}_{(n)} = \mathbf{U}^{(n)}\mathbf{S}^{(n)}\mathbf{V}^{(n)\,T}$. The orthogonal matrix $\mathbf{U}^{(n)}$ represents the leading left singular vectors of $\mathbf{Y}_{(n)}$ and it will be the factor matrix for the mode n (i.e., $\mathbf{A}^{(n)} = \mathbf{U}^{(n)}$).
2. Compute the core tensor as

$$\boldsymbol{\mathcal{G}} \approx \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)\mathrm{T}} \times_2 \mathbf{A}^{(2)\mathrm{T}} \cdots \times_N \mathbf{A}^{(N)\mathrm{T}} \tag{2.27}$$

The T-HOSVD is hence computed by means of N standard singular value decompositions. This method is shown in Algorithm 2

Algorithm 2 Trucated Higher-Order SVD

Require: A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and the number of components of for mode $R_1, R_2, \ldots, R_N$

Ensure: Core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and factor matrices $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$

1: **for** $n = 1, 2, \ldots, N$ **do**
2: $\quad \mathbf{A}^{(\mathrm{n})} \leftarrow R_n$ leading left singular vectors of $\mathbf{X}_{(\mathrm{n})}$
3: **end for**
4: $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)\mathrm{T}} \times_2 \mathbf{A}^{(2)\mathrm{T}} \cdots \times_N \mathbf{A}^{(\mathrm{N})\mathrm{T}}$
5: **return** $\boldsymbol{\mathcal{G}}$, $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$

The HOSVD does not minimize the loss function $\|\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}\|_F^2$, and does not produce an optimal lower rank-$R_1, R_2, ..., R_N$ approximation to $\boldsymbol{\mathcal{X}}$, since it optimizes for each mode separately without taking into account interactions among the modes. Nevertheless, the HOSVD often produces a close to optimal low rank approximation and is relatively fast in comparison with other iterative algorithms, including those discussed next.

The computation of the best rank approximation of a tensor requires an iterative ALS algorithm called Higher-Order Orthogonal Iteration (HOOI) [44] (Algorithm 3). The HOOI uses the HOSVD to initialize the matrices. In each step of the iteration, only one of the basis matrices is optimized, while keeping others fixed. The HOOI algorithm has been introduced by De Lathauwer, De Moor and Vandewalle [44] and recently extended and implemented by Kolda and Bader in [103] in their MATLAB Tensor Toolbox.

Algorithm 3 Higher Order Orthogonal iterations (HOOI)

Require: A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and the number of components of for mode $R_1, R_2, \ldots, R_N$
Ensure: Core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and factor matrices $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$
1: initialize $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$ using HOSVD
2: **repeat**
3: **for** $n = 1, 2, \ldots, N$ **do**
4: $\boldsymbol{\mathcal{Y}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)\mathrm{T}} \cdots \times_{\mathrm{n-1}} \mathbf{A}^{(\mathrm{n-1})\mathrm{T}} \times_{\mathrm{n+1}} \mathbf{A}^{(\mathrm{n+1})\mathrm{T}} \cdots \times_N \mathbf{A}^{(\mathrm{N})\mathrm{T}}$
5: $\mathbf{A}^{(\mathrm{n})} \leftarrow R_n$ leading left singular vectors of $\mathbf{Y}_{(\mathrm{n})}$
6: **end for**
7: **until** convergence or maximum number of iterations is reached
8: $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)\mathrm{T}} \times_2 \mathbf{A}^{(2)\mathrm{T}} \cdots \times_N \mathbf{A}^{(\mathrm{N})\mathrm{T}}$
9: **return** $\boldsymbol{\mathcal{G}}$, $\mathbf{A}^{(\mathrm{n})} \in \mathbb{R}^{I_n \times R}$ for $n = 1, 2, \ldots, N$

Tucker decomposition applications

Examples of Tucker decomposition applications are provided by Henrion [80] in chemical analysis and by Kiers and Van Mechelen [96] in psychometrics. De Lathauwer and Vandewalle [45] and Muti and Bourennane [133] applied Tucker decomposition in signal processing. Vasilescu and Terzopoulos [196] pioneered Tucker decomposition in the context of computer vision.

In data mining, Tucker decomposition was applied in the handwritten digits identification by Savas and Eldén [173]. Acar et al.[1, 2] applied Tucker decomposition in the context of discussion detanglement in online chat rooms. Tucker decomposition was also used by Sun et al. [187] to analyze web site click-through data. Liu et al. [117] proposed an extension of the vector space model. Further applications of Tucker decomposition can be found in [103].

Non-negative Tucker decomposition

Another type of Tucker decomposition is the *nonnegative Tucker decomposition* (NTD) [97, 146, 148], which is characterized by the presence of nonnegative constraints (i.e., factor matrices and core tensor have nonnegative entries). Such a decomposition has already found some applications in neuroscience, bioinformatics and chemometrics.

Algorithms that implement an NTD are usually ALS algorithms and are characterized by multiplicative global learning rules and in which, generally, at each iteration of the decomposition, each of the factor matrices is updated by multiplying it with a term that is function of the core tensor and of the other factor matrices; the update rule for the core tensor is similarly formulated [97, 98, 62, 132]. Large-scale problems, with the raw data tensor and its temporary variables stored in memory, are very large-scale and often cause memory overflow error during the decomposition process. To avoid this problem, one possible solution is to process and update the tensor and

its factors following block-wise or vector procedures, instead of operating on whole matrices or tensors [146, 148, 147]. This approach is referred to as a *local decomposition* or *local learning rule*.

A major advantage due to the application of such multiplicative rules is that the decomposition algorithm is able to more easily capture correlations among all modes of the tensor. Moreover, as discussed in [35], NTDs can be profitably exploited to produce clustering solutions that might be meaningful on each particular mode. Recently, a particularly effective formulation of NTD that uses a beta divergence has been successfully applied for PCA and clustering, robust ICA and robust NMF/NTF [34].

The ALS algorithms for NTD often require the initialization of the factor matrices; the HOSVD and HOOI are often used as initialization method, especially in those cases in which the factor matrices are sparse, orthogonal or close to orthogonality. An advantage in using HOSVD and HOOI approaches over standard ALS is that they are able to estimate the dimension of the core tensor by analyzing the singular values.

3

Tensor-based Clustering for Multiple Document Classifications

3.1 Introduction

Nowadays, with the diffusion of the new technologies, a huge amount of document is produced and in many real-world applications multiple clustering solutions might be available for the same document collection, therefore a challenge is to effectively cope with this knowledge to provide a unique yet meaningful clustering solution.

In this chapter we are interested in extending the task of document clustering, which is traditionally performed according only to the textual content information of the documents, to the case in which a single clustering is desired starting from multiple organizations of the documents. Such existing document organizations can be seen as multiple views over a document collection which might correspond to user-provided, possibly alternative organizations, or to the results separately obtained by one or more document clustering algorithms or supervised text classifiers. For example, news articles can be clustered based on the topics they discuss, or by citation-links, or to reflect some existing categorization of major themes or different types of meta-information (e.g., author, newswire source) they are related to. In the following, we refer to the existing multiple document organizations simply as document classifications.

The underlying assumption of our approach is that, when the documents can be naturally grouped in multiple ways, a single new clustering encompassing all existing document classifications can be obtained by integrating the textual content information with knowledge on the groupings of the documents through the available classifications. However, since no information about any labels of the available groups of documents is assumed to be required, our key idea to accomplish the task relies on the identification of frequent co-occurrences of documents in the groups across the existing classifications, in order to capture how documents tend to be grouped together orthogonally to the different views. Based on the discovered frequent associations of the documents as well as on the usual term-document representation

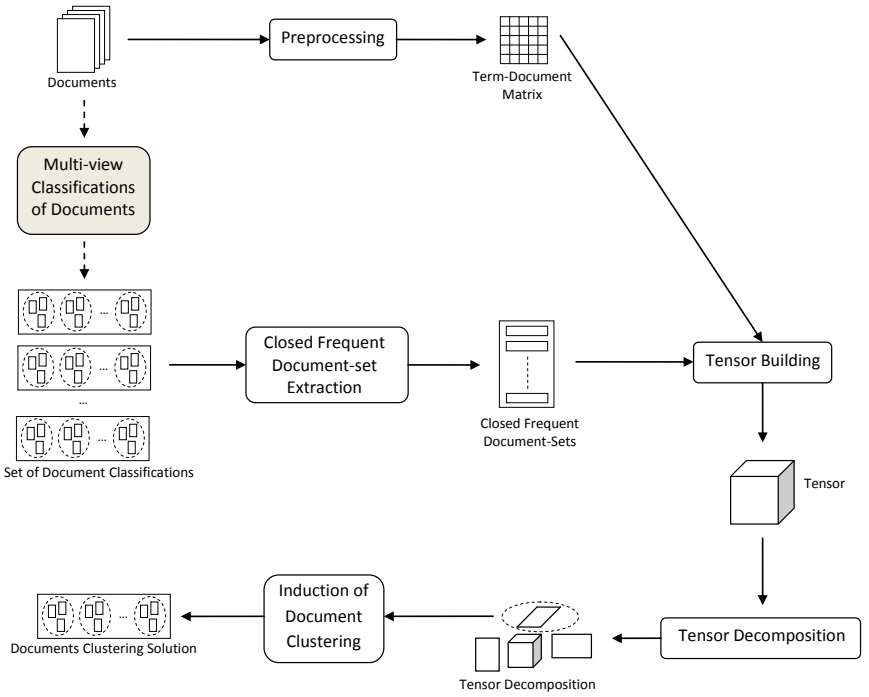


Fig. 3.1: Overview of the tensor-based clustering approach for multiple document organizations

of the text contents, a novel tensor model is built and decomposed to finally establish a unique clustering of documents that might be suited to reflect the multidimensional structure of the initial document classifications.

Figure 3.1 shows the main modules and data flows in the proposed framework. For a given document collection, a set of classifications of the documents is assumed to be available as a result of an independent process of multi-view document categorization. The collection of documents is initially subject to a standard preprocessing step, which yields the usual vector-space representation of the collection in the form of a term-document matrix, while, inspired by the classic task of frequent pattern discovery in transactional data, another module is in charge of discovering frequently occurring subsets of documents across the multiple classifications, in the form of *closed frequent document-sets*. The central part of the framework consists of the steps for the construction and the decomposition of the tensor model. A third-order tensor is built over the outputs of the two previous steps, so that three dimensions are considered together in the tensor, namely the terms, the documents, and the closed frequent document-sets extracted from the multiple views. A new term-relevance weighting scheme is also developed to compute the tensor entries. The tensor

decomposition module is in charge of producing a new tensor and three factor matrices, where the new tensor (core-tensor) is much smaller than the original one and expresses the hidden interactions among the three dimensions, and the factor matrices express the strength of each term, document, and frequent document-set, respectively, along the components pertaining to a specific dimension of the data. Finally, a single encompassing document clustering is induced by analyzing a factor matrix of the decomposed tensor.

The chapter is organized as follows. In Section 3.2 basic notions on frequent pattern mining will be given. Then, Section 3.3 describes in detail the proposed approach. Subsequently, in Section 3.4 the experimental evaluations and the results are shown and Section 3.5 contains the related works. Finally, Section 3.6 summarizes the chapter.

3.2 Preliminaries on frequent itemset mining

The frequent itemset mining problem is well-known in data mining research as it aims to discover all frequent itemsets in a transactional dataset. Given a set $\mathcal{I}$ of categorical values, or *items*, a transactional dataset $\mathcal{T}$ is a multiset of *transactions*, such that each transaction t is a subset of the item domain. Given any subset of $\mathcal{I}$, or itemset, its support is the number of transactions in $\mathcal{T}$ that contain it. An itemset is said frequent if if its support is not lower than a user-specified minimum-support threshold, *minsup*.

The number of frequent itemsets may be huge since the problem is exponential with the number of items. To reduce the number of patterns to be mined, some work has shifted toward the mining of closed and maximal itemsets. An itemset is *closed* if it is frequent and none of its supersets has exactly the same support. For a given support, the complete set of frequent itemsets can be obtained from the (typically much smaller) set of closed itemsets (including their support information). Moreover, a closed itemset is called *maximal* if none of its proper supersets is frequent. All frequent itemsets can be obtained from the set of maximal frequent itemsets, although not their support. Therefore, the set of maximal frequent itemsets is only an approximation of the information in the set of all frequent itemsets.

A particularly efficient method for mining closed frequent itemsets is CHARM algorithm [209] that uses a search space called *IT-Tree* in which each node is an IT-Pair, i.e., a pair *itemset-tid-set* (where *tid-set* is a set of transactions IDs in which the itemset appears). Despite CHARM can reduce the search space by pruning techniques, the large number of items can still make the task intractable in practice. By contrast, "bottom-up" approaches based on *intersections between transactions* can be more suited for solving the frequent itemset problem in the case that the number of transactions is comparable or even larger than the number of items. This case has been however much less studied in the literature. Two algorithms that follow this strategy are described in [22]. The first one works by enumerating sets of transactions

and intersecting them to find the frequent closed itemsets. The second one is based on a repository of all closed itemsets which is updated by intersecting it with the transaction.

3.3 Proposed method

We are given a collection $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$ of documents, which are represented over a set $\mathcal{V} = \{w_1, \ldots, w_{|\mathcal{V}|}\}$ of terms. We are also given a set of alternative organizations of the documents in $\mathcal{D}$, denoted as $\mathcal{CS} = \{\mathcal{C}_1, \ldots, \mathcal{C}_H\}$, such that each $\mathcal{C}_h = \{C_1, \ldots, C_{n_h}\}$ represents a set of n_h homogeneous groups. We hereinafter generically refer to each of the document organization as a *document clustering* and to each of the homogeneous groups of documents as *document cluster*.[1]

In the following we describe in detail the proposed framework. As previously shown in Figure 3.1, we organize the presentation into four main steps, namely extraction of closed document-sets from multiple document organizations, construction of the tensor model, decomposition of the tensor, and induction of a document clustering.

3.3.1 Extracting closed frequent document-sets

In our setting, an item $e \in \mathcal{I}$ corresponds to a document $d \in \mathcal{D}$, hence an itemset is a *document-set*. A transaction $t \in \mathcal{T}$ corresponds to a cluster C that belongs to any of the clusterings in $\mathcal{CS}$. As a transactional dataset is a multiset of transactions, there will be as many transactions as the number of clusters over all document clusterings in $\mathcal{CS}$. A frequent pattern mining algorithm applied to the above defined transactional dataset will extract document-sets that frequently occur over the clusters in the available document clusterings. Moreover, the frequent document-sets being discovered need to be closed, since we desire to minimize the size of the set of patterns discovered, while ensuring the completeness of such a set (cf. Section 3.2). Note also that we could not deal with maximal frequent patterns (as to further minimize the size of the set of patterns discovered), since we need to keep the support information to define the term relevance weighting function used for the tensor construction, which will be clarified in the next section.

As previously mentioned, a peculiarity of our transactional context is that, unlike any typical scenario of transactional data, the size of the transactional dataset (i.e., the number of document clusters) is much lower than the size of the item domain (i.e., the number of documents). As a consequence, in order to extract (closed) frequent document-sets, a traditional (closed) frequent

[1] Although the input document organizations might derive from a supervised text categorization task, no class labels or label correspondence scheme are assumed to be available.

itemset mining approach could be prohibitive, as it would require a cost which is exponential with the number of documents.

Let $T \subseteq \mathcal{T}$ denote a set of transactions (transaction-set) and $I_T = \bigcap_{t \in T} t$ denote the itemsets (i.e., documents) shared by the transactions (i.e., clusters) in T. A transaction-set containing d transactions is said a d-transaction-set, with $1 \leq d \leq |\mathcal{T}|$. Moreover, we assume that a total ordering exists among the transactions in the input dataset $\mathcal{T}$ based on the transaction ids (TIDs); this ordering is also exploited to compare transaction-sets: for any two transaction-sets T_i and T_j, it holds that $T_i < T_j$ if and only if there exists an index $p \in [1..\min\{T_i, T_j\} - 1]$ such that $t_{i_p} = t_{j_p}$ and $t_{i_{p+1}} = t_{j_{p+1}}$.

Algorithm 4 shows the proposed closed frequent itemset miner, which uses a level-wise search where d-transaction-sets are used to explore $(d+1)$-transaction-sets. To perform the search, an enumeration tree is incrementally built such that each node represents a pair of the form (T, I_T); initially, each individual transaction (and its set of items) forms a pair in its own (Line 2). This initial set of 1-transaction-sets is used to compute $(1 + it)$-transaction-sets, at each iteration it of the *search* procedure; this procedure terminates after $|\mathcal{T}|$ levels, i.e., when all transactions have been considered in a single union set. To avoid redundant unions among transaction-sets (hence, intersections among their itemsets), the ordering between the first transactions of any two transaction-sets is involved at each iteration (Lines 10 and 12). Note that, as the search space is being explored, the support of the itemsets obtained by the intersection of a growing number of transactions is monotonically non-decreasing. Therefore, every candidate closed itemset (Line 13) is checked to be a frequent itemset (Line 15).

The *merge* function (Line 5) searches for all pairs that have the same common itemset and yields a single pair containing the union of the transaction-sets, formally for each (T, I_T) and $(T', I_{T'})$ such that $I_T = I_{T'} = I$ and $T \neq T'$, the two pairs are replaced with the new pair $(T \cup T', I)$. Finally, the set CI of all closed frequent itemsets from $C_{IT\text{-}P}$ is returned (Line 6). Figure 3.2 illustrates an example of extraction of closed frequent itemsets from four transactions using Algorithm 4.

3.3.2 Building a tensor for multiple document organizations

We define a third-order tensor to model a set of documents contextually to multiple available organizations of the documents. Our key idea is to represent the content information of each document (based on the vocabulary terms) over each frequent aggregation of the document across the various organizations. Within this view, we assign the three modes the following meaning:

- Mode-1: the closed frequent document-sets extracted from the set of document organizations;
- Mode-2: the terms representing the document contents;
- Mode-3: the documents.

Algorithm 4 Intersection-based Closed Frequent Itemset Miner

Input: A transactional dataset $\mathcal{T}$, a minimum support threshold *minsup*
Output: A set CI of closed frequent itemsets

$C_{IT\text{-}P} \leftarrow \emptyset$
$P \leftarrow \{(\{t\}, t) \mid t \in \mathcal{T}\}$
$P_0 \leftarrow P$
search$(P_0, P, C_{IT\text{-}P})$
$C_{IT\text{-}P} \leftarrow$ *merge*$(C_{IT\text{-}P})$
$CI \leftarrow$ *flatten*$(C_{IT\text{-}P})$

procedure *search*$(P', J_1, C_{IT\text{-}P})$
for all $(T, I_T) \in P'$ **do**
  let t be the first transaction in T
  $P'' \leftarrow \emptyset$
  for all $(\{t_i\}, t_i) \in J_1, t < t_i$ **do**
    $T_j \leftarrow T \cup \{t_i\}, \quad I_{T_j} \leftarrow I \cap \{t_i\}$
    $P'' \leftarrow IP'' \cup \{(T_j, I_{T_j})\}$
    if $sup(I_{T_j}) \geq minsup$ **then**
      remove from $C_{IT\text{-}P}$ all (T_k, I_{T_k}) such that $T_j \supseteq T_k$ and $I_{T_k} = I_{T_j}$
      if I_{T_j} is a closed itemset for the itemsets in $C_{IT\text{-}P}$ **then**
        $C_{IT\text{-}P} \leftarrow C_{IT\text{-}P} \cup \{(T_j, I_{T_j})\}$
      end if
    end if
  end for
  search$(P'', P, C_{IT\text{-}P})$
end for

Formally, we define a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, where I_1 is the number of mined closed frequent document-sets, $I_2 = |\mathcal{V}|$ is the number of terms, and $I_3 = |\mathcal{D}|$ is the number of documents. We hereinafter denote as $\mathcal{CDS} = \{CDS_1, \ldots, CDS_{I_1}\}$ the set of closed frequent document-sets extracted from $\mathcal{CS}$.

Figure 3.3 shows our proposed three-order tensor. The i_3-th slice of the tensor refers to document d_{i_3} and is represented by a matrix of size $I_1 \times I_2$, where the (i_1, i_2)-th entry will be computed to determine the relevance of term w_{i_2} in document d_{i_3} contextually to the document-set CDS_{i_1}.

Given a document d, a term w, and a frequent document-set CDS (we omit here the subscripts for the sake of readability of the following formulas), our aim is to incorporate the following aspects in the term relevance weight:

1. the popularity of the term in the document;
2. the rarity of the term over the collection of documents;
3. the rarity of the term locally to the frequent document-set;
4. the support of the frequent document-set.

Aspects 1 and 2 refer to the notions of *term frequency* and *inverse document frequency* that compose the classic $tf.idf$ term relevance weighting func-

TID	*Transaction*
1	abcdefghilm
2	aghilmnopq
3	abcdefnopq
4	abcghimn

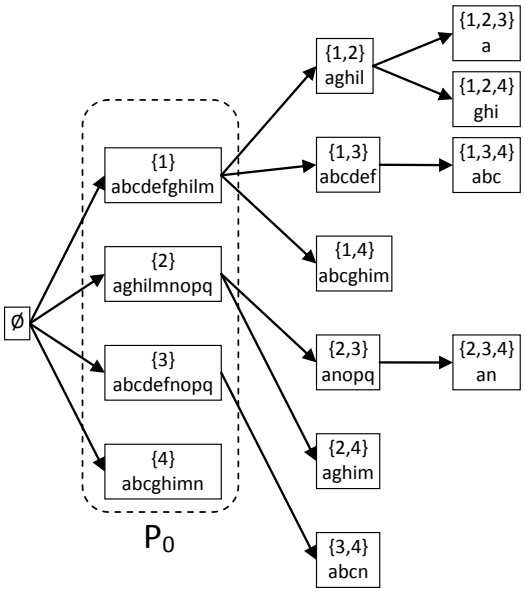


Fig. 3.2: Example of intersection-based closed frequent itemset extraction.

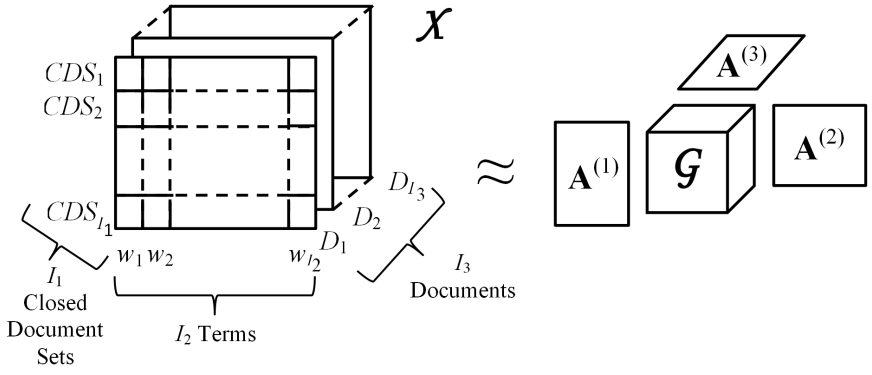


Fig. 3.3: Our tensor model for multiple document organizations.

tion. Formally, the frequency of term w in document d, denoted as $tf(w, d)$, is equal to the number of occurrences of w in d. The inverse document frequency of term w in the document collection is defined as $idf(w) = \log(|\mathcal{D}|/N(w))$, where $N(w)$ is the number of documents in $\mathcal{D}$ that contain w.

To account for aspect 3, we introduce an *inverse document-set frequency* factor:

$$idsf(w, CDS) = \log\left(1 + \frac{|CDS|}{N(w, CDS)}\right)$$

where $|CDS|$ is the number of documents belonging to the frequent document-set CDS, and $N(w, CDS)$ denotes the number of documents in CDS that contain w. Moreover, the $idsf$ weight is defined to be equal to zero if term w is absent in all documents of CDS; otherwise, note that $idsf$ weight is always a positive value even in case of maximum popularity of the term in the frequent document-set.

Finally, to account for aspect 4, we exploit the support of the frequent document-set:

$$s(CDS) = \exp\left(\frac{supp(CDS)}{\max_{CDS \in \mathcal{CDS}} supp(CDS')}\right)$$

where $supp(CDS)$ is the support of CDS, i.e., the number of document groups (clusters) in every $\mathcal{C} \in \mathcal{CS}$ that contain CDS. Note that the support of a document set is bounded by the number of document organizations (i.e., size of $\mathcal{CS}$) in case each document is originally assigned to only one group (cluster), in each of the organizations.

By combining all four factors, the overall term relevance weighting function has the form:

$$weight(CDS, w, d) = tf(w, d)\ idf(w)\ idsf(w, CDS)\ s(CDS)$$

It can be noted that the proposed weighting function increases with the popularity of a term in a document, with the rarity of a term in the document collection, with the rarity of a term in a frequent document-set, and with the support of a closed frequent document-set.

3.3.3 Tensor decomposition

Recently, a particularly effective formulation of NTD that uses a beta divergence has been successfully applied for PCA and clustering, robust ICA and robust NMF/NTF [35]. We chose this type of NTD, known as Fast Beta NTD, to define our tensor decomposition algorithm.

Figure 3.4 shows our modified Fast Beta NTD algorithm for a third-order tensor. In the figure, the symbol $\mathbf{A}^{\otimes -n}$ denotes the Kronecker product between all factor matrices except $\mathbf{A}^{(n)}$, i.e.,

$$\mathbf{A}^{\otimes -n} = \mathbf{A}^{(1)} \otimes \ldots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \ldots \otimes \mathbf{A}^{(N)}$$

The expression $\boldsymbol{\mathcal{G}} \times \{\mathbf{A}\}$ denotes the product $\boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times \ldots \times_N \mathbf{A}^{(N)}$.

The Fast Beta NTD algorithm has multiplicative update rules defined in function of the tensor $\boldsymbol{\mathcal{X}}$ and its current approximation $\hat{\boldsymbol{\mathcal{X}}}$. Unfortunately, $\hat{\boldsymbol{\mathcal{X}}}$ is a large yet dense tensor and hence it cannot be easily kept in primary

Modified Fast Beta NTD Algorithm

Input:
$\boldsymbol{\mathcal{X}}$: input data of size $I_1 \times I_2 \times I_3$,
J_1, J_2, J_3: number of basis for each factor,
β: divergence parameter.

Output:
three factors $\mathbf{A}^{(1)} \in \mathbb{R}_+^{I_1 \times J_1}$, $\mathbf{A}^{(2)} \in \mathbb{R}_+^{I_2 \times J_2}$, $\mathbf{A}^{(3)} \in \mathbb{R}_+^{I_3 \times J_3}$
core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$

begin
1. Nonnegative ALS initialization for all $\mathbf{A}^{(n)}$ and $\boldsymbol{\mathcal{G}}$
2. **repeat**
3. $\hat{\boldsymbol{\mathcal{X}}}' = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)}$
4. $\boldsymbol{\mathcal{X}}'' = computeStep1(\boldsymbol{\mathcal{X}}, \hat{\boldsymbol{\mathcal{X}}}', \mathbf{A}^{(3)}, n, \beta)$ // compute $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}$
5. **for** n = 1 to 3 **do**
6. $\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} \circledast computeStep2(\boldsymbol{\mathcal{X}}'', \mathbf{A}, \boldsymbol{\mathcal{G}}, n) \oslash computeStep3(\hat{\boldsymbol{\mathcal{X}}}', \mathbf{A}, \boldsymbol{\mathcal{G}}, \beta, n)$
7. $\mathbf{a}_{j_n}^{(n)} \leftarrow \mathbf{a}_{j_n}^{(n)} / \parallel \mathbf{a}_{j_n}^{(n)} \parallel_p$
8. **end**
9. $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{G}} \circledast \left[\boldsymbol{\mathcal{X}}'' \times \{\mathbf{A}^T\}\right] \oslash \left[\hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta]} \times \{\mathbf{A}^T\}\right]$
10. **until** *a stopping criterion is met*

end

Fig. 3.4: Modified Fast Beta NTD Algorithm

memory. To avoid this issue, we decompose the tensor following the lead of the approach proposed in [104]. Hereinafter,

Let us consider the update rule for the factor matrices $\mathbf{A}^{\otimes_{-n}}$:

$$\mathbf{A}^{(n)} \leftarrow \mathbf{A}^{(n)} \circledast \left[\left(\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}\right) \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T\right] \oslash \left(\hat{\mathbf{X}}_{(n)}^{\cdot[\beta]} \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T\right) \tag{3.1}$$

In the above rule, the most expensive operations are $\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}$, $\mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ and $\hat{\mathbf{X}}_{(n)}^{\cdot[\beta]} \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$, which clearly rely on the large numbers I_1, I_2, and especially I_3. In order to cope with the computational difficulties of this update rule, we decompose the problem into three smaller steps. In the first step, the product $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$ is computed by taking into account only the nonzero entries of $\boldsymbol{\mathcal{X}}$. In the second step, the product $\left(\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}\right) \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ is computed in a block-wise manner, so that we can control the use of primary memory. The third step, which is required to compute $\hat{\mathbf{X}}_{(n)}^{\cdot[\beta]} \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$, is performed analogously to the second step.

To avoid storing the entire tensor $\hat{\boldsymbol{\mathcal{X}}}$, we keep in memory only an intermediate result $\hat{\boldsymbol{\mathcal{X}}}' = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)}$ (Line 3), and then partially compute the final approximated tensor as $\hat{\boldsymbol{\mathcal{X}}} = \hat{\boldsymbol{\mathcal{X}}}' \times_3 \mathbf{A}^{(3)}$ only for a limited number of slices at time, for each mode.

We observe that the products $\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}$ for $n \in \{1,2,3\}$ are the matricizations of the same tensor element-wise product $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$ hence we can compute such a product once and then obtain the matricizations. Moreover, since $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$ is the element-wise product of a sparse tensor with a dense one, the resulting tensor will also be a sparse tensor whose nonzero entries are in the same positions as those within $\boldsymbol{\mathcal{X}}$. Another aspect that should be considered is that the matricization of a tensor is a matrix in which the slices are placed side by side (cf. Section 2.4.1).

By taking into account the above considerations, in order to compute $\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}$ for $n \in \{1,2,3\}$, we first obtain $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$ by considering only the nonzero entry of $\boldsymbol{\mathcal{X}}$ (Line 4), and hence by computing only the corresponding entry of $\hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$ starting from intermediate result $\hat{\boldsymbol{\mathcal{X}}}$', and then we obtained the matricizations of the resulting tensor. For instance, the matricization along the mode 1 of the tensor resulting from the product $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$ will have the same form of $\mathbf{Y}_{(n)}$ and will be exactly $\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}$, with $n = 1$.

If we consider that $\mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ is exactly the transpose of the matricization along the mode n of the tensor resulting from $\boldsymbol{\mathcal{G}} \times_{-n} \{\mathbf{A}\}$, it can be noted that $\left(\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}\right) \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ is the product between the matricizations along the mode n of two tensors ($\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$ and $\boldsymbol{\mathcal{G}} \times_{-n} \{\mathbf{A}\}$) that have the same number of slices along the mode n. For instance, for $n = 1$, $\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}$ will be formed by I_3 slices ($I_1 \times I_2$) and $\mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ will be formed by I_3 slices ($J_1 \times I_2$). Thus, it can be observed that $\left(\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}\right) \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ will be exactly the sum of I_3 matrix products, such that each slice of $\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}$ will be multiplied with the corresponding slice of $\mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$. If we denote with $\mathbf{L}_{i_3}$ the i_3-th slice that forms $\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}$ and $\mathbf{M}_{i_3}$ the i_3-th slice that forms $\mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$, then $\left(\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}\right) \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T = \sum_{i_3=1}^{I_3} \mathbf{L}_{i_3} \mathbf{M}_{i_3}^T$ and this operation is computed at Line 6 (*computeStep*2()) for each mode. This product, for mode 1, is shown in Figure 3.5

The computation of $\hat{\mathbf{X}}_{(n)}^{\cdot[\beta]} \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ (Line 6, *computeStep*3()) is analogous: $\hat{\mathbf{X}}_{(n)}^{\cdot[\beta]}$ is a matrix in which the slices of $\hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta]}$ are placed side by side, so that $\hat{\mathbf{X}}_{(n)}^{\cdot[\beta]} \mathbf{A}^{\otimes_{-n}} \mathbf{G}_{(n)}^T$ can be rewritten as the sum of a certain number of matrix products.

In the computation of the core tensor update rule (Line 9), the most expensive operation is $\hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta]} \times \{\mathbf{A}^T\}$. In this case we compute a normal mode-n product but each entry of $\hat{\boldsymbol{\mathcal{X}}}^{\cdot\beta}$ is computed starting from the intermediate result $\hat{\boldsymbol{\mathcal{X}}}$'.

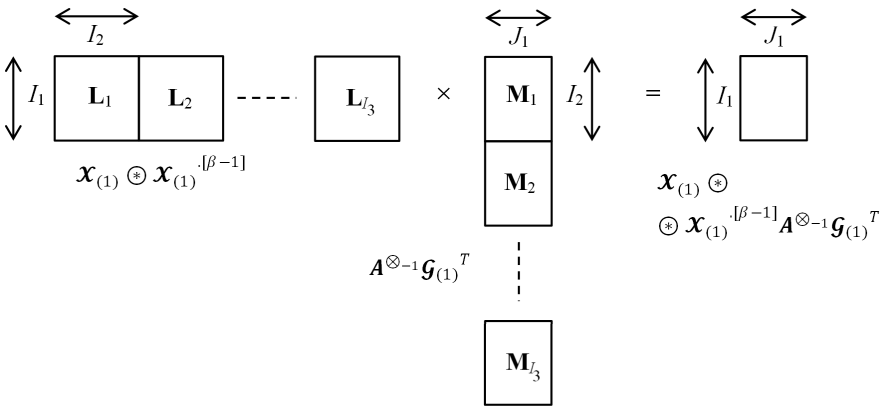


Fig. 3.5: $\left(\mathbf{X}_{(n)} \circledast \hat{\mathbf{X}}_{(n)}^{\cdot[\beta-1]}\right) \mathbf{A}^{\otimes -n} \mathbf{G}_{(n)}^T$ computation.

3.3.4 Induction of document clustering

We consider different ways of inducing a document clustering solution from the decomposed tensor. One simple way is to derive a monothetic clustering from the third factor matrix ($\mathbf{A}^{(3)}$) by assigning each document to the component (cluster) corresponding to the highest relevance value stored in the matrix. A direct way is to input a standard document clustering algorithm with $\mathbf{A}^{(3)}$. An alternative way, which does not explicitly involve $\mathbf{A}^{(3)}$, is to consider a clustering solution obtained by applying a document clustering algorithm to the projection of the matrix of the term-frequencies (over the original document collection) to $\mathbf{A}^{(2)}$—the rationale here is to project the original document vectors of term-frequencies along the mode-2 components, which express discriminative information for the term grouping, hence deriving a clustering of the documents that are mapped to a lower dimensional space. We hereinafter refer to the different ways as *monothetic*, *direct*, and *tf-projected* document clustering, respectively.

3.4 Evaluation and results

Reuters Corpus Volume 1 (RCV1) [114] is a major benchmark for text classification/clustering research, which consists of thousands of newswire stories in XML format. RCV1 lends itself particularly well for our case study since every news, besides its plain-text fields (i.e., body and headlines) is originally provided with alternative categorizations according to three different category fields (metadata): Topics (i.e., major subjects of a news), Industries (i.e., types of businesses discussed), and Regions (i.e., geographic locations as well as economic/political information about a news).

Function computeStep1

Input:
$\boldsymbol{\mathcal{X}}$: input data of size $I_1 \times I_2 \times I_3$,
$\hat{\boldsymbol{\mathcal{X}}}$': intermediate result for $\hat{\boldsymbol{\mathcal{X}}}$computation
$\mathbf{A}^{(3)}$: the third factor matrix of size $I_1 \times J_1$
n: the selected mode
β: the β-divergence coefficient
Output:
$\boldsymbol{\mathcal{X}}$": the tensor resulting from the product $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}$
begin
11. **for each** $x_{i_1 i_2 i_3}$ such that $x_{i_1 i_2 i_3} \neq 0$
12. compute $\hat{x}_{i_1 i_2 i_3}$ starting from $\hat{\boldsymbol{\mathcal{X}}}$'and $\mathbf{A}^{(3)}$
13. set $x''_{i_1 i_2 i_3} = x_{i_1 i_2 i_3} \hat{x}^{\beta-1}_{i_1 i_2 i_3}$
14. **end**
end

Function computeStep2

Input:
$\boldsymbol{\mathcal{X}}$": the tensor resulting from the product $\boldsymbol{\mathcal{X}} \circledast \hat{\boldsymbol{\mathcal{X}}}^{\cdot[\beta-1]}$
$\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}$: the factor matrices
$\boldsymbol{\mathcal{G}}$: the core tensor
n: the selected mode
Output:
$\mathbf{R}$: the matrix resulting from $\left(\boldsymbol{\mathcal{X}}_{(n)} \circledast \hat{\boldsymbol{\mathcal{X}}}^{[\beta-1]}_{(n)}\right) \mathbf{A}^{\otimes_{-n}} \boldsymbol{\mathcal{G}}^T_{(n)}$
begin
15. Let ms be the maximum number of slices that can be loaded into primary memory
16. Let fs be the first slice under consideration
17. Let R be an all-zero matrix of size $I_n \times J_n$
18. **repeat**
19. Let be S_1 sub-tensor built by taking $\boldsymbol{\mathcal{X}}$"slices from the fs-th one to the $(fs+ms)$-th one
20. Let be S_2 sub-tensor built by taking $\boldsymbol{\mathcal{G}} \times_{-n} \{\mathbf{A}\}$ slices from the fs-th one to the $(fs+ms)$-th one
21. $R = R + S_1 S_2^T$
22. **until** *all slices are computed*
end

Function computeStep3

Input:
$\hat{\boldsymbol{\mathcal{X}}}$': the tensor resulting from the product $\boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)}$
$\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}$: the factor matrices
$\boldsymbol{\mathcal{G}}$: the core tensor
β: the β-divergence coefficient
n: the selected mode
Output:
$\mathbf{R}$: the matrix resulting from $\left(\boldsymbol{\mathcal{X}}_{(n)} \circledast \hat{\boldsymbol{\mathcal{X}}}^{[\beta-1]}_{(n)}\right) \mathbf{A}^{\otimes_{-n}} \boldsymbol{\mathcal{G}}^T_{(n)}$
begin
23. Let ms be the maximum number of slices that can be loaded into primary memory
24. Let fs be the first slice under consideration
25. Let R be an all-zero matrix of size $I_n \times J_n$
26. **repeat**
27. Let be S_1 sub-tensor built by taking $\left[\hat{\boldsymbol{\mathcal{X}}}' \times_3 \mathbf{A}^{(3)}\right]^{\cdot\beta}$ slices from the fs-th one to the $(fs+ms)$-th one
28. Let be S_2 sub-tensor built by taking $\hat{\boldsymbol{\mathcal{G}}} \times_{-n} \{\mathbf{A}\}$ slices from the fs-th one to the $(fs+ms)$-th one
29. $R = R + S_1 S_2^T$
30. **until** *all slices are computed*
end

Fig. 3.6: Tensor decomposition functions for the modified Fast Beta NTD algorithm.

From the whole RCV1 collection, we filtered out very short news (i.e., XML documents with size less than 6KB), and any news that did not have at least one value for each of the three category fields. Then we selected the news labeled with one of the Top-5 categories for each of the three category fields. This resulted in a dataset of 3081 news. From the textual components of the

Table 3.1: Document classification sets.

	news fields	*text params*	*clust. params*	*size*
CS1	headline	$lf = 0$	$k \in [5..20]$	4 (50)
	body	$lf = \{0, 1, 5\}$	$k \in [5..20]$	12 (150)
CS2	headline + body	$lf = 5$	$k \in [5..43]$	20 (480)
CS3	headline	$lf = 0$	$k \in [5..20]$	4 (50)
	body	$lf = 0$	$k \in [5..20]$	4 (50)
	metadata	–	–	3 (19)

news, we discarded strings of digits, retained alphanumerical terms, performed removal of stop-words and word stemming (based on Porter's algorithm[2]), and filtered out terms with a document-frequency greater than 50%.

We generated various sets of classifications obtained over the RCV1 dataset, according to the textual content information as well as to the Topics/Industries/Regions metadata. For the purpose of generating the text-based classifications, we used the *bisecting k-means* algorithm implemented in the well-known CLUTO clustering toolkit [91] to produce clustering solutions of the documents represented over the space of the terms contained in the body and/or headlines. Table 3.1 summarizes the main characteristics of the three sets of document classifications used in our evaluation. Columns *text params* and *clust. params* refer to the lower document-frequency cut threshold (lf, percent) used to select the terms for the document representation, and to the number of clusters (k, with increment of 5 in CS1,CS3 and 2 in CS2) taken as input to CLUTO to generate the text-based classifications. Moreover, column *size* reports the number of classifications (and relating groups of documents, within brackets) that rely on the same type of information (i.e., body, headline, metadata).

For each of the three document classification sets, we derived different tensors according to different settings of the closed frequent document-set extraction. More specifically, we varied not only the *minsup* threshold (with increments of 0.1%) but also the minimum length desired for a closed frequent document-set CDS (with increments of 10), until the full coverage of the documents was ensured by the patterns discovered. We observed how the average number of patterns per document varied consequently, and finally we selected up to four configurations that corresponded to the best coverage of patterns per document. Table 3.2 reports on details about the tensors built upon the selected configurations. Note that, in each of the tensors, mode-2 corresponded to the space of terms extracted from the body and headline of the news (2692 terms) and mode-3 to the average number of clusters in the corresponding classification sets (i.e., 13 for CS1, 24 for CS2, and 11 for CS3).

For each of the tensors constructed, we run the algorithm in Figure 3.4 with different settings to obtain two decompositions: the first one led to a core-tensor with a number of components on mode-3 equal to the average

[2] http://www.tartarus.org/∼martin/PorterStemmer/.

Table 3.2: Tensors and their decompositions for the various document classification sets.

	min length of CDS	*no. of CDS*	*avg % of CDS per doc.*	_TD-S *size*	_TD-L *size*
CS1_Ten1	50	17443	3.29%	$174 \times 27 \times 13$	$1740 \times 270 \times 130$
CS1_Ten2	100	5871	5.25%	$58 \times 27 \times 13$	$580 \times 270 \times 130$
CS1_Ten3	150	2454	7.12%	$24 \times 27 \times 13$	$240 \times 270 \times 130$
CS1_Ten4	200	1265	8.53%	$12 \times 27 \times 13$	$120 \times 270 \times 130$
CS2_Ten1	50	12964	3.78%	$129 \times 27 \times 24$	$1290 \times 270 \times 240$
CS2_Ten2	100	7137	4.87%	$71 \times 27 \times 24$	$710 \times 270 \times 240$
CS2_Ten3	150	3129	5.89%	$31 \times 27 \times 24$	$310 \times 270 \times 240$
CS2_Ten4	180	918	7.53%	$9 \times 27 \times 24$	$90 \times 270 \times 240$
CS3_Ten1	50	2806	3.09%	$28 \times 27 \times 11$	$280 \times 270 \times 110$
CS3_Ten2	100	843	5.15%	$8 \times 27 \times 11$	$80 \times 270 \times 110$
CS3_Ten3	150	326	7.15%	$3 \times 27 \times 11$	$30 \times 270 \times 110$

Table 3.3: Results on the various tensor decompositions.

	clustering	F	$Q_{tf.idf}$	Q_{tensor}		*clustering*	F	$Q_{tf.idf}$	Q_{tensor}
CS1_Ten1_TD-S	monoth.	0.509	0.603	0.658	CS1_Ten1_TD-L	direct	0.556	0.601	0.665
	tf-proj.	0.610	0.838	0.891		tf-proj.	0.665	0.881	0.951
CS1_Ten2_TD-S	monoth.	0.534	0.599	0.654	CS1_Ten2_TD-L	direct	0.570	0.603	0.665
	tf-proj.	0.625	0.838	0.889		tf-proj.	0.688	0.884	0.949
CS1_Ten3_TD-S	monoth.	0.542	0.598	0.652	CS1_Ten3_TD-L	direct	0.586	0.601	0.666
	tf-proj.	0.624	0.835	0.886		tf-proj.	0.689	0.889	0.944
CS1_Ten4_TD-S	monoth.	0.533	0.598	0.651	CS1_Ten4_TD-L	direct	0.579	0.605	0.665
	tf-proj.	0.624	0.838	0.887		tf-proj.	0.687	0.837	0.946
CS2_Ten1_TD-S	monoth.	0.494	0.603	0.659	CS2_Ten1_TD-L	direct	0.599	0.604	0.669
	tf-proj.	0.569	0.847	0.898		tf-proj.	0.625	0.893	0.957
CS2_Ten2_TD-S	monoth.	0.496	0.603	0.658	CS2_Ten2_TD-L	direct	0.556	0.601	0.660
	tf-proj.	0.561	0.843	0.892		tf-proj.	0.629	0.889	0.952
CS2_Ten3_TD-S	monoth.	0.495	0.603	0.657	CS2_Ten3_TD-L	direct	0.560	0.604	0.660
	tf-proj.	0.570	0.846	0.894		tf-proj.	0.635	0.895	0.953
CS2_Ten4_TD-S	monoth.	0.497	0.604	0.656	CS2_Ten4_TD-L	direct	0.555	0.602	0.658
	tf-proj.	0.577	0.848	0.895		tf-proj.	0.639	0.890	0.957
CS3_Ten1_TD-S	monoth.	0.556	0.597	0.653	CS3_Ten1_TD-L	direct	0.619	0.600	0.661
	tf-proj.	0.617	0.837	0.890		tf-proj.	0.677	0.888	0.937
CS3_Ten2_TD-S	monoth.	0.556	0.597	0.651	CS3_Ten2_TD-L	direct	0.619	0.599	0.658
	tf-proj.	0.620	0.837	0.888		tf-proj.	0.686	0.839	0.930
CS3_Ten3_TD-S	monoth.	0.553	0.597	0.650	CS3_Ten3_TD-L	direct	0.610	0.596	0.656
	tf-proj.	0.620	0.837	0.886		tf-proj.	0.680	0.887	0.933

number of clusters in the original classification set, whereas the other two modes were set equal to the number of closed document-sets and number of terms, respectively, scaled by a factor of 0.01; the second decomposition was devised to obtain a larger core-tensor with components of each mode equal to an increment of a multiplicative factor of 10 w.r.t. the mode in the core-tensor obtained by the first decomposition. The last group of two columns in Table 3.2 contains details about the tensor decompositions; note that we use suffixes _TD-S and _TD-L to denote the first (smaller) and second (larger) decompositions of a tensor, respectively. Note that, while choosing the number of components for the modes of a tensor would deserve an extensive experimentation (since, in general, there are no available specialized techniques for it), the objective of our evaluation was to observe how the clustering performance varies from a configuration in which the core-tensor mode-3 is chosen as equal to the average number of clusters per classification-set to a configuration in which the core-tensor is proportionally larger on all modes.

From the result of a _TD-S decomposition, we derived a monothetic or, alternatively, a tf-projected clustering solution, with number of clusters equal to

the number of mode-3 components; analogously, from a _TD-L decomposition, we derived a direct or a tf-projected clustering solution (cf. Section 3.3.4).

All clustering solutions were evaluated in terms of both standard *external* and *internal* validity criteria. Precisely, we computed the average *F-measure* (F) between a clustering solution derived from the tensor model and each of the input document classifications. We also computed the inter-cluster similarity as the average of the pair-wise similarities of the cluster centroids, and the intra-cluster similarity as the average of the weighted (by the cluster size) summations of the similarities between the cluster centroid and the documents assigned to the cluster. An overall "quality" score of the clustering is finally given as the difference between the intra-cluster similarity and the inter-cluster similarity; as for the vectorial representation of the documents, we resorted to the original $tf.idf$ representation (based on the text of body plus headline fields of the news) and, alternatively, to the representation derived by averaging the row vectors of a frontal slice of the tensor, for each document—recall that the tensor entries are computed using our proposed weighting function (cf. Section 3.3.2). We will denote the corresponding quality scores as $Q_{tf.idf}$ and Q_{tensor}, respectively. Note also that, when the CLUTO algorithm was used, multiple runs (50) were executed so that an average performance score was finally presented, for any given data setting.

Table 3.3 shows our main experimental results. At a first glance, performance evaluation in terms of Q_{tensor} was generally better than those in terms of $Q_{tf.idf}$, which is not surprising at all since the clusterings were induced by taking into account the term relevance scores computed by tensor-based weighting function instead of the standard $tf.idf$.

Looking at each classification-set tensors, both for the monothetic vs. tf-projected clustering case and the direct vs. tf-projected clustering case, we observed that a lower average percentage of closed document-sets generally led to slightly better performance for classification-sets characterized by conceptually different views (i.e., CS1 and CS3), whereas an inverse tendency occurred for a more homogeneous classification-set (CS2). However, we also observed no significant differences in the overall average performance obtained by varying the number of components in mode-1, which would indicate a relatively small sensitivity of the tensor approximation to the mode-1 (i.e., space of the mined closed document-sets). Also, the F-measure evaluation for the CS3 tensors was comparable or even better than for the other tensors, which would suggest the ability of our tensor model to handle document classification sets which express possibly alternative views (i.e., different content-based views along with metadata-based views).

Comparing the performance of the different types of induced clustering, the tf-projected solutions achieved higher quality than the monothetic clusterings (for the case TD-S) and the direct clusterings (for the case TD-L); this was particularly evident in terms of internal quality scores (gains up to 0.24 for the case TD-S and up to 0.29 for the case TD-L).

A final remark is that the relatively small scores (especially for F) were partly due to a certain overlap of the closed document-sets across the various clusters of a given clustering solution: note that this fact, if on the one hand it reflects the frequent associations of the documents through the multiple classifications, on the other hand it would likely be better captured from a soft clustering scheme and validation.

3.5 Related work

One of the earliest proposals to bring tensor models into text representation is presented in [117]. A corpus of N documents is modeled as a character-level higher-order tensor where each document is represented by an (n-1)-order tensor, if an n-gram model is chosen to capture the word information. HOSVD is applied for dimensionality reduction.

In [102], a tensor model is used to integrate the anchor text information in the well-known HITS method for web hyperlink analysis. Each frontal slice in the tensor corresponds to the adjacency matrix obtained by using only a particular term for the web structure representation.

In the context of scientific publication clustering, [175] introduces a tensor decomposition called Implicit Slice Canonical Decomposition, and used it for grouping publications with multiple similarities. This decomposition is shown to be equivalent to CP and each frontal slice is implicitly stored as the product between two sparse matrices, an object-feature matrix and its transpose. In [54], multiple link types are assumed to build a tensor by stacking the adjacency matrix for each link type to form a three-dimensional array. CP decomposition is used to extract features vectors, and applied for the analysis of publication data where each link type corresponds to a particular similarity measure. [118] proposes an extension of graph clustering based on Tucker-2 model to cluster scientific publications by integrating information on citation-links and lexical similarities of the documents. The method consists of three steps: construction of a similarity tensor from a set of similarity matrices, Tucker-2 truncated decomposition of the similarity tensor, and final partitioning of the obtained optimal subspace for producing the clustering solution.

[210] defines a nonnegative Tucker decomposition for third-order tensors which aims to integrate the subspace identification (i.e., the low-dimensional representation with a common basis for a set of matrices) and the detection of the cluster structure in the data. This model, which can be seen as an extension of Tri-factor NMF, has been applied to author clustering in bibliographic data, image clustering, and image reconstruction.

[110] proposes a tensor to represent structural and content information from XML documents. A set of common sub-trees is extracted from a collection of XML documents and then a clustering method is used to group similar sub-trees. These clusters of common sub-trees represent a dimension of the

third-order tensor being built. The other two dimensions are documents and terms. Each frontal slice corresponds to a single document and each entry represents the number of occurrences of a particular term in a certain cluster of common sub-trees for that document. SVD decomposition is applied to the unfolded tensor and the factorized matrices are used to partition the set of documents.

Comparison with Ensemble Clustering and Multi-view Clustering. We would like to point out that the problem of extracting a single clustering from multiple existing ones is actually not novel. A large corpus of research in advanced data clustering has been developed to address the problem of *consensus clustering*, also known as *ensemble clustering*: given an ensemble of clustering solutions, the goal is to derive a consensus clustering as a (new) clustering by the optimization of a certain objective function which expresses how well any candidate consensus clustering complies with the solutions in the ensemble (see [66] for an overview). In this work we face the problem of extraction of a unique clustering from an available set of multiple clusterings from a different perspective, which relies on an integrated representation of all aspects in the set of clusterings and enables the induction of a single meaningful clustering from the unfolding of the multi-aspect representation. Moreover, our approach relaxes a main assumption in ensemble clustering methods, which limits the optimization of the consensus function to only use information on the object-to-cluster and feature-to-cluster assignments, whereas the feature relevance values are assumed to be unavailable.

It is also worth noticing that the problem addressed in this paper is opposite to what is known as *multi-view clustering* [20, 89], which seeks multiple clusterings in different subspaces of a data space, thus uncovering disparate or alternative clusterings that reflect the different groupings inherent in the data and are also decorrelated (i.e., alternative clusterings under the constraint of orthogonality of the subspaces). No assumption of decorrelation among the existing document organizations is required in our approach.

3.6 Chapter review

This chapter has presented a tensor-based approach to deal with multiple organizations of a document collection in order to produce a unique organization encompassing the available ones. This approach is motivated by many real-world clustering-based applications which are increasingly demanding for taking into account some knowledge about the multi-faceted nature of document collections when performing the clustering task.

Besides considering the usual information on the text-based content of the documents, the key idea was to exploit frequent associations of the documents in the groups across the existing classifications, in order to capture how documents tend to be grouped together orthogonally to the different views.

The main contribution of this study was the definition of a third-order tensor model that takes into account both the document content information and the ways the documents are grouped together across the available classifications. For this purpose, a (closed) frequent itemset mining algorithm (which is suited for dealing with transactional datasets in which the number of items is much larger than the number of transactions) was developed to discover the frequent associations of the documents through the existing classifications. Furthermore, a novel weighting fuction, capable to cope with this knowledge, was defined for the tensor construction.

A third-order tensor for the document collection was defined over both the space of terms and the space of the discovered frequent document-associations, and then it was decomposed to finally establish a unique encompassing clustering of documents. To this purpose, a fast-beta NTD algorithm has been used for the tensor decomposition, whereas different ways of inducing a single document clustering solutions have been shown.

Experiments conducted on a document clustering benchmark have shown the potential of the approach to capture the multi-view structure of existing organizations for a given collection of documents.

4

Tensor-based Clustering for View-Segmented Documents

4.1 Introduction

The study described in this chapter leverages on a common property that many documents have in reflecting a target set of views or aspects already defined over the data. These views can be of different kinds. Views can refer to an *application target*, which is the case of different rating aspects in the review of a certain product or service. For example, in a hotel review, different portions of the textual review typically refer to location, cleanliness, checkin/front desk, business service, etc. Other times, views can refer to user-driven goals, e.g., microblogs posted by the same user. Therefore, a multi-view document is a document that can still be seen in terms of its constituent segments modeling the different views.

Our research stems from an interest in exploring the effect, and presumed benefits, of a combination of document segmentation with a multidimensional data structure, the result being able to capture a segment-grained view-oriented topical representation of the documents. This representation will be eventually used for clustering purposes.

In this chapter we describe a tensor-based document clustering framework that explicitly treats multi-view documents in terms of their constituent, view-based text segments, while representing them under a multidimensional data structure. Important features of the proposed framework are *modularity*, since alternate methods for document clustering and tensor analysis can in principle be applied, and *domain versatility*. Note also that the text segmentation is actually descoupled from the construction of the tensor model, i.e., any text segmentation algorithm can be applied in order to extract topically-cohesive segments (e.g., [190, 152]).

The framework has been evaluated over different real-world scenarios, using document collections with different notions of "view".

Note that the problem we arise in this work is also related to the field of *multi-view clustering* [20], where the goal is to produce a partition of the instances exploiting all the different representations/views describing them.

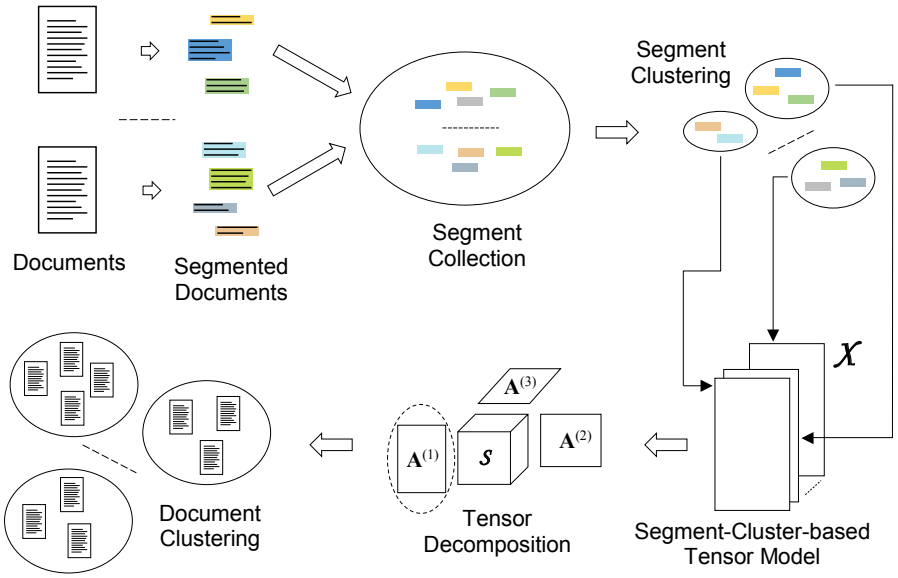


Fig. 4.1: The view-segmented document clustering framework.

Multi-view clustering methods have also demonstrated to be effective on document clustering tasks (e.g., [158, 87]). In particular, co-clustering approaches have shown to be a valuable tool to cluster sparse data. We will hence compare our approach with a recently proposed *multi-view co-clustering* approach which is specifically conceived to deal with text data [87].

The remainder of this chapter is structured as follows. Section 4.2 describe the proposed framework, giving details about the various steps. Then, in Section 4.3 the experimental settings are shown and Section 4.4 illustrates the results. Finally, in Section 4.5, a review of chapter is reported.

4.2 View-segmented document clustering

Given a collection of documents $\mathcal{D} = \{d_i\}_{i=1}^{N}$, we assume that each document is relatively long to be comprised of smaller textual units each of which can be considered cohesive w.r.t. a *view* over the document. The type of view is domain-dependent, and the way views are recognized in a document is supposed either to follow a topic detection approach or to reflect a metadata-level structuring of the documents. In the latter case, metadata can be of logical or descriptive type, such as, e.g., paragraph boundaries, specific subjects of discussion, or user-oriented aspects. We hereinafter refer to the view-oriented parts of a document with the term view-segment, or more simply *segment*.

The proposed view-segmented document clustering framework is shown in Fig. 4.1. The framework can be summarized by the following steps, which are discussed in detail next:

1. Clustering the document segments, or exploiting meta-data to derive a grouping of the segments;
2. Computing a representation of the original document collection for each of the obtained clusters of segments;
3. Computing a third-order tensor for the document collection, upon the segment-cluster based representations.
4. Decomposing the tensor using a Truncated HOSVD;
5. Performing any document clustering algorithm on the mode-1 factor matrix to obtain the final document clustering solution.

Inducing view-segment clusters

We are given a collection of segments $\mathcal{S} = \{s_j\}_{j=1}^{n}$ over $\mathcal{D}$, which can in principle be disjoint or not.

The first step of our framework is in charge of producing a clustering of the segments $\mathcal{C} = \{C_s\}_{s=1}^{k}$, by applying any document clustering algorithm over the segment collection $\mathcal{S}$. The obtained clusters of segments can be disjoint or overlapping.

Segment-cluster based representation

Upon the segment clustering, each document is represented by its segments assigned to possibly multiple segment clusters. Therefore, we derive a document-term matrix for each of the k segment clusters. For this purpose, we define four alternative approaches which are described next.

The basic approach is based on term-frequency information. Given a segment cluster C_s and the relating set of feature terms $\mathcal{F}(C_s)$, the representation of any document d_i in that cluster is defined as a vector of length $|\mathcal{F}(C_s)|$ that results from the sum of the feature vectors of the d_i's segments belonging to C_s; the feature vector of a segment is a vector of term-frequencies. We hereinafter refer to this approach as *TF*. An intuitive refinement of the TF model is to weight the appearance of a document in a cluster based on its segment-based portion covered in the cluster. The weighted TF model (henceforth *WTF*) is thus defined in such a way that the document vector of any d_i for a cluster C_s is multiplied by a scalar representing the portion of d_i's terms that appear in the segments belonging to C_s. Further alternative models can be obtained by normalizing each term-column in the document-term matrix obtained for each cluster via either TF or WTF model. We refer to these models as *NTF* and *NWTF*, respectively.

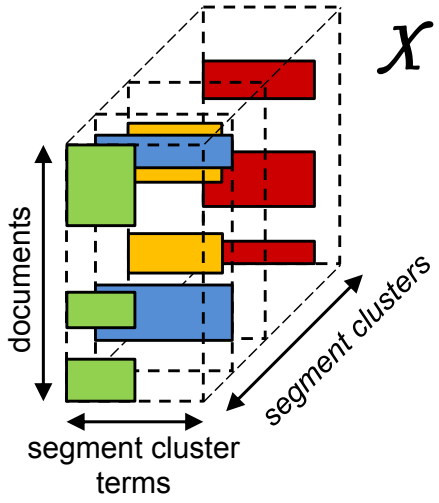


Fig. 4.2: The third-order tensor model for the document collection representation based on the produced view-segment clusters.

Tensor model

The document-term matrices corresponding to the k segment-clusters are used to form a third-order tensor. Recall that a tensor is a multidimensional array $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_D}$, and the number D of dimensions (or modes) is called order of the tensor. A two-dimensional fragment of tensor defined by varying two indices and keeping the rest fixed is a 2-mode tensor called slice.

Our third-order tensor model is built by arranging as frontal slices the k segment-cluster matrices. However, since the segment clusters have possibly different feature subspaces and might cover different subsets of the document collection, the resulting matrices will have a different number of rows/columns. Therefore, in order to build the tensor, each matrix needs to be properly filled with as many zero-valued rows as the number of non-covered documents and with as many zero-valued columns as the number of non-covered feature terms. The resulting tensor will be $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, with $I_1 = |\mathcal{D}|$, $I_2 = \max_{1 \leq s \leq k} |\mathcal{F}(C_s)|$, and $I_3 = k$. The proposed tensor model is sketched in Fig. 4.2.

Tensor decomposition

Although several tensor decomposition algorithms have been proposed [43, 44, 35, 103], most of them have an iterative nature and are designed to reach the best fit in terms norm of the difference, hence, they could be not suitable for clustering task.

The third-order tensor is decomposed through a Truncated Higher Order SVD (T-HOSVD) [43, 44] in order to obtain a low-dimensional representation

Table 4.1: Evaluation datasets.

Dataset	*Type of Document*	*Segment granularity*	*Segment clue*	*# docs*	*# segs*	*fraction of nonzero entries**	*avg # terms per doc*	*avg # terms per seg*
RCV1	news article	paragraph	paragraph boundaries	15,813	128,031	4.6E-4	200	25
TripAdvisor	hotel review	rating aspect	metadata	170,867	810,314	1.1E-3	75	16
Twitter	user's tweets	tweets by hashtag	keywords	9,289	36,763	1.4E-3	147	37

* It refers to the segment-term matrix of the dataset.

of the segment-cluster-based representation of the document collection; for document clustering purposes, we will consider the mode-1 factor matrix. Recall that T-HOSVD is a generalization of SVD [43, 44], as it approximates a tensor into an orthogonal component matrix along each mode and a smaller all-orthogonal and ordered core tensor.

If we denote with r the number of output components for each mode required by T-HOSVD, the decomposed tensor is defined as $\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)}$ (cf. Section 2.4.3). It is worth also noting that the key idea of T-HOSVD is to capture the variation in each of the modes independently from other ones, which makes T-HOSVD particularly appropriate for clustering purposes.

Document clustering

The mode-1 factor matrix is provided in input to a clustering method to obtain a final organization of the documents into K clusters. Note that there is no principled relation between the number K of final document clusters and k, however K is expected to reflect the number of views of interest for the document collection. Also, possibly but not necessarily, the same clustering algorithm used for the segment clustering step can be employed for this step.

4.3 Experimental evaluation

4.3.1 Data

We used three collections of documents that fall into very different application scenarios (Table 4.1). The peculiarities of each dataset prompted us to identify segments at different granularity levels, exploiting different clues.

Scenario 1: Documents with paragraph-based views

We used a subset of the Reuters Corpus Volume 1 (RCV1) [114]. We filtered out very short news (i.e., original XML documents with size less than 3KB) and highly structured news (e.g., lists of stock prices), then we performed tokenization, stopword removal and word stemming. Each paragraph in a

news article was regarded as a segment.[1] We exploited the availability of topic-labels associated with the news articles (i.e., values of Reuters TOPICS field) to sample the original dataset in order to select documents that satisfy certain requirements on the set of covered topics. For this purpose, we followed the lead of a methodology introduced in [151], whereby topic-sets are induced as sets of topic-labels that may overlap, whereas documents are kept organized in disjoint groups. Therefore, the assignment of topic-sets to documents results in a multi-topic, hard classification for the documents in the dataset. Moreover, we kept only the second-level topic-labels in order to ensure that there are no relations of containment between the topic-labels used to form the topic-sets. Upon the evaluation of frequency distribution of each possible set of second-level topic-labels occurring in the documents, we selected only topic-sets having at least two topic-labels and covering at least 1% of documents. Once the topic-sets were extracted, we collected all associated documents. The final dataset was composed of 15,813 documents belonging to 18 topic-sets.

Scenario 2: Documents with metadata-based views

We used the full DAIS TripAdvisor dataset[2], which is a collection of 170,867 hotel reviews. A nice feature of this dataset is that all text reviews are already provided as segmented according to eight rating-aspects, namely "Overall rating", "Value", "Rooms", "Location", "Cleanliness", "Check in/front desk", "Service" and "Business Service".

Scenario 3: Documents with user-driven views

We used a collection of tweets from the Twitter UDI dataset [115]. Our key idea was to consider all tweets of a particular user as a document and to exploit the appearance of hashtags to group related tweets in the user's thread. However, many tweets can have a very few number of words, and the distribution of hashtags over the tweets can be very sparse. Therefore, we imposed constraints on the number of words per tweet and on the number of hashtags per user, in order to reflect the characteristic average length of tweet (around 5.9 in the original dataset) and to ensure high variability in the hashtags utilized by a user. We hence selected the users who posted at least 5 tweets having at least 6 content words, and for which at least 3 hashtags were used in his/her tweets. We then computed the user popularity of hashtags and we selected the top-25 most popular ones. Finally, we collected all tweets related to the users for which the above constraints are still valid. The dimensions of the final dataset are 161,623 tweets related to 9,289 users, with average number of tweets per user of 17.4, average number of hashtags per user of 3.96,

[1] Note that clearly one can resort to text segmentation algorithms to induce segments at sentence level, and we indeed adopted this approach in previous work [190]; however, text segmentation typically requires the setting of several interrelated parameters, whose tuning is not an easy task.

[2] `http://sifaka.cs.uiuc.edu/wang296/Data/index.html`.

and average number of tweets per segment of 4.8. Note also that we found a negligible average degree of segment overlapping (0.064 shared tweets).

Reference classification.

In RCV1, each of the 18 topic-sets is a document class. Similarly, in Twitter, each of the top-25 selected hashtags defines a class, and a user's tweet-document is assigned to the class that corresponds to the most frequently occurring hashtag in the tweets of that user. In TripAdvisor, the reviews are categorized according to the 8 rating aspects. Each review document is assigned to the class that corresponds to the most descriptive aspect of that review, i.e., the aspect described by the largest portion of terms.

4.3.2 Competing methods

We compared our approach with two baseline clustering methods and a multi-view co-clustering algorithm.

In the first baseline method, dubbed *DocClust*, documents were represented by the conventional vector-space model equipped with the tf.idf term relevance weighting scheme. Clustering of the documents was performed by using the *Bisecting K-Means* [183] algorithm, which is widely known to produce high-quality (hard) clustering solutions in high-dimensional, large datasets.

The second baseline method, dubbed *SegClust*, utilizes the same text representation model and clustering scheme as the first baseline, however it applies on the collection of document segments. Once computed the clustering of segments, a document clustering solution is finally induced via majority voting. This approach was first explored in [190] and has shown to improve the final document clustering performance for relatively long multi-topic documents.

The multi-view co-clustering algorithm *CoStar* [87] searches for a solution that maximizes cross-association of the objects given the different views and viceversa. The clustering algorithm is formulated as a Pareto optimization problem and it does not require any parameter as input. Moreover, *CoStar* explores the search space choosing automatically the number of clusters.

4.3.3 Parameter settings and assessment criteria

Our approach requires the setting of two parameters, namely the number k of view-segment clusters, and the number of output components r for each mode by T-HOSVD. We varied k from 2 to $\sqrt{n}$ (n is the total number of segments over D) with increment of 2 and r from 5 to 100 with increment of 5.

We resorted to standard clustering validation criteria, namely F-Measure (FM), Entropy (E), Purity (Pty), and Normalized Mutual Information (NMI). We recall here that a larger (smaller) value is desirable for FM, Pty, and NMI (resp. E) to indicate better clustering quality. The interested reader is referred to [183] and [213] for details on the various assessment criteria.

Table 4.2: Best performance scores on RCV1. The best-performing setup of k and r is reported for each assessment criterion and tensor-slice representation model.

model	FM	k	r	E	k	r	Pty	k	r	NMI	k	r
TF	0.581	10	80	0.412	2	55	0.589	2	55	0.582	2	55
WTF	0.606	2	95	0.424	2	95	**0.602**	2	95	0.568	2	95
NTF	**0.608**	2	50	**0.394**	2	50	**0.602**	2	50	**0.601**	2	35
NWTF	0.566	4	75	0.418	12	70	0.579	4	75	0.582	12	70

Table 4.3: Best performance scores on TripAdvisor. The best-performing setup of k and r is reported for each assessment criterion and tensor-slice representation model.

model	FM	k	r	E	k	r	Pty	k	r	NMI	k	r
TF	0.406	10	5	0.686	8	5	0.496	8	5	0.127	8	5
WTF	0.531	8	5	0.618	6	5	0.578	8	5	0.210	8	5
NTF	0.475	12	10	0.635	6	5	0.540	8	5	0.176	6	5
NWTF	**0.558**	8	5	**0.580**	6	5	**0.606**	8	5	**0.250**	6	5

4.4 Results

Tables 4.2–4.4 report on the best performance results by our method, with corresponding parameter settings, where the number of final document clusters was set equal to the number of dataset-specific reference classes. It can be noted that the tensor-slice representation models with normalization mostly led to higher quality scores. More precisely, NWTF was always the best-performing model on TripAdvisor and Twitter, while NTF led to the best results on RCV1. The latter would hint that in RCV1 the segments are more uniformly distributed along the segment clusters, thus reducing the weighting factor's influence.

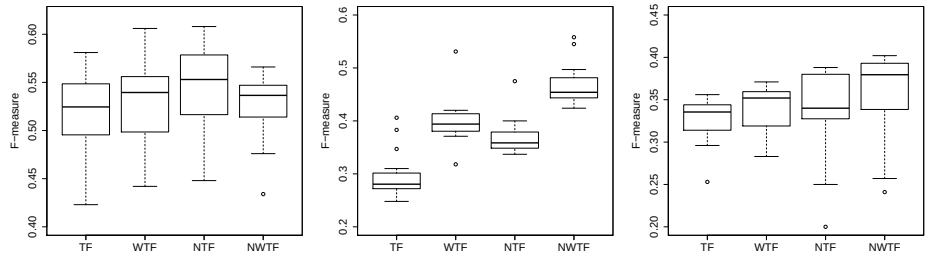
A major remark is that, on all datasets, the best results were consistently achieved by using a quite small number of segment clusters; more precisely, k was mostly below 10, or even equal to the minimum value (i.e., 2). Moreover, in TripAdvisor, the best-performing results also occurred with very few tensor components (i.e., 5). This in general was not the case for the other datasets as well. In this regard, Figures 4.3–4.4 provide more insights by comparing the various tensor-slice representation models. In the figures, the distributions of performance scores are plotted over different numbers of tensor components. It can be noted that NWTF or NTF models generally corresponded to better performance on average. (Only results for FM and E criteria were shown due to space limits, but analogous remarks could be done for the other criteria.)

Comparison with baselines

Table 4.5 compares the best performance obtained by our approach and the two baseline methods, DocClust and SegClust. Our method outperformed the baselines according to all assessment criteria, with gains up to 0.256 FM,

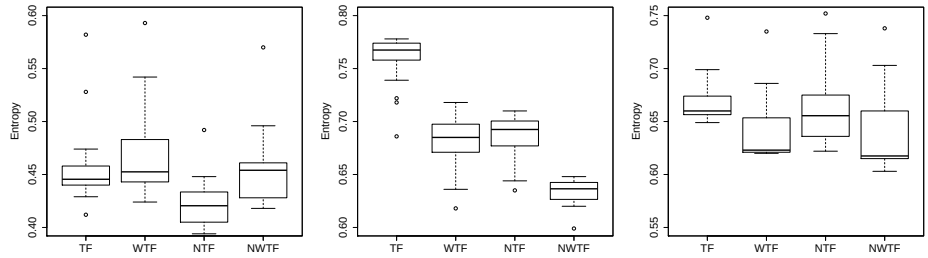
Table 4.4: Best performance scores on Twitter. The best-performing setup of k and r is reported for each assessment criterion and tensor-slice representation model.

model	FM	k	r	E	k	r	Pty	k	r	NMI	k	r
TF	0.356	12	100	0.649	12	100	0.414	12	80	0.294	12	90
WTF	0.371	12	40	0.62	10	35	0.434	12	90	0.333	10	35
NTF	0.388	22	85	0.622	22	100	0.42	22	100	0.328	10	20
NWTF	**0.402**	22	55	**0.603**	20	95	**0.438**	26	80	**0.346**	10	15



(a) RCV1 (b) TripAdvisor (c) Twitter

Fig. 4.3: F-measure distribution over different numbers of tensor components.



(a) RCV1 (b) TripAdvisor (c) Twitter

Fig. 4.4: Entropy distribution over different numbers of tensor components.

0.198 E, 0.247 Pty, and 0.223 NMI. It should be emphasized that the maximum gains achieved by our method corresponded to TripAdvisor, where both the documents and segments are more than in the other datasets, but also shorter on average (cf. Table 4.1). From a qualitative viewpoint, by exploring the cluster descriptions in the form of top-ranked descriptive and discriminating terms [183], we observed an evident ability of our approach to detect clusters that better capture and separate the expected view-based classes. For instance, on TripAdvisor, DocClust and SegClust found clusters that mainly corresponded only to the classes "Overall rating" and "Rooms", while our method was able to discriminate also among the other classes.

Table 4.5: Best performance comparison with baseline algorithms.

criteria	RCV1			TripAdvisor			Twitter		
	DocClust	*SegClust*	our method	*DocClust*	*SegClust*	our method	*DocClust*	*SegClust*	our method
FM	0.504	0.523	**0.608**	0.315	0.302	**0.558**	0.366	0.225	**0.402**
E	0.474	0.467	**0.394**	0.762	0.778	**0.580**	0.659	0.787	**0.603**
Pty	0.501	0.521	**0.602**	0.396	0.359	**0.606**	0.398	0.255	**0.438**
NMI	0.528	0.540	**0.601**	0.045	0.027	**0.250**	0.293	0.135	**0.346**

Comparison with CoStar

As previously discussed in Section 4.3.2, CoStar automatically detects the number of clusters. When applied to our evaluation datasets, CoStar consistently obtained a larger number of clusters (e.g., 50 clusters on Twitter) than the dataset-specific reference classes. Therefore, in order to fairly compare our method with CoStar, we carried out an agglomerative hierarchical clustering method over the partition originally produced by CoStar, cutting the dendrogram at the level corresponding to the right number of reference classes. Moreover, since CoStar has a non-deterministic clustering behavior, we ran it multiple times (50) and hence evaluated each of the final CoStar clustering solutions w.r.t. the reference classification. Upon this, by comparing the best performance scores obtained by CoStar and by our method, we observed a similar behavior on Twitter and TripAdvisor, with marginal improvements by our method (i.e., order of 0.05 or less, for each assessment criterion). However, on RCV1, our method outperformed CoStar, with the following gains: 0.11 FM, 0.10 E, 0.09 Pty, and 0.09 NMI.

We conducted a further experimental session in which we constrained the number of document clusters to be produced by our method as equal to the number of clusters originally produced by CoStar. Using NMI (which is a symmetric evaluation index), we compared each of the multiple CoStar clusterings with those produced by our method, at varying parameter values. Results (here not shown due to space limits of this paper) have revealed a moderate alignment between the two methods, up to around 0.6 on average; also, the best alignment was again consistently achieved by very low values of our method's parameters, (from $k = 6$, $r = 10$ on RCV1 to $k = 8$, $r = 20$ on Twitter).

4.5 Chapter review

In this chapter we have addressed the problem of multi-view document clustering, by proposing a tensor-based clustering framework for view-segmented documents. The framework is designed to exploit a view-based document segmentation into a third-order tensor model, whose decomposition result can enable any standard document clustering algorithm to better reflect the multi-faceted and multi-topic nature of the documents. Experimental results on document collections featuring paragraph-based, metadata-based, or user-driven

views have shown the significance of the proposed approach, highlighting performance improvement in the document clustering task.

5

Tensor-based Semantic Clustering for Multilingual Documents

5.1 Introduction

A major challenge in document clustering research arises from the growing amount of text data that are written in different languages, also due to the increased popularity of a number of tools for collaboratively editing through contributors across the world. *Multilingual document clustering* (MDC) aims to detect clusters in a collection of texts written in different languages. This can aid a variety of applications in cross-lingual information retrieval, including statistical machine translation and corpora alignment.

Existing approaches to MDC rely on structural and linguistic characteristics of the target evaluation corpora. More specifically, research works can be divided in two broad categories, depending on whether a parallel corpus rather than a comparable corpus is used [108]. A *parallel corpus* is typically comprised of documents with their related translations [99]. These translations are usually obtained through machine translation techniques based on a selected anchor language. Within this view, a common approach is to represent documents written in different languages into a common feature space (e.g., the union of language-specific term spaces [201]). This would simplify the clustering task, since any standard document clustering technique could be applied once all documents are made comparable over the same feature space. However, this approach also requires a preliminary translation of all documents in an anchor language, which is time-consuming and error-prone. Conversely, a *comparable corpus* is a collection of multilingual documents written over the same set of classes [141, 207] without any restriction about translation or perfect correspondence between documents. To mine this kind of corpus, external knowledge is employed to map concepts or terms from a language to another [108, 107], which enables the extraction of cross-lingual document correlations. In this case, a major issue lies in the definition of a cross-lingual similarity measure that can fit the extracted cross-lingual correlations. Also, from a semi-supervised perspective, other works attempt to define must-link constraints to detect cross-lingual clusters [207]. This implies that, for each

different dataset, the set of constraints needs to be redefined; in general, the final results can be negatively affected by the quantity and the quality of involved constraints [42].

To the best of our knowledge, existing clustering approaches for comparable corpora are customized for a small set (two or three) of languages [130]. Most of them are not generalizable to many languages as they employ bilingual dictionaries and the translation is performed sequentially considering only pairs of languages. Therefore, the order in which this process is done can seriously impact the results. Another common drawback concerns the way most of the recent approaches perform their analysis: the various languages are analyzed independently of each other (possibly by exploiting external knowledge like Wikipedia to enrich documents [108, 107]), and then the language-specific results are merged. This two-step analysis however may fail in profitably exploiting cross-language information from the multilingual corpus.

We address the problem of MDC by proposing a framework that features three key elements, namely: (1) to model documents over a unified conceptual space, with the support of a large-scale multilingual knowledge base; (2) to decompose the multilingual documents into topically-cohesive segments; and (3) to describe the multilingual corpus under a multidimensional data structure.

The first key element prevents loss of information due to the translation of documents from different languages to a target one. It enables a conceptual representation of the documents in a language-independent way preserving the content semantics. BabelNet [139] is used as multilingual knowledge base. To the extent of our knowledge, this is the first work in MDC that exploits BabelNet.

The second key element, document segmentation, enables us to simplify the document representation according to their multi-topic nature. Previous research has demonstrated that a segment-based approach can significantly improve document clustering performance [190]. Moreover, the conceptual representation of the document segments enables the grouping of linguistically different (portions of) documents into topically coherent clusters.

The latter aspect is leveraged by the third key element of our proposal, which relies on a tensor-based model [103] to effectively handle the high dimensionality and sparseness in text. Tensors are considered as a multi-linear generalization of matrix factorizations, since all dimensions or modes are retained thanks to multi-linear structures which can produce meaningful components. The applicability of tensor analysis has recently attracted growing attention in information retrieval and data mining, including document clustering (e.g., [118, 162]) and cross-lingual information retrieval (e.g., [32]).

The rest of the chapter is organized as follows. We describe our proposal in Section 5.2. Data and experimental settings are described in Section 5.3, while results are presented in Section 5.4. We summarize our main findings in Section 5.5, finally Section 5.6 concludes the chapter.

Algorithm 5 *SeMDocT* (**Se**gment-based **M**ultiLingual **Doc**ument Clustering via **T**ensor Modeling)

Input: A collection of multilingual documents $\mathcal{D}$, the number k of segment clusters, the number r of tensorial components.
Output: A document clustering solution $\mathcal{C}$ over $\mathcal{D}$.
1: Apply a text segmentation algorithm over each of the documents in $\mathcal{D}$ to produce a collection of document segments $\mathcal{S}$.
2: Represent $\mathcal{S}$ in either a bag-of-words (BoW) or a bag-of-synsets (BoS) space.
3: Apply any document clustering algorithm on $\mathcal{S}$ to obtain a segment clustering $\mathcal{C}^{\mathcal{S}} = \{C_i^s\}_{i=1}^k$.
4: Represent $\mathcal{C}^{\mathcal{S}}$ in either a bag-of-words (BoW) or a bag-of-synsets (BoS) space.
5: Model $\mathcal{S}$ as a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, with $I_1 = |\mathcal{D}|$, $I_2 = |\mathcal{F}|$, and $I_3 = k$.
6: Decompose the tensor using a Truncated HOSVD.
7: Apply a document clustering algorithm on the mode-1 factor matrix to obtain the final clusters of documents $\mathcal{C} = \{C_i\}_{i=1}^K$.

5.2 Proposed approach

5.2.1 Multilingual document clustering framework

We are given a collection of multilingual documents $\mathcal{D} = \bigcup_{l=1}^{L} \mathcal{D}_l$, where each $\mathcal{D}_l = \{d_{l_i}\}_{i=1}^{N_l}$ represents a subset of documents written in the same language, with $N = \sum_{l=1}^{L} N_l = |\mathcal{D}|$. Our framework can be applied to any multilingual document collection regardless of the languages, and can deal with balanced as well as unbalanced corpora. Therefore, no restriction is given on both the number L of languages and the distribution of documents over the languages (i.e., $N_i \lessgtr N_j$, with $i, j = 1..L$, $i \neq j$).

Real-world documents often span multiple topics. We assume that each document in $\mathcal{D}$ is relatively long to be comprised of smaller textual units, or *segments*, each of which can be considered cohesive w.r.t. a topic over the document. This represents a key aspect in our framework as it enables the use of a *tensor model* to conveniently address the multi-faceted nature of the documents.

Our overall framework, named *SeMDocT* (**Se**gment-based **M**ultiLingual **Doc**ument Clustering via **T**ensor Modeling), is shown in Algorithm 5. In the following, we shall describe in details each of the steps involved in *SeMDocT*.

Computing within-document segments

Text segmentation is concerned with the fragmentation of an input text into multi-paragraph, contiguous and disjoint blocks that represent subtopics. Regardless of the presence of logical structure clues in the document, linguistic

criteria [18] and statistical similarity measures [79, 33, 37] have been mainly used to detect subtopic boundaries between segments. A common assumption is that terms that discuss a subtopic tend to co-occur locally, and a switch to a new subtopic is detected by the ending of co-occurrence of a given set of terms and the beginning of the co-occurrence of another set of terms.

Our *SeMDocT* does not depend on a specific algorithmic choice to perform text segmentation; in this work, we refer to the classic *TextTiling* [79], which is the exemplary similarity-block-based method for text segmentation.

Inducing document segment clusters

The result of the previous step is a collection of document segments, henceforth denoted as $\mathcal{S}$. Each segment in $\mathcal{S}$ is represented as a vector of feature occurrences, where a feature can be either *lexical* or *semantic*. This corresponds to two alternative representation models: the standard *bag-of-words* (henceforth BoW), whereby features correspond to lemmatized, non-stopword terms, and the obtained feature space results from the union of the vocabularies of the different languages; and *bag-of-synsets* (henceforth BoS), whereby features correspond to BabelNet synsets. We shall devote Section 5.2.2 to a detailed description of our proposed BoS representation.

The segment collection $\mathcal{S}$ is given in input to a document clustering algorithm to produce a clustering of the segments $\mathcal{C}^{\mathcal{S}} = \{C_i^s\}_{i=1}^k$. The obtained clusters of segments can be disjoint or overlapping. Again, our *SeMDocT* is parametric to the clustering algorithm as well; here, we resort to a state-of-the-art clustering algorithm, namely *Bisecting K-Means* [183], which is widely known to produce high-quality (hard) clustering solutions in high-dimensional, large datasets [211]. Note however that it requires as input the number of clusters. To cope with this issue, we adopt the method described in [172], which explores how the within-cluster cohesion changes by varying the number of clusters. The number of clusters for which the slope of the plot changes drastically is chosen as a suitable value for the clustering algorithm.

Segment-cluster based representation

Upon the segment clustering, each document is represented by its segments assigned to possibly multiple segment clusters. Therefore, we derive a document-feature matrix for each of the k segment clusters. The features correspond either to the BoW or BoS model, according to the choice made for the segment representation.

Let us denote with $\mathcal{F}$ the feature space for all segments in $\mathcal{S}$. Given a segment cluster C^s, the corresponding document-feature matrix is constructed as follows. The representation of each document $d \in \mathcal{D}$ w.r.t. C^s is a vector of length $|\mathcal{F}|$ that results from the sum of the feature vectors of the d's segments belonging to C^s. Moreover, in order to weight the appearance of a document in a cluster based on its segment-based portion covered in the cluster, the

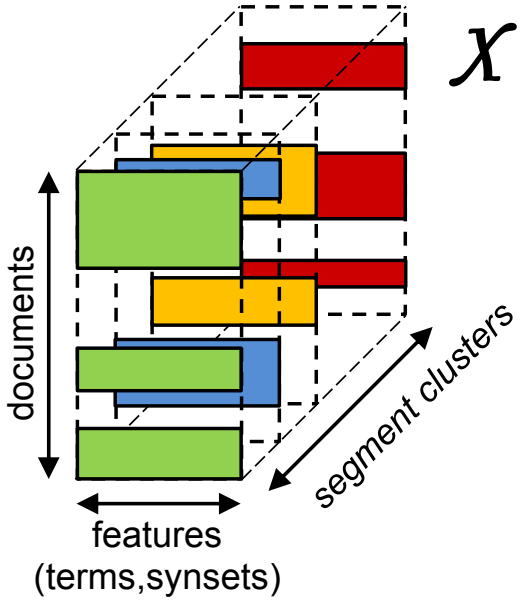


Fig. 5.1: The third-order tensor model for the representation of a multilingual document collection based on segment clusters.

document vector of d w.r.t. C^s is finally obtained by multiplying the sum of the segment-vectors by a scalar representing the portion of d's features that appear in the segments belonging to C^s. The document-feature matrix of C^s resulting from the previous step is finally normalized by column.

Tensor model and decomposition

The document-feature matrices corresponding to the k segment-clusters are used to form a third-order tensor.

Our third-order tensor model is built by arranging as frontal slices the k segment-cluster matrices. The resulting tensor will be of the form $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, with $I_1 = |\mathcal{D}|$, $I_2 = |\mathcal{F}|$, and $I_3 = k$. The proposed tensor model is sketched in Fig. 5.1.

The resulting tensor is decomposed through a Truncated Higher Order SVD (T-HOSVD) [43] in order to obtain a low-dimensional representation of the segment-cluster-based representation of the document collection. For further details about T-HOSVD, please refer to Section 2.4.3.

The ability of T-HOSVD in effectively capturing the variation in each of the modes independently from the other ones, is particularly important to alleviate the problem of concentration of distances, thus making T-HOSVD well-suited to clustering purposes. In this work, in order to obtain a final

clustering solution of the documents, we will consider the mode-1 factor matrix $\mathbf{A}^{(1)}$ of the T-HOSVD.

Document clustering

The mode-1 factor matrix is provided in input to a clustering method to obtain a final organization of the documents into K clusters, i.e., $\mathcal{C} = \{C_i\}_{i=1}^{K}$. Note that there is no principled relation between the number K of final document clusters and k. However, K is expected to reflect the number of topics of interest for the document collection. Also, possibly but not necessarily, the same clustering algorithm used for the segment clustering step (i.e., Bisecting K-Means) can be employed for this step.

5.2.2 Bag-of-synset representation

In the BoS model, our objective is to represent the document segments in a conceptual feature space instead of the traditional term space. Since we deal with multilingual documents, this task clearly relies on the multilingual lexical knowledge base functionalities of BabelNet. Conceptual features will hence correspond to BabelNet synsets.

The segment collection $\mathcal{S}$ is subject to a two-step processing phase. In the first step, each segment is broken down into a set of lemmatized and POS-tagged sentences, in which each word is replaced with related lemma and associated POS-tag. Let us denote with $\langle w, POS(w)\rangle$ a lemma and associated POS-tag occurring in any sentence sen of the segment. In the second step, a WSD method is applied to each pair $\langle w, POS(w)\rangle$ to detect the most appropriate BabelNet synset σ_w for $\langle w, POS(w)\rangle$ contextually to sen. The WSD algorithm is carried out in such a way that all words from all languages are disambiguated over the same concept inventory, producing a language-independent feature space for the whole multilingual corpus. Each segment is finally modeled as a $|\mathcal{BS}|$-dimensional vector of BabelNet synset frequencies, being $\mathcal{BS}$ the set of retrieved BabelNet synsets.

As previously discussed in Section 2.3.3, BabelNet provides WSD algorithms for multilingual corpora. More specifically, the authors in [140] suggest to use the Degree algorithm [138], as it showed to yield highly competitive performance in a multilingual context as well. Note that the Degree algorithm, given a semantic graph for the input context, simply selects the sense of the target word with the highest vertex degree. Clearly, other graph-based methods for (unsupervised) WSD, particularly PageRank-style methods (e.g., [124, 6, 206, 191]), can be plugged in to address the multilingual WSD task based on BabelNet. An investigation of the performance of existing WSD algorithms for a multilingual context is however out of the scope of this chapter.

RCV2 Topics	*English*	*French*	*Italian*
Balanced Corpus			
C15 - PERFORMANCE	850	850	850
C18 - OWNERSHIP CHANGES	850	850	850
E11 - ECONOMIC PERFORMANCE	850	850	850
E12 - MONETARY/ECONOMIC	850	850	850
M11 - EQUITY MARKETS	850	850	850
M13 - MONEY MARKETS	850	850	850
Total	5 100	5 100	5 100
Unbalanced Corpus			
C15 - PERFORMANCE	850	850	0
C18 - OWNERSHIP CHANGES	850	850	0
E11 - ECONOMIC PERFORMANCE	0	850	850
E12 - MONETARY/ECONOMIC	850	0	850
M11 - EQUITY MARKETS	0	850	850
M13 - MONEY MARKETS	850	0	850
Total	3 400	3 400	3 400

Table 5.1: Number of documents for each topic and language.

5.3 Evaluation methodology

In order to evaluate our proposal we need a multilingual comparable document collection with annotated topics. For this reason, we used *Reuters Corpus Volume 2* (RCV2), a multilingual corpus containing news articles in thirteen language.[1] In the following, we present the corpus characteristics and competing methods used in our analysis.

5.3.1 Data preparation

We consider a subset of the RCV2 corpus corresponding to three languages: *English*, *French* and *Italian*. It covers six different topics, i.e., different labels of the RCV2 TOPICS field. Topics are chosen according with their coverage in the different languages. The language-specific documents were lemmatized and POS-tagged through the Freeling library [142] in order to obtain a suitable representation for the WSD process.

To assess the robustness of our proposal, we design two different scenarios. The first (*Balanced Corpus*) is characterized by a completely balanced dataset. Each language covers all topics and for each pair language/topic the same number of documents is selected. The second scenario corresponds to an *Unbalanced Corpus*. Starting from the balanced corpus, we removed for each topic all the documents belonging to one language. In this way, we obtained a corpus in which each topic is covered by only two of the three languages.

Main characteristics of both evaluation corpora are reported in Table 5.1 and Table 5.2. In the latter table, we report the number of documents, number of terms, number of synsets and the dataset density for both representations. To quantify the density of each corpus/representation combination, we

[1] http://trec.nist.gov/data/reuters/reuters.html

Statistics	*Balanced Corpus*	*Unbalanced Corpus*
# of docs	15 300	10 200
# of terms	58 825	44 535
# of synsets	16 395	14 339
BoW Density	1.5×10^{-3}	2.0×10^{-3}
BoS Density	2.6×10^{-3}	3.1×10^{-3}

Table 5.2: Main characteristics of the corpora.

RCV2 Topics	*English*	*French*	*Italian*
C15 - PERFORMANCE	3.41	3.67	3.27
C18 - OWNERSHIP CHANGES	3.20	3.32	2.40
E11 - ECONOMIC PERFORMANCE	4.89	3.17	2.07
E12 - MONETARY/ECONOMIC	5.22	3.69	2.05
M11 - EQUITY MARKETS	4.29	2.94	2.15
M13 - MONEY MARKETS	3.31	3.12	2.10

Table 5.3: Average number of document segments, for each topic and language.

	English		*French*		*Italian*	
RCV2 Topics	avg *BoS* seg. leng.	avg *BoW* seg. leng.	avg *BoS* seg. leng.	avg *BoW* seg. leng.	avg *BoS* seg. leng.	avg *BoW* seg. leng.
C15	21.76	36.32	11.54	34.92	10.58	37.75
C18	20.94	36.87	10.94	35.62	11.24	41.20
E11	22.90	37.24	11.47	34.73	11.96	38.60
E12	22.70	37.70	11.50	37.44	12.59	43.63
M11	22.04	36.83	10.91	32.76	11.57	42.39
M13	22.22	36.97	11.34	34.75	11.72	39.36

Table 5.4: Average length of document segment in the BoW and BoS spaces, for each topic and language.

counted the non-zero entries of the induced document-synset matrix (alternatively, document-term matrix) and we divided this value by the size of such matrix. This number provides an estimate about the density/sparseness of each dataset. Lower values indicate more sparse data. We can note that BoS model yields more dense datasets for both *Balanced Corpus* and *Unbalanced Corpus*.

As our proposal explicitly models document segments, we also report statistics, considering both topics and languages, related to the average number of segments per document (Table 5.3), and the average length of segments per document (Table 5.4). The latter statistic is computed separately for BoW and BoS representations. We made this distinction because a term cannot have a mapping to a synset, or it can be mapped to more than one synset in the BoS space during the WSD process (Section 5.2.2).

Looking at the average number of segments per document in Table 5.3, it can be noted that English documents contain, for all topics, a larger number of segments. This means that English documents are generally richer than the ones in the other languages. Italian language corresponds to the smallest documents, each of them containing between 2 and 3.2 segments on average. A sharper difference appears in the *MONETARY/ECONOMIC* topic for which English documents contain 5.2 segments, while the Italian ones are composed, on average, by only 2 segments.

Table 5.4 shows the average length of segments per document for both space representations. Generally, segments in the BoS representation are smaller than the corresponding segments in the BoW space. More in detail, if we consider the ratio between the segment length in BoS and the one in BoW, this ratio is around 2/3 for the English language, while for both French and Italian it varies between 1/4 and 1/3. This disequilibrium is induced by the multilingual concept coverage of BabelNet, as stated by its authors [139], [140]. In particular, the WSD process tightly depends from the concept coverage supplied from the language-specific knowledge base.

5.3.2 Competing methods and settings

We compare our *SeMDocT* with two standard approaches, namely *Bisecting K-Means* [183], and *Latent Semantic Analysis (LSA)*-based document clustering (for short, *LSA*). Given a number K of desired clusters, *Bisecting K-Means* produces a K-way clustering solution by performing a sequence of K-1 repeated bisections based on standard K-Means algorithm. This process continues until the number K of clusters is found. *LSA* performs a decomposition of the document collection matrix through Singular Value Decomposition in order to extract a more concise and descriptive representation of the documents. After this step, *Bisecting K-Means* is applied over the new document space to get the final document clustering.

All the three methods, *SeMDocT*, *Bisecting K-Means* and *LSA* are coupled with either BoS or BoW representation models. The comparison between BoS and BoW representations allows us to evaluate the presumed benefits that can be derived by exploiting synsets instead of terms for the multilingual document clustering task.

Both *SeMDocT* and *LSA* require the number of components as input; as concerns specifically *SeMDocT*, we varied r_1 (cf. Section 5.2.1) from 2 to 30, with increments of 2. To determine the number of segment clusters k, we employed an automatic way as discussed in Section 5.2.1. By varying k from 2 to 40, for *Balanced Corpus* and *Unbalanced Corpus*, respectively, the values of k obtained were 22 and 23 under BoS, and 25 and 11 under BoW.

As concerns the step of text segmentation, TextTiling requires the setting of some interdependent parameters, particularly the size of the text unit to be compared and the number of words in a token sequence. We used the setting suggested in [79] and also confirmed in [190], i.e., 10 for the text unit size and 20 for the token-sequence size.

Performance of the different methods are evaluated using two standard clustering validation criteria, namely *F-Measure* and *Rand Index*.

Note that for each method, results were averaged over 30 runs and the number of final document clusters K was set equal to the number of topics in the document collection (i.e., 6).

5.4 Results

We present here our main experimental results. We first provide a comparative evaluation of our *SeMDocT* with the competing methods, on both balanced and unbalanced corpus evaluation cases. Then we provide a per language analysis focusing on *SeMDocT*.

5.4.1 Evaluation with competing methods

Evaluation on balanced corpus

Figure 5.2 shows FM and RI results obtained by the various methods coupled with the two document representations on the *Balanced Corpus*. Several remarks stand out. First, the BoS space positively influences the performance of all the employed approaches. This is particularly evident for *Bisecting K-Means* and *LSA* that clearly benefit from this kind of representation. The former almost doubles its performance in terms of FM and significantly improves its result w.r.t. RI. *LSA* shows improvements in both cases. *SeMDocT*-BoS generally outperforms all the competitors for both FM and RI when the number of components is greater than 16. Note that, under the BoW model, *SeMDocT*-BoW still outperforms the other methods.

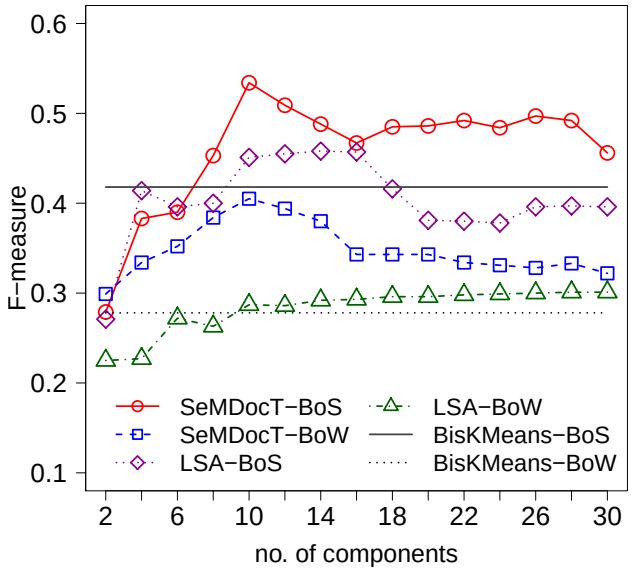
Evaluation on unbalanced corpus

Figure 5.3 reports results for the *Unbalanced Corpus*. Also in this evaluation, the best performances for all the methods are reached using the BoS representation. *SeMDocT*-BoS shows similar behavior according to the two measures. It always outperforms the competitors considering a number of components greater than or equal to 12. More precisely, *SeMDocT*-BoW obtains a gain of 0.047 and 0.103 in terms of FM and 0.006 and 0.058 in terms of RI, w.r.t. *LSA*-BoW and *Bisecting K-Means*-BoW, respectively. Similarly, *SeMDocT*-BoS obtains improvements of 0.05 in terms of FM w.r.t. both BoS competitors, while in terms of RI the differences in performance are 0.012 and 0.019 for *LSA*-BoS and *Bisecting K-Means*-BoS, respectively.

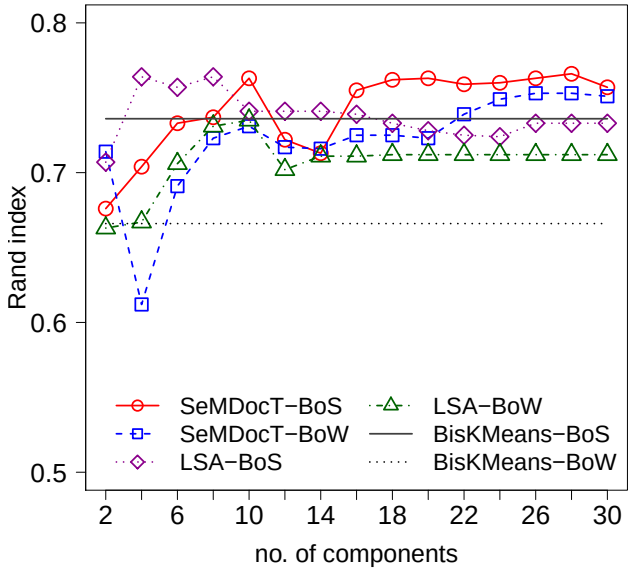
5.4.2 Per language evaluation of *SeMDocT*-BoS

Starting from the clustering solutions produced by *SeMDocT*-BoS in both balanced and unbalanced cases, for each language we extracted a language-specific projection of the clustering. After that, we computed the clustering validation criteria according to language specific solutions to quantify how well the clustering result fits each specific language. The results of this experiment are reported in Fig. 5.4 and Fig. 5.5.

On the *Balanced Corpus*, *SeMDocT*-BoS shows comparable performance for English and French documents, while it behaves slightly worse for Italian



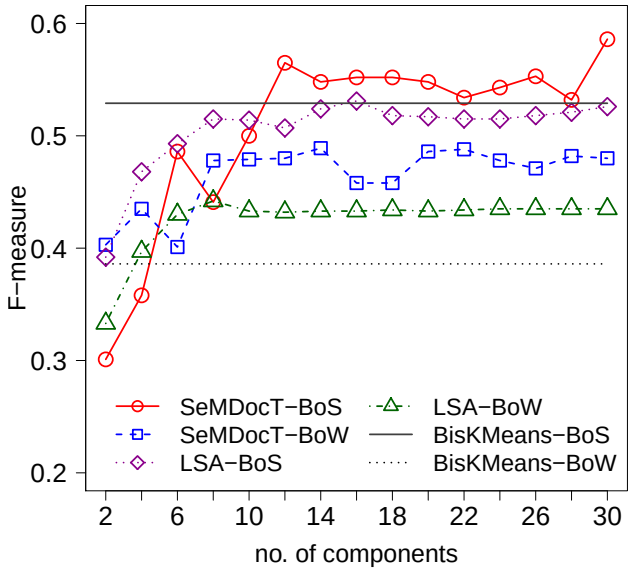
(a)



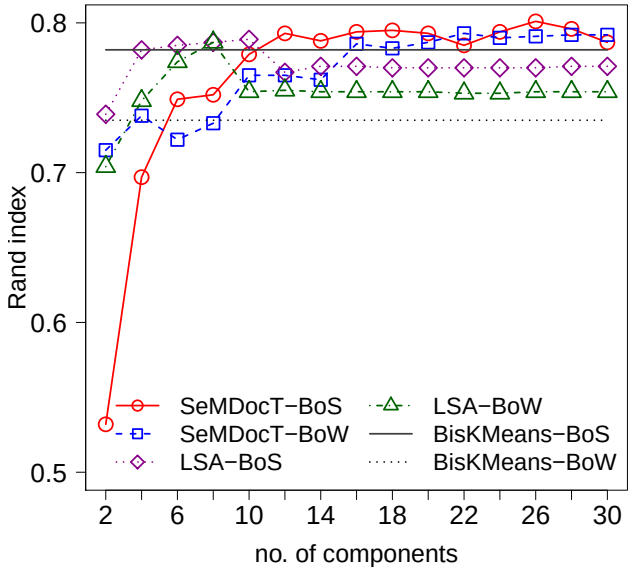
(b)

Fig. 5.2: Average F-Measure (a) and Rand Index (b) on the *Balanced Corpus* using BoW and BoS document representation and varying the number of components for both *SeMDocT* and *LSA*.

texts. This trend is highlighted for both clustering evaluation criteria. In-



(a)



(b)

Fig. 5.3: Average F-Measure (a) and Rand Index (b) on the *Unbalanced Corpus* using BoW and BoS document representation and varying the number of components for both *SeMDocT* and *LSA*.

specting the results for the *Unbalanced Corpus*, we observe a different trend.

Dataset	*Language*	*BoW size*	*BoS size*	*avg # synsets per term* (β)
Balanced	English	29 999	12 065	0.4021
	French	17 826	5 310	0.2978
	Italian	16 951	4 471	0.2637
Unbalanced	English	19 432	10 387	0.5345
	French	14 439	4 431	0.3068
	Italian	14 743	4 012	0.2721

Table 5.5: Balanced corpus: language statistics.

Results obtained for the English texts are generally better than the results for the French and Italian documents. For this benchmark, *SeMDocT*-BoS obtains similar results for documents written in French and in Italian.

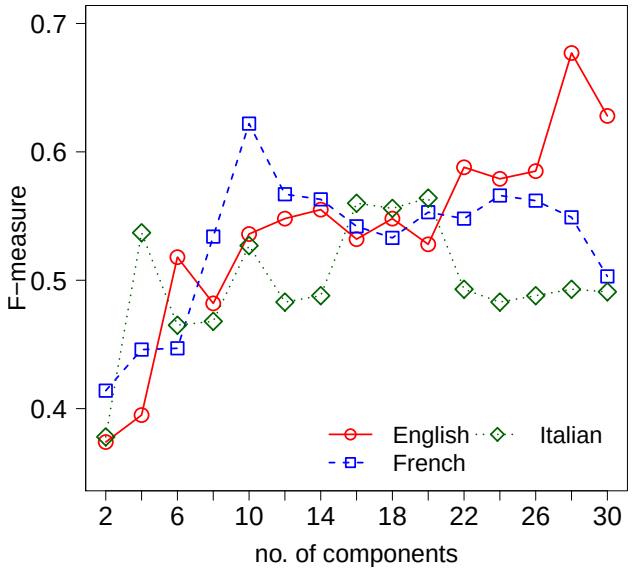
We gained an insight into the above discussed performance behaviors by computing some additional statistics that we report in Table 5.5: for each language and each dataset, the size of the term and synset dictionaries and the average number of synsets per lemma (β) we retrieved with BabelNet according to the related corpus. More in detail, β is the ratio between the BoS and the BoW dictionaries. This quantity roughly evaluates how many synsets are produced per term during the multilingual WSD process (Section 5.2.2). As we can observe, this value is always smaller than one, which means that not all the terms have a corresponding mapping to a synset. The β ratio can explain the discrepancy in (language-specific) performances in the two scenarios. In particular, the difference in the β statistic between English and the other languages is more evident for the *Unbalanced Corpus* (i.e., 0.23 between English and French), while it is lower for the *Balanced Corpus* (around 0.1). The relatively large gap in β between the first and the second language (respectively, English and French) for the *Unbalanced Corpus* reduces the relative gap between the second and the third languages (respectively, French and Italian) while this trend is less marked for the *Balanced Corpus* as β range is narrower. In summary, we can state that our framework works well if BabelNet knowledge base provides a good coverage of the terms in the analyzed language. Experimental evidence shows that, if this condition is met, *SeMDocT*-BoS provides better clustering results w.r.t. the competing approaches.

5.4.3 Runtime of tensor decomposition

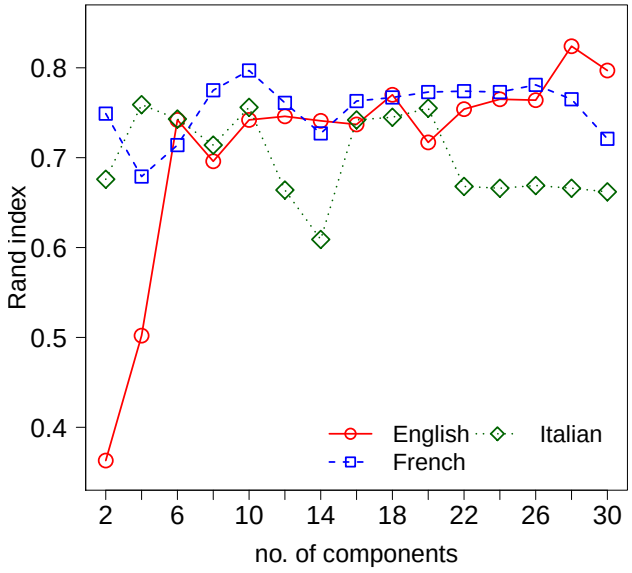
As previously discussed, T-HOSVD of a third-order tensor can be computed through three standard SVDs. Furthermore, for clustering purposes, we considered only the mode-1 factor matrix of the decomposition. To compute the SVD, we used the *svds()* function of MATLAB R2012b, which is based on an iterative algorithm.[2] Experiments were carried out on an Intel Core I7-3610QM platform with 16GB DDR RAM.

Figure 5.6 shows the execution time of the SVD over the mode-1 matricization of our tensor for the *Balanced Corpus*, by varying the number

[2] http://www.mathworks.it/it/help/matlab/ref/svds.html



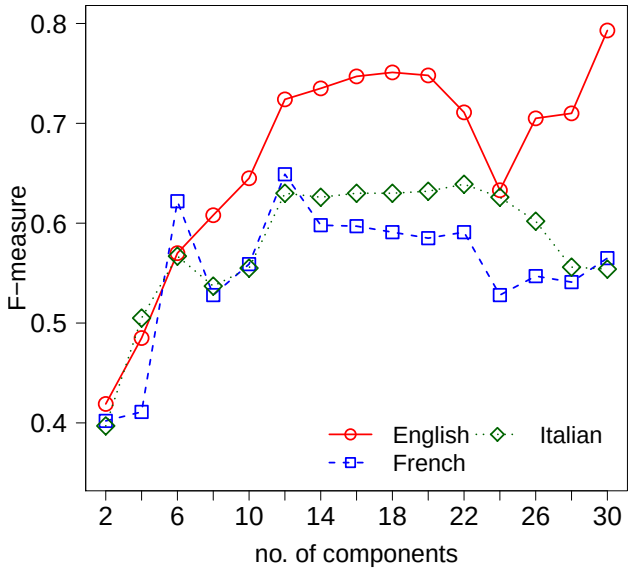
(a)



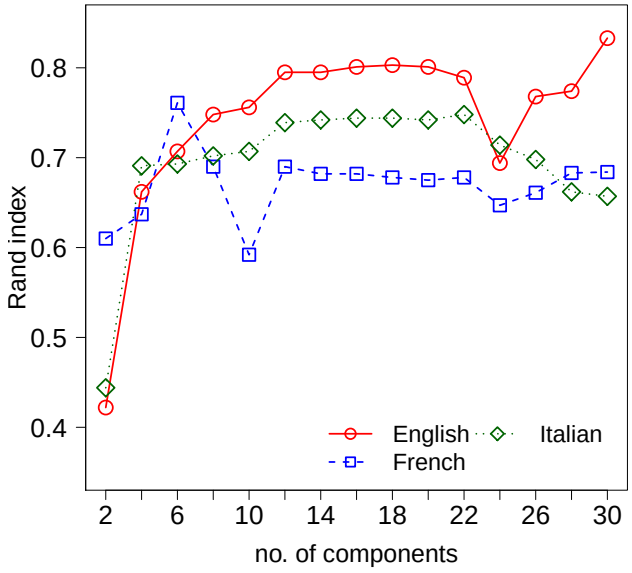
(b)

Fig. 5.4: Average F-Measure (a) and Rand Index (b) for language specific solutions on the *Balanced Corpus* obtained by *SeMDocT*-BoS.

of components, for both BoW and BoS representation models. As it can be observed, in both cases the runtime is linear in the number of components.



(a)



(b)

Fig. 5.5: Average F-Measure (a) and Rand Index (b) for language specific solutions on the *Unbalanced Corpus* obtained by *SeMDocT*-BoS.

However, the SVD computation in the BoS setting is one order of magnitude faster than time performance in the BoW setting. This is mainly due to a large

difference in size between the feature spaces of BoW and BoS (cf. Table 5.2), since the selected number of segment clusters (k) was nearly the same (25 for BoW, and 22 for BoS). Therefore, by providing a more compact feature space, BoS clearly allows for a much less expensive SVD computation for our tensor decomposition.

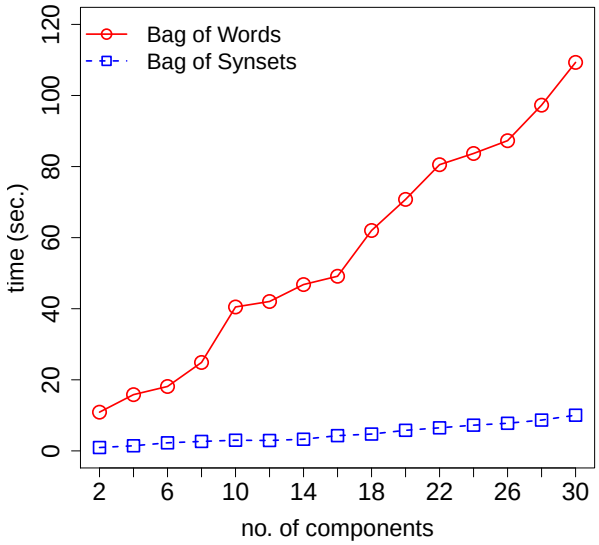


Fig. 5.6: Time performance of SVD over the mode-1 matricization of the *Balanced Corpus* tensor.

5.5 Discussion

Our work paves the way for the use of a multilingual knowledge base to deal with the multilingual document clustering task. Here we sum up our main findings.

***SeMDocT* vs. *LSA*.** *LSA* achieved its best results for a number of components generally smaller than the one for which *SeMDocT* obtained its maximum. This is due to the initial information that the two methods summarize. *LSA* tries to capture the variation of the initial document-term (alternatively, document-synset) matrix representing the texts in a lower space, whereas *SeMDocT* does the same starting from a richer representation of the documents (i.e., a third-order tensor model). For this reason, *SeMDocT* tends

to employ relatively more components in order to summarize the documents content; however, a number of components between 16 and 30 is generally enough to ensure good performance of *SeMDocT*. Moreover, in most cases, the highest performance results by *SeMDocT* are better than the highest performances of *LSA*. for

BoS vs. BoW. Our results have highlighted the better quality in multilingual clustering supplied by synsets compared with the one provided by terms. BoS produces a smaller representation space over which documents are projected, but it is enough rich to well capture the documents content. In particular, BoS benefits from the WSD process that is able to discriminate the same term w.r.t. the context in which it appears.

BabelNet. BabelNet is a recent project that supports many different languages. As the intention of the authors is to enrich this resource, in the future our framework will benefit of this fact. Moreover, our framework can deal with documents written in many different languages as they are represented through the same space; the only constraint is related to the available language support in BabelNet. On the other hand, we point out that any other multilingual knowledge base and WSD tools can in principle be integrated in our framework.

5.6 Chapter review

This chapter has focused on the problem of clustering documents written in different languages. A major issue in clustering multilingual documents arises from the computation of the similarity the documents. Comparing documents with respect to the original language-specific dictionaries result to be impracticable due to poor lexical overlap between the languages. Therefore, the problem is traditionally approached by translating all documents in an anchor language or making use of bilingual dictionaries, to evaluate document similarities. However, the required transformations may alter the original document semantics, especially in case of poor-resourced languages.

To cope with this issue, the framework presented in this chapter (i) exploits a large-scale multilingual knowledge base, (ii) takes advantage of the multi-topic nature of the text documents, and (iii) employs a tensor-based model to deal with high dimensionality and sparseness. The proposed approach has been compared with common document clustering approaches on a real multilingual corpus considering both balanced and unbalanced language coverage per topic. Experiments have shown the significance of the proposed approach, and its better performance w.r.t. classic document clustering approaches, in both a balanced and an unbalanced corpus evaluation.

6

Conclusion and future work

In this thesis we have addressed document clustering problems, focusing on the fact that many text repositories over the Web and other online information sources often contain documents that discuss multiple thematic subjects (i.e., *multi-topic* documents), cover different aspects of some of the topics or other external information (i.e., *multi-view* documents), and can be written in different languages (i.e., *multilingual* documents). A distinctive, unifying aspect of our study is the definition of document clustering framework that rely on a multidimensional representation model paradigm for the documents in the input collection. This paradigm refers to the definition of tensor models (and decomposition techniques) which are a powerful tool to explicitly capture the characteristics and relations between the different facets of high-dimensional data. Tensors are in fact particularly appropriate to provide an effective representation model for multi-topic, multi-view documents; moreover, tensor decompositions, which can be viewed as a generalization of multi-linear matrix factorizations, allow us to extract a low-dimensional yet meaningful representation which incorporates, in addition to the content information, the information related to the views' interactions.

In Chapter 3, the underlying assumption of the approach presented is that, when the documents can be naturally grouped in multiple ways, a single new clustering encompassing all existing document classifications can be obtained by integrating the textual content information with knowledge on the groupings of the documents through the available classifications. However, since no information about any labels of the available groups of documents is assumed to be required, our key idea to accomplish the task relies on the identification of frequent co-occurrences of documents in the groups across the existing classifications, in order to capture how documents tend to be grouped together orthogonally to the different views. Based on the discovered frequent associations of the documents as well as on the usual term-document representation of the text contents, a tensor model is built and decomposed to finally establish a unique clustering of documents that might be suited to reflect the multidimensional structure of the initial document classifications.

Experiments conducted on a document clustering benchmark have shown the potential of the approach to capture the multi-view structure of existing organizations for a given collection of documents.

We presented a tensor-based clustering framework for view-segmented documents in Chapter 4. Multi-view documents are treated in terms of their constituent, view-based text segments, while representing them under a multidimensional data structure. The goal was to explore the benefits deriving from a combination of document segmentation with a multidimensional data structure, the result being able to capture a segment-grained view-oriented topical representation of the documents. Experimental results have supported our intuition on the clustering improvement performance over different challenging datasets.

Finally, we addressed the multi-topic and multilingual document clustering problem in Chapter 5. In this context, we defined a language-independent representation model with the support of a multilingual lexical knowledge base (BabelNet). Thematic alignment across documents is accomplished by obtaining semantically coherent cross-lingual topics, so to enable language-independent preserving of the content semantics. The multilingual documents are decomposed into topically-coherent segments in order to enable the grouping of linguistically different portions of documents by content. Moreover, these two key aspects are integrated into a multidimensional data structure in order to better capture the multi-topic multilingual nature of the documents. We evaluated our approach w.r.t. standard document clustering methods, using both term and synset representations. Results have shown the benefits deriving from the use of a multilingual knowledge base in the analysis of comparable corpora, and also shown the significance of our approach in both a balanced and an unbalanced corpus evaluation. Our tensor-based representation of topically-segmented multilingual documents can also be applied to cross-lingual information retrieval or multilingual document categorization.

Future research. There are many open questions and emerging challenges that we would like to address. In the following we briefly provide pointers for future research.

- *Tensor-based corpus modeling* – In this thesis our focus was only on two tensor decompositions, namely Non-negative Tucker decomposition and Higher-order SVD. Therefore, a natural extension of our work is to be devoted to an investigation of the opportunities offered by other tensor decomposition models and their capabilities (and limitations) as concerns clustering performances.
  Tensor decompostions provide a meaningful and low-dimensional representation for each dimension. A further study we plan to conduct is the jointly exploitation of mode-1 and mode-2 factor matrices in order to perform co-clustering tasks.
  Furthermore, another challenge is the interpretability of the factorizations provided by tensor decompositions. In this context, it would be interesting

to investigate regularization methods in order to support the above aspect. Other study should be devoted to the development of methods that can support the detection of a suitable number of components for the tensor decomposition in our domains of interest.

- *Tensor models for supervised and semi-supervised learning tasks* – The multilingual document classification task is traditionally addressed in a inductive learning setting by translating all documents in an anchor language or with the support of bi-lingual resources. A major issue in this setting is the availability of a full-labeled corpora.
  Our view of the problem of multilingual document classification is different from the standard inductive learning setting: on the one hand, we look at a supervised learning fashion as we want to exploit the available a-priori knowledge on the topic-classes assigned to the (multilingual) documents; on the other hand, we recognize that high-quality labeled datasets are difficult to obtain due to costly and time-consuming annotation processes. This particularly holds for the multilingual scenario where the documents belong to different languages, and hence more, language-specific experts need to be involved in the annotation process. *Transductive learning* offers an effective approach to deal with this issue. We have started to study the problem, and preliminary evaluation has shown promising results [165]. As a further step, we plan to investigate the definition of a tensor model capable of providing a suitable low-dimensional representation of multilingual documents in a transductive learning framework.
- *Enhancing multilingual document modeling* – Our use of BabelNet was restricted to its multilingual lexical dictionary capabilities. However, BabelNet also offers ontological functionalities. Therefore, an urgent enhancement of our conceptual document representation approach will be devoted to the enrichment of our bag-of-synsets model with information such as the relations between the synsets.
- *Applications* – While the study described in this thesis has focused on relatively long documents, there exist several application domains characterized by a huge amount of short documents; this is for instance the case social media networks, like Twitter, which offers microblogging services. Mining short documents leads to several new challenges, mainly because of the poor content information which characterizes this kind of documents. Therefore, it would be interesting to investigate the development of tensor models that are capable of coping with short documents, also in multilingual contexts.
  Although collaborative editing environments, like Wikipedia, are powerful tools for sharing knowledge, in multilingual contexts often there is not exact alignment of the topical knowledge across the various languages. This is the case of resource-poor languages, in which it easily happens that a topic is not as well covered as in other languages. We believe that, by exploiting the ability of tensor models and decomposition techniques in leveraging interactions between different facets of documents in a collec-

tion, multilingual document clustering can support document fertilization, i.e., recommendation of topics poorly covered by documents written in a given language.

Information networks and their evolution over time also represent challenging scenarios due to the multiple perspectives through which they can be analyzed. Tensor-based network models can be helpful in developing new approaches to several tasks, such as relation prediction, community detection, and topic evolution.

References

1. E. Acar, S. A. Çamtepe, M. S. Krishnamoorthy, and B. Yener. Modeling and multiway analysis of chatroom tensors. In P. Kantor, G. Muresan, F. Roberts, D. Zeng, F.-Y. Wang, H. Chen, and R. Merkle, editors, *Intelligence and Security Informatics*, volume 3495 of *Lecture Notes in Computer Science*, pages 256–268. Springer Berlin Heidelberg, 2005.
2. E. Acar, S. A. Çamtepe, and B. Yener. Collective sampling and analysis of high order tensors for chatroom communications. In *Intelligence and security informatics*, pages 213–224. Springer, 2006.
3. G. W. Adamson and J. Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10:253–260, 1974.
4. C. C. Aggarwal, S. C. Gates, and P. S. Yu. On the merits of building categorization systems by supervised clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 352–356. ACM, 1999.
5. C. C. Aggarwal and C. X. Zhai. A survey of text clustering algorithms. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pages 77–128. Springer US, 2012.
6. E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the International Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–41, 2009.
7. J. Allen. *Natural Language Understanding*. Benjamin/Cummings, 1995.
8. D. C. Anastasiu, A. Tagarelli, and G. Karypis. *Data Clustering: Algorithms and Applications*, chapter Document Clustering: The Next Frontier, pages 305–338. Taylor & Francis Publisher, 2013.
9. C. M. Andersen and R. Bro. Practical aspects of parafac modeling of fluorescence excitation-emission data. *Journal of Chemometrics*, 17(4):200–215, 2003.
10. R. K. Ando and L. Lee. Iterative Residual Rescaling: An Analysis and Generalization of LSI. In *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–162, 2001.
11. J. Anil K. and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

12. C. J. Appellof and E. Davidson. Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. *Analytical Chemistry*, 53(13):2053–2056, 1981.
13. Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Spectral Analysis of Data. pages 619–626, 2001.
14. B. W. Bader, M. W. Berry, and M. Browne. Discussion tracking in enron email using parafac. In M. Berry and M. Castellanos, editors, *Survey of Text Mining II*, pages 147–163. Springer London, 2008.
15. B. W. Bader and T. G. Kolda. Algorithm 862: Matlab tensor classes for fast algorithm prototypin. *ACM Trans. Math. Software*, 32:635–653, 2006.
16. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Books. Addison-Wesley, 1999.
17. A. Barrón-Cedeño, M. L. Paramita, P. Clough, and P. Rosso. A comparison of approaches for measuring cross-lingual similarity of wikipedia articles. pages 424–429, 2014.
18. D. Beeferman, A. Berger, and J. Lafferty. Statistical models for text segmentation. *Machine learning*, 34(1-3):177–210, 1999.
19. M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, 37(4):573–595, 1995.
20. S. Bickel and T. Scheffer. Multi-view clustering. In *Proceedings of IEEE International Conference on Data Mining*, pages 19–26, 2004.
21. D. Boley. Principal direction divisive partitioning. *Data mining and knowledge discovery*, 2(4):325–344, 1998.
22. C. Borgelt, X. Yang, R. Nogales-Cadenas, P. Carmona-Saez, and A. Pascual-Montano. Finding closed frequent item sets by intersecting transactions. In *Proceedings ACM International Conference on Extending Database Technology*, pages 367–376, 2011.
23. T. Brants, F. Chen, and I. Tsochantaridis. Topic-based document segmentation with probabilistic latent semantic analysis. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 211–218. ACM, 2002.
24. E. Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21(4):543–565, 1995.
25. R. Bunescu and M. Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–16, 2006.
26. M. A. Bunge. *Treatise on Basic Philosophy: Vol. 3: Ontology I: The Furniture of the World*. Reidel, 1977.
27. M. A. Bunge. *Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems*. Reidel, 1979.
28. J. D. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35(3):283–319, 1970.
29. M. Carullo, E. Binaghi, and I. Gallo. An online document clustering technique for short web contents. *Pattern Recognition Letters*, 30(10):870–876, 2009.
30. R. B. Cattell. parallel proportional profiles and other principles for determining the choice of factors by rotation. *Psychometrika*, 9(4):267–283, 1944.
31. R. B. Cattell. The three basic factor-analytic research designstheir interrelations and derivatives. *Psychological Bulletin*, 49(5):499–452, 1952.

32. P. A. Chew, B. W. Bader, T. G. Kolda, and A. Abdelali. Cross-language information retrieval using PARAFAC2. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 143–152. ACM, 2007.
33. F. Y. Y. Choi, P. Wiemer-Hastings, and J. Moore. Latent semantic analysis for text segmentation. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 109–117, 2001.
34. A. Cichocki and S.-i. Amari. Families of alpha- beta- and gamma- divergences: Flexible and robust measures of similarities. *Entropy*, 12(6):1532–1568, 2010.
35. A. Cichocki, A. H. Phan, and R. Zdunek. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, 2009.
36. M. Ciocoiu and D. S. Nau. Ontology-based Semantics. In *Proceedings of the International Conference on Principles on Knowledge Representation and Reasoning*, pages 539–546, 2000.
37. N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2-3):127–152, 2002.
38. W. B. Croft, H. R. Turtle, and D. D. Lewis. The Use of Phrases and Structured Queries in Information Retrieval. In *Proceedings of the 14th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 32–45, 1991.
39. M. Cuadros and G. Rigau. Quality assessment of large scale knowledge resources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 534–541, 2006.
40. M. Cuadros and G. Rigau. Knownet: Building a large net of knowledge from the web. In *In Proceedings of International Conference on Computational Linguistics*, 2008.
41. D. R. Cutting, J. Kupiec, J. O. Pedersen, and P. Sibun. A Practical Part-of-Speech Tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 133–140, 1992.
42. I. Davidson, K. Wagstaff, and S. Basu. Measuring constraint-set utility for partitional clustering algorithms. In *Proceedings of European conference on Principle and Practice of Knowledge Discovery in Databases*, pages 115–126, 2006.
43. L. De Lathauwer, B. De Moor, and J. Vandewall. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
44. L. De Lathauwer, B. De Moor, and J. Vandewall. On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.
45. L. De Lathauwer and J. Vandewalle. Dimensionality reduction in higher-order signal processing and rank-(r1,r2, . . . ,rn) reduction in multilinear algebra. *Linear Algebra and its Applications*, 391(0):31 – 55, 2004.
46. G. de Melo and G. Weikum. Menta: Inducing multilingual taxonomies from wikipedia. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1099–1108, 2010.
47. M. Deepa, P. Revathy, and P. G. Student. Validation of document clustering based on purity and entropy measures. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(3):147–152, 2012.

48. S. C. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
49. E. Dermatas and G. Kokkinakis. Stochastic Tagging of Natural Language Texts. *Computational Linguistics*, 21(2):137–153, 1995.
50. S. J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, 1988.
51. M. Dillon and A. S. Gray. FASIT: A Fully Automatic Syntactically Based Indexing System. *Journal of the American Society for Information Science*, 34(2):99–108, 1983.
52. C. Ding, X. He, H. Zha, M. Gu, and H. Simon. Spectral min-max cut for graph partitioning and data clustering. *Lawrence Berkeley National Lab. Tech. report*, 47848, 2001.
53. B. E. Dom. An information-theoretic external cluster-validity measure. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 137–145, 2002.
54. D. M. Dunlavy, T. G. Kolda, , and K. W. Philip. Multilinear algebra for analyzing data with multiple linkages. In *Graph Algorithms in the Language of Linear Algebra*, Fundamentals of Algorithms, pages 85–114. SIAM, 2011.
55. D. A. Evans, K. Ginther-Webster, M. Hart, R. G. Lefferts, and I. A. Monarch. Automatic Indexing Using Selective NLP and First-Order Thesauri. In *Proceedings of the Conference on Intelligent Text and Image Handling*, pages 624–643, 1991.
56. J. L. Fagan. The Effectiveness of a Nonsyntactic Approach to Automatic Phrase Indexing for Document Retrieval. *Journal of the American Society for Information Science*, 40(2):115–132, 1989.
57. S. Faralli and R. Navigli. A new minimally-supervised framework for domain word sense disambiguation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1411–1422, 2012.
58. C. Fellbaum. English Verbs as a Semantic Net. *International Journal of Lexicography*, 3(4):278–301, 1990.
59. C. Fellbaum. Towards a Representation of Idioms in WordNet. In *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, pages 52–57, 1998.
60. C. Fellbaum, editor. *WordNet: An Electronic Database*. MIT Press, 1998.
61. W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, 1992.
62. M. P. Friedlander and K. Hatz. Computing non-negative tensor factorizations. *Optimization Methods Software*, 23:631–647, 2008.
63. N. Fuhr. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3):243–255, 1992.
64. G. W. Furnas, S. C. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information Retrieval using a Singular Value Decomposition Model of Latent Semantic Structure. *Proceedings of the 11th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 465–480, 1988.
65. E. Gabrilovich and S. Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34(1):443–498, 2009.

66. J. Ghosh and A. Acharya. Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4):305–315, 2011.
67. G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.
68. G. H. Golub and C. F. van Van Loan. *Matrix Computations*. Johns Hopkins University Press.
69. D. Gross and K. J. Miller. Adjectives in WordNet. *International Journal of Lexicography*, 3(4):265–277, 1990.
70. T. R. Gruber. A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.
71. S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.
72. S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521. IEEE, 1999.
73. M. A. Hafer and S. F. Weiss. Word Segmentation by Letter Successor Varieties. *Information Storage and Retrieval*, 10:371–385, 1974.
74. M. A. K. Halliday. *Spoken and Written Language*. Oxford University Press, 1989.
75. A. Hanene, C. Guinot, and G. Venturini. Anttree: A web document clustering using artificial ants. In *Proceedings of the 16th European Conference on Artificial Intelligence*, 2004.
76. S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtrieval Conference*, pages 479–488, 2000.
77. R. A. Harshman. Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):1–84, 1970.
78. J. Hastad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.
79. M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64, 1997.
80. R. Henrion. N-way principal component analysis theory, algorithms and applications. *Chemometrics and Intelligent Laboratory Systems*, 25(1):1–23, 1994.
81. F. L. Hitchcock. *The Expression of a Tensor Or a Polyadic as a Sum of Products*. Contributions from the Department of Mathematics. 1927.
82. F. L. Hitchcock. Multiple invariants and generalized rank of a p-way matrix or tensor. *Journal of Mathematical Physics*, 7:39–79, 1927.
83. T. Hoffmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.
84. F. B. Holt. Subspace Representations of Unstructured Text. In *Proceedings of the IEEE ICDM Workshop on Text Mining*, 2001.
85. D. A. Hull. Stemming Algorithms: A Case Study for Detailed Evaluation. *Journal of the American Society for Information Science*, 47(1):70–84, 1996.
86. 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.

87. D. Ienco, C. Robardet, R. G. Pensa, and R. Meo. Parameter-less co-clustering for star-structured heterogeneous data. *Data Mining Knowledge Discovery*, 26(2):217–254, 2013.
88. D. Ingaramo, M. Errecalde, and P. Rosso. A general bio-inspired method to improve the short-text clustering task. In *Computational Linguistics and Intelligent Text Processing*, pages 661–672. Springer, 2010.
89. P. Jain, R. Meka, and I. S. Dhillon. Simultaneous unsupervised learning of disparate clusterings. *Statistical Analysis and Data Mining*, 1(8):195–210, 2008.
90. A. Kapteyn, H. Neudecker, and T. Wansbeek. An approach ton-mode components analysis. *Psychometrika*, 51(2):269–275, 1986.
91. G. Karypis. Cluto - software for clustering high-dimensional datasets. http://www.cs.umn.edu/ cluto, 2002/2007.
92. G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
93. M. S. G. Karypis and V. Kumar. A comparison of document clustering techniques. In *TextMining Workshop at KDD 2000*, 2000.
94. L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.
95. H. A. L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105—122, 2000.
96. H. A. L. Kiers and I. V. Mechelen. Three-way component analysis: Principles and illustrative application. *Psychological methods*, 6(1):84, 2001.
97. Y.-D. Kim and S. Choi. Nonnegative tucker decomposition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
98. Y.-D. Kim, A. Cichocki, and S. Choi. Nonnegative tucker decomposition with alpha-divergence. In *Proceedings IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 1829–1832, 2008.
99. Y.-M. Kim, M.-R. Amini, C. Goutte, and P. Gallinari. Multi-view clustering of multilingual documents. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 821–822, 2010.
100. B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 62(317):86–101, 1967.
101. T. G. Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories, 2006.
102. T. G. Kolda and B. W. Bader. The TOPHITS model for higher-order web link analysis. In *Proceedings of the SIAM Data Mining Conference Workshop on Link Analysis, Counterterrorism and Security*, 2006.
103. T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3), 2009.
104. T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *Proceedings Eighth IEEE International Conference on Data Mining*, 2008.
105. W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, 1996.
106. J. B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and its Applications*, 18(2):95 –138, 1977.

107. N. K. Kumar, G. S. K. Santosh, and V. Varma. Effectively mining wikipedia for clustering multilingual documents. In *Proceedings of International Conference on Applications of Natural Language to Information Systems*, pages 254–257, 2011.
108. N. K. Kumar, G. S. K. Santosh, and V. Varma. Multilingual document clustering using wikipedia as external knowledge. In *Multilingual Document Clustering Using Wikipedia as External Knowledge*, volume 6653, pages 108–117, 2011.
109. K. Kummamuru, A. Dhawale, and R. Krishnapuram. Fuzzy co-clustering of documents and keywords. In *Proceedings of 12th IEEE International Conference on Fuzzy Systems*, volume 2, pages 772–777. IEEE, 2003.
110. S. Kutty, R. Nayak, and Y. Li. Xml documents clustering using a tensor space model. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (1)*, pages 488–499, 2011.
111. D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communitions of the ACM*, 38(11):33–38, 1995.
112. J. Levin. *Three-mode factor analysis*. PhD thesis, University of Illinois, Urbana, 1963.
113. D. D. Lewis and K. Sparck Jones. Natural Language Processing for Information Retrieval. *Communications of the ACM*, 39(1):92–101, 1996.
114. D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5, 2004.
115. S. Li, Rui amd Wang, H. Deng, R. Wang, and K. C.-C. Chang. Towards social user profiling: unified and discriminative influence model for inferring home locations. In *Proceedings of the ACM SIGKDD 18th International Conference on Knowledge Discovery and Data Mining*, pages 1023–1031, 2012.
116. L. V. Lita, W. A. Hunt, and E. Nyberg. Resource analysis for question answering. In *Proceedings of the acl 2004 on interactive poster and demonstration sessions*, page 18. Association for Computational Linguistics, 2004.
117. N. Liu, J. Zhang, Benyu amd Yan, Z. Chen, W. Liu, F. Bai, and L. Chien. Text representation: From vector to tensor. In *Proceedings of IEEE International Conference on Data Mining*, pages 725–728, 2005.
118. X. Liu, W. Glänzel, and B. De Moor. Hybrid clustering of multi-view data via tucker-2 model and its application. *Scientometrics*, 88(3):819–839, 2011.
119. J. B. Lovins. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11(1):22–31, 1968.
120. H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.
121. J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967.
122. C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
123. L. Marquez, L. Padro, and H. Rodriguez. A Machine Learning Approach to POS Tagging. *Machine Learning*, 39(1):59–91, 2000.
124. R. Mihalcea, P. Tarau, and E. Figa. Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

125. G. A. Miller. Nouns in WordNet: A Lexical Inheritance System. *International Journal of Lexicography*, 3(4):245–264, 1990.
126. G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.
127. H. Misra, F. Yvon, J. M. Jose, and O. Cappe. Text segmentation via topic modeling: an analytical study. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1553–1556, 2009.
128. J. Mocks. Topographic components model for event-related potentials and some biophysical considerations. *IEEE Transactions on Biomedical Engineering*, 35(6):482–484, 1988.
129. M. F. Moens. *Automatic Indexing and Abstracting of Document Texts*. Kluwer Academic Publishers, 2000.
130. S. Montalvo, R. Martinez, A. Casillas, and V. Fresno. Multilingual news clustering: Feature translation vs. identification of cognate named entities. *Pattern Recognition Letters*, 28(16):2305–2311, 2007.
131. A. Moro and R. Navigli. Wisenet: Building a wikipedia-based semantic network with ontologized relations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1672–1676, 2012.
132. M. Mørup, L. K. Hansen, and S. M. Arnfred. Algorithms for sparse nonnegative tucker decompositions. *Neural Computation*, 20:2112–2131, August 2008.
133. D. Muti and S. Bourennane. Multidimensional filtering based on a tensor approach. *Signal Processing*, 85(12):2338–2353, 2005.
134. V. Nastase. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 763–772, 2008.
135. V. Nastase, M. Strube, B. Borschinger, C. Zirn, and A. Elghafari. Wikinet: A very large scale multi-lingual concept network. In *Proceedings of 7th International Conference on Language Resources and Evaluation*, pages 1015–1022, 2010.
136. R. Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10:1–10:69, 2009.
137. R. Navigli, S. Faralli, A. Soroa, O. de Lacalle, and E. Agirre. Two birds with one stone: Learning semantic models for text categorization and word sense disambiguation. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2317–2320, 2011.
138. R. Navigli and M. Lapata. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):678–692, 2010.
139. R. Navigli and S. P. Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.
140. R. Navigli and S. P. Ponzetto. Multilingual wsd with just a few lines of code: The babelnet api. In *Proceedings of the ACL 2012 System Demonstrations*, pages 67–72, 2012.
141. X. Ni, J.-T. Sun, J. Hu, and Z. Chen. Cross lingual text classification by mining multilingual topics from wikipedia. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 375–384.

142. L. Padró and E. Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Eight International Conference on Language Resources and Evaluation*, 2012.
143. C. D. Paice. Another Stemmer. *SIGIR Forum*, 24(3):56–61, 1990.
144. C. D. Paice. A Method for the Evaluation of Stemming Algorithms based on Error Counting. *Journal of the American Society for Information Science*, 47(8):632–649, 1996.
145. C. H. Papadimitriou, P. Raghavan, Raghavan, and S. Vempala. Latent Semantic Indexing: A Probabilistic Analysis. *Journal of Computer and System Sciences*, 61(2):217–235, 2000.
146. A. H. Phan and A. Cichocki. Fast and efficient algorithms for nonnegative tucker decomposition. In *Proceedings International Symposium on Neural Networks: Advances in Neural Networks*, pages 772–782, 2008.
147. A. H. Phan and A. Cichocki. Multi-way nonnegative tensor factorization using fast hierarchical alternating least squares algorithm (hals). In *Proceedings of The 2008 International Symposium on Nonlinear Theory and its Applications*, 2008.
148. A. H. Phan and A. Cichocki. Local Learning Rules for Nonnegative Tucker Decomposition. In *Proceedings International Conference on Neural Information Processing*, pages 538–545, 2009.
149. E. Pianta, L. Bentivogli, and C. Girardi. Developing an aligned multilingual database. In *Proceedibgs 1st Intenrnational Conference on Global WordNet*, 2002.
150. D. Pinto, J.-M. Bened, and P. Rosso. Clustering narrow-domain short texts by using the kullback-leibler distance. In *Computational Linguistics and Intelligent Text Processing*, pages 611–622. Springer, 2007.
151. G. Ponti and A. Tagarelli. Topic-based hard clustering of documents using generative models. In *Proceedings of the 18th International Symposium on Foundations of Intelligent Systems*, pages 231–240, 2009.
152. G. Ponti, A. Tagarelli, and G. Karypis. A statistical model for topically segmented documents. In *Proceedings of the 14th International Conference on Discovery Science*, pages 231–240, 2011.
153. S. P. Ponzetto and M. Strube. Knowledge derived from wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30(1), 2007.
154. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
155. P. Proctor, editor. *Longman Dictionary of Contemporary English*. Longman Group, 1978.
156. V. V. Raghaven and S. K. M. Wong. A critical analysis of vector space model for information retrieval. *Journal of the American Society for Information Science*, 37(5):279–287, 1986.
157. A. Rahman and V. Ng. Narrowing the modeling gap: A cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research*, 40(1):469–521, 2011.
158. D. Ramage, P. Heymann, C. D. Manning, and G.-M. Hector. Clustering the tagged web. In *Proceedings of Web Search and Data Mining*, pages 54–63, 2009.
159. W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66:846–850, 1971.
160. S. E. Robertson and K. Sparck Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

161. P. M. Roget. *Rogets International Thesaurus,1st edition*. Cromwell, 1911.
162. S. Romeo, A. Tagarelli, F. Gullo, and S. Greco. A tensor-based clustering approach for multiple document classifications. In *Proceedings of International Conference on Pattern Recognition Applications and Methods*, pages 200–205, 2013.
163. S. Romeo, A. Tagarelli, and D. Ienco. Clustering view-segmented documents via tensor modeling. In *Proceedings of International Symposium on Methodologies for Intelligent Systems*, volume 8502 of *Lecture Notes in Computer Science*, pages 385–394. 2014.
164. S. Romeo, A. Tagarelli, and D. Ienco. Semantic-based multilingual document clustering via tensor modeling. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 600–609, 2014.
165. S. Romeo, A. Tagarelli, and D. Ienco. Knowledge-based representation for transductive multilingual document classification. To Appear in *Proceedings of 37th European Conference on Information Retrieval*, 2015.
166. M. Ruiz-Casado, E. Alfonseca, and P. Castells. Automatic assignment of wikipedia encyclopedic entries to wordnet synsets. In *Proceedings of the Third International Conference on Advances in Web Intelligence*, pages 380–386, 2005.
167. G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.
168. G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
169. G. Salton, C. Buckley, and M. Smith. On the Application of Syntactic Methodologies in Automatic Text Analysis. *Information Processing and Management*, 26(1):73–92, 1990.
170. G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.
171. G. Salton, C. C. Yang, and C. T. Yu. A Theory of Term Importance in Automatic Text Analysis. *Journal of the American Society for Information Science*, 26(1):33–44, 1975.
172. S. Salvador and P. Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Proceedings of 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 576–584, 2004.
173. B. Savas and L. Eldén. Handwritten digit classification. *Pattern Recognition*, 40(3):993–1003, 2007.
174. C. Schwarz. Automatic Syntactic Analysis of Free Text. *Journal of the American Society for Information Science*, 41(6):408–417, 1990.
175. T. M. Selee, T. G. Kolda, W. P. Kegelmeyer, and J. D. Griffin. Extracting clusters from large datasets with multiple similarity measures using imscand. In *CSRI Summer Proceedings 2007*, pages 87–103, 2007.
176. C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423/623–656, 1948.
177. A. Shashua and A. Levin. Linear image coding for regression and classification using the tensor-rank principle. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–42. IEEE, 2001.

178. W. M. Shaw, R. Burgin, and P. Howell. Performance Standards and Evaluations in IR Test Collections: Cluster-based Retrieval Models. *Information Processing and Management: an International Journal*, 33(1):1–14, 1997.
179. N. D. Sidiropoulos, R. Bro, and G. B. Giannakis. Parallel factor analysis in sensor array processing. *IEEE Transactions on Signal Processing*, 48(8):2377–2388, 2000.
180. P. H. A. Sneath and R. R. Sokal. *Numerical taxonomy. The principles and practice of numerical classification.* Freeman, 1973.
181. J. F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks Cole, 2000.
182. K. Sparck Jones. Index Term Weighting. *Information Storage and Retrieval*, 9:619–633, 1973.
183. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.
184. A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *High Performance Computing*, pages 525–536. Springer, 2000.
185. A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.
186. F. M. Suchanek, G. Ifrim, and G. Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 712–717, 2006.
187. J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: A novel approach to personalized web search. In *Proceedings of the 14th International Conference on World Wide Web*, pages 382–390, 2005.
188. Q. Sun, R. Li, D. Luo, and X. Wu. Text segmentation with lda-based fisher kernel. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 269–272. Association for Computational Linguistics, 2008.
189. M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307, 2011.
190. A. Tagarelli and G. Karypis. A segment-based approach to clustering multi-topic documents. *Knowledge and Information Systems*, 34(3):563–595, 2013.
191. G. Tsatsaronis, I. Varlamis, and K. Norvag. Semanticrank: Ranking keywords and sentences using semantic graphs. In *Proceedings of the International Conference on Computational Linguistics*, pages 1074–1082, 2010.
192. L. R. Tucker. *Problems in Measuring Change*, chapter Implications of factor analysis of three-way matrices for measurement of change, pages 122–137. University of Wisconsin Press, 1963.
193. L. R. Tucker. The extension of factor analysis to three-dimensional matrices. In H. Gulliksen and N. Frederiksen, editors, *Contributions to mathematical psychology*. Holt, Rinehart and Winston, 1964.
194. L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.
195. D. Tufis, D. Cristea, and S. Stamou. Balkanet: Aims, methods, results and perspectives. a general overview. *Romanian Journal on Science and Technology of Information*, pages 3–4, 2004.

196. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Computer Vision*, volume 2350, pages 447–460. 2002.
197. P. Vossen, editor. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, 1998.
198. X. Wan. Bilingual co-training for sentiment classification of chinese product reviews. *Computational Linguistics*, 37(3):587–616, 2011.
199. P. Wang and C. Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–721, 2008.
200. Y. Wang, Y. Jia, and S. Yang. Short documents clustering in very large text databases. In *Web Information Systems–WISE 2006 Workshops*, pages 83–93. Springer, 2006.
201. C.-P. Wei, C. C. Yang, and C.-M. Lin. A latent semantic indexing-based approach to multilingual document. *Decision Support Systems*, 45(3):606–620, 2008.
202. P. M. Wiemer-Hastings. How Latent is Latent Semantic Indexing? In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 932–941, 1999.
203. F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th International Conference on World Wide Web*, pages 635–644, 2008.
204. J. Xu and W. B. Croft. Corpus-based Stemming Using Coocurrence of Word Variants. *ACM Transactions on Information Systems*, 16(1):61–81, 1998.
205. Y. Yang and C. G. Chute. An Example-based Mapping Method for Text Categorization and Retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.
206. E. Yeh, D. Ramage, C. D. Manning, E. Agirre, A. Soroa, and I. Taldea. Wikiwalk: Random walks on wikipedia for semantic relatedness. In *Workshop on Graph-based Methods for Natural Language Processing*, pages 41–49, 2009.
207. D. Yogatama and K. Tanaka-Ishii. Multilingual spectral clustering using document similarity propagation. In *Proceeding of Conference on Empirical Methods in Natural Language Processing*, pages 871–879, 2009.
208. C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *Computers, IEEE Transactions on*, 100(1):68–86, 1971.
209. M. J. Zaki and C.-J. Hsiao. CHARM: An Efficient Algorithm for Closed Itemset Mining. In *Proceedings SIAM International Conference on Data Mining*, pages 457–473, 2002.
210. Z.-Y. Zhang, T. Li, and C. Ding. Non-negative tri-factor tensor decomposition with applications. *Knowledge and Information Systems*, 34(2):243–265, 2013.
211. Y. Zhao and G. Karypis. Empirical and theoretical comparison of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.
212. Y. Zhao and G. Karypis. Soft clustering criterion functions for partitional document clustering: a summary of results. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 246–247, 2004.
213. S. Zhong and J. Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.
214. G. K. Zipf. *Human Behavior and the Principle of Least Effort: an Introduction to Human Ecology*. Addison-Wesley, 1949.