Fisher Kernels and Probabilistic Latent Semantic Models

THÈSE N^{O} 4647 (2010)

PRÉSENTÉE LE 23 AVRIL 2010

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE D'INTELLIGENCE ARTIFICIELLE

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Emmanuel ECKARD

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Dr M. Rajman, Dr J.-C. Chappelier, directeurs de thèse
Dr L. Azzopardi, rapporteur
Prof. E. Gaussier, rapporteur
Dr O. Lévêque, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2010

Abstract

Tasks that rely on semantic content of documents, notably Information Retrieval and Document Classification, can benefit from a good account of document context, i.e. the semantic association between documents. To this effect, the scheme of *latent semantics* blends individual words appearing throughout a document collection into *latent topics*, thus providing a way to handle documents that is less constrained than the conventional approach by the mere appearance of such or such word.

Probabilistic latent semantic models take the matter further by providing assumptions on how the documents observed in the collection would have been generated. This allows derivation of inference algorithms that can fit the model parameters to the observed document collection; with their values set, these parameters can then be used to compute the similarities between documents. The *Fisher kernels*, similarity functions rooted in information geometry, constitute good candidates to measure the similarity between documents in the framework of probabilistic latent semantic models.

In this context, we study the use of Fisher kernels for the Probabilistic Latent Semantic Indexing (PLSI) model. By thouroughly analysing the generative process of PLSI, we derive the proper Fisher kernel for PLSI and expose the hypotheses that relate former work to this kernel. In particular, we confirm that the Fisher information matrix (FIM) should not be approximated by the identity in the case of PLSI. We also study the impact on the performances of the Fisher kernel of the contribution of the latent topics and the one of the distribution of words among the topics; eventually, we provide empirical evidence, and theoretical arguments, showing that the Fisher kernel originally published by Hofmann, corrected to account for FIM, is the best of the PLSI Fisher kernels. It can compete with the strong BM25 baseline, and even significantly outperforms it when documents sharing few words must be matched.

We further study of PLSI document similarities by applying the Language model approach. This approach shuns the usual IR paradigm that considers documents and queries to be of a similar nature. Instead, they consider documents as being representative of language models, and use probabilistic tools to determine which of these models would have generated the query with highest probability. Using this scheme in the framework of PLSI provides a way to bypass the issue of query representation, which constitutes one of the specific challenges of PLSI. We find the Language model approach to perform as well as the best of the Fisher kernels when enough latent categories are provided.

Eventually, we propose a new probabilistic latent semantic model consisting in a mixture of Smoothed Dirichlet distributions which, by better modeling word burstiness, provides a more realistic model of empirical observations on real document collections than the usually used Multinomials.

Keywords

Information retrieval, document classification, Fisher kernels, language modeling, PLSI, Smoothed Dirichlet, latent semantics, probabilistic models.

Résumé

La prise en compte du contexte, c'est-à-dire des associations sémantiques entre documents, peut avoir un effet bénéfique dans le cadre de tâches qui reposent sur le contenu sémantique de documents, comme la Recherche d'Information ou la Classification de Documents. Dans cette perspective, la théorie de la *sémantique latente* vise à fusionner les contributions de mots qui occurrent dans la collection de documents en formant des *catégories latentes*; ceci produit une quantité qui permet un traitement des documents qui dépend moins de la simple apparition d'un mot que n'en dépend l'approche traditionelle.

Les modèles à sémantique latente probabilistes poussent l'argument encore plus loin, par des hypothèses explicites sur la façon dont les documents observés sont créés. On peut, de là, dériver des algorithmes d'inférence pour adapter les valeurs des paramètres du modèle à l'observation. Ces valeurs peuvent alors s'utiliser pour comparer les documents. Les *noyaux de Fisher*, fonctions de proximité qui ont leurs racines dans la géométrie de l'information, constituent de bons candidats pour servir de similarités dans ce cadre.

Dans ce contexte, nous étudions le modèle *Probabilistic Latent Semantic Indexing* (PLSI), proposé par Hofmann. En analysant en profondeur le processus de génération de PLSI, nous mettons à jour le réel noyau de Fisher pour PLSI, et montrons sous quelles hypothèse le noyau d'Hofmann devient une approximation du noyau réel. Nous confirmons que la Matrice d'Information de Fisher (FIM) ne s'approche pas par la matrice identité dans le cas de PLSI. Nous menons une étude détaillée des différents termes qui constituent le noyau de Fisher, et essayons un éventail de variantes. L'expérience prouve que le noyau d'Hofmann ajustée pour FIM est le noyau le plus performant pour PLSI. Il se compare favorablement à la référence BM25, et surclasse même nettement les modèles classiques comme `tf-idf` ou BM25 lorsque des documents doivent être rapprochés alors qu'ils ne partagent que peu de termes.

Nous poursuivons en proposant une proximité fondée sur les modèles de langue. Au lieu de comparer des objets de même nature représentant les documents et les requêtes, comme en RD classique, cette approche envisage les documents comme des réalisations de modèles de langue, et utilisent des outils probabilistes pour décider lequel des modèles en question est le plus susceptible d'avoir généré la requête. Dans PLSI, cette approche permet de contourner élégamment la question de la représentation de la requête, qui est l'un de ses points faibles. L'approche à modèles de langue s'avère aussi performante que les meilleurs noyaux de Fisher lorsque le nombre de catégories latentes est suffisant.

Pour finir, nous proposons un nouveau modèle constitué d'un mélange de distributions de Dirichlet lissées. Ces distributions simulent la tendance des mots rares à apparaître en rafales, et modélisent ainsi mieux la génération des termes que les multinomiales habituelles. À la différence de PLSI, ce modèle est pleinement génératif, ce qui permet la représentation des requêtes et l'indexation incrémentale.

Mots-clefs

Recherche d'information, classification de documents, noyaux de Fisher, modèles de langue, PLSI, distribution de Dirichlet lissée, sémantique latente, modèles probabilistes.

Contents

List of Figures

List of Tables

Notations

Collection (or "corpus")

- C: collection, in the sense of all the word occurrences through all documents
- $|C|$: total number of occurrence of all terms in all documents: $|C| = \sum_d \sum_w n(d,w)$

Latent topics (or "categories", "semantic categories")

- Z: set of the latent topics
- $P(z)$: probability of occurrence of topic $z \in Z$.

Documents

- D: set of all documents d that appear in the collection
- $P(d)$: probability of document $d \in D$ occurring in the collection
- $|d|$: number of term occurrences in document d. $|d| = \sum_w n(d,w)$

Terms (or "words")

- V: vocabulary, set of all terms appearing in the documents
- $P(w)$: probability of term $w \in V$ occurring in the collection
- $n(d,w)$: number of occurrences of term w in document d
- $P(d,w)$: probability of term $w \in V$ occurring in document $d \in D$, or probability of the (d,w) indice pair in the case of PLSI
- $\widehat{P}(d,w)$: "empirical" or "observed" value of $P(d,w)$. $\widehat{P}(d,w) = n(d,w)/|C|$
- $\widehat{P}(w|d)$: "empirical" or "observed" value of $P(w|d)$. $\widehat{P}(w|d) = n(d,w)/|d|$

Fisher Information matrix

$G(\theta)$: Fisher Information matrix in the canonical parametrisation (i.e. without square root re-parametrisation)

$G(\rho)$: Fisher Information matrix in the square root parametrisation

component for K_z^{U}: $\alpha^{\mathrm{U}}(z) = \sum_d \left(\frac{P(d,z)}{\sqrt{P(z)}} \right)^2 = \sum_d P^2(d|z)P(z)$

component for K_w^{U}: $\gamma^{\mathrm{U}}(w,z) = \sum_d \left(\widehat{P}(d,w) \frac{P(z|d,w)}{\sqrt{P(w|z)}} \right)^2$

component for K_z^{H}: $\alpha^{H}(z) = \sum_d \left(\frac{P(d|z)}{\sqrt{P(z)}} \right)^2 = \sum_d \frac{P^2(d|z)}{P(z)}$

component for K_w^{H}: $\gamma^{H}(w,z) = \sum_d \left(\widehat{P}(w|d) \frac{P(z|d,w)}{\sqrt{P(w|z)}} \right)^2$

Identity matrix: $\mathbb{I}$

Kernels

Hofmann kernel: $K^{\mathrm{H}} = K_z^{\mathrm{H}} + K_w^{\mathrm{H}} = \sum_z k_z^{\mathrm{H}} + \sum_w k_w^{\mathrm{H}}$

Unnormalised kernel: $K^{\mathrm{U}} = K_z^{\mathrm{U}} + K_w^{\mathrm{U}} = \sum_z k_z^{\mathrm{U}} + \sum_w k_w^{\mathrm{U}}$

"Vector Space" kernel: $K^{\mathrm{VS}} = K_z^{\mathrm{VS}} + K_w^{\mathrm{VS}} = \sum_z k_z^{\mathrm{VS}} + \sum_w k_w^{\mathrm{VS}}$

Chapter 1

Introduction

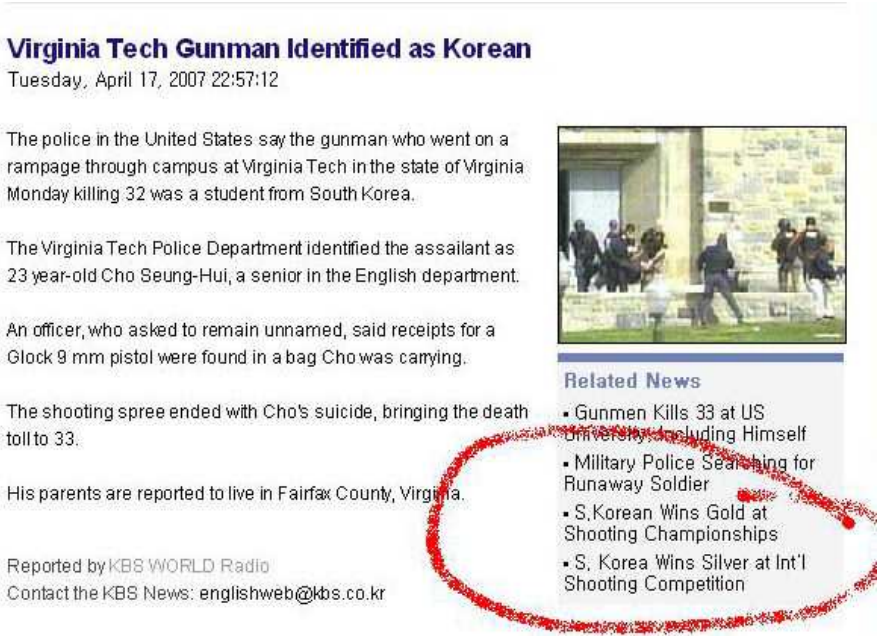
> I learn over the course of the morning that "library work" covers such an enormous area of information management that back during the dark ages, before libraries became self-organising constructs, people used to devote their entire (admittedly short) lives to studying the theory of how best to manage them.
>
> — Charles Stross, *Glasshouse*

Information Retrieval (IR) has been an area of enormous development since the 1950s. With the advent of the Internet, the generalisation of personal computers and the rise of distributed databases, information has become so abundant that managing it has become impossible without the help of automated systems. Furthermore, the democratisation of the World Wide Web has been a catalyst for a searing demand from the general public for a variety of Natural language processing services, ranging from assistance to translation (Altavista – Babel Fish Translation) to automated retrieval of documents relevant to a given query (Google, Yahoo,...), a task known as *ad hoc retrieval*.

Automated information retrieval is an attempt to supplement manual searching of the vast quantities of available information, an overwhelming task for a human, with the processing power of computers. A system is provided a mathematical representation of the available document collection, to which it compares a representation of the query issued by the user. Scores of relevance are produced based on document-query similarity functions, and the user is returned with a list of documents sorted by score.

While they are quicker than Humans to process large amounts of information, machines are in this case plagued by their inability to understand the meaning of the data they process. The fallout can be observed in especially vivid light when synonymy or polysemy are involved, but more generally amount to overall failure to take context into account. Figure 1.1 shows an example of automatically generated text where two distinct events are erroneously brought together: the victory of the Southern Korean team at an international shooting championship is considered to be related to an announcement that the demented gunman of the Virginia Tech massacre of 17 April 2007 was from South Korea. The mistaken relation is caused by the occurrence of the terms "Korean" and "shooting" in both newsfeeds.

Incorporating context into ad hoc Information retrieval has thus been one of the foremost tracks in recent research on the field [3]. The present work positions itself in this trend.



Virginia Tech Gunman Identified as Korean

Tuesday, April 17, 2007 22:57:12

The police in the United States say the gunman who went on a rampage through campus at Virginia Tech in the state of Virginia Monday killing 32 was a student from South Korea.

The Virginia Tech Police Department identified the assailant as 23 year-old Cho Seung-Hui, a senior in the English department.

An officer, who asked to remain unnamed, said receipts for a Glock 9 mm pistol were found in a bag Cho was carrying.

The shooting spree ended with Cho's suicide, bringing the death toll to 33.

His parents are reported to live in Fairfax County, Virginia.

Reported by KBS WORLD Radio
Contact the KBS News: englishweb@kbs.co.kr

Figure 1.1: Screenshot from *KBS*, 17 April 2007. In a typical example of computer-generated, unintended dark humour, the automated system that classifies newfeeds did not distinguish between the demented Korean shooter of the Virginia Tech incident, and the Korean shooters of the national sporting team.

1.1 Motivations

The family of *topic-based* representations of documents is a way to circumvent the problems of IR that stem from failure to account for context. By considering semantic categories, or *topics*, as a parameter of the document collection, on the same foot than words and documents themselves, these representations allow better matching documents of similar content — even if they share few actual word.

One of the foremost manners to model a document collection is to assume it to be the realisation of a stochastic process of some sort. When this process features latent topics, the model is a *probabilistic latent semantic model*. This family of models provides a set of parameters that describe the collection, and the process by which they contribute to produce data. More specifically, Information Retrieval systems based on probabilistic latent semantic models are constituted of three elements (see figure 1.2, p. 1.2):

a *generative model*. This model defines a number of *hidden parameters* and describes how they generate documents. It typically comprises

- a model of how topics z are generated, defined by probability $P(z)$
- a model of how words w are generated knowing each topic z, defined by the probability $P(w|z)$

a *machine learning scheme*, derived from the model. It allows estimating to the parameters as to fit the observed data

a *document similarity*, derived from the model. It uses the parameters and their values to determine how similar two elements of the observed data are.

For instance, the so-called "Naive Bayes" model assumes that for each document d, a topic z is draw with multinomial probability $P(z)$, and that the words to appear in this document are then drawn with probabilities $P(w|z)$ (see section 3.2.1, page 38 for more details). Probabilistic Latent Semantic Indexing (PLSI) [27] is another example of probabilistic latent semantic models, in which the generative model furthermore comprises an element that generates document indices, in addition to topics and to words knowing a given topic (see section 3.3.1, p. 42 for more details).

Table 1.1 provides a classification of a few probabilistic latent semantic models, according to the nature of their "topic generator" and of their "word generator". For the Machine learning part of the system, we used standard schemes, namely the Expectation-Maximisation algorithm, which can be derived from the generative model. Finally, the semantic similarity constitute the essential part of this work (chapters 4 and following).

To benefit from the representations based on latent topics in the framework of a task such as Information Retrieval or Text Classification, ways to compare the document representations between each others are needed. In the vector space representation of documents, defined from the number of occurrences of each word in each document, semantic similarities between documents are typically measured through the use of standard geometric similarities, like cosine. In the case of stochastic models, the semantic similarity should be founded on the generative process; *Fisher kernel* constitute a good candidate in this respect [29, 2].

The present thesis contributes to the field of probabilistic latent semantic models by studying the associated document similarities; it also proposes an approach to the "word knowing topic" aspect of generative models based on the Smoothed Dirichlet distribution. Information Retrieval was chosen as

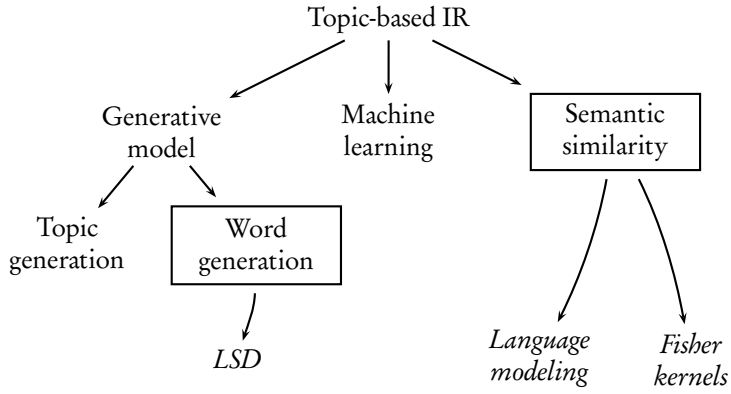


Figure 1.2: Components of a PLSM-based IR system: the generative model, which comprises a topic generator and a "word knowing topics" generator, defines parameters and the generation process, and determines a machine learing scheme and similarities. The machine learing scheme, derived from the generative model, fits values the of the parameters to observed document collection. The semantic similarity, derived from the generative model, uses the parameters to compare documents of the observed collection. The elements of the system addressed in this work are framed; the solutions studied for these elements are in italics.

an evaluation framework, though the methods exposed thereafter can be applied to a range of other textual or non-textual tasks such as clustering, categorisation, etc.

In the course of the work, we formally develop the Fisher kernel for PLSI from the ground up, uncovering its rigorous form stemming from the IID nature of the PLSI generative process. This completes the previously published study of PLSI Fisher kernels. We relate the IID-based kernel to previously published Fisher kernels for PLSI by explicitly giving transformations that lead from the IID-based kernel to previously published kernel.

We study in details the relative contributions of the different parts of the Fisher kernel for PLSI, as well as of the role of the Fisher information matrix, and quantify their contribution to the performances of the overall system. this explains previously published results and determines which version of the Fisher kernel is the best to use as a document similarity in a PLSI framework.

We evaluate a language model approach to PLSI similarities, and provide a similarity based only on observable features of the query. Reliance on only observable features of the query dispenses from the need for latent parameters for the query; this constitutes a theoretically grounded manner of bypassing the problem of "folding in" the queries.

We also herald the approximations needed to make the exact form of the PLSI Fisher kernel convenient for computation, and to link it to previously published literature.

The second contribution of this thesis is *Latent-based Smoother Dirichlet* (LSD, see table 1.1), a probabilistic latent semantic model where the generative model is constituted of a mixture of Smoothed Dirichlet distributions. his provides a fully generative latent topics model built on a strong model of word generation. We provide the generative model, posterior inference of the parameters and the Fisher kernel appropriate for it.

1.2 Structure of the thesis

This thesis is structured in three parts:

Generation of TOPICS through PROPORTIONS	TERMS: Multinomial	TERMS: DCM [51]	TERMS: **Smoothed Dirichlet** [58]
$P(z)$ given as parameter	**Naive Bayes** **PLSI** [27]	–	**LSD**
$P(z)$ generated by Dirichlet distribution	LDA [7]	DCMLDA [16]	—
$P(z)$ generated by Logistic normal distribution	Correlated Topics [6]	–	—

Table 1.1: Generative models classified by their generation of latent topics and of words, with the models addressed in this thesis in bold face. Naive Bayes is detailed in section 3.2 (p. 37); PLSI, in section 3.3 (p. 42); and Smoothed Dirichlet and LSD, in chapter 6 (p. 71).

1. The first part describes the context and previous work:
   - Chapter 2 gives an outlook of the context of the systems that we use as a benchmark.
   - Chapter 3 describes the Naive Bayes and PLSI models in detail, notably including the derivations of the posterior inference through EM and the formerly published Fisher kernels.
2. The second part contains our major contributions:
   - Chapter 4 describes the Fisher kernel that stems directly from a literal application of the generative process of PLSI. Its connections to previously published kernels are studied and explained.
   - Chapter 5 studies the individual contributions of the different constituents of the Fisher kernel. It also details legacy kernels published in previous works and put them in perspective from each other.
   - Chapter 6 proposes a novel probabilistic latent semantic model called *Latent-based Smoothed Dirichlet*; it uses a mixture of Smoothed Dirichlet distributions to model the document collection in a completely generative manner.

   Chapter 7 provides experimental results for the previous chapters.
3. The last part, constituted by chapter 8, concludes this work and proposes outlooks.

1.3 Contributions

The foremost contributions of this thesis are

- a formal derivation of the Fisher kernel for PLSI, uncovering its rigorous form stemming from the IID nature of the PLSI generative process; in addition to completing the previously published study of PLSI Fisher kernels, this puts previously published kernels into perspective and explicits which transformations lead from the IID-based kernel to previously published results (chapter 4, page 53).
- application of the language model approach to PLSI similarities, providing a similarity based only on observable features of the query. This has the effect of bypassing the problem of "folding in" unknown documents in a theoretically grounded manner (section 7.5, page 85);

- a study of the relative contributions of the different parts of the Fisher kernel, as well as of the role of the Fisher information matrix; this explains previously published results and determines which version of the Fisher kernel is the best to use as a document similarity in a PLSI framework (chapter 5, page 63).
- a proposition for a novel, fully generative probabilistic latent semantic model called Latent-based Smoothed Dirichlet, where the generative model is constituted of a mixture of Smoothed Dirichlet distributions. We provide the generative model, posterior inference of the parameters and the Fisher kernel appropriate for it (chapter 6, page 71).

1.4 Publications

This thesis has yielded four publications:

1. Jean-Cédric Chappelier and Emmanuel Eckard. "Rôle de la matrice d'information et pondération des composantes dans les noyaux de Fisher pour PLSI". In *Actes du colloque Coria 09*, pages 279-294.
2. Jean-Cédric Chappelier and Emmanuel Eckard. "Utilisation de PLSI en recherche d'information", 16ème conference sur le Traitement Automatique des Langues Naturelles, 2009
3. Jean-Cédric Chappelier and Emmanuel Eckard. "PLSI: the true Fisher Kernel and beyond". In *Proceedings of the European Conference on Machine Learning and Principles and Practise of Knowledge Discovery in Databases 2009*.
4. Jean-Cédric Chappelier and Emmanuel Eckard. "An Ad Hoc Information Retrieval Perspective on PLSI through language model identification". 2nd International Conference on the Theory of Information Retrieval, Cambridge, 2009.

Chapter 2

Context and state of the art in document representation

Abstract

This chapter is intended to give an overview of the context in which the present work is positioned. It present the problematic of Information Retrieval and Text Classification, describes the toolchains used in this field, and briefly introduced Vector Space and latent topic-based Information Retrieval, along with document similarity measures. Eventually, it lists software tools that provide some of these components, as well as ressources for evaluation.

2.1 Introduction

This thesis positions itself in the framework of Information Retrieval (IR) [84], and more specifically of latent-based IR. It studies the semantic similarities adapted to Probabilistic Latent Semantic Indexing (PLSI), selected as an instance of stochastic latent semantic model.

IR consists in recovering documents of a collection that match a given query; Text Clustering and Text Classification (TC) [71] consist respectively in regrouping documents of similar semantic content, and in distributing documents amongst pre-defined semantic categories. IR and TC share a number of techniques [71] such as *indexing*, which transforms documents into objects usable by a computer; *document similarity*, which allow comparing documents and judging how semantically close they are to each other; and *evaluation*, which compares the output of the system with a Human-made reference to judge how well the system performs.

It has been argued that Information Retrieval systems *are* in fact classification systems[45, 57], where documents are classified into the "relevant" and "non-relevant" classes. However, fundamental differences do exist:

1. information retrieval depends on queries that are a priori unknown. This can entail specific consequences, notably when document representation is ill-defined for queries.[1]
2. queries have specific traits, well distinct from documents. One notable aspect is *short queries*, which need to be expanded (*query expansion* [67]) before good results can be obtained.

[1] This is the case, for instance, for dimensionality reduction models that are not entirely generative, like PLSI. The learning phase constitutes a representation process defining a latent space into which queries must be projected — a non-trivial problem when this operation is not clearly defined.

There is no question that text classification and information retrieval are very similar, enough to permeate each other — for instance, variants of query expansion have been used in TC [94]. Nevertheless, if a specific trait of IR queries can be blurred by circumstances or by a point of view, the fact that they are a priori unknown yields fundamental differences from the known documents in TC.

Recently, beginning with the Latent Semantics Indexing (LSI) [15], Information Retrieval has been performed by transforming the word vector space as to make it more favourable to Information retrieval, notably by extracting latent semantic features of the text.

Probabilistic models of a collection of documents can be constructed by providing simple hypothesis on "what causes documents to happen", or "how they have been generated". For example, Hoffman's Probabilistic Latent Semantic Indexing (PLSI) model postulates that documents collections are constituted of documents d, terms w and topics z. Documents are considered to be observable realisations from which the hidden parameters $P(z)$, $P(d|z)$ and $P(w|z)$ can be inferred through Expectation-Maximisation. These parameters are then used in document similarities that allow IR to be performed.

2.2 The Information Retrieval processing chain: a functional breakdown

IR systems aim to process data that is in nature intended for human usage; this data is usually not written with computers in mind, and needs some sort of formating before becoming exploitable by a machine. This formatting phase, called *indexing*, may contain a number of steps: recognising where words begin and end, or identifying singular and plural forms as occurrences of one same word, are typical examples of such tasks.

The toolchain for IR or TC is composed of a sequence of steps that depends on the application (more precisely, depending on the intended result, available data , language in which the data is written, ...).

A toolchain for IR in English will typically perform the following steps:

1. translate the text of each document into a mathematical object $D \in S$, S being the representation space. For instance, a good and often-used document representation D is a vector of term frequencies.

2. translate the query into a similar mathematical object $Q \in S$.

3. choose a semantic similarity, i.e. function $f : S \times S \to \mathbb{R}$ of a pair of indexed documents; compute the value of the function of each (document – query) pair and associate the result to the (document – query) pair.

4. compare the results of the function for all (document – query) pairs given a pre-determined criteria; for each query, order the (document – query) pairs.

5. documents are ordered

6. for each query, return the corresponding list of documents

Each of these steps can be refined to some extend. For instance, before the first step, it might be relevant to ponder whether or not we wish to distinguish between two different forms of the same word (plural or singular of one work, or accusative vs nominative); between words of the same family, like nouns, verbs, adjectives; between several different spellings of a same word. This first step is called *indexing*.

Indexing consists in associating documents to points in a high-dimensional representation space. This step is what allows an actual semantic metric to be used, rather than performing a mere comparison as to whether words occur or not in a particular text [5]. Indexing entails two choices: that of a set of *indexing terms*, and that of their *weighting*. Indexing terms are the vocabulary tokens used to describe a document — typically "words", but more accurately stems, noun phrases, etc.; term weighting expresses the relative importance that is given to one indexing term with, compared to others, as some indexing term appear more sense-bearing than other.

Families of different indexing schemes can be typified along two criteria [45]:

- choice of indexing terms: human or machine;
- number of terms: indefinite or fixed a priori.

In indexing with an indefinite number of terms, new terms can be added on the fly as new documents are being indexed, and their contribution can be accounted for straightforwardly. In indexing with a fixed number of terms, the system does not attempt to account for new terms. In this case, the terms of the vocabulary can be chosen either before indexing, or during indexing; but if adding a document to the index is possible, new terms that it might be bearing would be ignored.

Polysemy, synonymy and term relevance field[2] hinder the implementation of algorithms before representation [45]. A naive approach consists in using surface forms of the terms, without accounting for polysemy or synonymy, either with binary weighting (accounting only for the presence or absence of a word), or more advanced representation that uses statistical features, like the number of times a word appears in a document (*term frequency*) or the number of documents in which a word appears (*document frequency*). This naive approach is extremely cost-effective: as much in IR [69] as in TC, it has been shown that the most sophisticated representations do not necessarily yield better results than those based on term frequency and document frequency [76, 17, 14].

Using *phrases* or passages as indexing terms has been tried. Phrases can be defined either syntactically [43] or statistically [12], using terms which statistically tend to appear close together rather than terms which bear a syntactical meaning. Research in these directions is underway [53, 54, 1].

To illustrate the indexing phase, we take a sample of text and apply typical indexing steps. The first articles of the Universal Declaration of Human Rights on 1948 go

> All human beings are created free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood. Everyone is entitled to all the rights and freedoms set forth in this Declaration, without distinction of any kind.

Tokenisation recognises the beginning and ends of the words and separates them; in the English version of the text, these positions are clearly marked by punctuation marks, and this step is straightforwards and leaves the text unchanged; it would no be so with the Chinese version.

Normalisation removes punctuation marks and capitalisation, yielding

> all human beings are created free and equal in dignity and rights they are endowed with reason and conscience and should act towards one another in a spirit of brotherhood everyone is entitled to all the rights and freedoms set forth in this declaration without distinction of any kind

Stopwords are words that are listed as items to ignore. They comprise the most commons of words bearing no semantic content, like determiners. Their removal yields

[2] That is the semantic field within which a term has a particular meaning, a question of particular interest for technical terms.

> human beings created free equal dignity rights endowed reason conscience should act towards another spirit brotherhood everyone entitled rights freedoms set forth declaration distinction kind

Stemming consists in reducing words to their semantic core, and removing inflexions like plural, cases, or verbal forms. Stemming by Porter's stemmer [85, 80] would yield

> human be creat free equal dign right endow reason conscien should act towards another spirit brotherhood everyone entitl right freedom set forth declaration distinct kind

On this basis, a mathematical object representing the document can be constructed.

2.3 IR models based on observable features

In IR, like in TC, documents are usually represented by vectors of weights associated to indexing terms [71].

Coordinates in the representation space are a function of whether an indexing term appears in the document or not; is some cases, of how many times they appear within a given document; and of how they populate the document collection.

The "bag of word" assumption consists in ignoring the position of words in a document. The representation of the document is degraded from a a *sequence* of words into a *multiset* (or bag), and the semantic content of the document is determined only by its term frequencies. To further indexing of the sample of the Universal Declaration of Human Rights, a bag of word representation of the stemmed Declaration would turn it into

(`declar`, 2) (`distinct`, 2) (`human`, 2) (`polit`, 2) (`right`, 3) (`status`, 2)

In this case, the terms are weighted by term frequency and the words appearing only once have been eliminated.

Making the Bag of Words assumption has consequences on the probability of a modeled document. For instance, a very simple model is an independent identically distributed (IID) process generating unigrammes (single words), and its probability is

$$P^{\text{IID}}(d) = \prod_{w \in V} P(w)^{n(w,d)}.$$

A similar model is the multinomial model, which generates documents with probability:

$$P^{\text{multinomial}}(d) = \frac{|d|!}{\prod_{w \in V} n(w,d)!} \prod_{w \in V} P(w)^{n(w,d)}$$

What distinguishes the IID generation of unigrammes from the multinomial model is the Bag of Words assumption, which alters the normalisation of the distribution as the order of the sequence becomes irrelevant.

The Bag of Words assumption is the basis for different forms of IR, notably *Boolean IR* and *Vector Space IR*.

Boolean IR attributes a binary weight (`0` or `1`) to a term, according to whether is appears in a document or not. The type of weighting allows performing boolean algebra with keywords. For instance, the keywords associated with the sample of the Universal Declaration of Human Rights would be

(`declar`) (`distinct`) (`human`) (`polit`) (`right`) (`status`)

Vector Space IR associates documents with vectors of a vector space whose dimensions represent the words contained in the collection. Comparison between two documents can then performed using similarities based on cosines.

The `tf-idf` weighting is a traditional and popular numerical weighting for Vector Space IR [79]. The `tf` refers to the t*erm* f*requency* within a given document. `idf` refers to the i*nverse* d*ocuments* f*requency* of the term[3]. A number of different flavours of `tf-idf` exist [74].

BM25 is a more complex similarity based on probabilistic retrieval models [66]. Its latest variant, BM25F, is considered as state-of-the-art in the IR community [64].

Research continues to find similar, more efficient weighting of the same nature [79].

2.4 IR models based on latent features

Using information based on latent semantic features could be a way to improve the retrieval by matching documents of the same context even if they share few common terms, by disambiguating between several meanings of a same surface form, and generally to overcome the problems posed by polysemy and synonymy.

The latent semantic method relies on the notion that *latent topics* can be inferred from the distribution of terms amongst documents; by relying, implicitly or explicitly, on the co-occurrence of terms, the context or a particular occurrence can be determined, and matched to similar contexts.

Latent semantic features were introduced with the *Latent Semantic Indexing* (LSI), in an attempt to reduce the dimensionality of the representation space. Some learning systems rely on dimensionality reduction to classify documents — for instance Linear Least Squares Fit (LLSF) [91]. In general, learning algorithms respond well to dimensionality reduction, as it tends to reduce problems of overfitting.

Dimensionality reduction can be spread amongst two families [44]:

- *Feature selection* : the reduced indexing space is constituted of a sub-set of the dimensions of the original indexing space. An example of such a configuration would be to select the most significant indexing terms.

- *Feature extraction* : the reduced indexing space is constituted of a new space, different from the original indexing space, and possibly of a completely different nature. For instance, LSI produces an index whose dimensions are "fusions" of the indexing terms of the original space, through linear combination amongst similar semantic fields.

Feature selection: Dimensionality reduction through term selection aims at extracting the subset of the original dimensions that contains most of the sense of the documents.

A straightforwards example is filtering terms according to their importance, through a numerical criterion: document frequency, or more sophisticated criteria based on Information Theory. Notable examples include DIA association factor [21], Chi-square [12, 22, 70, 72, 93, 92], NGL coefficient [59, 68], Information gain [12, 40, 43, 46, 55, 56, 93, 92], Mutual information [17, 39, 41, 46, 47, 56, 68, 81, 93], Odds ratio [12, 55, 68], Relevancy score [90], GSS coefficient [22] and DCA [10].

Feature extraction: Dimensionality reduction through term extraction reduces the orthogonality between dimensions that stem from the Standard Model by "blending" the dimensions that bear the same concepts into another. LSI fathered a lineage of models based on this principle. Term extraction allows for better management of synonymy [18].

[3] i.e. the inverse of the number of documents that contain one of more occurrences of the term.

The foremost representants of this trend are LSI and its derivatives, as well as Term clustering, which can be performed either supervised or non-supervised[4]. The following provides a short description of the most notables members of the Term extraction trend.

Latent Semantic Indexing (LSI) is a techniques of dimensionality reduction that creates a lower-dimensionality space by performing a Principal component analysis over the occurrence matrix representing the corpus of documents and keeping the N most significant dimensions. As such, it is an example of term extraction, since the resulting dimensions are linear combinations of the dimensions of the original space.

The performances of LSI match and often surpass those of term selection through χ^2 [70]. It has entailed a number of developments, such as Polynomial filtering, a matrixless variant [33]; preprocessing optimisations [82]; and non-linear structures [25].

Probabilistic Latent Semantic Indexing (PLSI, described in much deeper details in the next chapter, section 3.3, page 42) is also a technique of dimensionality reduction by term extraction. It is based on the postulate that documents are realisations of mixtures of conditional probabilities over topics, documents, and terms. Estimated by machine learning techniques performed on the document collection, these probabilities constitute parameters from which IR or TC systems can be built.

After the introduction of the PLSI model, the document similarities used to compare PLSI representations of document have quickly evolved from ad hoc similarities based on cosine [27] to the Fisher kernel [28]. To address a number of practical and theoretical concerns, a variety of flavours of Fisher kernels for PLSI have been proposed [61] (see also the kernels listed in chapter 5).

This thesis is largely focused on the Fisher kernels for PLSI. A detailed review of the kernels of the literature is provided in section 3.3.3 (p. 46), preceding the main contributions that we provide on the topic, starting from chapter 4 (p. 53).

PLSI is not a genuinely generative model, as it does not provide a generation scheme at the level of the document. To remedy this issue, Blei et al. [7] proposed Latent Dirichlet Allocation (LDA), a generative probabilistic model that attempts to describe exchangeability of both words and documents, i.e. their IID nature conditional to a latent topic. LDA defines finite mixture models on latent topics to generate the collection, every subject being modeled as a mixture over the probabilities of latent topics.

It has been shown that PLSI is a maximum a posteriori estimated LDA model under a uniform Dirichlet prior [23]. LDA is little sensitive to its Dirichlet priors [62].

2.5 Similarity functions for documents

Given document representations in a mathematical space, IR or TC tasks can be performed with a *document similarity function*.

2.5.1 General principles

In the 1980s, Bollman published a study of metrics between documents [9, 8], citing six criteria that a "reasonable retrieval function" should fulfil [19]:

1. *order relation* for pairs of documents: for any two documents of the same length, given a one-term query, the document containing the most occurrences is ranked highest;

[4] Of course, dimensionality reduction by term clustering would amount to term selection if the resulting clusters are centred on the indexing terms of the initial representation space.

2. *order relation* for differences between the term frequencies in three documents;

3. *term discrimination*: documents containing more occurrences of discriminating terms are favoured;

4. relation on influence of out-of-query terms;

5. relation on *document length* for two documents with the same term count for query terms;

6. constraint between `tf` and the document length;

These amount to two constraints on term frequencies (relations 1 and 2), a constraint on term discrimination (3), two constraints on document length normalisation (4 and 5), and one constraint on the interaction between `tf`.

Concretely, these relations are a formal description of the following intuitions:

- documents with the most occurrences of a desired term are favoured (1);
- documents that match several terms are favoured (2);
- a rise from 1 to 2 occurrences has more impact that from 100 to 101 (2);
- the respective effects of `tf` and `idf` are regulated and balanced (3);
- for a same term frequency, a shorter document will be favoured (4), but to a "reasonable" extend (5,6);
- the respective effects of `tf` and $|d|$, the document length, are regulated and balanced (6);

These criteria have been put in practise with the numerous variants of `tf-idf` of SMART[5] or in BM25 [66], notably.

In the last decade, an important development of document similarities for probabilistic models in IR has been based on the "Language model" approaches, presented in the next section.

2.5.2 Language models

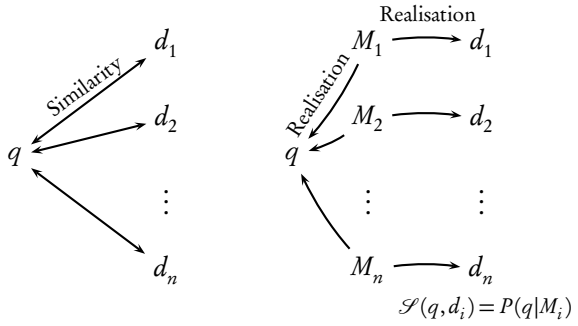
Language models [63, 96, 48] consist in a probabilistic profiles over the vocabulary. To a document d, language models associate a probability distribution over the vocabulary, $M_d = \{P(w|d) : w \in V\}$.

In the IR context, a query, represented as an empirical term distribution is considered to be a realisation of the document model M_d of one of the documents of the collection [4] (see figure 2.1). The document model distribution may be based on unigrammes, or on model accounting for positional information, such as passages [50]. The search for the most relevant document d for a given query q then amounts to maximising $P(q|d)$ over the d. This can take two courses:

1. Keeping in line with the assumption that the observed query $\widehat{q}$ be a realisation of M_d, the most relevant document can be chosen as the one that maximises the likelihood of this realisation: $L_d(\widehat{q}) = \log P(\widehat{q}|M_d)$ [88, 4], with

$$L(\widehat{q}) = \prod_{w \in q} P(w|M_d)^{n(q,w)}.$$

[5] see `http://people.csail.mit.edu/jrennie/ecoc-svm/smart.html`



(a) Classical IR (b) Language models

Figure 2.1: Classical IR (2.1a) chooses relevant documents using canonical document similarities; it compares documents and queries representations of the same nature. Language models (2.1b) infer the probabilistic models, of which documents are assumed to be realisations; the query is considered to be a realisation of these models, and the document whose model better explains the query is chosen as a best match.

2. The other course, equivalent in nature to the former [37], measures the divergence between the empirical distribution $P(\widehat{q})$ and a probability distribution of reference $P(d)$. The canonical example for such a measure is the Kullback-Leibler divergence [95, 35]

$$\mathscr{S}_{\mathrm{KL}}(d,q) = -\mathrm{KL}\left(\widehat{P}(w|q), P(w|d)\right) = \sum_{w \in q \cap d} \widehat{P}(w|q) \log \frac{P(w|d)}{\widehat{P}(w|q)}, \tag{2.5.1}$$

where $\widehat{P}(w|q) = n(q,w)/|q|$, the number of occurrences of word w in query q divided by the length $|q|$ of query q.

Smoothing

The former straightforwards expressions of $\mathscr{S}_{\mathrm{KL}}$ and $\mathscr{S}_{\mathrm{LogL}}$ use the normalised raw term count $\widehat{P}(w|q) = n(w,q)/|q|$ as estimator for $P(w|q)$. However, estimating the query model has been the subject of much work, that aim at preventing unseen features from being attributed a zero probability [36]. Proposed solutions include Jelinek-Mercer smoothing [32], shrinkage style estimators [63], Good-Turing estimate, used on higher order n-grammes [78], and a two-stage smoothing model [97]; clusters have also been used for smoothing [36], produced by k-means [48] or LDA [88]. Models provided by probabilistic latent semantic models, such as PLSI, can be used for smoothing, combining with the empirical probability estimates [4].

To illustrate, we describe the two simplest smoothing schemes:

Jelinek-Mercer smoothing is a way to balance between the normalised term count of a document (or, here, of a query) and the normalised term count amongst the entire document collection. The unconditioned ("general English") probability of a term w is estimated, for instance, as

$$P_{\mathrm{GE}}(w) = \sum_{d \in D} \widehat{P}(w,d) = n(C,w)/|C|, \tag{2.5.2}$$

where $n(C,w) = \sum_{d \in D} n(d,w)$ is the number of occurrence of word w in the entire corpus C, and $|C|$, the size of the corpus, is $|C| = \sum_w n(C,w)$. Using the above, the Jelinek-Mercer

estimator $\widetilde{P}(w|q)$ of $P(w|q)$ is written

$$\widetilde{P}(w|q) = (1-\lambda)\widehat{P}(w|q) + \lambda P_{\mathrm{GE}}(w), \tag{2.5.3}$$

with $\lambda \in [0,1]$.

Pseudo-feedback is another way to build a smoothed estimator for $P(w|q)$ [11, 38]. The idea is to estimate $P(w|q)$ from the top ranking documents from an initial retrieval phase, as done for instance in the Relevance Model [42]: a first retrieval is performed and the N highest documents of the rating, with a rather small N, are then used to estimate

$$\widetilde{P}^{\mathrm{PF}}(w|q) = \frac{1}{N}\sum_{i=1}^{N} P\left(d_i(q), w\right) \qquad \text{with } d_i(q) : i^{\text{th}} \text{ best match.}$$

Finally, a second retrieval phase is performed with the normalised term count replaced with $\widetilde{P}(w|q) = (1-\lambda)\widehat{P}(w|q) + \lambda\widetilde{P}^{\mathrm{PF}}(w|q)$. In recent work, λ is learnt automatically for every query [49].

Smoothing can be applied either to $\mathscr{S}_{\mathrm{LogL}}$ or $\mathscr{S}_{\mathrm{KL}}(d,q)$. Furthermore, the pseudo-feedback approach can be combined with the "direct" smoothing *à la* Jelinek-Mercer: the first retrieval can indeed use either raw term frequencies, or smoothed estimators.

2.5.3 Fisher kernels

In the classical approach of IR (section 2.5.1, p. 26), when documents are represented in the vector space model, similarities that rely only on observable features of documents can be used. However, when the document model stems from a probabilistic generative process, the *Fisher kernel* provides a similarity that also derives from the same generative process [76, 75].

In all generality, a *kernel* is a similarity measure between two elements X and Y in a vector space V; the kernel constitute a scalar product in an alternative space W. Given $s : V \to W$ a function casting $X \in V$ to its representation in the W space, $K(X,Y) = s(X)\cdot s(Y)$.

The *Fisher kernel* was defined in [29] as

$$K(d,q) = \nabla L_d(\theta) \quad G^{-1}(\theta) \quad \nabla L_q(\theta) \tag{2.5.4}$$

It can be derived by going through the following steps:

1. Compute the Fisher score: gradient over θ of the log-likelihood

$$U_X(\theta) = \nabla_\theta \log P(X|\theta) \tag{2.5.5}$$

where $P(X|\theta)$ is a generative probability model(X = observable data, θ = model parameters)

2. Compute the Fisher information matrix, expectation over X of the covariance matrix of the score:

$$G(\theta) = \mathbf{E}_X\left[U_X(\theta)\, U_X(\theta)^T\right] \tag{2.5.6}$$

It constitutes a metric tensor for the probabilistic representation space, characterising its topology.

3. The Fisher Kernel is given by

$$K(X_1, X_2) = [U_{X_1}(\theta)]^T \, G(\theta)^{-1} \, U_{X_2}(\theta)$$

Fisher kernels [31, 30] are an application of kernels to probabilistic models, defined as

$$K(X, Y) = U_X(\theta)^T G(\theta)^{-1} U_Y(\theta) \tag{2.5.7}$$

In probabilistic setting, they constitute a natural similarity measure, as they approximate the Kullback-Leibler divergence in the neighbourhood of a point θ of the probabilistic space:

$$KL\Big(P(\theta), P(\theta + d\theta)\Big) \simeq \frac{1}{2} d\theta^T G(\theta) d\theta$$

In the case of PLSI, the form of the kernel was first proposed by Hofmann [28]; this form of the kernel was studied by Nyffenegger et al. [61], yielding several alternative forms of this kernel that are detailed in depth in section 7.1 (p. 81). We take the mater further and propose a better grounded Fisher kernel for PLSI in chapter 4 (p. 53).

2.6 Application

Actually setting up an IR/TC system requires more than a theoretical framework: pre-processing the data (tokenising, stemming, ...) or simply efficiently holding and accessing it in memory necessitate software tools that require enormous amounts of work to write. Hence, reviewing the software already available and selecting a toolkit usually the first step when one wishes to build a complete system.

2.6.1 Tools

As of this writing, no particular software framework is accepted as *the* standard toolset for community-wide usage in Information Retrieval. A number of specific laboratories promote their own tools, developed in different programming languages, focusing on different aspects of Information Retrieval, and made available under a variety of different licences.

Adopting a software framework is critical for several reasons: efficient access, storage and handling of data is a complex, specific task which requires optimisations falling way beyond the scope of Information Retrieval research; indexing of data might require steps for lemmatisation, chunking, word sense disambiguation, etc. (see section 2.2, p. 22), which usually also falls outside of the scope of the intended research; off-the-shelf software provide complete toolchain allowing to perform experiments with known models, and hence allowing baseline comparisons. Software under Free licences ("open-source") is especially well-suited for these tasks because

- the absence of charges allows testing several tools;
- the availability of the code makes it possible (if not always easy) to implement new algorithms;
- the general availability of the software framework makes it possible for third parties to use applications developed during the research, if the application is suitable.

A general trait apparent in most available software is the effort made to produce applications capable of scaling up to hundreds of Gigabytes of data, in consistency with the Terabyte track of TREC [77, 13], which explores the behaviour of Information Retrieval models when used on very large quantities of data.

TREC[6] is an evaluation campaign to study the efficiency of Information Retrieval methods in English, co-sponsored by the National Institute of Standards and Technology (NIST) and the US Department of Defence. Its aim is to provide NLP researchers with sample corpora, queries and tools to benchmark retrieval systems, particularly on large test collections. Each of the collections consists of a set of documents, a set of questions, and a corresponding set of reference files ("right answers").

A review of the most proeminent software follows. Synthetic tables of the results are available as tables 2.1, 2.2 and 2.3.

General information	LEMUR	TERRIER	XAPIAN
Latest revision	21 December 2009 (Lemur Toolkit version 4.11 and Indri version 2.11)	11 March 2010 (Terrier 3.0)	14 February 2010 (Xapian 1.0.18)
Focus			Probabilistic retrieval; large databases
Developers	Carnegie Mellon and University of Massachusetts [a]	University of Glasgow (under Keith van Rijsbergen) [b]	Dr. Martin Porter, BrightStation PLC (originally Cambridge University) [c]
Licence	BSD	Mozilla	GPL
Language	C / C++	Java	C++
Extendable	Library	"Hooks" for custom modules [d]	Library
Indexing capacity			4×10^9 documents or 256 Terabytes [e]

[a] See `http://www.lemurproject.org/contrib.php`
[b] See `http://ir.dcs.gla.ac.uk/`
[c] See `http://www.xapian.org/history.php`
[d] See `http://ir.dcs.gla.ac.uk/terrier/doc/terrier_develop.html`
[e] see `http://www.xapian.org/docs/scalability.html`

Table 2.1: General comparison of Lemur, Terrier and Xapian

2.6.2 Specific software

Lemur Lemur is a toolkit for information retrieval and language modeling. It is licenced under the BSD licence, and can be obtained from `http://www.lemurproject.org/`. The programme is well-maintained, and has been used as a baseline in research [58].

Lemur provides six retrieval models: `tf-idf`, Okapi (BM 25), KL-divergence, InQuery, CORI collection selection, cosine similarity, Indri SQL. Additionally, the optional classification toolkit of Lemur provides an implementation of Hoffman's Probabilistic Semantic Analysis [28] — or at least its

[6] Text REtrieval Conference, `http://trec.nist.gov/`

Available IR models	LEMUR	TERRIER	XAPIAN
Vector Space	Vector Space	–	tf-idf
LSI	No	No	No
PLSI	Classification only	–	Not native
Boolean	–	–	"TradWeight"
Probabilistic	Okapi (BM25)	BM25F, Divergence From Randomness	Okapi (BM25)

Table 2.2: Comparison of available IR models in Lemur, Terrier and Xapian

Miscellaneous	LEMUR	TERRIER	XAPIAN
Executable application	Indri	Terrier	Omega
TREC compatibility	Native	Native	Not native
Packaging	tar.gz ; pre-compiled packages for MS Windows	tar.gz	RPM, DEB, FreeBSD

Table 2.3: Practical notes on Lemur, Terrier and Xapian

the Expectation-Maximisation learning part [7] : as such, queries cannot be processed easily without an operation of *folding in*, which is neither implemented in Lemur, nor very well-defined in the literature.

Lemur in itself is a library. The package provides a stand-alone Information retrieval engine known as *Indri*. Indri is parallelisable, can be used as a filter, and scales to the terabyte.

Lemur provides indexers able to read PDF, HTML, XML, and TREC syntax. UTF-8 is supported.

Lucene Lucene is an Information retrieval library. It is supported by the Apache Software Foundation[8] and is available under the Apache Software Licence from `http://lucene.apache.org/java/docs/index.html`.

Lucene was written in Java, but can be used with Delphi, Perl, C#, C++, Python, Ruby and PHP.

The LucQE Lucene Query Expansion Module allows using Lucene for TREC experiments[9].

Terrier (TERabyte RetrIEveR) Terrier is an IR system for large quantities of data. It is written in Java and published under the Mozilla Free licence[10].

Terrier is said to have "full TREC capabilities including the ability to index, query and evaluate the standard TREC collections, such as AP, WSJ, WT10G, .GOV and .GOV2."

Terrier provides tf-idf, Okapi's BM25 and Rocchio's query expansion. It has been tested to scale to all TREC collections.

Development tips are given at `http://ir.dcs.gla.ac.uk/terrier/doc/terrier_develop.html`. Terrier has a framework application, which defines a `main()` function; the Terrier user thus cannot use

[7] We have implemented another version of the learning algorithm on top of Xapian. Our implementation has been tested as giving more accurate results than the Lemur version. See section 2.6.2 and figure A.1, p. 114.

[8] Apache is the most widely used HTTP server on the World Wide Web. It is Free software, available under the Apache Software Licence.

[9] `http://lucene-qe.sourceforge.net/`

[10] Terrier is available from `http://ir.dcs.gla.ac.uk/terrier/download.html` (rather oddly for Free software, subscription is required); documentation is available at `http://ir.dcs.gla.ac.uk/terrier/documentation.html`

it as a standard C++ library, but has to write an `appmain()`" application. Options for the application are given as XML documents rather than with the command line.

Zettair Zettair is a textual Information retrieval engine published RMIT University under a BSD licence[11].

Zettair allows indexation of text, HTML and TREC formats. A tutorial is available at `http://www.seg.rmit.edu.au/zettair/start.html` Zettair outputs query logs in the TrecEval format.

Zettair has been tested on 426 GB database of the TREC Terabyte track.

Zettair is written in C.

Zebra Zebra is an indexation and retrieval engine available under the GPL from `http://www.indexdata.dk/zebra/`. Zebra was tested as scaling up to dozens of GB.

Zebra is written in C.

Xapian Xapian is an Information retrieval library focusing on probabilistic retrieval. A stand-alone retrieval engine named Omega is provided. Xapian is available under the GPL from `http://www.xapian.org/`. It is written in C++, and can be interfaced with Perl, Python, PHP, Java, TCL and C#.

Xapian provides pre-compiled software packages for the main Linux distributions (`rpm` and `deb`).

We decided to use Xapian as a basis for our work, as it is well-documented, easy to use as a library, and written in C++. Its licence and its native BM25 were also advantages. On the other hand, we had to write an indexer for TREC-formatted data, and the EM learning scheme for PLSI.

Evaluation: Treceval

After IR has been performed on a database, the results must be evaluated to judge of the quality of the system.

Among the ressources provided by TREC is TrecEval , a text utility to evaluate the efficiency of retrieval programmes, which has become a *de facto* standard among researchers. TrecEval allows evaluation of TREC results using the evaluation procedures of the NIST[12].

2.6.3 Resources

To evaluate the performances of experimental Information Retrieval systems, standardised collections of documents and queries are made available to research, along with referentials. The TREC conference issues guidelines regarding evaluations, usually requesting at 1000 documents to be returned for every query, and a minimum of 50 queries [87].

The following is a review of the collections used in this work.

Smart collection

The SMART collection is composed of six evaluation sets for information retrieval, each complete with a document collection, a query set and a referential (see table 2.6.3 for statistical information). The collection is available for no charge from `ftp://ftp.cs.cornell.edu/pub/smart/`; it is composed of

ADI: a 82-document collection. This collection is very small collection by modern standards, which can be useful for development and testing.

[11] available from `http://www.seg.rmit.edu.au/zettair/`

[12] the US National Institute of Standard and Technology, `http://www.nist.gov/data/nistsd22.htm`

	ADI	CACM	CISI	CRAN	MED	TIME
# Terms	2402	14181	16067	12029	20177	35619
# Stems	887	4911	5545	4063	7688	13367
V	2789	90927	87067	120973	76571	114850
Documents						
#	82	1587	1460	1398	1033	425
avg. $\|d\|$	31.0	56.8	56.7	85.1	73.8	268.6
Queries						
#	35	64	112	225	30	83
avg. $\|q\|$	7.1	12.7	37.7	8.9	11.4	8.2

Stemming was performed using the Porter algorithm of Xapian

Table 2.4: Statistics of the SMART collections

CACM: 3204 document abstracts from CACM journal. This collection provides only the titles of the documents.

CISI: 1460 document abstracts from the Institute for Scientific Information. This collection provides a specialised library science context. It is remarkable because some expected query-document matches are between queries and documents that share no significant term; it is thus particularly suited to test the extend to which a retrieval models is robust to synonymy.

CRAN: 1400 document abstracts from the Cranfield Institute of Technology. This collection provides a specialised aeronautic context.

MED: a collection of 1033 articles abstracts from Medline, the National Library of Medicine. This collection provides a specialised medical context.

TIME: a collection of 425 articles of *Time magazine*, with 83 queries. This collection provides a general English context, with an emphasis on political and economic news.

Reuters-21578

The Reuters-21578 collection is a categorisation corpus, composed of 21578 Reuters newswire articles. Being a TC ressource, it does not comprise queries. It is available from `http://daviddlewis.com/resources/testcollections/reuters21578/`. Reuters-21578 indexes with 100174 terms (31844 stems when stemmed through the Porter algorithm of Xapian).

The collection comes in 22 files, named `reut2-000.sgm` through `reut2-021.sgm`. Each file contains 1000 documents, except for the last one which contains the last 578 documents.

AP collection from TREC

The Trec AP collection[13] is a text retrieval annotated corpus. It is constituted of 242 918 news stories published by the Associated Press in 1988, 1989 and 1990. The data is provided on several CDs available from TREC for researchers only. A version of this collection also exists for text categorisation.

Queries are provided in separate 50-query files; they can be applied to any of the collections present on the CDs (unlike SMART query sets which are specific to a corpus). The matching referentials

[13] `http://www.daviddlewis.com/resources/testcollections/trecap/`

	AP88	AP89	AP90	ap89_0101-0109	ap89_01xx
# Terms	328 600	348 764	327 019		
# Stems	118 574	126 698	117 912	6 592	13 379
V	14 569 545	34 028 514	14 330 736	326 112	1 321 482
Documents					
#	79 919	84 678	78 321	1 896	7 466
avg. $\|d\|$	183.7	403.5	189.3	172.3	177.2

Table 2.5: Statistics of the AP collections from TREC

provide answers relevant to all TREC collections (the AP88, AP89 and AP90 data sets, but also DOE, FR, WSJ and ZF), and must be edited accordingly before use.

ap89_0101-0109 and ap89_01xx These two collection were custom-made during this work to provide datasets larger than the SMART bases, but still manageable with PLSI.

The `ap89_0101-0109` collection is a subset of AP89 comprising all the documents whose identification starts by 0101 through 0109 (documents AP890101-0001 to AP890109-0349). Similarly, ap89_01xx comprises all documents whose identification starts with 01 (documents AP890101-0001 to AP890131-0311).

2.7 Conclusions

IR has known an enormous rise in interest in the last decades due to the growth of information and the popularisation of the World Wide Web. Classical IR models produce mathematical representions of documents and queries from statistical traits such as the presence of number of occurrences of a word in the document and in the collection. An important component of IR system, at the core focus of this work, is document similarities, which measure the semantic association between queries and documents.

Another specific concern regarding semantic association between queries and documents is that the retrieval performances of classical IR systems are hindered by an overall difficulty to account for context. This notably yields drops in recall in the face of synonyms, or drops in precision in the face of polysemy [20, 86, 73]. To remedy this issue, the trend of latent semantics has attempted to describe document collections not only in terms of documents and words, but also in terms of topics. These topics, assumed to exist as a latent trait in the collection, must be acquired by machine learning techniques. Once the new representation have has been acquired through machine learning techniques, appropriate similarities come into play. We have seen that in the case of PLSI, a notable latent semantics model, issues remain open regarding its document similarities, notably Fisher Kernels [61]. This issue constitutes the core of this thesis.

The next chapter is devoted to an in-depth review of the PLSI model. The main contributions of this thesis follow in the latter chapters, answering the questions left unsolved regarding document similarities for PLSI, especially Fisher kernels.

Chapter 3

A review of notable models

Abstract

The previous chapter has described the Information Retrieval (IR) and Text Categorisation (TC) frameworks, and introduced the way in which IR based on latent topic works, with a generative model from which a learning algorithm and a document similarity are derived.

This chapter reviews in depth the generative models, the derivation of the learning algorithm and the derivation of the Fisher kernel for the mixture of multinomials, or Naive Bayes model, and for Probabilistic Latent Semantic Indexing (PLSI).

The study of PLSI and its Fisher kernel will allow us to take a new point of view on the Fisher Kernel for PLSI, detailed in the next chapter (p. 53).

3.1 Introduction

In the previous chapter, we presented a general overview of IR and TC frameworks, focusing on probabilistic latent semantic models. IR based on Probabilistic latent semantic models is three-fold:

- a generative model for the documents is given as an axiom or hypothesis; it defines latent parameters assumed to define how documents are generated.
- an inference process is derived from the generative model; it permits inferring the values of the parameters from an observable document collection.
- a document similarity is derived from the generative model; using the values of the parameters, it permits the retrieval of documents.

This chapter describes in details these three components for two models: the mixture of multinomials, or Naive Bayes model, a simple example; and PLSI, a notable model, at the core of this work.

3.2 Naive Bayes

"Naive Bayes" is the name sometimes associated to the latent topic-based generative model that models the document collection as a mixture of multinomials. Each document is assumed to stem from one of the latent categories, and the words are generated by multinomial distributions associated with the categories [65].

3.2.1 The Naive Bayes generative process

The Naive Bayes model assumes that for each document, a latent category is first chosen according to the category probabilities $P(z)$; terms are then generated by multinomial distributions $P(w|z)$ (see figure 3.1). Probability of occurrence of a document, as observed in the learning sample, is given by

$$P(d) = \sum_{z \in Z} P(d|z)P(z) \qquad \text{with} \qquad P(d|z) = \prod_{w \in V} P(w|z)^{n(w,d)} \tag{3.2.1}$$

The parameters of the model are thus $\Theta = \{P(z), P(w|z)\}$ for all $z \in Z$ and $w \in V$.

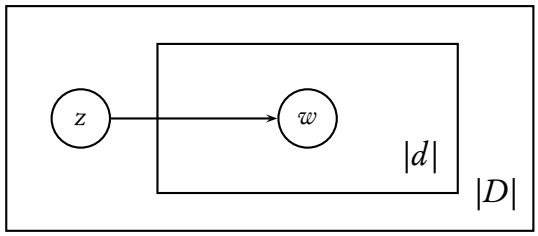


Figure 3.1: Graphical representation of the Naive Bayes stochastic model. For each document d in the document collection D, a category z is first drawn with probability $P(z)$; for each word in document d, a token d is drawn with probability $P(w|z)$.

3.2.2 Learning parameters of Naive Bayes

The parameters of Naive Bayes can be inferred, given the document collection, through an EM process in which the expectation and maximisation steps take the following forms:

E-Step

$$P(z|d) = \frac{P(z)[P(d|z)]^{\beta}}{\sum_{\zeta \in Z} P(\zeta)[P(d|\zeta)]^{\beta}}$$

with $P(d|z) \sim \prod_w P(w|z)^{n(d,w)}$. The parameter β is used to introduce simulated annealing in the inference process.

M-Step

$$P(z) = \frac{\sum_d P(z|d)}{\sum_{\zeta \in Z} \sum_d P(\zeta|d)} = \frac{\sum_d P(z|d)}{|D|}$$

$$P(w|z) = \frac{\sum_d n(w,d)P(z|d)}{\sum_{\omega \in V} \sum_d n(\omega,d)P(z|d)}$$

Proof

The Expectation-Maximisation algorithm aims at maximising the expectation of the joint likelihood of the observable over the parameters θ:

$$Q(\Theta,\Theta^{\text{old}}) = \mathbb{E}_Y\left[\log P(X,Y|\Theta)\middle|X,\Theta^{\text{old}}\right] \tag{3.2.2}$$

with X the observable and Y the hidden variables. In the case of Naive Bayes, the observable are the sequence of word realisations :

$$X = \underbrace{w_1,\ldots,w_{|d_1|}}_{\widehat{d_1}}, \underbrace{w_{|d_1|+1},\ldots,w_{|d_1|+|d_2|}}_{\widehat{d_2}}, \cdots, \underbrace{w_{|C|-|d_{|D|}|},\ldots,w_{|C|}}_{\widehat{d}_{|D|}}$$

while the hidden variables are the categories that generate these observables:

$$Y = \underbrace{z_1,\ldots,z_{|d_1|}}_{=z(\widehat{d_1}) \text{ by hypothesis}}, \underbrace{z_{|d_1|+1},\ldots,z_{|d_1|+|d_2|}}_{=z(\widehat{d_2}) \text{ by hypothesis}}, \cdots, \underbrace{z_{|C|-|d_{|D|}|},\ldots,z_{|C|}}_{=z(\widehat{d}_{|D|}) \text{ by hypothesis}}$$

For the entire collection, equation 3.2.2 becomes:

$$\begin{aligned}
&\mathbb{E}_Y[\log P(X,Y|\Theta)|X,\Theta^{\text{old}}] = && (3.2.3)\\
&= \sum_{z_1\in Z}\cdots\sum_{z_{|D|}\in Z} P(z_1,\ldots,z_{|D|}|w_1^{|C|},\Theta^{\text{old}})\cdot\log P(w_1^{|C|},z_1,\ldots,z_{|D|}|\Theta) && (3.2.4)\\
&= \sum_{z_1\in Z}\cdots\sum_{z_{|D|}\in Z}\left[\prod_{d\in D} P(z_d|d,\Theta^{\text{old}})\right]\cdot\left[\sum_{\delta\in D}\log P(\delta,z_\delta|\Theta)\right] && (3.2.5)\\
&= \sum_{z_1\in Z}\cdots\sum_{z_{|D|}\in Z}\sum_{\delta\in D}\left[\prod_{d\in D} P(z_d|d,\Theta^{\text{old}})\right]\cdot\log P(\delta,z_\delta|\Theta) && (3.2.6)\\
&= \sum_{\delta\in D}\sum_{z\in Z} P(z|\delta,\Theta^{\text{old}})\cdot\log P(\delta,z|\Theta) && (3.2.7)
\end{aligned}$$

We thus arrive at

$$Q(\Theta,\Theta^{\text{old}}) = \sum_{d\in D}\sum_{z\in Z} P(z|d,\Theta^{\text{old}})\log P(d,z|\Theta) \tag{3.2.8}$$

The Estimation step amounts to evaluating $Q(\Theta,\Theta^{\text{old}})$. The Maximisation step thus aims at finding

$$\Theta^{\text{new}} = \arg\max_\Theta Q(\Theta,\Theta^{\text{old}})$$

As the parameters of the model probabilise their respective spaces, we have

$$\sum_z P(z) = 1 \quad \text{and} \quad \sum_w P(w|z) = 1$$

We can introduce these constraints into the maximisation of $Q(\Theta,\Theta^{\text{old}})$ through Lagrange multipliers, creating the auxiliary function

$$\Lambda\Big(P(z),P(w|z),\lambda,\mu\Big) = Q(\Theta,\Theta^{\text{old}}) + \lambda\left(\sum_{w\in V} P(w|z) - 1\right) + \mu\left(\sum_{z\in Z} P(z) - 1\right).$$

Derivation by $P(z)$:

$$\frac{\partial \Lambda}{\partial P(z=\zeta)} = \sum_{d \in D} P(z|d,\Theta^{\text{old}}) \frac{\cancel{P(w|z)}}{\cancel{P(w|z)} P(z=\zeta)} + \lambda \tag{3.2.9}$$

$$\Longrightarrow P(z) \sim \sum_{d \in D} P(z|d) \tag{3.2.10}$$

Finally, we obtain

$$\boxed{P(z) = \frac{\sum_{d \in D} P(z|d)}{\sum_{\zeta \in Z} \sum_{d \in D} P(\zeta|d)}} \tag{3.2.11}$$

Derivation by $P(w|z)$:

$$\frac{\partial \Lambda}{\partial P(w=\omega|z=\zeta)} = \sum_{d \in D} P(z|d,\Theta^{\text{old}}) \frac{1}{P(d,z|\Theta)} \cdot \frac{\partial P(d|z,\Theta)}{\partial P(w=\omega|z=\zeta)} + \mu \tag{3.2.12}$$

$$= \sum_{d \in D} P(z|d,\Theta^{\text{old}}) \frac{1}{\cancel{P(d,z|\Theta)}} \frac{\cancel{P(d,z|\Theta)} n(d,w)}{P(w,z)} + \mu \tag{3.2.13}$$

$$\Longrightarrow P(w|z) \sim \sum_{d \in D} n(w,d) P(z|d,\Theta^{\text{old}}) \tag{3.2.14}$$

Finally, we obtain

$$\boxed{P(w|z) = \frac{\sum_{d \in D} n(w,d) P(z|d)}{\sum_{\omega \in V} \sum_{d \in D} n(\omega,d) P(z|d)}} \tag{3.2.15}$$

Notes on implementation

A practical difficulty can arise in the $P(d|z)$ term of the E-step because it contains a product of $P(w|z)$; as a product over terms all lesser than one (and, depending on the w and z couple, potentially considerably so), this term yields very low figures. In the eventual output, this very small term is divided by a sum of similar terms, resulting in figures commensurable to 1; however, computation of the intermediary figures is not tracktable for long documents.

One possible solution to this problem takes advantage of the basic premises of Naive Bayes. As Naive Bayes generates documents by first choosing a topic and then producing a burst of words, the $P(z|d)$ have in practise binary values: in effect, $P(z|d)$ is a mask that tells which unique latent category z corresponds to a given document d. Taking advantage of this property, a value of 1 could be attributed to the term that maximises $P(d|z)$ over z for a given d; a value of 0 would be attributed to all others terms.

However, we refrain from using this shortcut in our implementation, for two reasons:

- estimation of $P(z|d)$ could yield values between 0 and 1, reflecting uncertainty with respect to the data;
- we want to leave open the possibility to study models in which $P(z|d)$ has values other than 1 or 0 — i.e. actual mixture models, in which every *document* is itself a mixture of topics.

We thus propose another course of action: for a given document d, the $P(d|z)$ will be computed following the classical trick of re-normalising $P(d|z)$ with $\exp\left(\log P(d|z) - \log P(d|z_0)\right)$, where

$z_0 = \arg\max_z P(d|z_0)$. The term in $P(d|z_0)$ eventually cancels out later in the renormalisation.

More in details, $P(z|d)$ is obtained as

$$P(z|d) = P(z) \cdot \exp\left(\log P(d|z) - \log P(d|z_0) - P(d)\right) = P(z) \cdot \exp\Big(A(z) - B\Big),$$

where $A(z) = \log P(d|z) - \log P(d|z_0)$ and $B = P(d)$.

- a first loop over $z \in Z$
  - finds, z_0 the z such as $P(d|z_0) = \arg\max_z P(d|z)$
  - computes the values $A(z) = \log P(d|z) = \sum_w n(d,w) \log P(w|z)$
- the term $\log P(d)$ is computed with the help of a second loop over $z \in Z$, which computes $C = \sum_z \frac{P(z)}{P(z_0)} \exp\Big(A(z) - A(z_0)\Big)$. We then compute $B = \log P(z_0) + \log P(d|z_0) + \log(C) = \log P(d)$.

With this way of writing, C has the interesting property of measuring the degree to which the $P(z|d)$ form a mask: if $C = 1$, the $P(z|d)$ form a perfect mask; if $C = 1/|Z|$, then all the $P(z|d)$ are equal.

In practise, B is better computed as

$$\log P(d|z_0) + \log \sum_z P(z) \exp\Big(A(z) - A(z_0)\Big),$$

thus dispensing with the division by $P(z_0)$ and the addition of $\log P(z_0)$, which can induce numerical errors if $P(z_0) \sim 0$.

3.2.3 Fisher kernels for Naive Bayes

This section is devoted to a formal derivation of the Fisher kernel for mixtures of multinomials.

The log-likelihood for a document in the Naive Bayes model is

$$L = \log P(d) = \log \sum_{z \in Z} P(z) P(d|z) = \log \sum_{z \in Z} P(z) \prod_{w \in V} P(w|d)^{n(w,d)} \tag{3.2.16}$$

We apply a *square root re-parametrisation* by replacing

- $P(z)$ with $\rho(z) = 2\sqrt{P(z)}$
  entailing $P(z) = \frac{1}{4}\rho^2(z)$ and $\frac{\partial P(z)}{\partial \rho(z)} = \frac{1}{2}\rho(z) = \sqrt{P(z)}$
- $P(w|z)$ with $\rho(w|z) = 2\sqrt{P(w|z)}$
  entailing $P(w|z) = \frac{1}{4}\rho^2(w|z)$ and $\frac{\partial P(w|z)}{\partial \rho(w|z)} = \frac{1}{2}\rho(w|z) = \sqrt{P(w|z)}$

We obtain the Fisher score by gradient of the log-likelihood, that is by deriving L_d by $\rho(z)$ for all $z \in Z$ on one hand; and on the other, by deriving L_d by $\rho(w|z)$ for all (w,z) pairs.

The derivation by $\rho(z)$, we have

$$\frac{\partial L_d}{\partial \rho(z=\zeta)} = \frac{\partial L_d}{\partial P(z)} \frac{\partial P(z)}{\partial \rho(z)} = \sqrt{P(z)} \frac{P(d|\zeta)}{P(d)} = \sqrt{P(z)} \frac{P(z|d)}{P(z)} = \frac{P(z|d)}{\sqrt{P(z)}}$$

For the derivation by $\rho(w|z)$, let us begin by explicating the derivation by $P(w|z)$:

$$\frac{\partial L_d}{\partial P(w=\omega|z=\zeta)} = \frac{P(d|\zeta)}{P(\omega|\zeta)^{n(d,\omega)}} n(d,\omega) P(\omega|\zeta)^{n(d,\omega)-1} = P(d|\zeta)\frac{n(d,\omega)}{P(\omega|\zeta)}.$$

The complete derivation by $\rho(w,z)$, similar to that by $\rho(z)$, then gives:

$$\frac{\partial L_d}{\partial \rho(w=\omega|z=\zeta)} = n(d,\omega)\cdot\frac{P(\zeta|d)}{\sqrt{P(\omega|\zeta)}}.$$

Eventually, the Fisher score for a (d,q) couple is the sum of the above components for all w and z:

$$K^{\mathrm{NB}}(d,q) = \sum_{z\in Z}\frac{P(z|d)P(z|q)}{P(z)} + \sum_{z\in Z}\sum_{w\in V}\frac{n(d,w)n(q,w)P(z|d)P(z|q)}{P(w|z)}$$

3.3 PLSI

PLSI is a latent topic-based generative model for textual document classification and Information Retrieval, introduced in [27].

This section will review the generative process of PLSI, describing its parameters and how they are assumed to generate the observed documents. From the log-likelihood of the model, we derive the classical EM machine learning scheme for PLSI. Eventually, we derive the Fisher kernel as a document-query similarity.

3.3.1 The PLSI generative process

PLSI models the documents as realisations of a random choice of term-document pairs (d,w) knowing some topics $z \in Z$: a topic is first chosen with probability $P(z)$, then a term indice w and a document indice d are chosen, with probabilities $P(w|z)$ and $P(d|z)$ respectively, as w and d are assumed to be conditionally independent knowing z (see Fig. 3.2).

A realised document $\widehat{d}$ is then modeled as the set of all (d,w) pairs sharing the same document indice d: $\widehat{d_0} = \{(d,w) \in C | d \in D_0\}$. Because it holds documents indices to be part of its parameters, PLSI makes a distinction between *observed documents* and *document indices*.

The document indices d and words w are supposed to be independent knowing category z, i.e. $P(d,w|z) = P(w|z)P(d|z)$. Thus, a pair (d,w) is given by

$$\begin{aligned} P(d,w) &= \sum_z P(z,d,w) = \sum_z P(z)P(d,w|z) \\ &= \sum_z P(z)P(w|z)P(d|z) \end{aligned} \tag{3.3.1}$$

The parameters of the PLSI model are thus

- $P(d|z)$: probability of a document d knowing a category z;
- $P(w|z)$: probability of a term w knowing a category z;
- $P(z)$: probability of a latent category z.

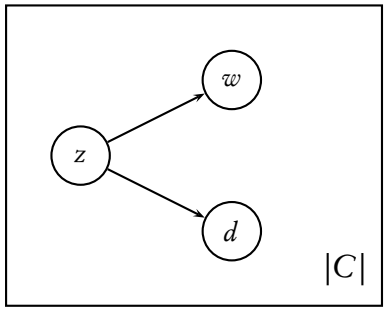


Figure 3.2: Graphical representation of the PLSI stochastic model. Category z chosen, then a (d,w) pair is chosen knowing z. w and d are furthermore assumed to be conditionally independent knowing z. The process is repeated a total of $|C|$ times, where $|C|$ is the number of term occurrences in the entire collection.

3.3.2 Learning parameters of PLSI

The parameters for PLSI are estimated using the EM algorithm, maximising the likelihood of the observed document collection C as follows :

E-Step To avoid overfitting, a tempered version of EM is used (TEAM), implementing simulated annealing for quicker convergence [27].

$$P_\beta(z|d,w) = \frac{P(z)[P(d|z)P(w|z)]^\beta}{\sum_{\zeta\in Z} P(\zeta)[P(d|\zeta)P(w|\zeta)]^\beta}$$

for some parameter β which is controlled during learning.

M-Step

$$P(w|z) = \frac{\sum_{d\in D} n(d,w)P(z|d,w)}{T(z)}, \quad \text{with} \quad T(z) = \sum_{d,w} n(d,w)P(z|d,w)$$

$$P(d|z) = \frac{\sum_{w\in V} n(d,w)P(z|d,w)}{T(z)},$$

$$P(z) = \frac{\sum_{w\in V}\sum_{d\in D} n(d,w)P(z|d,w)}{|C|},$$

where $n(d,w)$ is the number of occurrences of term w in document d, and $|C| = \sum_{d,w} n(d,w)$.

Proof Let us recall that EM aims at finding

$$\max_\Theta \mathbb{E}_Y \left[\log P(X,Y|\Theta)\middle|X,\Theta^{\text{old}}\right] \doteq \max_\Theta \sum_Y \log P(X,Y|\Theta)\cdot P(Y|X,\Theta^{\text{old}})$$

with X is the observable and Y, the hidden data.

In the case of PLSI, the observable is $X = \{(w_i, d_i) | i = 1, \ldots, |C|\}$, the set of all the (w_i, d_i) pairs that compose the document collection. The hidden variable is $Y = \{z_i | i = 1, \ldots, |C|\}$, the categories associated with the pairs of indices. In this notation, i denotes the indice of the experiment that generates first z_i and then (w_i, d_i).

To fix ideas, let us write the expectation for one single such experiment:

$$\mathbb{E}_z = \sum_{z \in Z} \log P(w, d, z) \cdot P^{\text{old}}(z|w, d)$$

Now, for a document collection comprising $|C|$ such experiments, we have

$$\mathbb{E}_Y = \sum_{z_1 \in Z} \cdots \sum_{z_{|C|} \in Z} \log P\Big((w_1, d_1, z_1), \ldots, (w_{|C|}, d_{|C|}, z_{|C|})\Big) \cdot P^{\text{old}}\Big(z_1, \ldots, z_{|C|} \Big| (w_1, d_1), \ldots, (w_{|C|}, d_{|C|})\Big)$$

The IID nature of the PLSI process allows us to write $P\Big((w_1, d_1, z_1), \ldots, (w_{|C|}, d_{|C|}, z_{|C|})\Big) = P(w_1, d_1, z_1) \cdot \ldots \cdot P(w_{|C|}, d_{|C|}, z_{|C|})$ and $P\Big(z_1, \ldots, z_{|C|} \Big| (w_1, d_1), \ldots, (w_{|C|}, d_{|C|})\Big) = P(z_1|w_1, d_1) \cdot \ldots \cdot \ldots \cdot P(z_{|C|}|w_{|C|}, d_{|C|})$ Thus,

$$\mathbb{E}_Y = \sum_{z_1 \in Z} \cdots \sum_{z_{|C|} \in Z} \left[\left(\sum_{i=1}^{|C|} \log P(z_i, w_i, d_i) \right) \cdot \left(\prod_{j=1}^{|C|} P^{\text{old}}(z_j|w_j, d_j) \right) \right]$$

For each experiment i, we can reunite the relevant $P^{\text{old}}(z_i|w_i, d_i)$ term with the $\log P(z_i, w_i, d_i)$, as

$$\mathbb{E}_Y = \sum_{z_1 \in Z} \cdots \sum_{z_{|C|} \in Z} \left[\sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i, d_i) \cdot \log P(z_i, w_i, d_i) \right] \cdot \prod_{j \neq i}^{|C|} P^{\text{old}}(z_j|w_j, d_j)$$

The sums $\sum_{z_1 \in Z} \ldots \sum_{z_{|C|} \in Z}$ that do not concern experiment i can then be moved inside the product. As each $\sum_{z_j \in Z} P^{\text{old}}(z_i|w_i, d_i) = 1$, the product cancels away, leaving

$$\mathbb{E}_Y = \sum_{i=1}^{|C|} \sum_{z_i \in Z} P^{\text{old}}(z_i|w_i, d_i) \cdot \log P(z_i, w_i, d_i)$$

Since, from PLSI, $P(z_i, w_i, d_i) = P(z)P(w|z)P(d|z)$, we can write a final expression of

$$\mathbb{E}_Y = \sum_{i=1}^{|C|} \sum_{z_i \in Z} P^{\text{old}}(z_i|w_i, d_i) \cdot \Big(\log P(z) + \log P(w|z) + \log P(d|z) \Big)$$

To maximise this quantity, we cancel the derivative, under the constraints

$$\sum_{z \in Z} P(z) = 1 \qquad \sum_{w \in V} P(w|z) = 1 \qquad \sum_{d \in D} P(d|z) = 1$$

The Lagrange auxiliary function to derive is thus

$$\Lambda = \sum_{i=1}^{|C|} \sum_{z_i \in Z} P^{\text{old}}(z_i|w_i,d_i) \cdot \Big(\log P(z) + \log P(w|z) + \log P(d|z) \Big) + \lambda \Big(\sum_z P(z) - 1 \Big) + \mu \Big(\sum_w P(w|z) - 1 \Big) + \nu \Big(\sum_d P(d|z) - 1 \Big)$$

Derivation by $P(z)$, at $z = \zeta$ given:

$$\frac{\partial \Lambda}{\partial P(\zeta)} = \sum_{i=1}^{|C|} P^{\text{old}}(\zeta|d,w) \frac{1}{P(\zeta)} + \lambda \doteq 0$$

entailing that

$$P(\zeta) = -\frac{\sum_{i=1}^{|C|} P(\zeta|d,w) \frac{1}{P(\zeta)}}{\lambda}$$

Knowing that $\sum_z P(z) = 1$, we can introduce this expression into the constraint to obtain

$$\sum_{z \in Z} -\frac{\sum_{i=1}^{|C|} P^{\text{old}}(z|d,w) \frac{1}{P(z)}}{\lambda} = 1 \longrightarrow \lambda = -\sum_{z \in Z} \sum_{i=1}^{|C|} P^{\text{old}}(z|d,w)$$

yielding an expression for $P(z)$ of

$$P(z) = \frac{\sum_{i=1}^{|C|} P(z|d,w)}{\sum_{z \in Z} \sum_{i=1}^{|C|} P^{\text{old}}(z|d,w)}$$

Let us now observe that the sum over the experiments, $\sum_{i=1}^{|C|}$, can be expressed as a sum over the occurrences of the different words in the different documents of the collection:

$$\sum_{i=1}^{|C|} P(z|d,w) = \sum_{w \in V} \sum_{d \in D} n(w,d) P^{\text{old}}(z|d,w)$$

Furthermore, $\sum_{z \in Z} P(z|d,w) = 1$. Eventually, we arrive at a final expression of

$$P(z) = \frac{\sum_{w \in V} \sum_{d \in D} n(w,d) P^{\text{old}}(z|d,w)}{\sum_{w \in V} \sum_{d \in D} n(w,d)} = \frac{\sum_{w \in V} \sum_{d \in D} n(w,d) P^{\text{old}}(z|d,w)}{|C|}$$

Derivation by $P(w|z)$, at $z = \zeta$ and $w = \omega$ given:

$$\frac{\partial \Lambda}{\partial P(\omega|\zeta)} = \sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i,d_i) \cdot \frac{1}{P(\omega|\zeta)} + \mu$$

yielding

$$P(\omega|\zeta) = -\frac{\sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i,d_i)}{\mu}$$

The value of μ is computed as above by introducing the expression above into the constraint $\sum_{w\in V} P(w|z) = 1$, yielding an overall expression for $P(\omega|\zeta)$ of

$$P(\omega|\zeta) = \frac{\sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i,d_i)}{\sum_{w\in V}\sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i,d_i)}$$

Now, for a given word ω the sum $\sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i,d_i)$ will run only over those of the w_i that happen to be ω:

$$\sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i,d_i) = \sum_{i=1}^{|C|} \delta_{w_i,\omega} P^{\text{old}}(z_i|w_i,d_i) = \sum_{w\in V}\sum_{d\in D} \delta_{w_i,\omega} P^{\text{old}}(z_i|w_i,d_i)$$

Clearly, the sum over the w can be removed to yield

$$\sum_{i=1}^{|C|} P^{\text{old}}(z_i|w_i,d_i) = \sum_{d\in D} n(w,d) P^{\text{old}}(z|w,d)$$

From this stems the final expression for $P(w|z)$:

$$P(w|z) = \frac{\sum_{d\in D} n(w,d) P^{\text{old}}(z|w,d)}{\sum_{w\in V}\sum_{d\in D} n(w,d) P^{\text{old}}(z|w,d)}$$

Derivation by $P(d|z)$, at $z = \zeta$ and $d = \delta$ given, will yield a development entirely parallel to that detailed above, yielding a final expression of

$$P(d|z) = \frac{\sum_{w\in W} n(w,d) P^{\text{old}}(z|w,d)}{\sum_{d\in D}\sum_{w\in V} n(w,d) P^{\text{old}}(z|w,d)} \qquad \square$$

3.3.3 Hofmann's Fisher kernels for PLSI

When the PLSI model was initially proposed, no document similarities well-suited for it were deployed. The first experiments were performed using two ad hoc variants of the cosine similarity, used in combination with traditional `tf-idf` similarities in the direct word representation space (as exposed in section 2.3, p. 24).

Soon afterwards, a variant of the Fisher kernel adapted to PLSI was proposed as a better theoretically grounded similarity, yielding significant improvements in retrieval performance [28]. This work was revisited on a number of points by Nyffenegger et al. [61], who re-examined several simplifying hypotheses[1] made by Hofmann and gave evidences that by restricting some of them, yet better results could be obtained.

[1] Notably one to the effect that the Fisher information matrix can be reduced to $\mathbb{I}$ with a square root re-parametrisation. All the hypotheses are examined in details in table 4.1 (p. 60).

As recalled in section 2.5.3 (p. 29), Fisher kernels are similarity measures between instances of probabilistic models [29]: for two instances X and Y of a given family of stochastic models $P(X|\theta)$ parametrised with θ, the Fisher kernel is defined as

$$K(X,Y) = U_X(\theta)^T G(\theta)^{-1} U_Y(\theta) \ , \tag{3.3.2}$$

where $U_X(\theta)$ is the gradient of the log-likelihood: $U_X(\theta) = \nabla_\theta \log P(X|\theta)$, and the Fisher information matrix $G(\theta)$ is the covariance of this score:

$$G(\theta) = \mathbf{E}_X[\, U_X(\theta)\, U_X(\theta)^T\,] \ .$$

The Fisher kernel for PLSI was introduced in [28] with the form

$$K^H(d,q) = \sum_z \frac{P(z|d)P(z|q)}{P(z)} + \sum_w \widehat{P}(w|d)\widehat{P}(w|q) \sum_z \frac{P(z|d,w)P(z|q,w)}{P(w|z)} \tag{3.3.3}$$

Derivation The following is a development of the Fisher kernel for PLSI, as done in Nyffenegger et al. [60]. The steps are largely identical to those taken by Hofmann, except for the Fisher information matrix.

In order to compute the Fisher information matrix $G(\Theta)$, Hofmann used a square root re-parametrisation.

In the case of mixtures of multinomials, or more generally with any distribution of the exponential family, the square root re-parametrisation entails that the metric tensor identifies with the identity matrix:

$$G(\Theta) \simeq \mathbb{I}. \tag{3.3.4}$$

However, PLSI is not a combination of multinomials. We believe that $G(\Theta)$ can be fairly well approximated by a diagonal matrix, but that this matrix is not the identity. We come back to this point in section 5.2 (p. 66).

We will nevertheless follow Hofmann's steps here, as to allow ourselves to discuss his kernel. Henceforth, we note this kernel K^{H} The square root re-parametrisation entails

- $P(z) \to \rho(z) = 2\sqrt{P(z)}$
  hence $P(z) = \frac{1}{4}\rho^2(z)$ and $\frac{\partial P(z)}{\partial \rho(z)} = \frac{1}{2}\rho(z) = \sqrt{P(z)}$
- $P(d|z) \to \rho(d|z) = 2\sqrt{P(d|z)}$
  hence $P(d|z) = \frac{1}{4}\rho^2(d|z)$ and $\frac{\partial P(d|z)}{\partial \rho(d|z)} = \frac{1}{2}\rho(d|z) = \sqrt{P(d|z)}$
- $P(w|z) \to \rho(w|z) = 2\sqrt{P(w|z)}$
  hence $P(w|z) = \frac{1}{4}\rho^2(w|z)$ and $\frac{\partial P(w|z)}{\partial \rho(w|z)} = \frac{1}{2}\rho(w|z) = \sqrt{P(w|z)}$

The *Fisher score* is obtained by the gradient of the likelihood with respect to θ:

$$U_{\widehat{d_0}}(\Theta) = \nabla_\theta L_{\widehat{d_0}}(\theta) = \nabla_\theta \sum_{w \in V} n(d_0,w) \log \left[\sum_z P(z)P(w|z)P(d_0|z) \right]$$

$$\frac{\partial L_d}{\partial \rho(z)} = \frac{\partial L_d}{\partial P(z)} \frac{\partial P(z)}{\partial \rho(z)} = \sqrt{P(z)} \sum_w n(d,w) \frac{P(d|z)P(w|z)}{\sum_\zeta P(\zeta)P(w|\zeta)P(d|\zeta)} \tag{3.3.5}$$

$$\frac{\partial L_d}{\partial \rho(\widehat{w}|z)} = \frac{\partial L_d}{\partial P(\widehat{w}|z)} \frac{\partial P(\widehat{w}|z)}{\partial \rho(\widehat{w}|z)} = \sqrt{P(\widehat{w}|z)} \cdot n(d,w) \cdot \frac{P(d|z)P(z)}{\sum_\zeta P(\zeta)P(\widehat{w}|\zeta)P(d|\zeta)} \tag{3.3.6}$$

Note that equation (3.3.6) does not contain a sum over w since the derivation with respect for one given $\widehat{w} \in V$ cancels all the terms where $w \neq \omega$.

The expectation is estimated as $\mathbb{E}_d[\nabla L_d \nabla L_d^T] \sim \sum_{d \in D} \cdot \nabla L_d \nabla L_d^T$. Let us define

$$\alpha_z = \sum_d \left(\frac{\partial L_d}{\partial \rho(z)} \right)^2, \qquad \gamma_{w,z} = \sum_d \left(\frac{\partial L_d}{\partial \rho(w|z)} \right)^2$$

Then[2]

$$G(\rho) = \begin{pmatrix} \alpha_1 & & & & & & \\ & \ddots & & & & & \\ & & \alpha_Z & & & & \\ & & & \gamma_{11} & & & \\ & & & & \ddots & & \\ & & & & & \gamma_{WZ} \end{pmatrix}$$

As $G(\rho)$ is assumed to be diagonal, $G(\rho)^{-1}$ is straightforwards, and this yields a two-component kernel where the contributions of the categories alone and the contributions of the words are distinct:

$$K^{\mathrm{H}}(d,q) = \underbrace{\sum_z \frac{\partial L_d}{\partial \rho(z)} \alpha_z^{-1} \frac{\partial L_q}{\partial \rho(z)}}_{K_z(d,q)\text{: contribution of the } z} + \underbrace{\sum_w \sum_z \frac{\partial L_d}{\partial \rho(w|z)} \gamma_{w,z}^{-1} \frac{\partial L_q}{\partial \rho(w|z)}}_{K_w(d,q)\text{: contribution of the } w}$$

We note the contributions of the categories alone and the contributions of the words K_z^{H} and K_w^{H}, respectively. We can now give an explicit expression of these terms, function of the parametres of PLSI: by introducing the expressions of equations 3.3.5 and 3.3.6 into the above kernel (and writing $\sum_z P(z)P(w|z)P(d|z)$ as $P(d,w)$ for simplify notations), we obtain

$$\begin{aligned} K_z^{\mathrm{H}}(d,q) = \sum_z \sqrt{P(z)} \cdot \sum_w n(d,w) \frac{P(d|z)P(w|z)}{P(d,w)} \cdot & \longleftarrow L_d \\ \sqrt{P(z)} \cdot \sum_w n(q,w) \frac{P(q|z)P(w|z)}{P(q,w)} \cdot & \longleftarrow L_q \\ \left(P(z) \sum_d \sum_w \left(n(d,w) \frac{P(d|z)P(w|z)}{P(d,w)} \right)^2 \right)^{-1} & \longleftarrow G^{-1} \end{aligned} \tag{3.3.7}$$

[2]For two given documents d and q, $d \neq q$, the vectors $\frac{\partial L}{\partial \rho(d|z)}$ and $\frac{\partial L}{\partial \rho(q|z)}$ are orthogonal, which yields a component

$$\sum_z \frac{\partial L_d}{\partial \rho(\delta|z)} G_{\delta,z}^{-1} \frac{\partial L_q}{\partial \rho(\delta|z)} = 0, \quad \forall \delta$$

.

and $K_w^{\mathrm{H}}(d,q)$ as

$$K_w^{\mathrm{H}}(d,q) = \sum_w \sum_z \sqrt{P(w|z)} \cdot n(d,w) \frac{P(d|z)P(z)}{P(d,w)} \cdot \qquad \longleftarrow L_d$$
$$\sqrt{P(w|z)} \cdot n(q,w) \frac{P(q|z)P(z)}{P(q,w)} \cdot \qquad \longleftarrow L_q \qquad (3.3.8)$$
$$\left(P(w|z) \sum_d \left(n(d,w) \frac{P(d|z)P(z)}{P(d,w)} \right)^2 \right)^{-1} \qquad \longleftarrow G^{-1}$$

For the sake of simplicity and tractability, we seek more simple expressions of $K_z^{\mathrm{H}}(d,q)$, $K_w^{\mathrm{H}}(d,q)$, α and γ.

K_z^{H} component

$$K_z^{\mathrm{H}} = \sum_z P(z) \cdot \sum_w n(d,w) \frac{P(d|z)P(w|z)}{P(d,w)} \cdot \sum_w n(q,w) \frac{P(q|z)P(w|z)}{P(q,w)}$$

Clearly the $P(d|z)$ and $P(q|z)$ are independent of w and can be taken out of the sums. Hence

$$K_z^{\mathrm{H}} = \sum_z \underbrace{P(z)P(d|z)P(q|z)}_{=\frac{P(d,z)P(q,z)}{P(z)}} \cdot \underbrace{\sum_w n(d,w) \frac{P(w|z)}{P(d,w)}}_{\cong 1/|C|} \cdot \underbrace{\sum_w n(q,w) \frac{P(w|z)}{P(q,w)}}_{\cong 1/|C|}$$

Hofmann introduces the following hypothesis:

- $P(z)P(d|z)P(q|z) = \frac{P(d,z)P(q,z)}{P(z)}$ straightforwardly from Bayes' rule
- $\sum_w n(d,w) \frac{P(w|z)}{P(d,w)} \cong 1/|C|$ because it is expected that, over all the w, the observed distribution $\widehat{P}(d,w)$ tends to fit the probability $P(d,w)$, and thus $\sum_w \frac{\widehat{P}(d,w)}{P(d,w)} \cong 1$. Furthermore, $\sum_w P(w|z) \equiv 1$. Hence $\sum_w \widehat{P}(d,w) \frac{P(w|z)}{P(d,w)} \cong 1$.

  As Hofmann makes the assumption with a $n(d,w)$ instead of a $\widehat{P}(d,w)$, a $|C|^{-2}$ factor is introduced

$$K_z^{\mathrm{H}} = \frac{1}{|C|^2} \frac{P(d,z)P(q,z)}{P(z)}$$

K_w^{H} component

$$K_w^{\mathrm{H}}(d,q) = \sum_w \sum_z P(w|z) \cdot n(d,w) \frac{P(d|z)P(z)}{P(d,w)} \cdot n(q,w) \frac{P(q|z)P(z)}{P(q,w)}$$

Clearly the terms $n(d,w)$ and $n(q,w)$ are independent from z and can be taken out of $\sum_z$. We also manipulate $P(w|z)$ to make the expression $P(w|z)P(d|z)P(z)$ appear twice inside the sum:

$$K_w(d,q) = \sum_w n(d,w)n(q,w) \sum_z \underbrace{\frac{P(w|z)P(d|z)P(z)}{P(d,w)}}_{P(z|d,w)} \underbrace{\frac{P(w|z)P(q|z)P(z)}{P(q,w)}}_{P(z|q,w)} \frac{1}{P(w|z)}$$

This is justified by the fact that in the PLSI model, $\sum_z P(w|z)P(d|z)P(z) = P(z,d,w)$. Hence, $\sum_z \frac{P(w|z)P(d|z)P(z)}{P(d,w)} = \frac{P(z,d,w}{P(d,w)} \overset{\text{Bayes}}{=} P(z|d,w)$

Substituting $\frac{n(d,w)}{|C|}$ for $\widehat{P}(d,w)$, we can write

$$K_w^{\mathrm{H}}(d,q) = |C|^2 \sum_w \widehat{P}(d,w)\widehat{P}(q,w) \sum_z \frac{P(z|d,w)P(z|q,w)}{P(w|z)}$$

α component

$$\alpha_z = \sum_d \left(\frac{\partial L_d}{\partial \rho(z)} \right)^2 = \sum_d \sum_z P(z) \left(\sum_w n(d,w) \frac{P(d|z) \cdot P(w|z)}{P(d,w)} \right)^2$$

Clearly the $P(d|z)$ can be taken out of the sum over w, and

$$\alpha_z = \sum_d \underbrace{\sum_z P(z) \cdot P^2(d|z)}_{\frac{P^2(d,z)}{P(z)}} \underbrace{\left(\sum_w n(d,w) \frac{P(w|z)}{P(d,w)} \right)^2}_{\cong 1/|C|^2}$$

With a $|C|^2$ factor like in K_z^H, we have

$$\alpha_z = |C|^2 \sum_d \frac{P^2(d,z)}{P(z)} \qquad \square$$

γ component

We have

$$\gamma_{w,z} = \sum_d P(d) \cdot \left(\frac{\partial L_d}{\partial \rho(w|z)} \right)^2 = \sum_d P(w|z) \left(n(d,w) \frac{P(d|z)P(z)}{P(d,w)} \right)^2,$$

which can be written as

$$\gamma_{w,z} = \sum_d \left(n(d,w) \frac{P(z|d,w)}{\sqrt{P(w|z)}} \right)^2 \qquad \square$$

Eventually, the kernel becomes

$$K(d,q) = \frac{\cancel{1}/\cancel{|C|^2} P(d,z)P(q,z)}{P(z)} \cdot \left[\cancel{|C|^2} \sum_d \frac{P^2(d,z)}{P(z)} \right]^{-1} + |C|^2 \sum_w \widehat{P}(d,w)\widehat{P}(q,w) \sum_z \frac{P(z|d,w)P(z|q,w)}{P(w|z)} \cdot \left[\sum_d n(d,w)^2 \frac{P(z|d,w)^2}{P(w|z)} \right]^{-1}$$

We see a kernel composed of two additive parts that match those of Hofmann's kernel (eq. 3.3.3); however, each of these terms is multiplied by a term that is absent from Hofmann's kernel. This term accounts for the contribution of the Fisher information matrix, which acts as a metric tensor on the new representation space.

Furthermore, we see that the $P(d|z)$ and $\widehat{P}(d|w)$ of Hofmann's kernel have been replaced with $P(d,z)$ and $\widehat{P}(d,w)$, respectively, because Nyffenegger et al. did not introduce the hypothesis on normalisation by document and query lengths.

The presence of a FIM term and the absence of document length normalisation distinguish Nyffenegger's kernels from Hofmann's. We discuss these points in details in chapter 5.

3.3.4 Query folding-in

Folding-in [26, 27] is a technique to estimate $P(d|z)$ parameters for unknown documents in PLSI; in an ad hoc information retrieval framework, queries are typically concerned.

The parameters $P(q|z)$ of the query are learnt by a simplified process that uses the $P(w|z)$ learnt from the corpus, by running EM on $P(q|z)$ while keeping the $P(z)$ and $P(w|z)$ fixed. The $P(d|z)$ are then rescaled to accommodate the new $P(q|z) > 0$.

This method engenders problems, such as inaccuracies in log-likelihood estimation of the test set [89], or their adequacy with the $P(d|z)$ of the training set.

In the course of our experiments, we have deliberately refrained from implementing a "folding in" system. Instead, we concatenate the queries and the corpus, and perform posterior estimation on both. We then separate the queries from the document collection and perform the retrieval as usual.

3.4 Conclusions

We have reproduced and detailed the derivation of Nyffenegger et al. for two proeminent probabilistic latent semantic models, Naive Bayes and PLSI. For both, the EM inference algorithm and the Fisher kernel are provided.

In the case of models of the exponential family, the Fisher information matrix may be reduced to $G(\Theta) = \mathbb{I}$ at the price of a square root re-parametrisation. However, PLSI does not belong to this family, and the Fisher information matrix for PLSI contributes to the Fisher kernel whatever the re-parametrisation.

Because PLSI depends on parameters $P(d|z)$, it is not a fully generative model, because it does not provide a readily available representation for new documents. This causes practical problems for queries, and prevents incremental indexing.

The kernels reproduced in this chapters are Hofmann's kernel, normalised by $|d|$ and $|q|$ and whose Fisher information matrix is approximated by the identity; and the unnormalised kernel by Nyffenegger et al., whose Fisher information matrix is also approximated by the identity; the "Vector Space" kernel of Nyffenegger et al. is a combination of terms coming from the two former kernels. We have also shown how the Fisher Information matrix can be approximated by its diagonal rather than by the identity, as was done by Nyffenegger et al. These elements will be useful when the derivation of the Fisher kernel will be redone from the ground up (chapter 4, p. 53), to study the contributions of the different parts of the kernels (chapter 5, p. 63), and when the kernels of Hofmann and the different kernels of Nyffenegger et al. (unnormalised, Vector Space and DFIM) will be evaluated (chapter 7, p. 81).

Chapter 4

PLSI: from I.I.D. processes to the Fisher kernel

Abstract

In the previous chapter, we gave an overview of how the postulates of a probabilistic latent semantics model yield the learning algorithms that generate the latent representations, as well as the Fisher kernels that constitute an appropriate similarity measure between them.

In the present chapter, we re-work the derivation of the Fisher kernel for PLSI from the ground up, i.e. from the very generative process of PLSI. From the observation that the generative process of PLSI is an Independent, Identically Distributed (IID) process, and through a lemma regarding the compositionality of Fisher kernels in this case; we derive a new kernel form, which constitutes the actual, rigorous PLSI Fisher kernel.

We complete the discussion by comparing this kernel to previously published forms and examining which conditions transform one into another. This will allow us to elaborate on the different variants of the Fisher kernel that have been proposed, and to judge their merits from on theoretical and practical points of views.

4.1 Introduction

In this chapter, we question the form of the Fisher kernel for PLSI from an even deeper level, and re-develop it from the ground up. By remaining tightly linked to the generative process of PLSI while developping the kernel, we take a different course than Hofmann and Nyffenegger et al., and reach another form of the kernel. We then show that this form is in fact very close to that found by Hofmann, bargaining exactly four hypothesis that we explain in detail (table 4.1, p. 60). By doing so, we can clearly understand where the previously published kernels position themselves with respect to the ideal, theoretically pure kernel.

4.2 Derivation of the Fisher kernel for IID models

As recalled in section 2.5.3 (p. 29), Fisher kernels are similarity measures between instances of probabilistic models [29]: for two instances X and Y of a given family of stochastic models $P(X|\theta)$ parametrised

with θ, the Fisher kernel is defined as

$$K(X,Y) = U_X(\theta)^T G(\theta)^{-1} U_Y(\theta) \ , \tag{4.2.1}$$

where $U_X(\theta)$ is the gradient of the log-likelihood: $U_X(\theta) = \nabla_\theta \log P(X|\theta)$, and the Fisher information matrix $G(\theta)$ is the covariance of this score:

$$G(\theta) = \mathbf{E}_X[U_X(\theta)\, U_X(\theta)^T] \ .$$

4.2.1 Fisher kernel of documents: the document log-likelihoods and how they combine

Writing the log-likelihood of documents is at the core of the derivation of the Fisher kernel, as the gradient of this log-likelihood intervenes both in the Fisher score and in the Fisher matrix.

In developping his Fisher kernel for PLSI (see section 3.3.3,p. 46), Hofmann uses a length-normalised log-likelihood of an observed document:

$$L_{\widehat{d_0}}(\theta) = \frac{1}{|\widehat{d_0}|} \log P(\widehat{d_0}) = \frac{1}{|\widehat{d_0}|} \sum_{w \in \widehat{d_0}} \log P(d_0, w) = \frac{1}{|\widehat{d_0}|} \sum_{w \in V} n(w, d_0) \log P(d_0, w) \tag{4.2.2}$$

We, on the other hand, propose to compute the Fisher kernel from the original nature of PLSI: an IID process on pairs of indices (d, w). Instead of writing an overall log-likelihood for a realised document and deriving a kernel from there, we will keep the log-likelihood associated to individual (d, w) instances, and derive the associated Fisher kernels; we go on to demonstrate that these elementary Fisher kernels combine into the overall Fisher kernel for documents.

To this effect, we provide a lemma on compositionality of Fisher kernels in an IID framework.

Lemma 4.2.1 *The Fisher kernel between two instances X_1^n and Y_1^m of an independent and identically-distributed (IID) stochastic process is the sum of the Fisher kernels between the individual random variables X_i and Y_j, divided by the number of variables in the processes:*

$$K(X_1^n, Y_1^m) = \frac{1}{n} \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{m} K(X_i, Y_j) \ . \tag{4.2.3}$$

Proof Let us consider X_1^n a n-element-long instance of an IID stochastic process. To simplify notations, we shall henceforth write $X = X_1^n$. Its log-likelihood is expressed by $\log P(X) = \sum_{i=1}^n \log P(X_i)$. Linearity of derivation operators entails that the corresponding Fisher score is written $U_X(\theta) = \sum_{i=1}^n U_{X_i}(\theta)$.

The Fisher information matrix for n-event instances of such an IID stochastic process is by definition $G(\theta) = \mathbb{E}_X \left[U_X(\theta)\, U_X(\theta)^T \right]$ and for a single instance: $G_1(\theta) = \mathbb{E}_{X_i} \left[U_{X_i}(\theta)\, U_{X_i}(\theta)^T \right]$ (which

exact X_i is independent of$i$, since the process is IID). $G(\theta)$ can be written as:

$$G(\theta) = \mathbb{E}_X \left[\left(\sum_{i=1}^{n} U_{X_i}(\theta) \right) \left(\sum_{j=1}^{m} U_{X_j}(\theta) \right)^T \right] \tag{4.2.4}$$

$$= \sum_{i=1}^{n} \left(\mathbb{E}_X \left[U_{X_i}(\theta) U_{X_i}(\theta)^T \right] + \sum_{j \neq i} \mathbb{E}_X \left[U_{X_i}(\theta) U_{X_j}(\theta)^T \right] \right) \tag{4.2.5}$$

$$= \sum_{i=1}^{n} \left(G_{X_i}(\theta) + \mathbb{E}_{X_i} \left[U_{X_i}(\theta) \right] \sum_{j \neq i} \underbrace{\mathbb{E}_{X_j} \left[U_{X_j}(\theta) \right]^T}_{=0} \right) = \sum_{i=1}^{n} G_1(\theta) \tag{4.2.6}$$

$$= n \cdot G_1(\theta) \ . \tag{4.2.7}$$

As in [29], the "natural gradient" ϕ_X is obtained from the ordinary gradient $U_X(\Theta)$ via

$$\phi_X = G(\Theta)^{-1} U_X(\Theta).$$

In the case of IID stochastic processes, the previous results lead to

$$\phi_X = G(\Theta)^{-1} U_X(\Theta) = \frac{1}{n} \cdot \sum_{i=1}^{n} G_1(\Theta)^{-1} U_{X_i}(\Theta) = \frac{1}{n} \cdot \sum_{i=1}^{n} \phi_{X_i} \ .$$

Eventually, the Fisher kernel between two instances X_1^n and Y_1^m of an IID process is given by

$$K(X_1^n, Y_1^m) = {\phi_X}^T G_1(\Theta) \phi_Y = \left(\frac{1}{n} \sum_{i=1}^{n} G_1(\Theta)^{-1} U_{X_i}(\Theta) \right)^T G_1(\Theta) \left(\frac{1}{m} \sum_{i=j}^{m} G_1(\Theta)^{-1} U_{Y_j}(\Theta) \right)$$

$$= \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{i=j}^{m} U_{X_i}^T(\Theta) G_1(\Theta)^{-1} U_{Y_j}(\Theta) = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{i=j}^{m} K(X_i, Y_j) \ .$$

□

With Lemma 4.2.1, we are empowered to develop the Fisher kernel for PLSI in a way that stems directly from the very nature of PLSI: an IID process of (d, w) pairs probabilised by[1]

$$P(d, w) = \sum_z P(z) P(w|z) P(d|z),$$

4.2.2 Derivation of the PLSI Fisher kernel

From the aforementioned derivation of the Fisher kernel (lemma 4.2.1, p. 54), in which the terms X_i and Y_i correspond to

$$X_i = (d, w) \qquad Y_i = (q, w') \quad ,$$

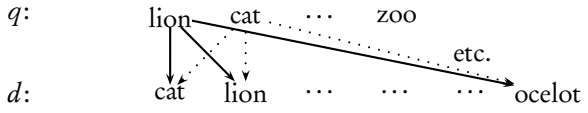
for two document models d and q, and two terms w and w' from vocabulary V, we have

$$K^{\text{IID}}(d, q) = \frac{1}{|d|} \frac{1}{|q|} \sum_{w \in V} \sum_{w' \in V} n(d, w) n(q, w') K\Big((d, w), (q, w') \Big) \ , \tag{4.2.8}$$

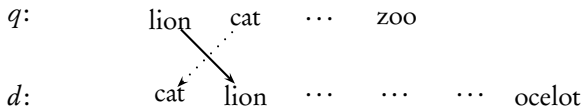
where $n(d, w)$ is the number of occurrences of word w in document d, $|d| = \sum_w n(d, w)$, and $K\big((d, w), (q, w')\big)$ is the "atomic kernel" between a pair of terms w and w' belonging to documents d and q respectively.

[1] See equation 3.3.1 (p. 42) for more details.

Note: Notice the two indices w and w' the summation above: in contrast to former formulations (section 3.3.3,p. 46), these pairs of (w,d) indices do not necessarily share the same w. The score associated to the match between two document models d and q is not only a term-by-term comparison (a single summation over the terms of the vocabulary); instead, it yields a *double* summation, as the contribution of each term of the query is itself a summation of the kernels of this query term against *all* document terms (see figure 4.1). In particular, this entails that the IID model can yield a non-zero contribution for a term present only in one of the documents but not in both.



(a) IID kernel



(b) Hoffman's kernel

Figure 4.1: Summation of contributions within the two forms of the Fisher kernel: in the IID kernel (4.1a), each word appearing in document q yields a summation over all the words in d, producing a double summation. Pairs of different words — for instance (`cat`,`ocelot`) — contribute, entailing that terms that appear in only one document contribute. In Hofmann's kernel (4.1b), each word appearing in document q yields one single term with from instances of the same word in d. Pairs of different words do not contribute, entailing that terms that appear in only one document are ignored.

4.2.3 Atomic kernel

Lemma 4.2.2 *For PLSI, the "atomic kernel" between two term occurrences is written as*

$$K\Big((d,w),(q,w')\Big) = \frac{P(d|z)}{P(d,w)}\frac{P(q|z)}{P(q,w')}\cdot\sum_z\Big[P(w|z)P(w'|z)\alpha(z)+\delta_{w,w'}\gamma(w,z)\Big] \quad , \tag{4.2.9}$$

with $\delta_{w,w'}$ the Kronecker delta between w and w'. The terms $\alpha(z)$ and $\gamma(w,z)$ denote the contribution of the Fisher Information Matrix:

$$\alpha^{-1}(z)=\sum_{d\in D}\sum_{w\in V}n(d,w)\left(\frac{P(w|z)P(d|z)}{P(d,w)}\right)^2 \qquad \gamma^{-1}(w,z)=\sum_{d\in D}n(d,w)\left(\frac{P(d|z)}{P(d,w)}\right)^2$$

Proof The derivation of the Fisher kernel goes in three stages: in the first stage, the *Fisher score* is computed from the log-likelihood associated with the model; in the second, the *Fisher Information Matrix* is computed, as the expectation of the Fisher Score over the known sample; eventually, the *Fisher kernel* itself is obtained by combining the Fisher Information Matrix with the Fisher scores of the elements to

be compared.

The Fisher score for one pair of indices (d,w) in PLSI is $U_{(d,w)}(\theta) = \nabla_\theta \log P(d,w)$. The gradient is, on our case, computed by performing derivations with respect to the parameters $P(z)$, $P(d|z)$ and $P(w|z)$ for all terms w, document indices d and categories z. Let us write δ and ω the indices of the document and the term with respect to which derivations of $P(d,w)$ are performed. Since $P(d,w) = \sum_z P(z)P(w|z)P(d|z)$, we have

$$U_{(d,w)}(\theta) \quad = \quad \nabla_\theta \log P(d,w) = \begin{pmatrix} \frac{\partial \log P(d,w)}{\partial P(z)} \\ \frac{\partial \log P(d,w)}{\partial P(\omega|z)} \\ \frac{\partial \log P(d,w)}{\partial P(\delta|z)} \end{pmatrix} = \begin{pmatrix} \frac{P(w|z)P(d|z)}{P(d,w)} \\ \delta_{\omega,w} \frac{P(d|z)P(z)}{P(d,w)} \\ \delta_{\delta,d} \frac{P(w|z)P(z)}{P(d,w)} \end{pmatrix} ,$$

with $\delta_{\omega,w}$ the Kronecker delta between ω and w, and similarly for δ and d.

When comparing two documents, a scalar product is performed on their Fisher scores. In this product, the "document dimensions"[2] can yield a non-zero contribution if, and only if, the documents being compared have the same document indice — that is if they are the same document model. In other cases, two different documents are always orthogonal in the "document dimensions". Since this only non-zero case is trivial, the "document dimensions" of the Fisher score can thus be ignored in the kernel.

At this stage, for the sake of consistency with former derivations (see section 3.3.3), a square root re-parametrisation can be introduced:

$$\theta(z) \to \rho(z) = 2\sqrt{P(z)} \quad \text{and} \quad \theta(w|z) \to \rho(w|z) = 2\sqrt{P(w|z)} \ . \tag{4.2.10}$$

With this re-parametrisation, $\frac{\partial P(z)}{\partial \rho(z)} = \frac{1}{2}\rho(z) = \sqrt{P(z)}$ and $\frac{\partial P(w|z)}{\partial \rho(\omega|z)} = \delta_{\omega,w}\frac{1}{2}\rho(w|z) = \delta_{\omega,w}\sqrt{P(w|z)}$. Hence,

$$U_{(d,w)}(\rho) = \begin{pmatrix} \sqrt{P(z)}\frac{P(w|z)P(d|z)}{P(d,w)} \\ \delta_{\omega,w}\sqrt{P(w|z)}\frac{P(d|z)P(z)}{P(d,w)} \end{pmatrix} .$$

The Fisher information matrix for one instance (d,w) is

$$G_1(\theta) = \mathbb{E}_{(d,w)}\left[U_{(d,w)}(\theta)\, U_{(d,w)}(\theta)^T\right] .$$

In general, this matrix is not tractable as such — in particular, computing its inverse is not practically feasible for document collections that contain over a few dozens documents and terms. We will thus introduce the *Diagonal Fisher Information Matrix* (DFIM) hypothesis, i.e. the notion that $G_1(\theta)$ can be approximated by its diagonal.

Let us consider the parts of the matrix $G_1(\theta)$ which stem from $U_{(d,w)}(z) = \frac{\partial \log P(d,w)}{\partial P(z)}$ (noted G_z) and from $U_{(d,w)}(\omega|z) = \frac{\partial \log P(d,w)}{\partial P(\omega|z)}$ (noted G_w) respectively. For a document collection C, the diagonal of $G_1(\theta)$ can be approximated by (e.g. [52]):

$$G_z(z) = \sum_{(d,w)\in C} U_{(d,w)}(z)^2, \qquad G_w(\omega,z) = \sum_{(d,w)\in C} U_{(d,w)}(\omega|z)^2 \ , \tag{4.2.11}$$

[2] That is the components of the Fisher score that pertain to the derivation by documents indices, $\frac{\partial \log P(w,d)}{\partial P(\delta|z)}$ and $\frac{\partial \log P(w,q)}{\partial P(\delta|z)}$.

which, for $G(\rho)$, leads to

$$G_z(z) = P(z)\sum_{d\in D}\sum_{w\in d} n(d,w)\left(\frac{P(w|z)P(d|z)}{P(d,w)}\right)^2 \tag{4.2.12}$$

$$G_w(w,z) = P(w|z)P(z)^2\sum_{d\in D} n(d,w)\left(\frac{P(d|z)}{P(d,w)}\right)^2 . \tag{4.2.13}$$

Note that the terms $P(z)$ in equation 4.2.12 and $P(w|z)P(z)^2$ in equation 4.2.13 stem solely from the square root re-parametrisation, and cancel out in the kernel when combined with their homologues of $U_d(\theta)$ and $U_q(\theta)$. This heralds the normalising role of $G(\theta)$.

The "atomic" Fisher kernel is finally obtained by assembling the Fisher scores $U_{(d,w)}(\Theta)$ and the Fisher Information matrix $G_1(\theta)$ as

$$\begin{aligned} K\big((d,w),(q,w')\big) &= U_{(d,w)}(\Theta)\, G_1^{-1}(\Theta)\, U_{(q,w')}(\Theta) \\ &= U_{(d,w)}(\rho)\, G_1^{-1}(\rho)\, U_{(q,w')}(\rho). \end{aligned}$$

The matrix $G_1(\Theta)$ is assumed to be diagonal. We can separate the contributions of the "categories dimensions" (the $\partial U/\partial P(z)$ terms) and those of the "term dimensions" (the $\partial U/\partial P(w|z)$ terms) by writting

$$K\big((d,w),(q,w')\big) = K_z\big((d,w),(q,w')\big) + K_w\big((d,w),(q,w')\big),$$

with:

$$\begin{aligned} K_z\big((d,w),(q,w')\big) &= \sum_z U_{(d,w)}(z)\, G_z(z)^{-1}\, U_{(q,w')}(z) \ \text{, and} \\ K_w\big((d,w),(q,w')\big) &= \sum_z\sum_\omega U_{(d,w)}(\omega|z)\, G_w(\omega,z)^{-1}\, U_{(q,w')}(\omega|z) \\ &= \delta_{ww'}\sum_z U_{(d,w)}(w|z)\, G_w(w,z)^{-1}\, U_{(q,w')}(w|z) \ . \end{aligned}$$

Combining these components with the general expression of the kernel, we eventually obtain that

$$\begin{aligned} K\Big((d,w),(q,w')\Big) = & \\ \sum_z\Big[& \frac{P(w|z)P(d|z)}{P(d,w)}\frac{P(w'|z)P(q|z)}{P(q,w')}P(z)G_z(z)^{-1} \\ & + \delta_{w,w'}\frac{P(d|z)P(q|z)}{P(d,w)P(q,w)}P(z)^2P(w|z)G_w(w,z)^{-1}\Big] \ . \end{aligned} \tag{4.2.14}$$

4.2.4 IID Fisher kernel

The final expression of the Fisher kernel for PLSI can now by obtained by injecting the expression of the "atomic kernels" (equation 4.2.14), pertaining to one pair of indices (d,w) and (q,w') indices, into the expression of the full IID kernel (equation 4.2.8). We find that

$$\begin{aligned} K^{\text{IID}}(d,q) = & \sum_{w\in V}\sum_{w'\in V}\widehat{P}(w|d)\cdot\widehat{P}(w'|q)\cdot\frac{P(d|z)}{P(d,w)}\frac{P(q|z)}{P(q,w')}\cdot \\ & \cdot\sum_z\Big[P(w|z)P(w'|z)\cdot\alpha(z)+\delta_{w,w'}\gamma(w,z)\Big] \ , \end{aligned} \tag{4.2.15}$$

where $\widehat{P}(w|d) = \frac{n(d,w)}{|d|}$.

Note that $\alpha = G_z(\theta)^{-1} = P(z)G_z(\rho)^{-1}$, and that $\gamma = G_w(\theta)^{-1} = P(w|z)P(z)^2 G_w(\rho)^{-1}$:

$$\begin{aligned}\alpha(z) &= \left(\sum_{d\in D}\sum_{w\in d} n(d,w)\left(\frac{P(w|z)P(d|z)}{P(d,w)}\right)^2\right)^{-1} = P(z)G_z(\rho)^{-1}\\ \gamma(w,z) &= \left(\sum_{d\in D} n(d,w)\left(\frac{P(d|z)}{P(d,w)}\right)^2\right)^{-1} = P(w|z)P(z)^2 G_w(\rho)^{-1}\end{aligned}$$

4.3 Relation between $K^{\mathrm{IID}}(d,q)$ and $K^{\mathrm{H}}(d,q)$

We now examine equation 4.2.15 related to Hofmann's original development of the Fisher kernel [28] [3]

$$K^H(d,q) = \sum_{z\in Z}\frac{P(z|d)P(z|q)}{P(z)} + \sum_{w\in V}\widehat{P}(w|d)\widehat{P}(w|q)\sum_{z\in Z}\frac{P(z|d,w)P(z|q,w)}{P(w|z)} \ , \tag{4.3.1}$$

Hofmann's kernel stems partly from the assumption that, averaged over the entire vocabulary of the document collection, the probabilities inferred by the learning system are good estimators of the observed statistical features of the collection. The exact assumption used by Hofmann is

$$\sum_{w\in V} P(w|z)\frac{\widehat{P}(w|d)}{P(w|d)} = \sum_{w\in V}\frac{n(d,w)}{|d|}\frac{P(w|z)}{P(w|d)} \simeq 1 \ . \tag{4.3.2}$$

We introduce a closely related term, distinct from the term that appear in Hofmann's assumption only by replacing the document length $|d|$ with $P(d)$:

$$\zeta(d,z) = \sum_{w\in V}\frac{n(d,w)}{P(d)}\frac{P(w|z)}{P(w|d)} = \sum_{w\in V} n(d,w)\frac{P(w|z)}{P(d,w)} \ ,$$

Assumption 4.3.2 then yields

$$\zeta(d,z) \simeq \frac{|d|}{P(d)} \tag{4.3.3}$$

We re-write equation 4.2.15 with the $\zeta(d,z)$ notation, yielding

$$\begin{aligned}K^{\mathrm{IID}}(d,q) &= \frac{1}{|d|}\frac{1}{|q|}\sum_{z\in Z} P(d|z)P(q|z)\cdot\\ &\quad\cdot\left[\alpha(z)\zeta(d,z)\zeta(q,z) + \sum_{w\in V}\frac{n(d,w)}{P(d,w)}\frac{n(q,w)}{P(q,w)}\gamma(w,z)\right] \ .\end{aligned} \tag{4.3.4}$$

Now we can combine 4.3.4 and 4.3.3. Furthermore, as we experimentally verified that

$$P(d) \simeq \frac{|d|}{|C|}, \qquad \text{where } |C| = \sum_{w\in V}\sum_{d\in D} n(d,w) \ , \tag{4.3.5}$$

to a precision inferior to 1%, we can state that

$$\zeta(d,z) \simeq |C|$$

[3] Our derivation for this kernel is given in section 3.3.3 (p. 46).

$\sum_{w\in V} P(w\|z)\frac{n(w,d)/\|d\|}{P(w\|d)} =$ $= \mathbb{E}_w\left[\frac{\widehat{P}(w\|d)}{P(w\|d)}\middle\|z\right] \simeq 1$	Over all words and all categories, $\widehat{P}(w\|d) = \frac{n(w,d)}{\|d\|}$ is a good estimator of $P(w\|d)$.	eq. 4.3.2 (p. 59)
$P(d) \simeq \frac{\|d\|}{\|C\|}$	The probability of a document is proportional to its length.	eq. 4.3.5 (p. 59)
$\theta(z) \to \rho(z) = 2\sqrt{P(z)}$ $\theta(w\|z) \to \rho(w\|z) = 2\sqrt{P(w\|z)}$	Square root re-parametrisation.	eq. 4.2.10 (p. 57)
$G(\rho) \simeq \mathbb{I}$	The Fisher information matrix can be approximated by $\mathbb{I}$.	eq. 3.3.4 (p. 47)

Table 4.1: The four assumptions that link Hofmann's kernel to the IID kernel.

For K_z: this entails that the K_z of the kernel above can be replaced by

$$K_z^{\text{appr}} = \sum_{z\in Z} \frac{P(d|z)}{P(d)} \frac{P(q|z)}{P(q)} \alpha(z);$$

assuming that $G = 1$, and assuming that α denotes $G(\rho)$ (i.e. is the result of a square root re-parametrisation), $\alpha = P(z)$, and

$$K_z^{\text{appr}} = \sum_{z\in Z} \frac{1}{P(z)} \underbrace{\frac{P(d|z)P(z)}{P(d)}}_{P(z|d)} \underbrace{\frac{P(q|z)P(z)}{P(q)}}_{P(z|q)} = \sum_{z\in Z} \frac{P(z|d)P(z|q)}{P(z)}$$

For K_w: with the same assumptions on G, we have that $\gamma = P(w|z)P(z)^2$; this entails that the K_w of the kernel above can be replaced by

$$\begin{aligned} K_w^{\text{appr}} &= \sum_{w\in V} \widehat{P}(w|d)\widehat{P}(w|q) \sum_{z\in Z} \frac{P(w|z)P(z)^2}{P(d,w)P(q,w)} \\ &= \sum_{w\in V} \widehat{P}(w|d)\widehat{P}(w|q) \sum_{z\in Z} \frac{1}{P(w|z)} \underbrace{\frac{P(w|z)P(z)}{P(d,w)}}_{=P(z|d,w)} \underbrace{\frac{P(w|z)P(z)}{P(q,w)}}_{=P(z|q,w)} \end{aligned}$$

Combining K_z^{appr} and K_w^{appr}, we find the Hofmann's kernel of equation 4.3.1.

Thus, at the price of four assumptions (table 4.1) Hofmann's "matrixless" kernel can be seen as an approximation of the IID one.

These approximations call for several comments:

Approximation 4.3.2 yields a very significant simplification of the computation by turning the double sum over the $w \in d$ and $w' \in q$ in equation 4.2.15 into a single sum over $w \in q$, like in equation 3.3.3 (see figure 4.1 p. 56). This yields computation of the order of 100 times faster[4], as, for every (d,q) pair, the IID computation in $O(|Z|\cdot|d|\cdot|q|)$ is replaced with a computation in $O(|Z|\cdot|q|)$.

[4]For instance, a run on the SMART base TIME, with 32 categories, takes about 9 seconds for Hofmann's kernel, and nearly 15 minutes for the IID kernel

Approximation 4.3.5 is mostly technical and furthermore has some good theoretical arguments for itself. It has been verified up to a small error.

On the other hand, that the Fisher information matrix can be approximated by $\mathbb{I}$ finds is not well justified. This point is discussed further in section 5.2 (p. 66).

4.4 Implementation of the IID Kernel

In practise, (4.3.4) is more efficiently computed by taking into account that $\alpha(z)$ depends only on z (neither on d nor on w): $\alpha(z)$ and $\zeta(d,z)$ can be pre-computed once for all for the entire corpus. $\gamma(w,z)$ and $\zeta(q,z)$ are best computed for each query, as to take advantage of the limited number of different terms present in a query: $\gamma(w,z)$ is computed only for $w \in q$ (i.e. $w \in V$ such as $n(w,q) > 0$). We have observed that processing is an order of magnitude quicker using these precomputations.

Furthermore, the computation of all Fisher kernels can be decomposed into two independent parts K_z and K_w which stem from the contributions of the latent categories and of the terms, respectively; for instance using (4.3.4):

$$K^{\mathrm{IID}}(d,q) = K_z^{\mathrm{IID}}(d,q) + K_w^{\mathrm{IID}}(d,q),$$

where

$$K_z^{\mathrm{IID}}(d,q) = \frac{1}{|d|}\frac{1}{|q|}\sum_{z\in Z} P(d|z)P(q|z)\,\alpha(z)\,\zeta(d,z)\,\zeta(q,z) \ ,$$

and

$$K_w^{\mathrm{IID}}(d,q) = \frac{1}{|d|}\frac{1}{|q|}\sum_{z\in Z}\left[P(d|z)P(q|z)\cdot\sum_{w\in V}\frac{n(d,w)}{P(d,w)}\frac{n(q,w)}{P(q,w)}\gamma(w,z)\right] \ .$$

In spite of the optimisation that we present here, computation of the IID kernel remains quite costly, as it yields a double loop on the terms of the query *and* the terms of the documents. In this perspective, it is interesting that a kernel such as Hofmann's would exist, as it gives insights for further approximations that would combine the theoretical foundations and the good results of the IID kernel with the computational advantages of Hofmann's kernel.

4.5 Conclusions

We developed the true Fisher kernel for PLSI, based on the IID nature of its generative process. The IID Fisher kernel features normalisation by document length and contributions of the Fisher information matrix; in Hofmann's original kernel, the first was present but not justified beyond an analogy to `tf-idf` models, while the second was absent.

The resulting kernel outperforms Hofmann's original kernel (see chapter 7, p. 81). It is much more expensive to compute, but the two kernels can be linked by four simplifying assumptions (table 4.1, p. 60). Examining these assumptions could result in a kernel combining the merits of both IID and Hofmann's kernel; this is the object of the following chapter.

Chapter 5

The components of the Fisher kernel for PLSI and their roles

Abstract

The previous chapter has heralded theoretically founded similarities for PLSI by deriving the Fisher kernel that stems directly from the IID nature of the PLSI generative process. In this chapter, we study in detail the contributions of the various terms that compose the kernel. In particular, we study the role of the Fisher information matrix, neglected by Hofmann. We also address the Fisher kernels "word component" K_w and "category components" K_z.

These studies allow us to answer the questions raised by Nyffenegger et al. [61]: what are the consequences of the normalisation of the document log-likelihood by document length, introduced by Hofmann in his legacy kernel? Should the Fisher information matrix be approximated by the identity, as Hofmann does, or not? We also address the so-called "Vector Space" kernel that attempted to bring balance in the Fisher kernel by normalising only its K_z part.

This study vindicates the validity of the normalisation by document length as introduced by Hofmann, and confirms the necessity to account for the Fisher information matrix.

5.1 Ratios between kernel components

In the preceding chapter, we introduced the "term dimensions" K_w and the "categories dimensions" K_z of the Fisher kernel. The complete Fisher kernel is a sum of these two terms:

$$K(d,q) = K_z(d,q) + K_w(d,q)$$

Nyffenegger et al. [61] studied in which proportions these two components contribute to the overall retrieval performance. This question stems from a normalisation by document length in Hofmann's legacy kernel [28], justified by analogy with the `tf-idf` model but not strongly theoretically grounded[1]. Questioning this normalisation led them to propose two alternative variants of the Fisher kernel:

[1] In chapter 4, we have shown how regarding PLSI as a stochastic IID process explains the origin of these terms, and justifies their introduction in a simplified kernel.

	K_z	K_w
Unnormalised kernel [61]	K_z^{U}	K_w^{U}
Hofmann [28]	$\frac{1}{P(d)P(q)}K_z^{\mathrm{U}}$	$\frac{\lvert C\rvert^2}{\lvert d\rvert\lvert q\rvert}K_w^{\mathrm{U}}$
Vector Space [61]	K_z^{U}	$\frac{\lvert C\rvert^2}{\lvert d\rvert\lvert q\rvert}K_w^{\mathrm{U}}$

Table 5.1: Components of Hofmann's kernel K^{H} and the Vector Space kernel K^{VS} with respect to those of the kernel K^{U} derived from unnormalised log-likelihoods

1. an un-normalised kernel, denoted K^{U}
2. a hybrid derivative called *Vector Space* (VS), in which only the second part of the kernel is normalised. This version is not justified beyond experimental considerations.

Nyffenegger et al. go on to compare K^{H} and the Vector Space Kernel K^{VS} — though giving no results for K^{U}.

We take a step further by evaluating K^{U}, the kernel where normalisation is removed from both components. Furthermore, for all these kernels, we provide experimental results for the components "categories part" and the "term part" alone (denoted K_z^{H} and K_w^{H}, and similarly for U and VS).

As mentioned in section 4.2.1, the term that Hofmann uses in his derivation is not exactly the log-likelihood for documents $L_{\hat{d}_0}(\theta)$ (eq 4.2.2), but rather $L_{\hat{d}_0}(\theta)/|d|$, which comprises a normalisation by document length. His legacy kernel, K^{H}, stems from this basis.

Derivation of the kernel from an un-normalised log-likelihood led to the kernels K^{U} and K^{VS}. These normalisations of log-likelihoods are echoed more of less explicitly in the kernel components K_z and K_w; table 5.1 shows the K_z and K_w components for Hofmann's kernel K^{H}, the Vector Space kernel K^{VS} and the unnormalised kernel K^{U}, taking K^{U} as a reference.

Given the typical orders of magnitude of $|C|$ (size of the document collection) versus typical $|d|$ and $|q|$, it quite clear that the VS kernel is largely dominated by its K_w component, which it inherits from K^{H}:

$$K^{\mathrm{VS}} \simeq K_w^{\mathrm{H}} \tag{5.1.1}$$

This was confirmed experimentally, as shown in detail in chapter 7.

When the log-likelihood is multiplied by a constant, the Fisher kernel is left unchanged. However, if the log-likelihood is multiplied by a function of d not depending on the parameters, i.e. $l_d^{\mathrm{new}}(\theta) = \lambda(d) l_d(\theta)$, s.t. $\nabla_\theta \lambda = 0$, then

$$\begin{aligned} G^{\mathrm{new}}(\theta) &= \mathbf{E}_d\left[(\nabla_\rho l_d^{\mathrm{new}}(\rho))(\nabla_\rho l_d^{\mathrm{new}}(\rho))^T\right] \\ &= \mathbf{E}_d\left[\lambda(d)^2(\nabla_\rho l_d(\rho))(\nabla_\rho l_d(\rho))^T\right], \end{aligned}$$

and

$$K_\theta^{\mathrm{new}}(d,q) = \lambda(d)\,\lambda(q)(\nabla_\theta l_d(\theta))^T\, G^{\mathrm{new}}(\theta)^{-1}(\nabla_\theta l_d(\theta)),$$

which cannot, in the most general case, be expressed in terms of $K_\theta(d,q)$.

However, if the contribution of the Fisher information matrix $G(\theta)$ is neglected (i.e. if $G(\theta) \simeq 1$), as in Hofmann's derivation, then the two kernels do come in relation:

$$K_\theta^{\mathrm{new}}(d,q) = \lambda(d)\,\lambda(q) K_\theta(d,q).$$

Thus, under the assumption that $G(\theta)$ is approximated by $\mathbb{I}$, we expect to find

$$K^{\mathrm{H}}(d,q) = \frac{1}{|d||q|} K^{\mathrm{U}}(d,q).$$

The reason why this is not verified in table 5.1 stems from slightly different assumptions made in their computations: with assumption 4.3.2 (p. 59), Hofmann states that

$$\sum_w \frac{\widehat{P}(w|d)}{P(w|d)} P(w|z) \approx 1 \qquad ,$$

where Nyffenegger et al. make a similar but slightly different assumption on the *joint* probabilities rather than on the *conditional* probabilities:

$$\sum_w \frac{\widehat{P}(d,w)}{P(d,w)} P(w|z) \approx 1.$$

The two relations are linked by

$$\sum_w \frac{\widehat{P}(w|d)}{P(w|d)} P(w|z) = P(d) \frac{|C|}{|d|} \sum_w \frac{\widehat{P}(d,w)}{P(d,w)} P(w|z).$$

Replacing the first approximation with the second amounts to turning $|d|/|C|$ into $P(d)$, well in line with assumption 4.3.5 (p. 59). Substitution leads directly to the formulae of table 5.1, as expected.

We could thus consider three similar "Hofmann"-like kernels: K^{H} as originally computed and described above, but also $K^{\mathrm{H}_1} = \frac{|C|^2}{|d||q|} K^{\mathrm{U}}$, which has the same k_z as K^{H} but a different k_w, and $K^{\mathrm{H}_2} = \frac{1}{P(d)P(q)} K^{\mathrm{U}}$, which has the same k_w as K^{H} but a different k_z.

We evaluated these three kernels and no change occurred in the results; observed differences in the document similarities were in the order of 10^{-4}%. This is because $|d|/|C|$ is indeed a very good estimator of $P(d) = \sum_z P(d|z)P(z)$.

For coherence with existing literature, we focused on K^{H} rather than K^{H_1} or K^{H_2}.

5.2 Role of the Fisher Information matrix

The Fisher kernel (eq. 2.5.7 p. 30) is defined as a dot product on an alternative representation space; as such, it contains a metric tensor that denotes the topology of this alternative space. This tensor, the *Fisher Information matrix* $G(\theta)$, is the covariance of the Fisher score $U_X(\theta)$ (see equations 2.5.5 and 2.5.6 p. 29):

$$G(\theta) = \mathbb{E}_X[U_X(\theta)U_X(\theta)^T]$$

The Fisher kernel for PLSI derived by Hofmann [28] was deemed to neglect the contribution of the Fisher information matrix $G(\theta)$: Hofmann identified $G(\theta)$ with the identity matrix through a re-parametrisation suited for multinomial models; however, PLSI is neither a multinomial, nor in an exponential family, and $G(\theta)$ may significantly differ from identity in such a case [61].

Besides, the formal derivation of the actual PLSI kernel, through the IID point of view as detailed in chapter 4, yields a tensor which is indeed not close to the identity matrix (see figure 5.1, p. 68, for an example).

All attempts to take $G(\theta)$ into account were carried out with the assumption that $G(\theta)$ can be approximated by its diagonal.

This can be justified by theory to some extent: equation 4.2.15 (p. 58) contains a $\delta_{ww'}$ term; the terms of $G(\theta)$ stem from a similar expression. $G(\theta)$ is thus genially diagonal in its "word part". We extend this hypothesis to the "topic part" also.

The reason for this is mostly computational: the complete tensor $G(\theta)$ holds $|Z|^2 + (|Z| \cdot |V|)^2$ terms, which in itself entails an enormous amount of computation. Furthermore, it is the *invert* of $G(\theta)$ that appears in the Fisher kernel: inverting such a matrix is conceivable only in the most succinct of document collections.

To distinguish clearly the different cases and hypothesis, we shall label the forms that derive from the postulate that $G(\theta)$ can be approximated by its diagonal as *DFIM* (for "Diagonal Fisher Information Matrix"); in contrast, models that do not take $G(\theta)$ into account will be called *IFIM* (for "Identity Fisher Information Matrix").

Nyffenegger et al. [61] present a comparative study of the Hofmann kernel K^{H}, the Vector Space Kernel K^{VS} — that amounts to K^{H}_w, see equation 5.1.1 (p. 64); and the DFIM-VS Kernel $K^{\text{DFIM-VS}}$ (which similarly amounts to $K^{\text{DFIM-H}}_w$). We took the matter further by introducing several other measures:

- $K^{\text{DFIM-H}}$, similarly built upon K^{H} (also known as $K^{\text{IFIM-H}}$);
- the unnormalised Fisher kernel K^{U}, already encountered in section 5.1 – which we also call $K^{\text{IFIM-U}}$ to underline the assumption on $G(\theta)$;
- the unnormalised Fisher kernel K^{U} with the Fisher information matrix enabled, denoted $K^{\text{DFIM-U}}$;
- all the relative topic and word parts of these kernels.

Table 5.2 lists the resulting expressions depending of the assumptions on $G(\theta)$ and on log-likelihood normalisation that are selected.

Examination of the DFIM family reveals the normalising role of $G(\theta)$:

- all the terms that are independent of d cancel out, as they can be factorised in $\alpha(z)$ or $\gamma(w,z)$. This is the factor which balances K_z with respect to K_w;

		k_z	k_w
IFIM-U	Theory	$\frac{P(d,z)P(q,z)}{P(z)}$	$\sum_z \widehat{P}(d,w)\widehat{P}(q,w)\frac{P(d\|z)P(q\|z)}{P(d,w)P(q,w)} \cdot P^2(z)P(w\|z)$
	Impl.	$P(d\|z)P(q\|z)P(z)$	$\frac{1}{\|C\|^2}\frac{n(d,w)}{P(d,w)}\sum_z \upsilon_q(w,z)P(d\|z)P(w\|z)P^2(z)$
IFIM-H	Theory	$\frac{P(z\|d)P(z\|q)}{P(z)}$	$\sum_z \widehat{P}(w\|d)\widehat{P}(w\|q)\frac{P(d\|z)P(q\|z)}{P(d,w)P(q,w)} \cdot P^2(z)P(w\|z)$
	Impl.	$\frac{1}{P(d)P(q)}P(d\|z)P(q\|z)P(z)$	$\frac{1}{\|d\|\|q\|}\frac{n(d,w)}{P(d,w)}\sum_z \upsilon_q(w,z)P(d\|z)P(w\|z)P^2(z)$
DFIM-U	Theory	$\frac{P(d,z)P(q,z)}{P(z)}\left(\sum_d \frac{P^2(d,z)}{P(z)}\right)^{-1}$	$\sum_z \widehat{P}(d,w)\widehat{P}(q,w)\frac{P(d\|z)P(q\|z)}{P(d,w)P(q,w)} \cdot \left[\sum_d \widehat{P}^2(d,w)\frac{P^2(d\|z)}{P^2(d,w)}\right]^{-1}$
	Impl.	$P(d\|z)P(q\|z)/\alpha^{\text{DFIM-F}}(z)$	$\frac{n(d,w)}{P(d,w)}\sum_z \frac{\upsilon_q(w,z)P(d\|z)}{\gamma^{\text{DFIM-F}}(w,z)}$
DFIM-H	Theory	$\frac{P(z\|d)P(z\|q)}{P(z)}\left(\sum_d \frac{P^2(z\|d)}{P(z)}\right)^{-1}$	$\sum_z \widehat{P}(w\|d)\widehat{P}(w\|q)\frac{P(d\|z)P(q\|z)}{P(d,w)P(q,w)} \cdot \left[\sum_d \widehat{P}^2(w\|d)\frac{P^2(d\|z)}{P^2(d,w)}\right]^{-1}$
	Impl.	$\frac{1}{P(d)P(q)}\frac{P(d\|z)P(q\|z)}{\alpha^{\text{DFIM-H}}(z)}$	$\frac{1}{\|d\|\|q\|}\frac{n(d,w)}{P(d,w)}\sum_z \frac{\upsilon_q(w,z)P(d\|z)}{\gamma^{\text{DFIM-H}}(w,z)}$

With

$P(d) = \sum_z P(d|z)P(z)$

$P(d,w) = \sum_z P(z)P(w|z)P(d|z)$

$\upsilon_q(w,z) = \frac{n(q,w)P(q|z)}{P(q,w)}$

$\alpha^{\text{DFIM-U}}(z) = \sum_d P^2(d|z)$

$\alpha^{\text{DFIM-H}}(z) = \sum_d \frac{P^2(d|z)}{P^2(d)}$

$\gamma^{\text{DFIM-U}}(w,z) = \sum_d \left(\frac{P(d|z)n(d,w)}{P(d,w)}\right)^2$

$\gamma^{\text{DFIM-H}}(w,z) = \sum_d \left(\frac{P(d|z)n(d,w)}{P(d,w)}\frac{1}{|d|}\right)^2$

Note: substitution of an expression by a variable in the implementation formula reflects actual pre-computation or separate computation of the relevant quantities in the code.

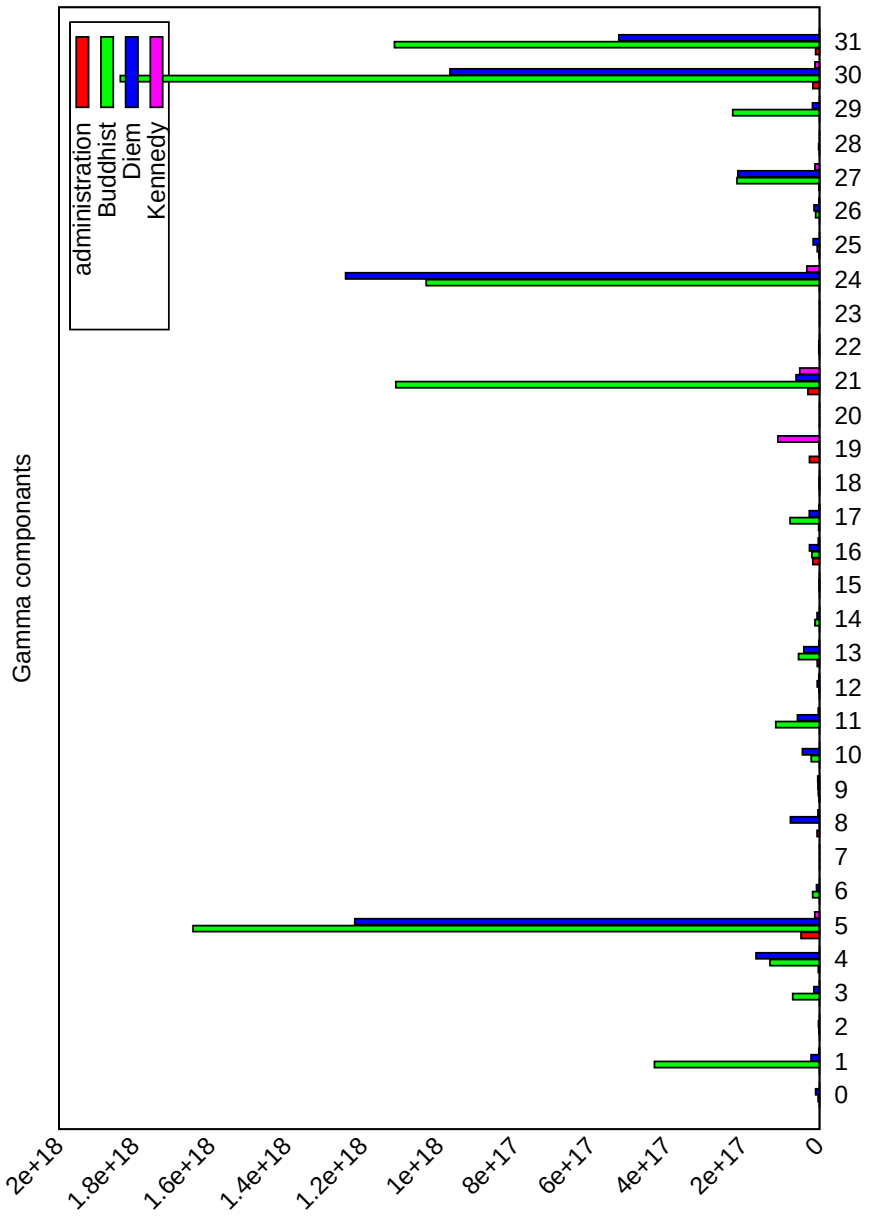


Figure 5.1: Values of the Fisher information matrix $G(\theta)$ for four in the TIME collection, on 32 categories. They are clearly not equal, and the approximation $G(\Theta) \simeq \mathbb{I}$ would not result in a good estimator of $G(\Theta)$.

- k_z and k_w are close to the form $\frac{a \cdot b}{a^2+b^2+c^2}$, which is bounded by 0.5 — this form is exact when $q \in C$. In practise, this allows implementing a trigger which cuts out impossible values stemming from numerical divergence caused by very small values, and diminishes the risks of numerical errors due to a form where both $U_d(z)U_q(z)$ and α, or both $U_d(w|z)U_q(w|z)$ and γ, would tend to zero.

Hence, in practise, the Fisher Information Matrix effectively prevents K_z from yielding larger and larger scores as the number of categories grows and from overwhelming K_w (see figure 5.1). This had been the original idea behind the "Vector Space" kernel: by normalising only its K_z component, K^{VS} managed to keep its values lower, thus obtain better results as the contributions of K_z could be seen. K^{VS} was however a crude attempt at balancing the kernel, and introducing the Fisher information matrix to build $K^{\text{DFIM-VS}}$ resulted in an unjustifiable kernel.

To reconnect with theoretical considerations of chapter 4, expressing the contribution of $G(\theta)$ comes closer to the actual PLSI Fisher kernel that stems from IID processes. There are thus both theoretical and practical reasons to properly accounting for the contribution of $G(\theta)$.

5.3 Conclusions

The normalisation by document length was introduced in Hofmann's kernel as an analogy with `tf-idf`, but had no further justification. It was then challenged by Nyffenegger et al., who proposed an unnormalised version. From the IID kernel of chapter 4, we know that this normalisation stems from the IID nature of the PLSI process.

Based on empirical reasons, Nyffenegger et al. also proposed a "Vector Space" version, where only the K_z part of the kernel was normalised. We show this version of the kernel appeared to work because it artificially moderates the contribution of K_z, giving more relative importance to K_w. Our study of the Fisher information matrix shows that such an effect is indeed desirable for both practical and theoretical reasons, but that the proper way to obtain it is to account for the contribution of the Fisher information matrix rather than approximate it by the identity.

We come to the conclusion that the Fisher kernel to deploy on PLSI, combining light computational requirements, adequate theoretical justifications and good performances, is $K^{\text{DFIM-H}}$:

$$K^{\text{DFIM-H}} = \frac{P(z|d)P(z|q)}{P(z)} \left(\sum_d \frac{P^2(z|d)}{P(z)} \right)^{-1} + \sum_z \widehat{P}(w|d)\widehat{P}(w|q) \frac{P(d|z)P(q|z)}{P(d,w)P(q,w)} \cdot \left[\sum_d \widehat{P}^2(w|d) \frac{P^2(d|z)}{P^2(d,w)} \right]^{-1}$$

Eventally, we also provide a table of equivalent formulations of the different kernels as to aid to practical implementation.

Chapter 6

Latent-based Smoothed Dirichlet

Abstract

Chapters 4 and 5 have discussed in details the similarities available for PLSI. We saw that the non-generative nature of PLSI hinder its use in IR as incremental indexing is impossible. These shortcomings warrant the study of alternative topic-based IR models; one of the requirements for the new models would be to be fully generative.

Generative models for IR are composed of two components: one generates the topics (the $P(z)$ part); the other generates the words (the $P(w|z)$ part). An abundant literature already exists for perfecting topic generation (LDA, Correlated Topics,...). On the other hand, word generation for topic models was paid relatively little attention, in spite of the literature available in the domain [83]. Notably, most topic models generate their words using multinomials, though they are known to account poorly for burstiness and markably underestimate the tail of Zipf's law.

In this chapter, we propose to include better word generation (one that would account for word burstiness) in a probabilistic latent semantic model. To this effect, we study the Smoothed Dirichlet distribution, designed specifically to account for burstiness, and advertised as usable as a "new building block for topical models". After experimenting with SD in a classical IR setting, we propose a generative topic model constituted by a simple mixture of SD distributions; we call this model *Latent-based Smoothed Dirichlet*. We provide the inference scheme as well as the Fisher kernel for it.

6.1 Introduction

The preceding chapters contribute to the framework of PLSI in two ways. On one hand, the accurate derivation of the PLSI Fisher kernel provides a better understanding of its nature and clearly heralds the links between it and the Hofmann's legacy kernel, provides markably better performances and theoretical justifications with the Fisher information matrix, and details the computational difficulties of implementing and running it. On the other hand, the similarity based on language models provides a theoretically sound alternative to the Fisher kernel that entirely bypass the process of "folding in", a step that features practical difficulties and is not well-justified.

Nevertheless, in spite of the theoretically sound IR scheme allowed by Language Models, PLSI itself retains undesirable features which make it difficult to deploy on large collections. Due to its $P(d|z)$ parameter, it is fundamentally not a completely generative model, in the sense that it cannot straightforwardly provide the representation of a new document without re-processing the entire database.

Furthermore, the learning phase runs over expectation of the parameters $P(z)$, $P(d|z)$ and $P(w|z)$, and the maximisation of a quantity $P(z|w,d)$, a three-dimensional tensor with $|V| \cdot |D| \cdot |Z|$ terms, a number that grows quickly as the collection grows; this forces to either re-compute the same terms over and over again, extending the running time of the learning algorithm, or storing large amounts of data in memory, which can consume tens of gigabytes of RAM with collection that are by no means exceptionally large.

These drawbacks call for new models that would

1. be fully generative
2. display performances surpassing those of PLSI[1]

The first requirement entails that the new models would be composed of only two components: the first one to generate the topics, and the second part to generate the words appearing in the documents of the collection. The second requirement calls for advances in either of these components or in both

Perfecting the generation of topics has been a subject of much recent research. The Latent Dirichlet Allocation (LDA) model [7] replaced the simple mixture of Naive Bayes with a Dirichlet distribution; using multinomials to generate words after choosing the topics with a Dirichlet couples the multinomial with its conjugate prior distributions. This makes the Dirichlet-multinomial couple an appealing package for a latent model.

Further proposals have been made to account for topic-specific phenomenons, like using the logistic distribution to account for the correlation between topics [6].

Contrasting with the literature devoted to sophisticated generation of topic models, there are no proeminent topic models putting a specific effort in an accurate generation of words in documents — though specific efforts are beginning to appear [16]. In non-topic IR, this issue has generated much debate since the early days of `tf-idf` and until recently [51].

We attempt to contribute to the issue of word generation in topic models by challenging the multinomial generation of terms and replacing it with a better term generating distribution, grounded in experimental evidences and observations on document collections. Specifically, we will attempt to accurately model word burstiness by importing techniques developed for non-topical IR and using them to generate words within a probabilistic latent semantic model.

6.2 Smoothed Dirichlet

Experimental observation of document collections indicate that rare terms tend to appear in "bursts" — that is, if a rare term does appear in a text, it is experimentally observed that it tends to appear more than once (by contrast to rare terms that do not appear at all). Indeed, appearance of a rare term often reflects that it is at the core the document topic, making it locally more likely to occur than what multinomials over the entire document collection predict. This phenomenon induces a marked underestimation of the tail of Zipf's law by multinomial distributions.

The *Dirichlet Compound Multinomial* [51] was proposed as a way to better match experimentally observed distributions of terms and improve estimation of the tail of Zipf's law. However, this model entails intractable non-closed-form expressions, which makes it impractical for actual utilisation. In order to actually deploy a well-suited term-distribution estimator based on DCM, approximations were proposed.

[1] we do not satisfy ourselves with Naive Bayes or with non-topic models

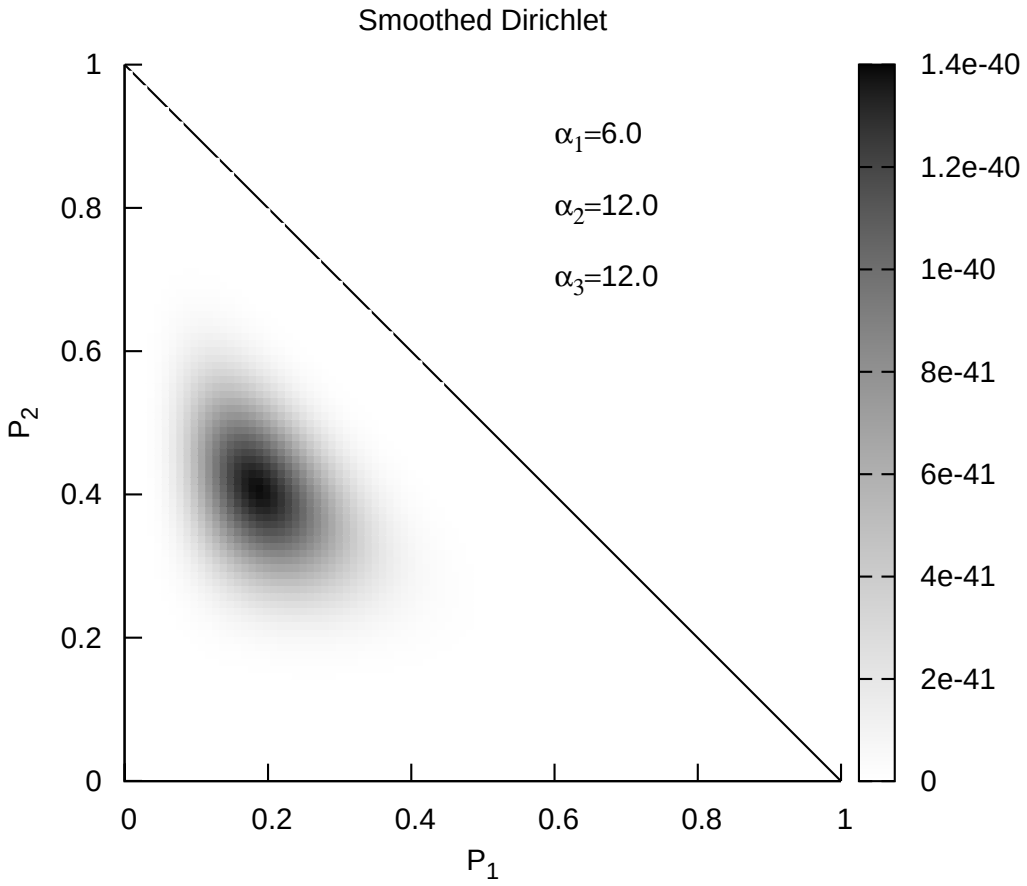


Figure 6.1: The Smoothed Dirichlet distribution simplex plotted for three variables of priors values as indicated.

The Smoothed Dirichlet Distribution constitutes such an approximation [58]. It is a generative process that provides theoretically founded approximations for the Dirichlet distribution and that yields closed-form expressions. These expressions are similar in complexity to those entailed by multinomials, but they retain the desired features of heavy-tailed distributions.

Nallapati's original work uses a scheme based on language models to compare documents and queries. A "relevant" model is built from the query, while a "non-relevant" model is built from the entire collection. To give more substance to the sample from which the "relevant" model is built, a second pass is made using a handful of the most relevant documents found during the first pass as data, in the fashion of pseudo-relevance feedback.

6.3 Latent Smoothed Dirichlet

6.3.1 The Latent Smoothed Dirichlet model

The Latent Smoothed Dirichlet model is an attempt to combine the advantages of Smoothed Dirichlet (realistic modeling of term generation) with those of generative latent semantics models. The framework in which we operate is that described in 1.1 (page 19). For a first attempt, in order to test the

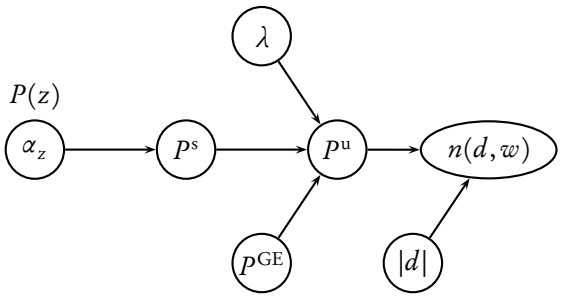


Figure 6.2: The Smoothed Dirichlet graphical model

validity of using Smoothed Dirichlet as a basic functional element for latent topic models, we chose the very simple topics scheme of the Naive Bayes model, modeling the document collection as a mixture of topics where each document stems from a single category.

The principle is to generate terms with several Smoothed Dirichlet distributions, each associated with a category z through its parameter α_z; concretely, this translates into terms $\alpha(w|z)$ whose role mimics that of the $P(w|z)$ in Naive Bayes.

The model thus summarises as:

1. draw a language model α_z with probability $P(z)$
2. draw a smoothed distribution P^{s} from Smoothed Dirichlet with parameter α_z
3. using λ and P^{GE}, unsmooth the distribution into P^{u} as done in [58]:

$$P^{\mathrm{u}} = \frac{1}{\lambda}\Big(P^{\mathrm{s}} - (a-\lambda)\cdot P^{\mathrm{GE}}\Big)$$

4. draw $n(d,w)$ from P^{u}

The original Smoothed Dirichlet distribution is

$$P(d|\alpha) = \frac{S^S}{Z^{\mathrm{SD}}} \prod_{w\in V} \Big(\theta^d(w)\Big)^{\alpha(w)-1} \tag{6.3.1}$$

with $Z^{\mathrm{SD}} = \prod_{w\in V} \alpha(w)^{\alpha(w)}$, $S = \sum_{w\in W} \alpha(w)$, and $\theta^d(w) = \lambda \cdot \frac{n(d,w)}{|d|} + (1-\lambda)P^{\mathrm{GE}}$

From eq. (6.3.1), we can write

$$P(d,z|\alpha) = \frac{P(z)}{Z^{\mathrm{SD}}(z)} \prod_{w\in V} \Big(\theta^d(w)\Big)^{\alpha_z(w)-1} \tag{6.3.2}$$

with $Z^{\mathrm{SD}}(z) = \prod_{w\in V} \alpha_z(w)^{\alpha_z(w)}$

6.3.2 Posterior inference and parameter estimation

The parameters Θ of the model are the $|Z|$ language priors α_z, and their respective probabilities $P(z)$. We observe the documents of the training set $d \in D$ by their term occurrences $Y = \{n(d,w)\}$; the length $|d|$ and P^{GE} are also known.

The Expectation-Maximisation algorithm is very much similar to the one for Naive Bayes, which we developed in details in section 3.2.2 (p. 39), except that the term $\frac{\partial Q}{\partial P(w|z)}$ of NB is replaced by a term in $\frac{\partial Q}{\partial \alpha_z(w)}$

Terms: We derive $Q(\Theta, \Theta^{\text{old}})$ with respect to the $\alpha_\zeta(w)$, for a given category $\zeta \in Z$ and word $\omega \in V$.

$$\frac{\partial Q(\Theta, \Theta^{\text{old}})}{\partial \alpha_z(w)} = \sum_d P_{\text{old}}(\zeta|d) \cdot \frac{\partial \log P(d|\zeta)}{\partial \alpha_\zeta(\omega)}$$

In the Smooth Dirichlet model $\log P(d|z)$ amounts to:

$$\log P(d|z) = \log(S_z{}^{S_z}) - \log(Z_z) + \sum_w \Big(\alpha_z(w) - 1\Big) \log \tilde{P}(w|z)$$

with $\log(Z_z) = -\sum_w \alpha_z(w) \log \alpha_z(w)$. The derivation goes

$$\frac{\partial Q(\Theta, \Theta^{\text{old}})}{\partial \alpha_z(w)} = \sum_d P_{\text{old}}(\zeta|d) \cdot \Big[-\big(\log \alpha_z(w) + 1\big) + \log \tilde{P}(w|d)\Big] = \mu$$

with μ the Lagrange multiplier and $\tilde{P}(w|d)$ the smoothed estimator of $P(w|d)$, typically $\tilde{P}(w|d) = \lambda \frac{n(d,w)}{|d|} + (1-\lambda) P_{\text{GE}}(w)$. With the usual probabilistic form for Lagrange multipliers $\alpha = \mu \cdot T \Longrightarrow \alpha = \frac{T}{\sum_i T_i}$, we find

$$\log \alpha_z(w) = \sum_d \frac{P(z|d)}{\sum_d P(z|d)} \cdot \log \tilde{P}(w|d) + K \tag{6.3.3}$$

with constant K such as $\sum_w \alpha_z(w) = S$ (S is a free parameter chosen by the user). We thus have

$$K = \frac{S}{\sum_w \bar{\alpha}(z,w)}$$

with $\bar{\alpha}(z,w) = \exp\big[\sum_d P(z|d) \log \tilde{P}(w|d)\big]$, the unnormalised values for the α.

Summary: EM learning for LSD can be implemented as:

E-step	$P(z\|d) = \frac{P(d,z)}{\sum_{\zeta \in Z} P(d,\zeta)}$
M-step	$P(z) = \frac{\sum_d P(z\|d)}{\sum_d \sum_z P(z\|d)}$ $\log \alpha_z(w) = \sum_d P(z\|d) \log \tilde{P}(w\|d) + K$

with:

- $P(d,z) = P(z) \frac{S_z^{S_z}}{Z(z)} \prod_w \tilde{P}(w|d)^{\alpha_z(w)-1}$
  $\rightarrow \log P(d,z) = \log P(z) + S \log(S) - \log Z(z) + \sum_w (\alpha_z(w) - 1) \log \tilde{P}(w|d)$
  with $\log Z(z) = \sum_w \alpha_z(w) \log \alpha_z(w)$
- $K =$ (normaliser)

Implementation

The E-step tends to yield very small values in its numerator and its denominator. To avoid situations where $P(z|d) \to \frac{0}{0}$, we consider the log of the expression: $\log P(z|d) = \log P(d,z) - \log \sum_z P(d,z)$. Because of the $\log \sum_z P(d,z)$ term, it is not tractable. We use the trick

$$\log \sum_{z \in Z} P(d,z) = \log \sum_{z \in Z} \frac{P(d,z)}{P(d,z_0)} + \log P(d,z_0)$$

with $\log P(d,z_0) = \max_z \log P(d,z)$. In this configuration, $\sum_{z \in Z} \frac{P(d,z)}{P(d,z_0)}$ is a sum of elements smaller than 1, which prevents it from diverging for $|Z|$ small (a few hundreds at most).

The parameter $P(z)$ poses no particular problem and can be written straightforwardly:

$$P(z) = \frac{\sum_d P(z|d)}{\sum_z \sum_d P(z|d)}$$

Additionally, $\sum_z \sum_d P(z|d) = |D|$, number of documents in the corpus.

The parameter $\alpha(z,w)$ is computed by is logarithm.

$$\log \alpha_z(w) = \sum_d P(z|d) \log \tilde{P}(w|d) - \log K$$

It can yield very large quantities, yielding a problem similar to that posed $P(z|d)$.

Again, the $\sum_w \bar{\alpha}(z,w)$ can yield large values. We thus write

$$K = \frac{K}{\varkappa}\varkappa = \frac{S}{\sum_w \bar{\alpha}(z,w) \cdot \varkappa} \cdot \varkappa$$

with $\varkappa$ a constant chosen as to keep $\bar{\alpha}(z,w) \cdot \varkappa = \exp\left[\sum_d P(z|d) \log \tilde{P}(w|d) + \varkappa\right]$ small enough to be tractable. We finally compute

$$\log \alpha(z,w) = \left[\sum_d P(z|d) \log \tilde{P}(w|d)\right] - \left[\log \frac{S}{\sum_w \exp\{\log \bar{\alpha}(z,w) + \varkappa\}}\right] + \log \varkappa.$$

We typically choose $\varkappa = -\max_z \bar{\alpha}(z,w)$

6.3.3 Fisher kernels for LSD

The fundamental equation for the LSD model is

$$P(d,z) = P(z) \frac{S_z^{S_z}}{Z(z)} \prod_w \tilde{P}(w|d)^{\alpha(w|z)-1}$$

This entails that the log-likelihood of a document for given values of the parameters is

$$L_d(\theta) = \log P(d) = \log \left[\sum_z \frac{P(z)}{Z(z)} \prod_w \tilde{P}(w|d)^{\alpha(w|z)-1}\right] + S_z^{S_z}$$

The Fisher kernel is given by

$$K(d,q) = U_d^T G^{-1} U_q = U_d^T \left[\mathbb{E}_d U_d U_d^T\right]^{-1} U_q =$$

with $U_d = \nabla L_d$. This amounts to deriving L_d by the parameters: $P(z)$ and $\alpha(w|z)$.

Derivation by $P(z)$:

$$\frac{\partial L_d}{\partial P(z)} = \frac{\partial}{\partial P(z)} \log P(d) = \frac{1}{P(d)} \frac{\partial P(d)}{\partial P(z)} = \frac{P(d|z)}{P(d)} = \frac{P(z|d)}{P(z)}$$

Derivation by $\alpha(w|z)$:

$$\frac{\partial L_d}{\partial \alpha(w|z)} = \frac{\partial}{\partial \alpha} \log P(d) = \frac{1}{P(d)} \frac{\partial}{\partial \alpha} P(d)$$

$$\frac{\partial L_d}{\partial \alpha(w|z)} = \frac{1}{P(d)} \frac{\partial}{\partial \alpha} \left[\sum_z P(z) \prod_w \frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} \right] S^S$$

Two observations arise:

- Derivation by $\alpha(w|\zeta)$ cancels out the terms of the sum for which $z \neq \zeta$
- Derivation by $\alpha(\omega|z)$ turns all the terms of the product for which $w \neq \omega$ into constants

This entails that

$$\frac{\partial L_d}{\partial \alpha(w|z)} = \frac{P(z)}{P(d)} \left(\prod_{w \neq \omega} \frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} \right) \cdot \frac{\partial}{\partial \alpha(w|z)} \frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} \cdot S^S \tag{6.3.4}$$

$$= \frac{P(z)}{P(d)} \left(P(d|z) \cdot \left[\frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} \right]^{-1} \right) \cdot \frac{\partial}{\partial \alpha(w|z)} \frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} \cdot S^S \tag{6.3.5}$$

The details of the derivation of $\frac{\partial}{\partial \alpha(w|z)} \frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha}$ is:

$$\frac{\partial}{\partial \alpha} \frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} = \frac{1}{\alpha^\alpha} \frac{\partial}{\partial \alpha} \tilde{P}^{\alpha-1} - \tilde{P}^{\alpha-1} (\alpha^\alpha)^{-2} \frac{\partial}{\partial \alpha} \alpha^\alpha \tag{6.3.6}$$

$$= \frac{\log \tilde{P} \cdot \tilde{P}^{\alpha-1}}{\alpha^\alpha} - \frac{\tilde{P}^{\alpha-1} \alpha^\alpha \cdot (\log \alpha + 1)}{(\alpha^\alpha)^2} \tag{6.3.7}$$

$$= \frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} \left(\log \tilde{P} - \log \alpha - 1 \right) \tag{6.3.8}$$

Therefore, injecting 6.3.5 into 6.3.8, we have

$$\frac{\partial L_d}{\partial \alpha(w|z)} = \frac{P(z)P(d|z)}{P(d)} \cancel{\frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha}} \left(\log \tilde{P} - \log \alpha - 1 \right) \cancel{\left(\frac{\tilde{P}^{\alpha-1}}{\alpha^\alpha} \right)^{-1}} \tag{6.3.9}$$

$$= P(z|d) \left(\log \tilde{P} - \log \alpha - 1 \right) \tag{6.3.10}$$

Fisher information matrix The components can be written

$$G_z(z) = \sum_d \frac{P^2(z|d)}{P^2(z)} = \frac{1}{P^2(z)} \sum_d P^2(z|d)$$

$$G_\alpha(w,z) = \sum_d P^2(z|d) \left(\log \tilde{P}(w|d) - \log \alpha(w|z) - 1 \right)^2$$

Fisher kernel With $u(w,z,d) = \log \tilde{P}(w|d) - \log \alpha(w|z) - 1$, the components can be written

$$K_z(d,q) = \sum_z \frac{P(z|d)P(z|q)}{\sum_\delta P^2(z|\delta)}$$

$$K_\alpha(d,q) = \sum_w \sum_z \frac{P(z|d)P(z|q)u(w,z,d)u(w,z,q)}{\sum_\delta P^2(z|\delta)u^2(w,z,\delta)} = \sum_z P(z|d)P(z|q) \sum_w \frac{u(w,z,d)u(w,z,q)}{G_{\alpha(w,z)}}$$

We thus eventually reach

$$K^{\mathrm{LSD}} = \sum_{z\in Z} P(z|d)P(z|q) \cdot \left[\frac{1}{\sum_{\delta\in D} P(z|\delta)^2} + \sum_{w\in V} \frac{u(w,z,d)u(w,z,q)}{\sum_{\delta\in D} [P(z|\delta)u(w,z,\delta)]^2} \right] \tag{6.3.11}$$

Note that the original Smoothed Dirichlet scheme relies on a Language model approach to rank documents. Here, we have changed the point of view and built a Fisher kernel that can be used in the usual IR scheme.

Unfortunately, shortage of time did not allow us to perform conclusive experiments to validate the model. Specifically, we have not yet been able to have the EM process converge towards non-trivial values.

We have noted that in his work, Nallapati sets the value of the S parameter to $S = \sum_{w\in V} \alpha_w = 1$. Doing so, he forces all α_w to have values lesser than 1. However, the domain $\alpha \in [0;1]$ is where the Smoothed Dirichlet approximation of the Γ function ceases to be a good estimator of Γ (see figure 6.3).

When $\alpha \to 0$, the $\prod_{w\in V} \left(\theta^d(w)\right)^{\alpha_z(w)-1}$ part yields a product over numerous terms ($\prod_{w\in V}$) smaller than zero ($\theta^d(w)$) powered by a term than tends to -1 ($\alpha_z(w) - 1$); therefore, this term tends to explode as $\alpha \to 0$. In the actual Dirichlet distribution, the behaviour is compensated by a similarly divergeant Γ function at the denominator. Bounding the approximation prevents this moderation from taking effect.

6.4 Conclusions

Building upon Smoothed Dirichlet, a state-of-the-art distribution for word generation, we propose a probabilistic latent semantic model whose word generation accounts for word burstiness, thus better modelling experimentally observed corpora than what can be done using multinomials. From the fundamental hypothesis of the generation model as a mixture of Smoothed Dirichlet distributions, we provide inference equations and derive the Fisher kernel adapted to the model. In the course of our developments, we raise the issue of the $S = \sum_{w\in V} \alpha(w)$ parameter, which Nallapati sets to 1, thereby concentrating on the domain where his distribution diverges from Dirichlet distribution.

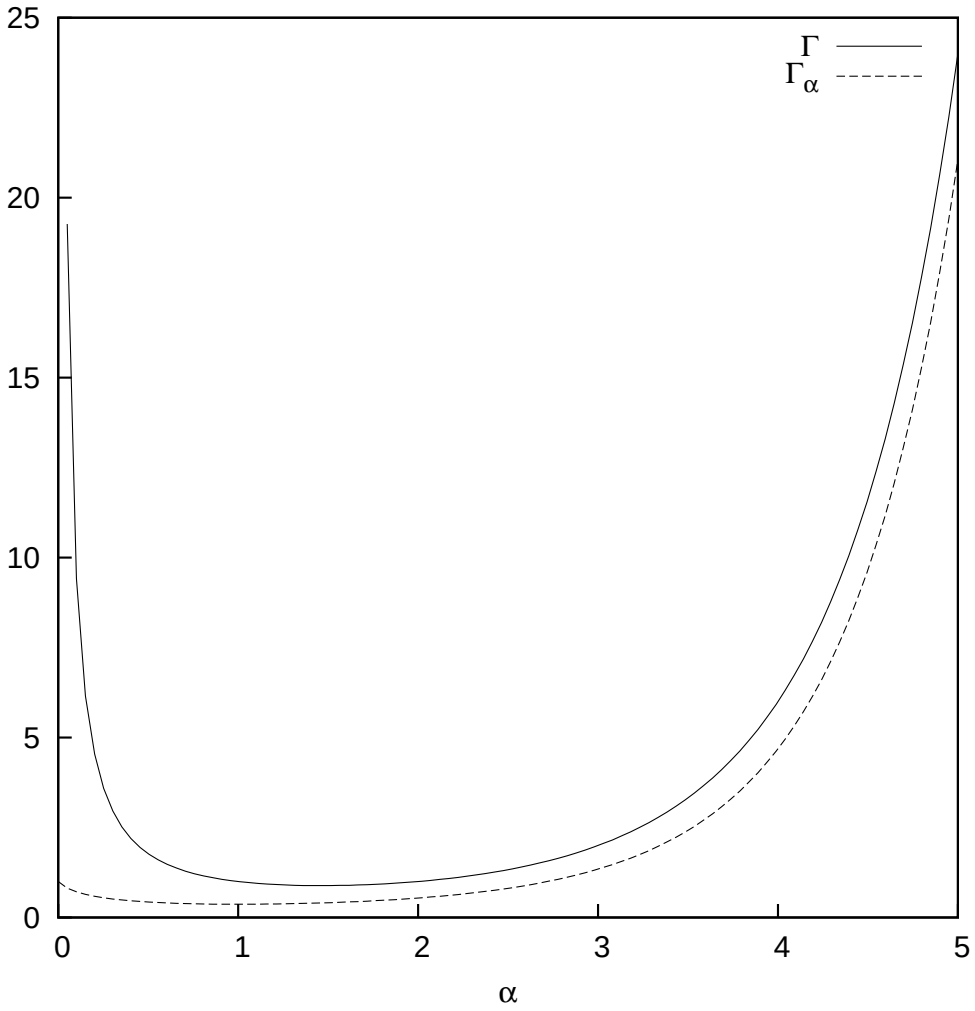


Figure 6.3: The Smoothed Dirichlet approximation of Γ. The approximation of Γ for $\alpha \to 0$, while Γ diverges.

Chapter 7

Experimental results

Abstract

The three previous chapters have proposed and discussed a variety of document similarities for probabilistic latent semantic models. Chapter 4 (p. 53) develops the Fisher kernel from the IID nature of the generative process, and links the resulting kernel to the kernels proposed by Hofmann in [28]. Chapter 5 (p. 63) discusses the respective contributions of the different parts that constitute the Fisher kernel for PLSI. Eventually, chapter 6 (p. 71) presented the Smoothed Dirichlet distribution and a new generative model constituted by a mixture of SD distributions.

This chapter provides comparatives results for these different models, both between each other and with respect to the strong baseline provided by the BM25 model [66], as well as the Language Model approach (section 2.5.2, p. 27) applied to PLSI.

We show that Hofmann's legacy kernel $K^{\text{DFIM-H}}$ mimics the behaviour of the K^{IID} kernel; that DFIM versions of kernels generally outperform IFIM versions; that K_w performs much better than K_z, to the point where the contribution of K_z may as well be discarded altogether; and that Language Models provide similarities that can match $K^{\text{DFIM-H}}$ given enough latent categories. The best document similarities, stemming either from Fisher kernels or from Language Models, can compete with BM25, and even outperform it in specific circumstances.

7.1 Introduction

The previous chapters provide us with a number of different document similarity measures:

- the Fisher kernels developed by Hofmann, K^{H} – section 3.3.3 (p. 46) – and by Nyffenegger et al. – section 5.1 (p. 63).
- the new IID-based Fisher kernel K^{IID} — equation 4.2.3 (p. 54);
- components K_z and K_w alone for the above kernels (section 5.1 p. 63);
- the Smoothed Dirichlet scheme and the mixture of SD, chapter 6 (p. 71).

In addition, we will consider similarities based on Language models (section 2.5.2, p. 27). We specifically developed Language Model-based similarities for PLSI and compared them to the various Fisher

kernels listed above.

This chapter experimentally addresses the following questions:

1. Are there significant differences between all the possible variants of the Fisher kernel? Which one is the best?
2. How does the IID kernel compete with variants of Hofmann's kernel on PLSI?
3. How do the document similarities based on language models fare, between each other and with respect to the Fisher kernels?
4. How do these compare to the IR baselines like BM25 [66]?

7.2 Experimental framework

To address the above questions, we experimented on the standard Information Retrieval benchmarks from the SMART collection[1]: CACM, CISI, MED, CRAN and TIME.

These bases might appear very small with respect to the current trend of "terabyte" collections. We use them because they are in line with previously published work on PLSI, thus allowing use to compare our work with that of Hofmann and others.

Another major experimental issue is the non-generative nature of PLSI for unknown document models (section 3.3.4, p. 51): because PLSI representation is not readily available for a particular document outside of the collection, the learning process to be applied to the entire collection as a whole — in other terms, it is not possible to perform an iterative indexation of the document collection.

Parameter estimation on the *entire* document collection quickly becomes intractable as the size of the collection increases. In practise, it is therefore impossible to use PLSI on collections of the same order of magnitude as those of TREC, for instance.

In order to explore the limits of PLSI learning tractability, we experimented on a collection significantly bigger than the SMART ones. The new collection is constituted of the 7466 first documents of the TREC–AP 89 corpus [24] [2], which amounts to over 5 times as many documents and 10 times as many word occurrences as in the SMART collection.

As such, the EM learning process at 128 categories took 45 hours of CPU time and used 6.7 Gb of RAM on a dedicated computer server with one octo-core 2-GHz Intel Xenon processor and 32 Gb of memory. This collection was evaluated against queries 1 to 50.

For all the experiments, stemming was performed using the Porter algorithm of Xapian [3]. Evaluation results were obtained using the standard `trec_eval` tool [4]. The main characteristics of the evaluation corpora (SMART and AP) are given in Table 7.2 (p. 83).

For experiments on the SMART collection, 6 runs with different learning initial conditions were performed for all the models, and for different numbers of topics: $|Z| \in \{1, 2, 8, 16, 32, 64, 128\}$ For the TREC-AP part, due to its size, a single run was performed for each $|Z| \in \{1, 32, 48, 64, 80, 128\}$.

[1] `ftp://ftp.cs.cornell.edu/pub/smart/`
[2] Documents AP890101-0001 to AP890131-0311.
[3] `http://xapian.org/`
[4] `http://trec.nist.gov/trec_eval/`

	CACM	CRAN	TIME	CISI	MED	AP89_01XX
# Terms (stems)	4 911	4 063	13 367	5 545	7 688	13 379
# occurrences ($\|C\|$)	90 927	120 973	114 850	87 067	76 571	1 321 482
Documents						
#	1 587	1 398	425	1 460	1 033	7 466
avg. $\|d\|$	56.8	85.1	268.6	56.7	73.8	177.2
Queries						
#	64	225	83	112	30	50
avg. $\|q\|$	12.7	8.9	8.2	37.7	11.4	79.3

Table 7.1: Characteristics of the document collections used for evaluation.

We mostly use the standard Mean Average Precision (MAP) to display the results: we plot the MAP against the number of latent topics $|Z|$, averaged over all experiments and with error bars corresponding to 1-standard deviation. The conclusions are exactly the same using either 5-point precision or R-precision, except on MED; we come back to this latter point at the end of this section.

7.3 The IID-based kernel and Hofmann's kernel

Chapter 4 (p. 53) introduces the IID-based Fisher kernel K^{IID} and relates it to Hofmann's kernels with four assumptions (see section 4.3, p. 59). The only assumption likely to bear a significant impact on the performances of the kernel (like seen with the VS and DFIM-VS kernels) is the approximation of FIM by the identity. This leads us to predict that

- There will be a qualitative difference in behaviour between the IFIM and DFIM kernels; the DFIM are expected by behave better than the IFIM kernels;
- For a same approximation of FIM, there is no significant difference between the K^{IID} and K^{H} families — i.e., $K^{\text{IFIM-IID}}$ is expected to behave in a similar manner as $K^{\text{IFIM-H}}$, and $K^{\text{DFIM-IID}}$ is expected to behave in a similar manner as $K^{\text{DFIM-H}}$.

Figure 7.1 presents the performances of the $K^{\text{IFIM-IID}}$, $K^{\text{IFIM-H}}$, $K^{\text{DFIM-IID}}$ and $K^{\text{DFIM-H}}$ kernels, evaluated on five of the SMART bases for different numbers of topics. The performance is assessed in terms of Mean Average Precision (MAP).

As predicted, we see the two IFIM kernels and the two DFIM kernels exhibiting similar behaviours. This experimentally validates the idea that for a same approximation of FIM, Hofmann's kernel (with normalisation by document and query length) constitutes an approximation of the IID-based kernel: $K^{\text{IFIM-IID}} \simeq K^{\text{IFIM-H}}$, and $K^{\text{DFIM-IID}} \simeq K^{\text{DFIM-H}}$.

This result has interesting practical implications, since computation of a single document-query pair is more costly for IID-based kernels than for Hofmann's kernels — $O(|d| \cdot |q| \cdot |Z|)$ for K^{IID}, but only in $O(|q| \cdot |Z|)$ for K^{H}. Since the FIM terms in DFIM kernels can be pre-computed for all topics (for K_z) and for all word-topic pairs (for K_w), the DFIM approximation does not critically impact computation costs in comparison to the IFIM versions of the kernels.

Regarding the DFIM and IFIM approximations, we confirm previously published results [61] indicating superior performances of the DFIM kernels over the IFIM kernels.

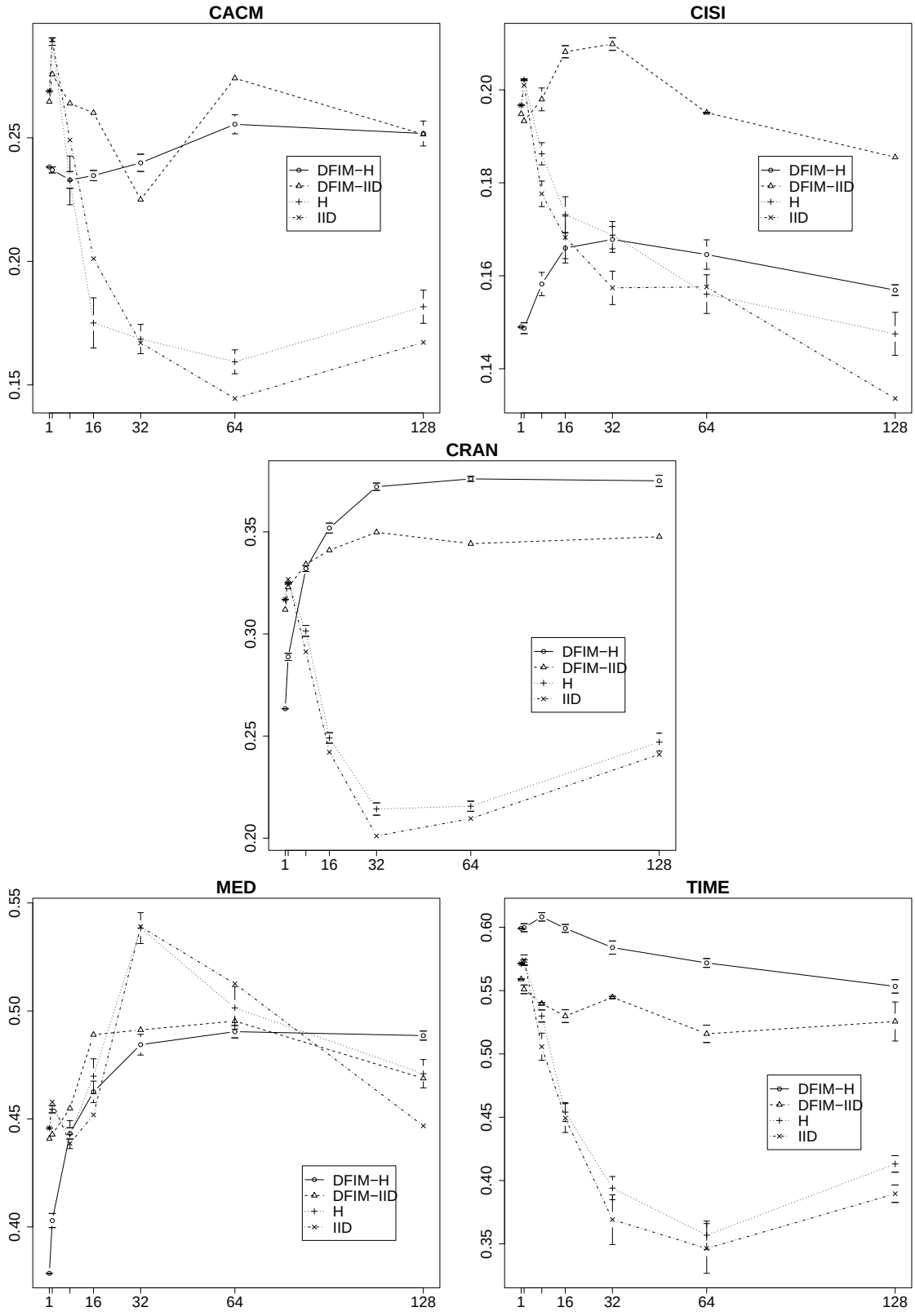


Figure 7.1: Behaviours (MAP vs $|Z|$) of the K^{IID} and K^{H} kernels, both for IFIM and DFIM variants. $K^{\text{IFIM-IID}}$ and $K^{\text{IFIM-H}}$ have very similar behaviours, as do $K^{\text{DFIM-IID}}$ and $K^{\text{DFIM-H}}$ to a slightly lesser extend. The DFIM kernels generally outperform the IFIM kernels. The IFIM kernels suffer from a markable drop in retrieval performance as $|Z|$ increases, while the DFIM kernels are much more stable to this respect.

7.4 K_z and K_w components of the kernels

The behaviour of the individual parts of the Fisher kernel, detailed in chapter 5 (p. 63), is worthy of specific examination. This study will explain the differences in behaviour between the different normalisations of the Fisher kernel: Hofmann's, where both K_z and K_w are normalised by $|d|$ and $|q|$; VS, where only the K_z is normalised; and the unnormalised kernel.

Figures 7.2 through 7.7 show results for Hofmann's kernel and the unnormalised kernel, both in IFIM and DFIM versions, as well as the K_z and K_w sub-parts of these 4 kernels. The VS kernel uses the K_z of the unnormalised kernel, and the K_w of Hofmann's kernel (see table 5.1, p. 64). These kernels and kernel parts are evaluated over the SMART collections, as well as over a subset of the AP collection from the TREC terabyte track.

These graphs all display three families of curves:

The K_z parts, which describe the contribution of the topics to the kernels, have a very low performance for low numbers of topics, and improve as the number of topics increases. After the number of topics has increased to a certain point, this performance reaches a plateau (evident on MED, figure 7.5). The maximum performance is nevertheless rather low. This behaviour can be observed consistently for all K_z parts, with IFIM or DFIM approximations of FIM, and with either normalisation (Hofmann's or unnormalised);

The K_w parts, which describe the contribution of the "words knowing topics" to the kernels, start with a reasonable performance, which may increase a bit until it reaches a plateau; this plateau is found quicker for K_w than it is found for K_z. Good illustrations of this behaviour can be seen on figure 7.4 or figure 7.5. On figure 7.7, there are some fluctuations of these components, but the overall behaviour still corresponds to this. Again, this behaviour can be observed consistently for all K_w parts, with IFIM or DFIM approximations of FIM, and with either normalisation (Hofmann's or unnormalised);

The complete kernels exhibit two strikingly contrasting behaviours: the DFIM kernels remain, overall consistant with the behaviour of the K_z parts; the IFIM kernels, on the other hand, start at low number of topics with performances comparable to those of the K_w parts, but collapse quickly until they reach a minimum, after which they rise slowly and converge with the performances of the K_z part.

The behaviour of the IFIM kernels brings a clue to experimentally understand the relative absolute values of the K_z and K_w parts of the kernel: from its results, it s clear that the IFIM kernels are dominated by their K_w part at a low number of topics, and by their K_z part at a high number of topics.

7.5 Language models

7.5.1 Log-likelihood and Kullback-Leibler divergence

We developed and tested two Language Model (LM) approaches (section 2.5.2, p. 27) for PLSI:

Maximisation of the query log-likelihood: as opposed to the usual approach, in the case of PLSI, we write the likelihood using the joint probability $P(d,w)$, in line with the fundamental assumptions of PLSI as an IID model generating (d,w) pairs (see section 3.3.1, p. 42), yielding $L(\widehat{q}) = \prod_{w\in V} P(d,w)^{n(q,w)}$. For a given query, we seek the document that maximises the log-likelihood with respect to the model $P(d,w)$:

$$\mathscr{S}_{\mathrm{LogL}}(d,q) = \sum_{w\in q\cap d} n(q,w)\log P(d,w), \tag{7.5.1}$$

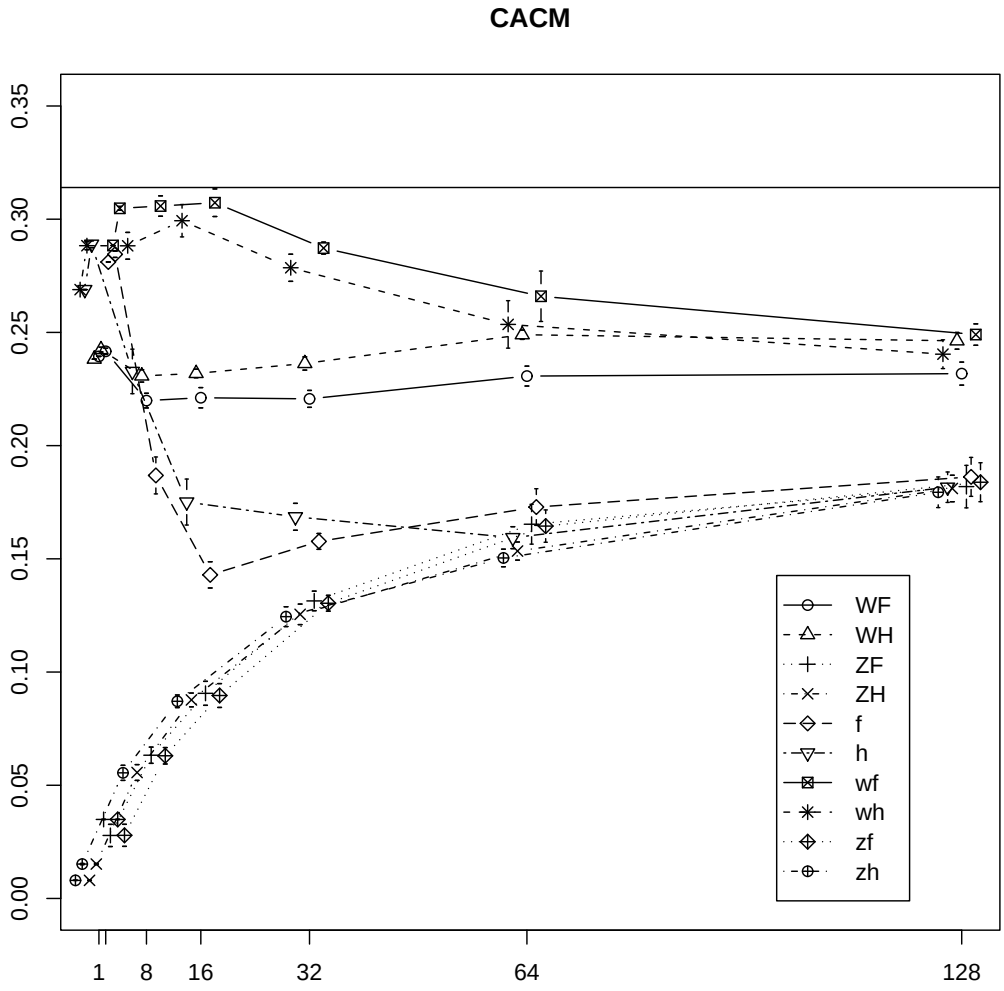


Figure 7.2: Relative performances of the subparts of the different variants of the Fisher kernel for PLSI (MAP vs $|Z|$), for the CACM collection; results for other collections are given in figures 7.3 through 7.7.
IFIM versions in minuscules, DFIM in capitals; "F" or "f" is the unnormalised version of the kernel, "H" or "h" is is the normalised version; the "W" or "w" prefix indicates that only the word part of the kernel is considered, and "Z" or "z" indicates the same for the topics part.
This illustrates that the latent-topic part K_z (ZF, ZH, zf and zh) performs poorly in comparison to the word part K_w (WF, WH, wf and wh), and impairs the combined kernels: notice how K^{H} (h) starts from K_w^{H} (wh) at low $|Z|$, and degrades to K_z^{H} (zh) as $|Z|$ grows.
The horizontal bar represents the MAP of state-of-the-art BM25 model (which does not depend on $|Z|$).

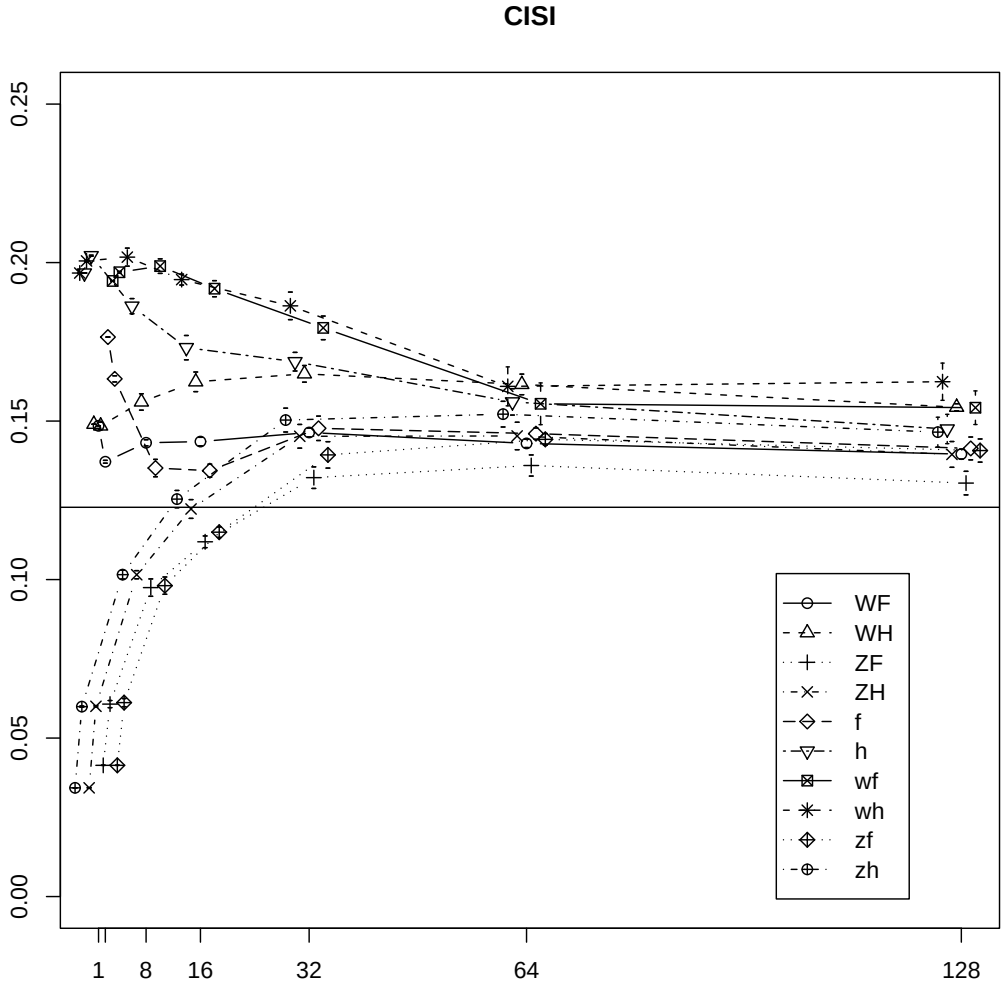


Figure 7.3: Same as figure 7.2 (p. 86), for the CISI collection.

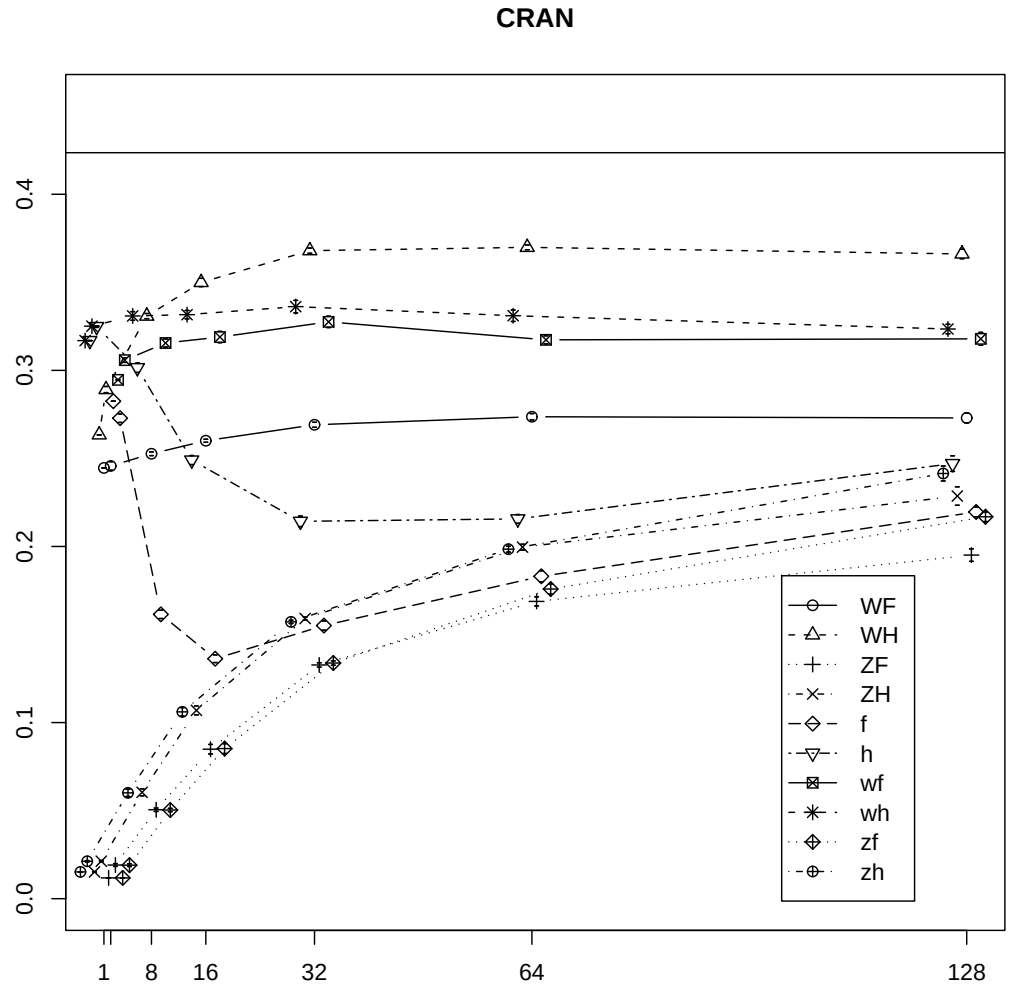


Figure 7.4: Same as figure 7.2 (p. 86), for the CRAN collection.

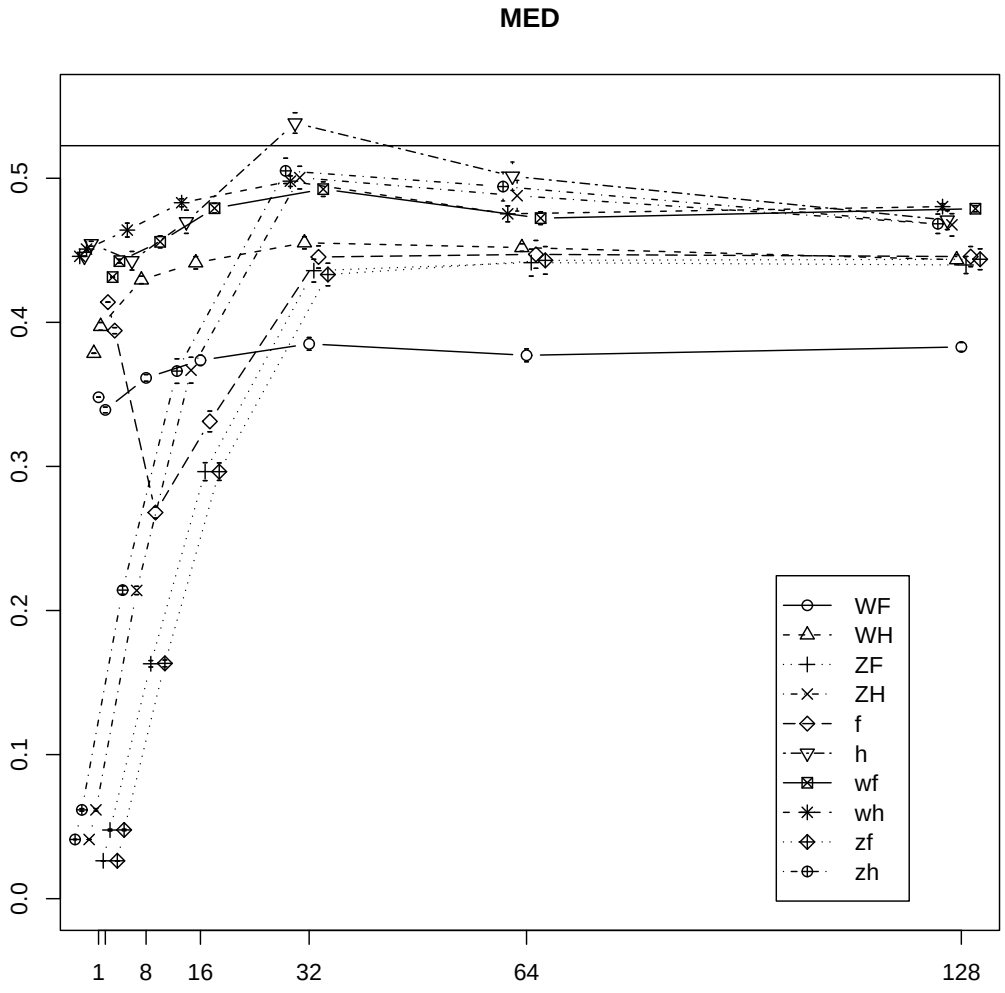


Figure 7.5: Same as figure 7.2 (p. 86), for the MED collection.

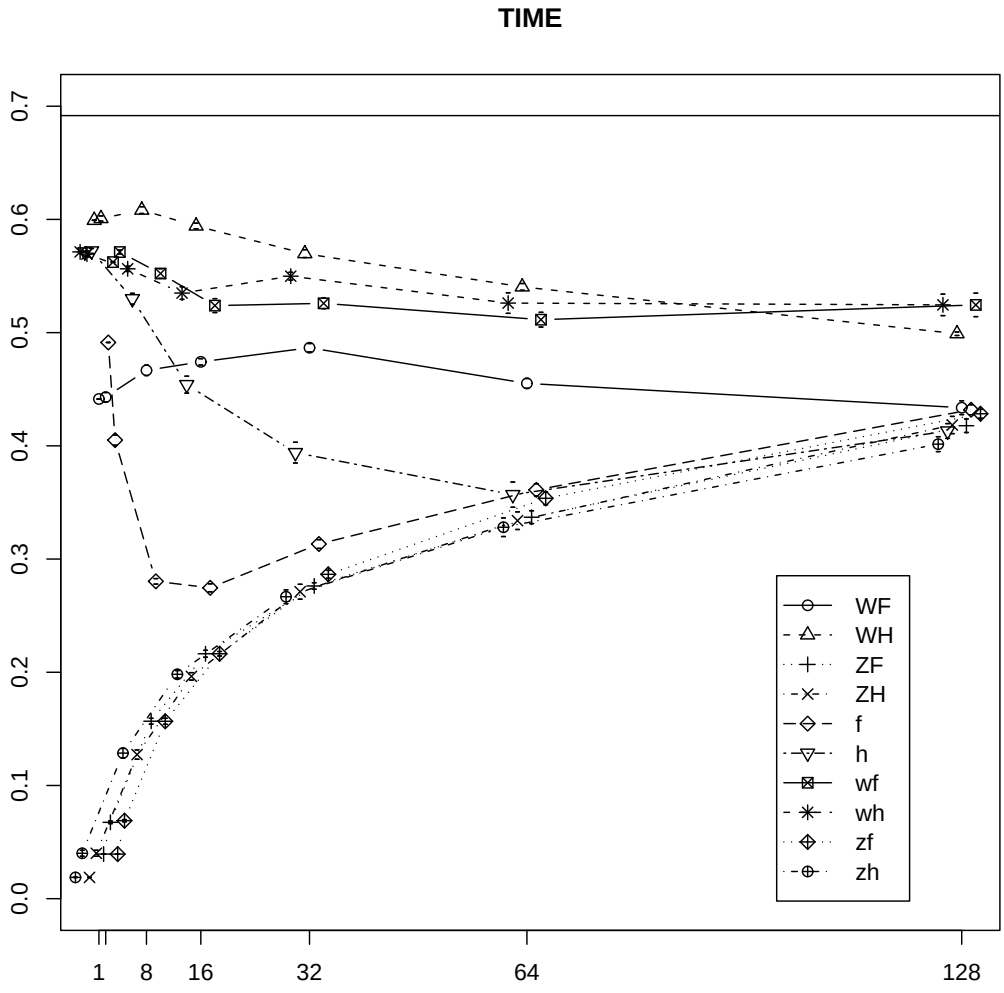


Figure 7.6: Same as figure 7.2 (p. 86), for the TIME collection.

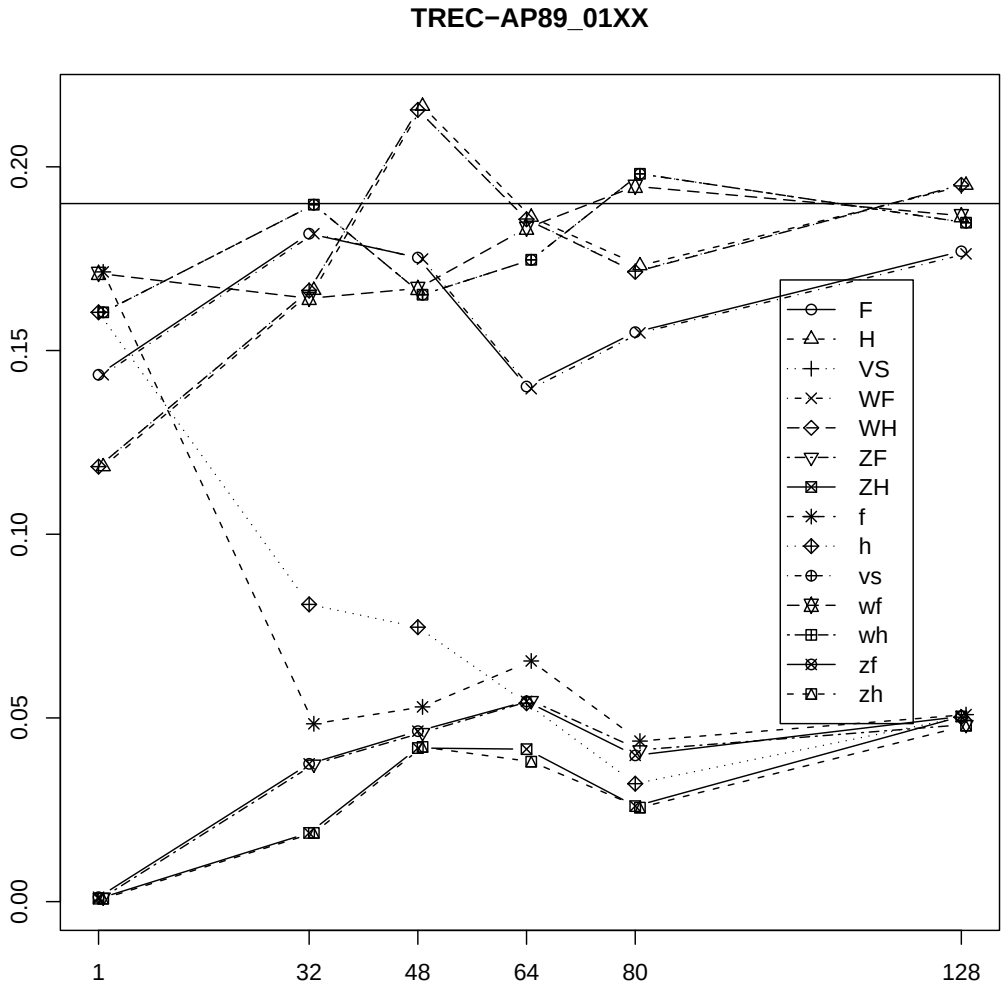


Figure 7.7: Same as figure 7.2 (p. 86), for the AP collection.

where $n(q,w)$ is the number of occurrences of word w in query q, and "$w \in q \cap d$" denotes all the words appearing in q (i.e. $n(q,w) > 0$) and such that $P(d,w) > 0$.

Minimisation of the Kullback-Leibler divergence, i.e. maximisation of

$$\begin{aligned}\mathscr{S}_{\mathrm{KL}}(d,q) &= -\mathrm{KL}\left(\widehat{P}(w|q), P(w|d)\right) \\ &= \sum_{w \in q \cap d} \widehat{P}(w|q) \log \frac{P(w|d)}{\widehat{P}(w|q)},\end{aligned} \tag{7.5.2}$$

where $\widehat{P}(w|q) = n(q,w)/|q|$, the number of occurrences of word w in query q divided by the length $|q|$ of query q.

The former straightforwards expressions of $\mathscr{S}_{\mathrm{KL}}$ and $\mathscr{S}_{\mathrm{LogL}}$ use the normalised raw term count $\widehat{P}(w|q) = n(w,q)/|q|$ as estimator for $P(w|q)$. However, any other estimator could be used in this place; in particular, smoother estimators could be tried [98]. We experimented with the two simplest smoothings: Jelinek-Mercer smoothing, and pseudo-feedback [11, 38].

Our experiments (Figures 7.11 and 7.12) show that in this case, both smoothings degrade the performances rather than improve them.

7.5.2 Relation between LogL and KL in the case of PLSI

Our log-likelihood-based and Kullback-Leibler-based similarities are related, as already mentioned by Lafferty and Zhai [37], but not exactly equivalent: in our case, the log-likelihood similarity is based on a joint probability (rather than a conditional probability), motivated by the PLSI model of generation of (w,d) pairs of indices. Thus:

$$\begin{aligned}\mathscr{S}_{\mathrm{KL}}(d,q) &= \sum_{w \in q \cap d} \widehat{P}(w|q) \log \frac{P(w|d)}{\widehat{P}(w|q)} \\ &= \sum_{w \in q \cap d} \frac{n(q,w)}{|q|} \left(\log \frac{P(d,w)}{P(d)} - \log \widehat{P}(w|q) \right) \\ &= \frac{1}{|q|} \left(\sum_w n(q,w) \log P(d,w) - |q| \log P(d) \right) - \sum_w \widehat{P}(w|q) \log \widehat{P}(w|q) \\ &= \frac{1}{|q|} \Bigg(\underbrace{\sum_w n(q,w) \log P(d,w)}_{=\mathscr{S}_{\mathrm{LogL}}(d,q)} - |q| \log P(d) \Bigg) - \underbrace{\sum_w \widehat{P}(w|q) \log \widehat{P}(w|q)}_{\text{indep. of } d} \\ &= \frac{1}{|q|} \Big(\mathscr{S}_{\mathrm{LogL}}(d,q) - |q| \log P(d) \Big) - f(q)\end{aligned}$$

When performing retrieval, that is maximising $\mathscr{S}(d,q)$ with respect to d for a given query q, the two approaches thus differ by an additive factor of $|q| \log P(d)$.

Using the conditional probability $P(w|d)$ instead of the joint probability would remove the $|q| \log P(d)$ term and make the two document similarities rank-equivalent[5]. As shown in figures 7.8 through 7.10, the similarity $\mathscr{S}_{\mathrm{KL}}$ based on $P(w|d)$ (Eq. 2.5.1) outperforms $\mathscr{S}_{\mathrm{LogL}}$ based on $P(d,w)$ (Eq. 7.5.1). Their

[5] Straightforwardly, $P(d,w) = P(w|d)P(d)$, and $\mathscr{S}_{\mathrm{LogL}}(d,q) = \sum_{w \in q \cap d} n(q,w) \log P(d|w) = \sum_{w \in q \cap d} n(q,w)[\log P(d,w) + \log P(d)]$.

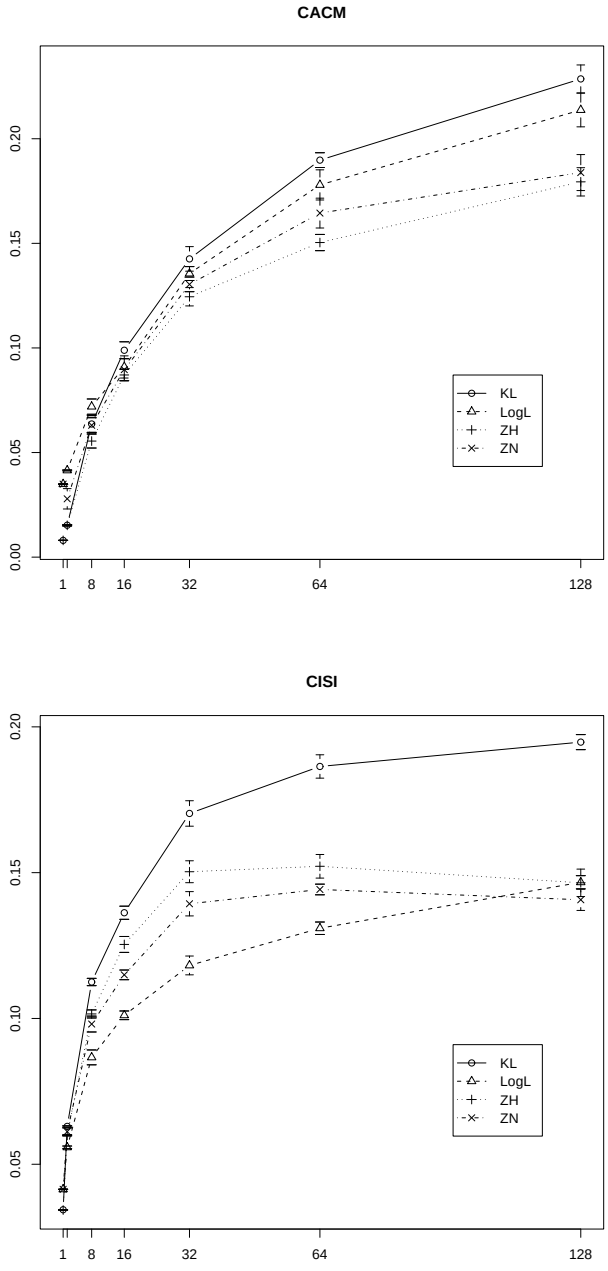


Figure 7.8: The similarity $\mathscr{S}_{\mathrm{KL}}$ based on $P(w|d)$ (Eq. 2.5.1, noted "KL" on the graph) outperforms $\mathscr{S}_{\mathrm{LogL}}$ based on $P(d,w)$ (Eq. 7.5.1, noted "LogL" on the graph). MAP averaged over 6 different runs and corresponding 1-standard deviation bars are displayed versus the number $|Z|$ of latent-topics for these two models. We also compare them with the latent-topic parts of Fisher kernels, K_z^{H} (ZH) and K_z^{N} (ZN). Results for CACM and CISI are provided here, and for other collections, in figures 7.9 and 7.10

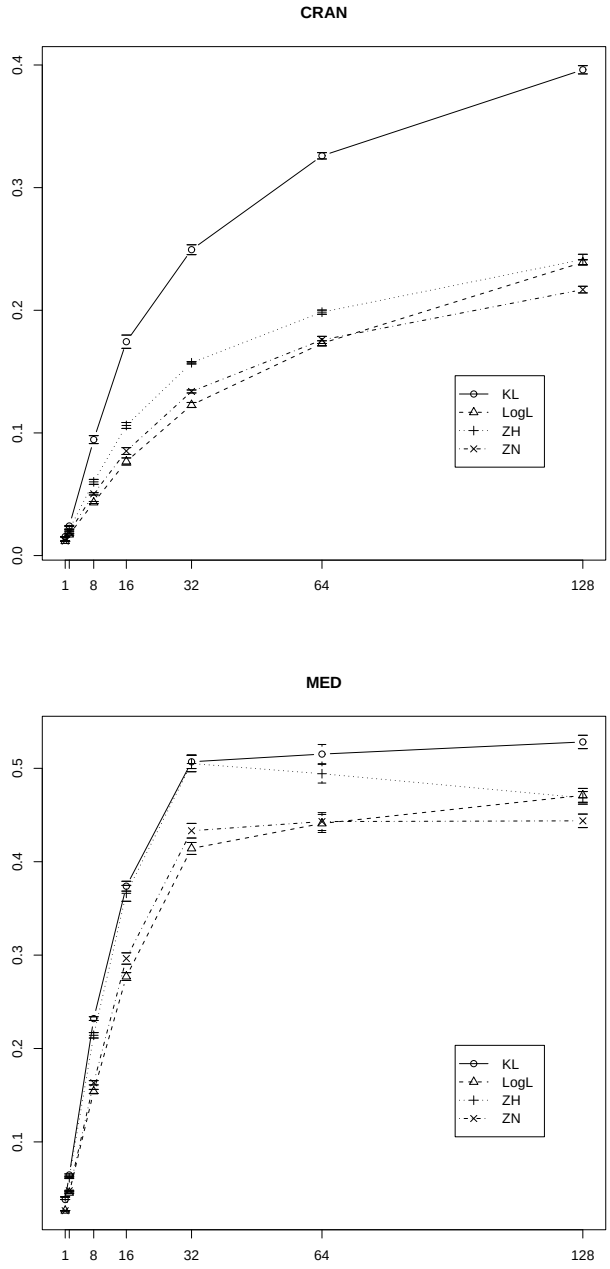


Figure 7.9: Same as figure 7.8 (p. 93), for CRAN and MED

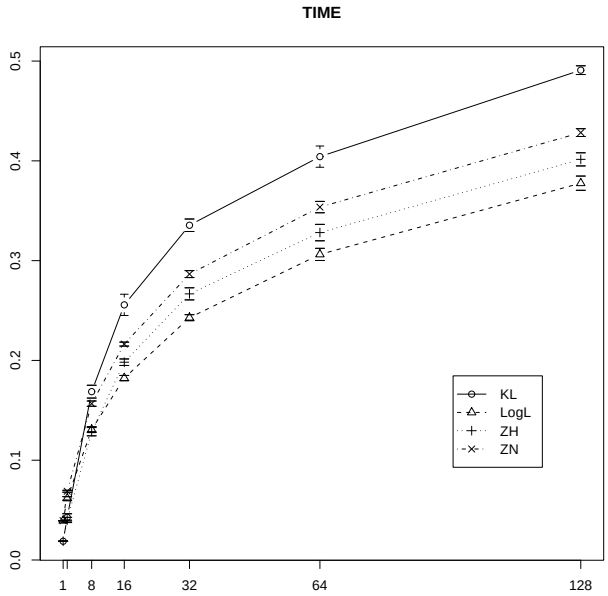


Figure 7.10: Same as figure 7.8 (p. 93), for TIME

general behaviour, low performance with few latent topics, rising asymptotically as the number of latent topics grow, is comparable to the behaviour of the latent-topic parts of Fisher kernels, K_z^{H} and K_z^{N}.

The joint probability of $\mathscr{S}_{\mathrm{LogL}}$ affects its ranking performances [34], making it perform poorly compared to $\mathscr{S}_{\mathrm{KL}}$ which is based on a conditional probability.

7.5.3 Language model similarities and Fisher kernels

We compare $\mathscr{S}_{\mathrm{KL}}$, the best of the language model similarities, to significant Fisher kernels: K^{H} (H), $K_w^{\text{DFIM-H}}$ (WDFIMH) and K_w^{H} (WH). As Figure 7.13 (p. 98) shows, at 128 categories, the performances of $\mathscr{S}_{\mathrm{KL}}$ often competes favourably with the best of the Fisher kernels. This illustrates the respective advantages and drawbacks of the Fisher kernels and the language model similarities: the best of the Fisher kernels, $K_w^{\text{DFIM-H}}$, exhibits stable performances with respect to the number of topics, but they require PLSI representation of the queries; the language model similarity $\mathscr{S}_{\mathrm{KL}}$ does not require a PLSI representation of the queries, but needs to large number of latent topics to perform well.

7.5.4 Conclusions on Language Models applied to PLSI

The parameters of PLSI can be used to implement similarities that stem from the language model paradigms. These similarities have simple forms and are quick to compute (unlike the IID-based Fisher kernel of chapter 4). We have discussed two straightforwards examples: query log-likelihood $\mathscr{S}_{\mathrm{LogL}}$ (based on $P(w,d)$), and Kullback-Leibler divergence $\mathscr{S}_{\mathrm{KL}}$. Of the two, $\mathscr{S}_{\mathrm{KL}}$ performs experimentally better (see figure 7.8, page 93); it is rank-equivalent with query log-likelihood based on $P(w|d)$.

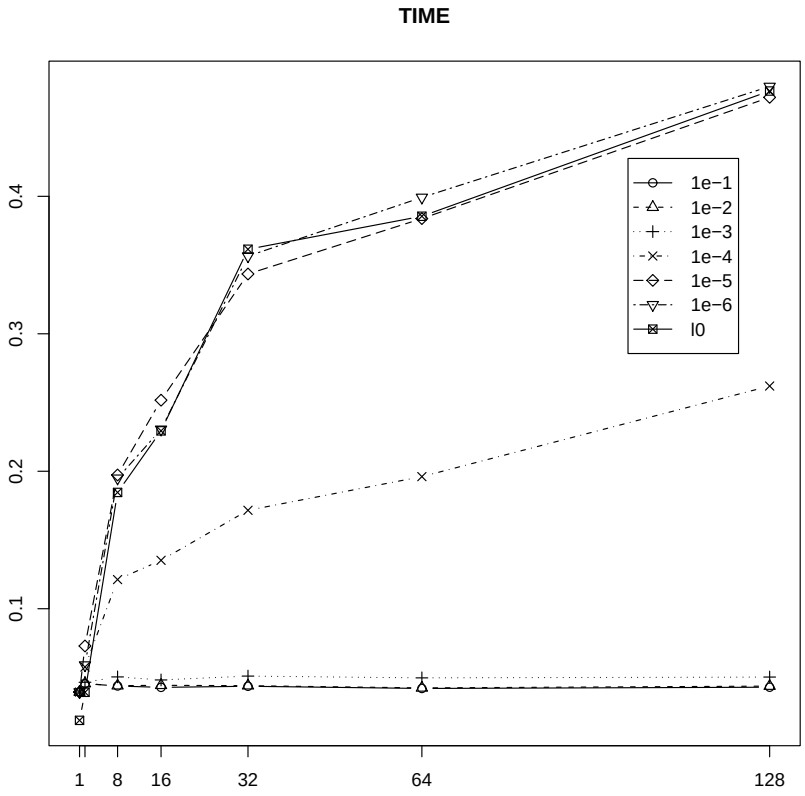


Figure 7.11: A typical example on TIME illustrating how Jelinek-Mercer-smoothed estimates for $P(q|w)$ degrade the performances of $\mathscr{S}_{\mathrm{KL}}$ and $\mathscr{S}_{\mathrm{LogL}}$ compared to raw estimate $\widehat{P}(q|w)$ ("l0"), for different values of λ.

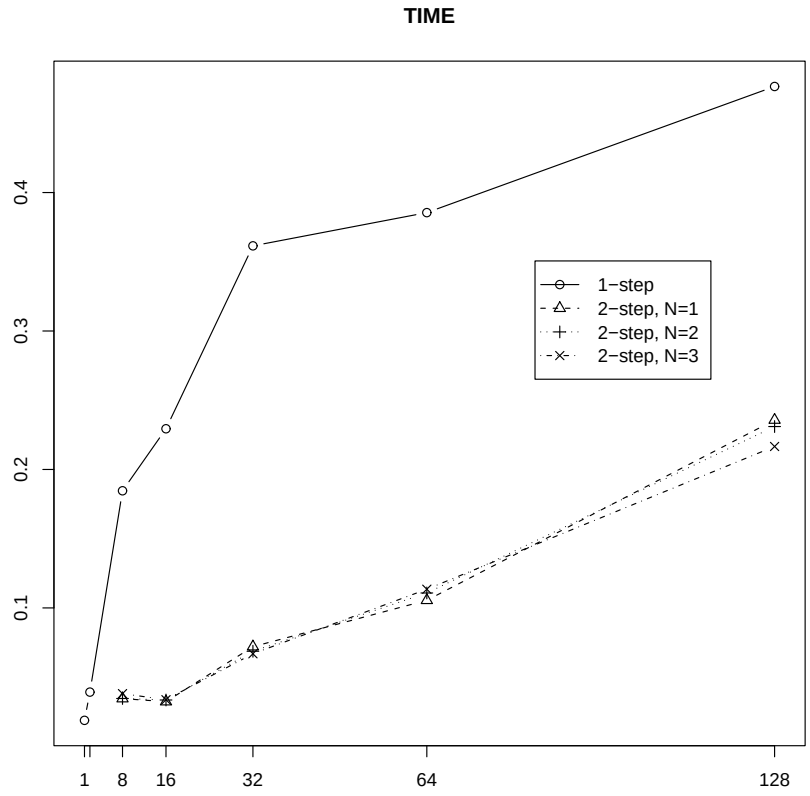


Figure 7.12: A typical example on TIME illustrating how pseudo-feedback retrieval (right) estimates for $P(q|w)$ degrade the performances of $\mathscr{S}_{\mathrm{KL}}$ and $\mathscr{S}_{\mathrm{LogL}}$, compared to raw estimate $\widehat{P}(q|w)$ ("1 step"), for different values of N.

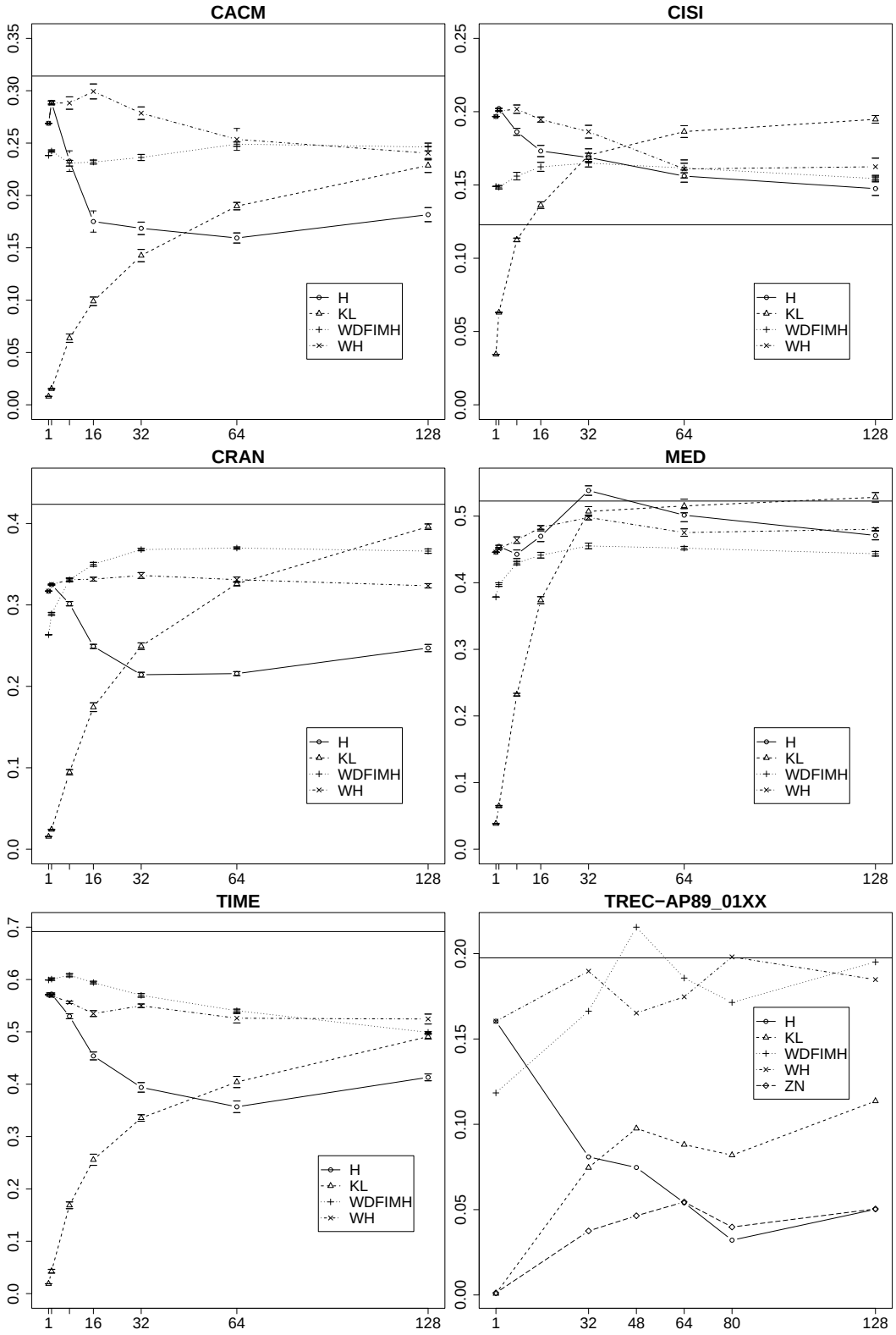


Figure 7.13: Results (MAP vs $|Z|$) obtained on the six corpora considered for different models: K^{H} (H), $\mathscr{S}_{\mathrm{KL}}$ (KL), $K_w^{\text{DFIM-H}}$ (WDFIMH), and K_w^{H} (WH). The performances of $K_z^{\text{DFIM-N}}$ (ZN) are also provided for the AP collection. The performances of $\mathscr{S}_{\mathrm{KL}}$ increase asymptotically as the number of latent topics grows. At 128 categories, $\mathscr{S}_{\mathrm{KL}}$ often competes favourably with the best of the Fisher kernels. The horizontal bar represents the MAP of state-of-the-art BM25 model (which does not depend on $|Z|$).

Noticeably, these similarities entirely bypass the problematic of query representation in the PLSI parameter space: in other words, they do not require $P(q|z)$ and thus dispence from query projection ("folding-in', mentioned in section 3.3.4, p. 51). Document similarities for PLSI based on language modeling are thus a theoretically sound answer to the question of folding-in.

7.6 Smoothed Dirichlet

We conducted experiments on the Smoothed Dirichlet distribution (see section 6.2, p. 72). As the Smoothed Dirichlet uses Jelinek-Mercer smoothing (section 2.5.2, p. 28), we produced results for different values of λ, where $\lambda = 0.1$ indicates reliance mostly on P^{GE} and $\lambda = 0.9$, mostly on $\tilde{P}(w|d)$. Figures 7.14 to 7.17 consistently indicate that a low value of λ produces the best results; only TIME, on figure 7.18, features the opposite tendency.

Comparison with TF-IDF and BM25 is favourable, with performances matching those of BM25 at a high precision; Smoothed Dirichlet even significantly outperforms BM25 on the difficult CISI collection (figure 7.15). TIME is the exception where Smoothed Dirichlet is outperformed by BM25 and struggles to equal the performances of TF-IDF (figure 7.18).

(The figures follow in the next pages)

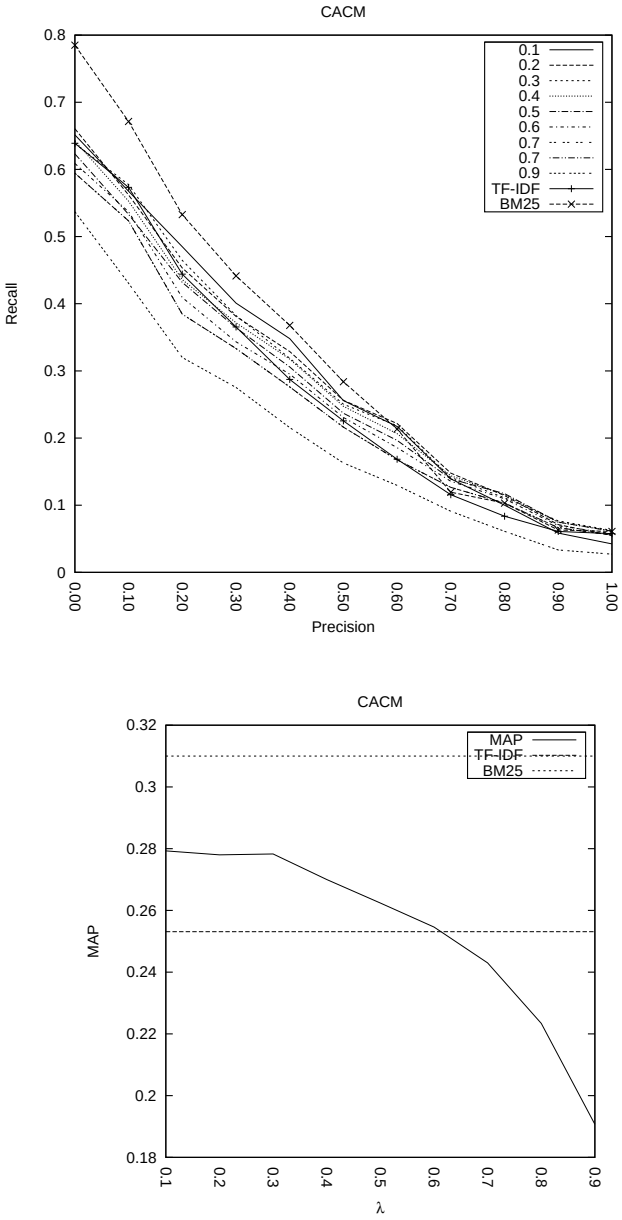


Figure 7.14: Top: precision-Recall curves for Smoothed Dirichlet at different values for λ. $\lambda = 0.1$ indicates reliance mostly on $P^{\mathbb{GE}}$ and $\lambda = 0.9$, mostly on $\tilde{P}(w|d)$. Bottom: MAP plotted against λ. The corresponding curves for TF-IDF and BM25 are given. Results for CACM here, results for the other collections in figures 7.15 through 7.18.

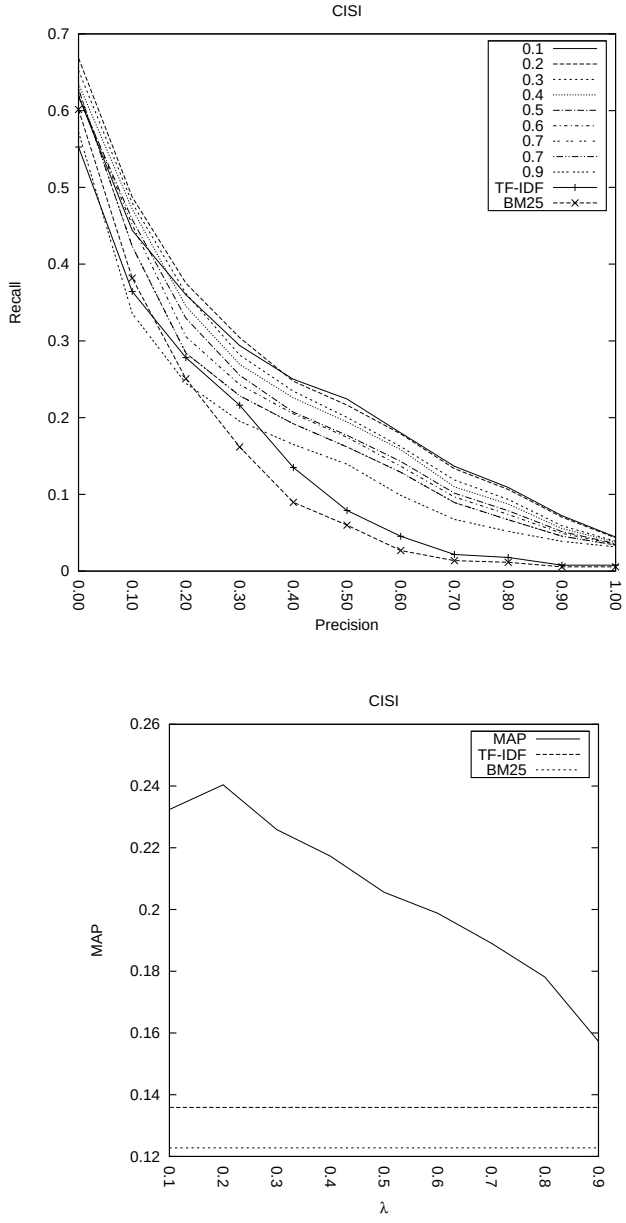


Figure 7.15: Same as figure 7.14 (p. 100), for CISI.

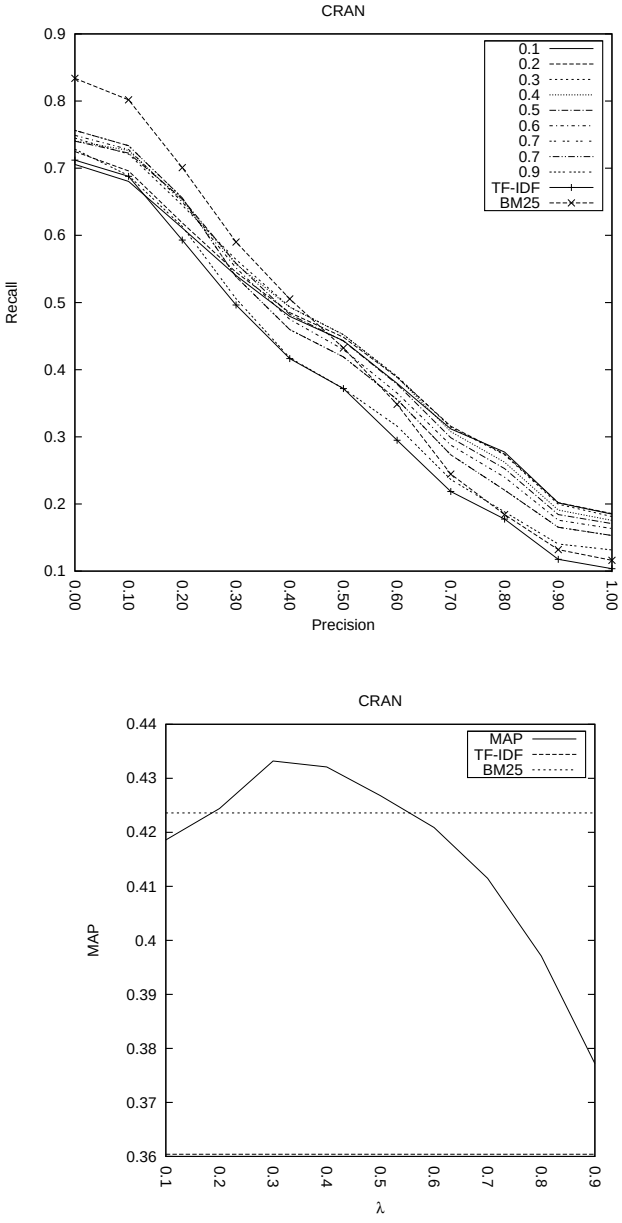


Figure 7.16: Same as figure 7.14 (p. 100), for CRAN.

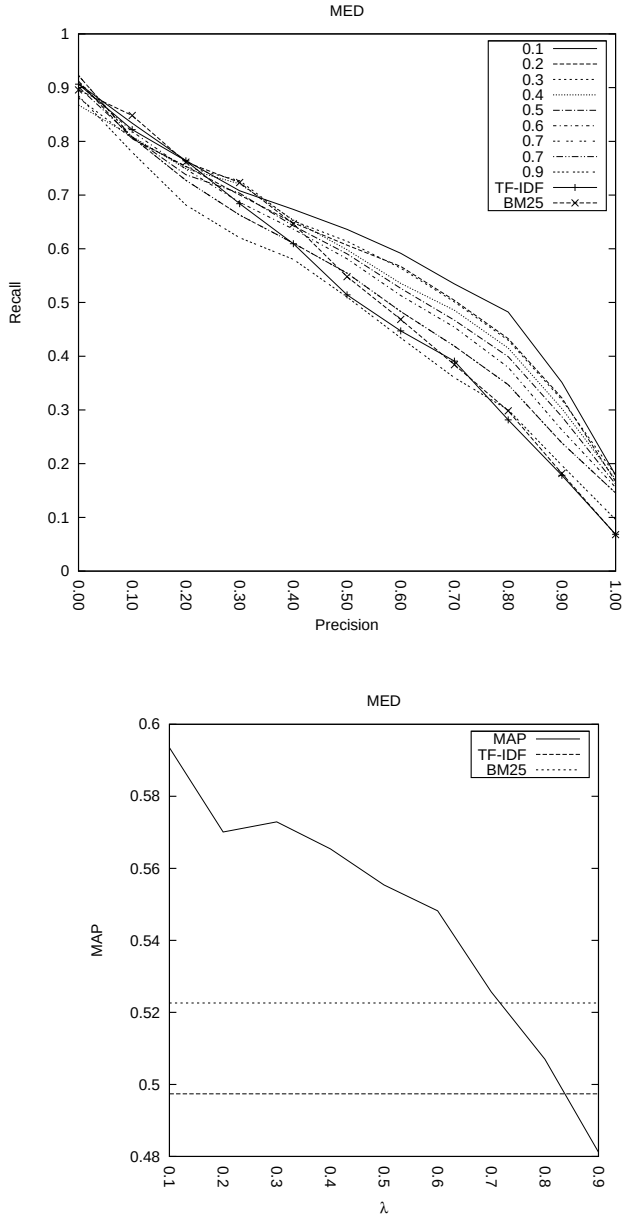


Figure 7.17: Same as figure 7.14 (p. 100), for MED.

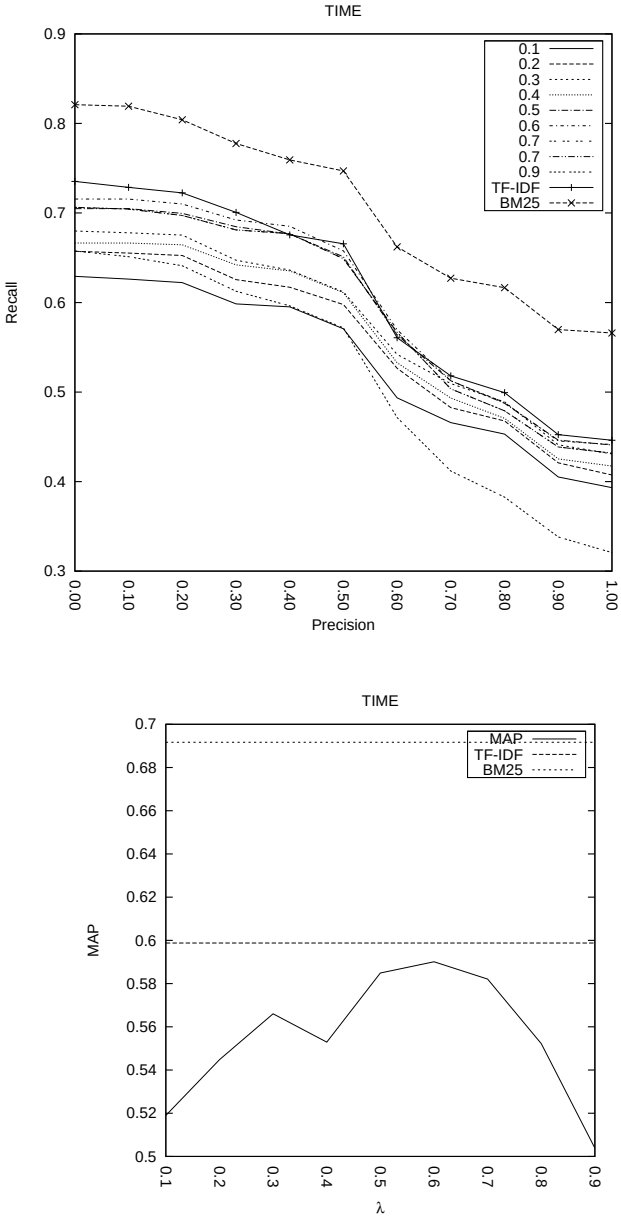


Figure 7.18: Same as figure 7.14 (p. 100), for TIME.

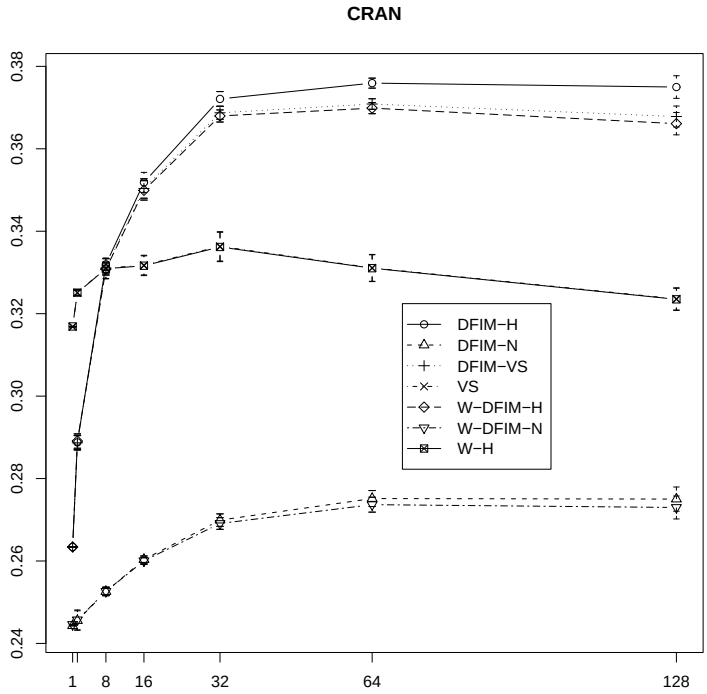


Figure 7.19: Typical example (MAP vs $|Z|$) illustrating that $K^{\text{DFIM-VS}}$(DFIM–VS) $\simeq$ $K_w^{\text{DFIM-H}}$(W–DFIM–H), $K^{\text{IFIM-VS}}$(VS) $\simeq$ $K_w^{\text{IFIM-H}}$(W–H), and $K^{\text{DFIM-N}}$(DFIM–N) $\simeq$ $K_w^{\text{DFIM-N}}$(W–DFIM–N).

7.7 Analysis

The main global results out of all these experiments, summarised in Table 7.2 (p. 108), are:

1. The kernels $K^{\text{IFIM-IID}}$ and $K^{\text{IFIM-H}}$ have very similar performances, which experimentally validates the theoretical result that $K^{\text{IFIM-H}}$ approximates $K^{\text{IFIM-IID}}$ (see Fig. 7.1). $K^{\text{DFIM-IID}}$ and $K^{\text{DFIM-H}}$ are a bit less similar, due to the slightly different form that $G(\theta)$ takes. However, the K^{IID} family is more computationally demanding than K^{H} — at least one order of magnitude slower.

2. As illustrated by Figure 7.19, $K^{\text{DFIM-VS}}$, $K_w^{\text{DFIM-H}}$, and K^{H} (resp. $K^{\text{DFIM-N}}$ and $K_w^{\text{DFIM-N}}$) have similar behaviours. The reason is that the normalising role of $G(\theta)$ makes $K_z \ll K_w$ for DFIM kernels. Moreover, $K^{\text{VS}} \simeq K_w^{\text{H}}$ since $\frac{|C|^2}{|d||q|} \gg 1$.

   The "VS" models [61], already not justified theoretically, are thus not worth considering, since the behaviours of $K_w^{\text{IFIM-VS}}$ and $K_w^{\text{DFIM-VS}}$ are akin to those of the $K_w^{\text{IFIM-H}}$ and $K_w^{\text{DFIM-H}}$ component, respectively.

3. Regarding the Fisher Information Matrix:

(a) We can confirm that the DFIM Fisher kernels for PLSI outperform by their original IFIM versions (Fig. 7.1). They are furthermore dominated by their K_w component.

(b) $K_w^{\text{DFIM-H}}$ and $K^{\text{DFIM-H}}$ have similar behaviours (Fig. 7.2 to 7.7). The reason is that the normalising role of $G(\theta)$ makes $K_z \ll K_w$ for DFIM kernels.

4. As illustrated by Figure 7.2 through 7.7 (pp. 86 through 91), K_z tends to deteriorate performances: used alone, it performs poorly; furthermore in K^{H} and K^{N}, as its role becomes more important for growing $|Z|$, the performances of these kernels decrease: starting from those of K_w at low $|Z|$, the performances of K^{H} and K^{N} reach down K_z at higher $|Z|$.

   On the other hand, K_w alone is always good, if not the best.

5. As illustrated by Figures 7.3 (p. 87) and 7.5 (p. 89), the best PLSI-based kernels can perform better than BM25 in some circumstances[6], especially on corpora which could be considered semantically more difficult: CISI, where few words are shared between queries and documents (thus particularly suited to test the extend to which a retrieval models is robust to synonymy, or the topics.)[7], MED (specialised vocabulary), and TREC–AP.

6. $\mathscr{S}_{\text{KL}}$ has the same type of behaviour as K_z (growing performance with $|Z|$), but performs notably better. It can even outperform the best Fisher kernel on CRAN, and reaches similar performances on MED and CISI. However it necessitates high numbers of latent categories to do so.

7. The Smoothed Dirichlet model $\mathscr{S}_{\text{SD}}$ (which is not a latent topics model) is performs well in comparison with BM25, particularly so on the difficult collections like CISI or MED (figures 7.14 and subsequent). The overall performance is better at low values for λ, indicating a strong dependence on P^{GE}.

The only collection where conclusions should be more nuanced is MED: the different models do not behave in the same way at different recall values (Fig. 7.20); some are better at low recall and others are better at higher recall. Global measures as MAP or R-Prec cannot represent such nuances.

7.8 Conclusions

These experiments show that the Fisher kernels that stem from our study of PLSI represent an improvement over the legacy kernel proposed by Hofmann. The experimental results confirm the adequacy of the IID kernel (chapter 4), as it displays superior performances; this validates the normalisation by document length, and accounting for the Fisher information matrix.

The similarity in behaviour between $K^{\text{DFIM-IID}}$ and $K^{\text{DFIM-H}}$ shows that the later is indeed an approximation of the first. We advise using $K^{\text{DFIM-H}}$ in practise as it is less computationally costly — or even $K_w^{\text{DFIM-H}}$ only, as K_z has poor performances and contributes little to the overall performances.

The language model approach is very dependant on the number of latent topics chosen in the model, in contrast to the best of the Fisher kernels that display quite stable performances. $\mathscr{S}_{\text{KL}}$ outperforms $\mathscr{S}_{\text{LogL}}$. For high numbers of latent topics, the performances of $\mathscr{S}_{\text{KL}}$ are on par with those of the best Fisher kernels.

[6] Although fine-tuning of the parameters of BM25 could negate the advantage to PLSI in these cases. We deliberately centre our study on "raw" models, without combining them or fine-tuning them to improve the results, and decide to leave techniques like boosting outside the scope of this work.

[7] CISI is remarkable in the sense that some query-document matches are expected between queries and documents that do not share any significant term (for instance query 1 with document 35).

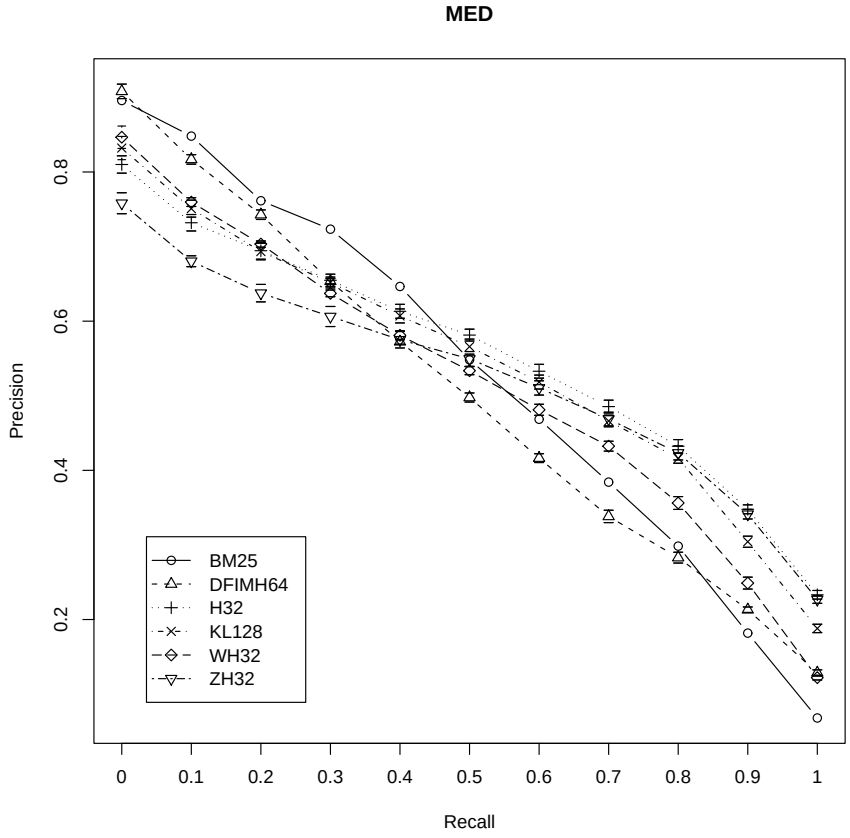


Figure 7.20: Precision vs Recall curves on MED for BM25, $K^{\text{DFIM-H}}$ for $|Z|$=64 (DFIMH64), K^{H} for $|Z|$=32 (H32), $\mathscr{S}_{\text{KL}}$ for $|Z|$=128 (KL128), K^{H}_{w} for $|Z|$=32 (WH32), and K^{H}_{z} for $|Z|$=32 (ZH32). We see that at a high recall, the Fisher kernels outperform BM25. Notice how the performances K^{H}_{z} fluctuate from being the worse of the models at low recall, to being the second best at high recall.

		CACM	CRAN	TIME	CISI	MED	AP89_01XX
Results	BM25 MAP	**31.4**	**42.4**	**69.2**	12.3	52.3	19.7
	Best PLSI-kernel MAP	30.7	39.6	60.8	**20.3**	**53.8**	**21.6**
	Best kernel is:	K_w^{N}	$\mathscr{S}_{KL}$	$K_w^{\text{DFIM-H}}$	K^{VS}	K^{H}	$K_w^{\text{DFIM-H}}$
	for $\lvert Z\rvert =$	16	128	8	8	32	48
	K_w^{H} MAP	**30.0**	33.6	55.6	**20.2**	**49.8**	16.5
	$K_w^{\text{DFIM-H}}$ MAP	23.2	37.0	**60.8**	15.6	45.5	**21.6**
	$\mathscr{S}_{\mathrm{KL}-128}$ MAP	22.9	**39.6**	49.1	19.5	**52.8**	11.4
Concl.	kernel PLSI > BM25?	No	No	No	**YES**	**yes**	**yes**
	K_z helps?	No	No	No	No	A bit	No
	DFIM $G(\theta)$ helps?	No	**Yes** (K_w)	**Yes** (K_w)	No	A bit	**Yes** (K_w)

Table 7.2: Main results and conclusions

The PLSI similarities ($K^{\text{DFIM-H}}$ and $\mathscr{S}_{\mathrm{KL}}$) can compete effectively with the strong baseline constituted by BM25. We show that for semantically difficult collections, such as CISI, the latent topics are an asset and allow PLSI document similarities to significantly outperform BM25.

The Smoothed Dirichlet distribution produces good results as a stand-alone IR system, outperforming BM25 on difficult collections.

Chapter 8

Perspectives and conclusions

Abstract

In this last chapter, we give a brief summary of the work performed all through the thesis and of the main results. We conclude by outlining trends for possible future work to further the advances.

8.1 Summary of the work

The framework of this thesis is the study of the way in which probabilistic latent semantics models can be used in Information Retrieval or Text classification, in which they can yield positive consequences by accounting for properties of a document collection underlying to the mere word occurrences and documents.

Probabilistic latent semantics models rely on assumptions about the process that generates the observed documents; from these hypothesis, given observable data, values for the model parameter can be inferred. Depending on the nature of the data and the validity of the assumptions of the model, inference provides underlying information on the document collection, the *latent topics*.

Document similarities are then used to match documents of similar content. These document similarities must be adapted to the nature of the model; the *Fisher kernel* features the necessary characteristics for this role.

One of the proeminent probabilistic latent semantic models is PLSI; we found that a number of questions relative to the document similarities for PLSI were still open, and decided to address them. We derive the Fisher kernel for PLSI, reviewing previously published discusions on Hofmann's original kernel, the role of the Information matrix therein, and issues about normalisation by document length. We expose Hofmann's original formulation to be linked to a more exact form of the kernel, a combination of "atomic" kernels on the elementary events that PLSI generates in an IID manner. Considering compositionality of Fisher kernels in an IID context, we combine these "atomic" kernels into a document kernel that accounts of the IID nature of the generative process. We discuss in details the four approximations that lead from this IID kernel to Hofmann's (table 4.1, p. 60). Doing so, we explain the results of the previous studies on PLSI Fisher kernels, validate the normalisation by document length, and vindicate the role of the Fisher information matrix within this kernel.

We also propose an alternative similarity measure based on language models, that exhibits competitive performances with the Fisher kernels. This application of language models also adverts the negative

consequences of the incompletely generative nature of the PLSI process: while PLSI provides assumptions on document generation that yield representations for known documents, these assumptions cannot be used to represent new documents or incoming queries. The language modeling approach provides a theoretically well-grounded document similarity that relies solely on statistic features of the query, entirely bypassing the issue of the PLSI representation of the query.

We complete our study of probabilistic latent semantic models by addressing the question of a latent model generating words in a realistic manner, by accounting for word burstiness.

In this perspective, we review Nallapati's Smoothed Dirichlet distribution. SD aims at providing a probabilistic distribution accounting for word generation in a realistic manner, and usable as a "module" in the framework of a wider probabilistic model. We attempt to present a model based on a mixtures of Smoothed Dirichlet distributions, the Latent-based Smoothed Dirichlet model. We provide the learning receipt and the Fisher kernel for this model. We were however unable not able to extract meaningful parameter values from the inference process.

8.2 Contributions

This thesis has contributed to several points.

- We derive the Fisher kernel for PLSI as a combination of elementary kernels in a way that reflects the IID nature of the PLSI generative process. This uncovers the rigorous form of the Fisher kernel, puts the previously existing Fisher kernels for PLSI into perspective, and explains previously published results.
- A language model approach to PLSI document similarities, providing a theoretically valid manner to perform IR based on PLSI parameters for documents and only statistic traits of queries. This bypasses the problem of "folding in" unknown documents or queries while remaining theoretically sound and providing competitive performances with Fisher kernels.
- A study of the contributions of the different parts of the Fisher kernel, as well as of the role of the Fisher information matrix. Together with the considerations on IID kernel, this part explains previously published results. It also shows why Hofmann's kernel complete with its Fisher information matrix, $K^{\text{DFIM-H}}$, is the best document similarity to deploy in practise.
- A proposition for a novel probabilistic latent semantics model, called Latent-based Smoothed Dirichlet, extending the Smoothed Dirichlet distribution to the realm of latent topics. This provides a fully generative latent topics model built on a strong model of word generation. We give the inference formulae and the Fisher kernel for the model.

8.3 Future work

A number of avenues remain to be explored and complete this work.

The study of document similarities for PLSI has been extensive, but this is not yet the case of the study of the Latent-based Smoothed Dirichlet (LSD). The discussion on this topic has been essentially theoretical, as the current implementation is not yet mature enough to produce experimental results. The present bottleneck is the inference process, which we have been unable to have produce non-degenerated values for the parameters.

It is possible that mating a smoothed model and a mixture model yield compatibility issues, as smoothing tends to bring points closer to the centre of the simplex, while the mixture models attempts to find well-contrasted points of the simplex around which documents would gather.

For large values of $S = \sum_{w \in V} \alpha_w$, the general silhouette of the Dirichlet distribution becomes close to that of the multinomial. The inference programme for Naive Bayes, which has been implemented and tested successfully, could in this perspective serve as a reference to tune the implementation of the LSD inference system.

The behaviour of the LSD model should then be studied with specific care for low values of S, as to benefit from the specificities of Dirichlet and answer the questions raised by the bounded behaviour of the approximation for the Γ function used by Nallapati.

Thanks to its fully generative nature, a working implementation of LSD would allow experimentation on large bases, like those of TREC: to this effect, a representative sample of the document collection should be extracted and used as a learning sample; the parameters learnt on this basis would then feed a straightforward re-indexing system running over the entire collection. This re-indexing could be incremental or run on a parallel architecture.

If experimental performances of the LSD model prove satisfying, then LSD would open interesting opportunities for further, more refined models: two foreseeable developments would entail replacing the simple mixture scheme of LSD with a latent Dirichlet allocation (similar to the ideas behind DCMLDA [16]), or possibly with a logistic distribution (similar to the ideas behind [6]) as to model the correlations between the latent topics.

Appendix A

Software developed for the thesis

Xapian[1] is a software framework for Information Retrieval. It does not natively provide PLSI, we have implemented a version of PLSI as part of a more general software layer. This version was compared against that of Lemur, and was found to yield more accurate results. 50 "documents" were generated from a given probability distribution, and the algorithms were applied to the result to see how they found the original distribution (see figure A.1).

One common technical problem of Information Retrieval engines is to hold and access to data. We choose to use Xapian, a performant retrieval library focusing on probabilistic retrieval.

Modern Information Retrieval does not limit itself at representing documents in a vector space via indexing. A transformation of the vector space can take place, often involving a reduction in dimensionality. Starting with the Latent Semantic Analysis (LSI) model, a number of models have been developed in the past 10 years, including Probabilistic Latent Semantic Analysis (PLSI), Naive Bayesian models, or Latent Dirichlet allocation.

We attempt at providing a general software framework to develop such models on the stable, fast and generally efficient basis of Xapian.

This document describes Topian ("Topic-based Model layer for Xapian"), a software layer intended to add support for topical models to Xapian.

A.1 Conception

Topian provides two trends to objects: objects directly linked to the documents, and abstract objects dealing with the model. Document representation is implemented in `DocsRep` objects which encapsulate Xapian objects like `Xapian::Database`. The `DocsRep` is what makes the link between the classical document representation, and features of the intended models held in `Parameters` objects. The `Parameters` are abstract objects which represent the model describing statistical features of the document topics and word topics. This architecture keeps the `Parameters` distinct and separated from the `DocsRep`, allowing for selective loading of the data when both direct data and model data are not needed for a given process.

For instance, the PLSI model is a statistical model based on the parameters $P(w|z)$ (probability that, given a certain topic z, work w occurs), $P(d|z)$ (probability that, given a certain topic z, document d occurs), and $P(z)$ (probability that a topic z occurs). On Topian, this translates into a `PLSARep` class inheriting from `DocsRep` data-wise, and a `PLSAParams` inheriting from `Parameters`, model-wise. The

[1] `http://xapian.org/`

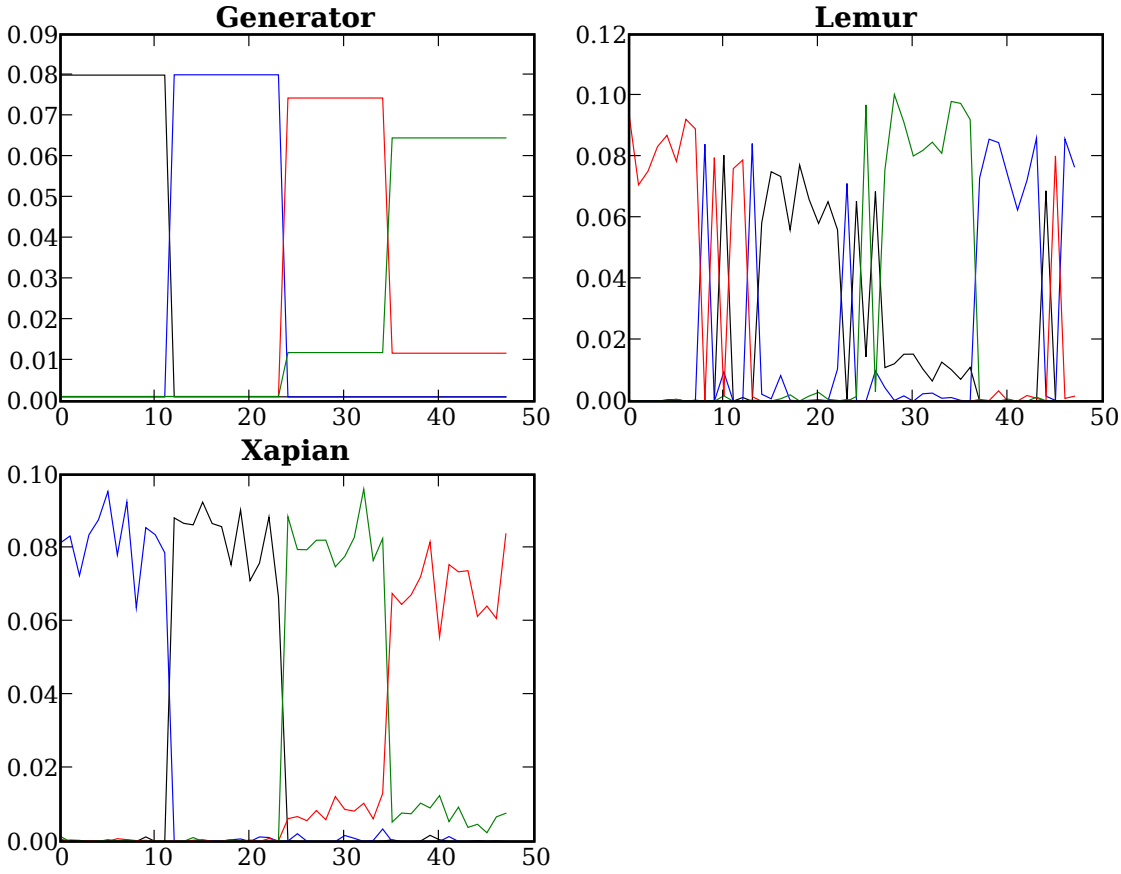


Figure A.1: Compared performances of the PLSA learning algorithm as implemented in Lemur, and as implemented by us on Xapian. The "true" distribution used to generate the data is also shown.

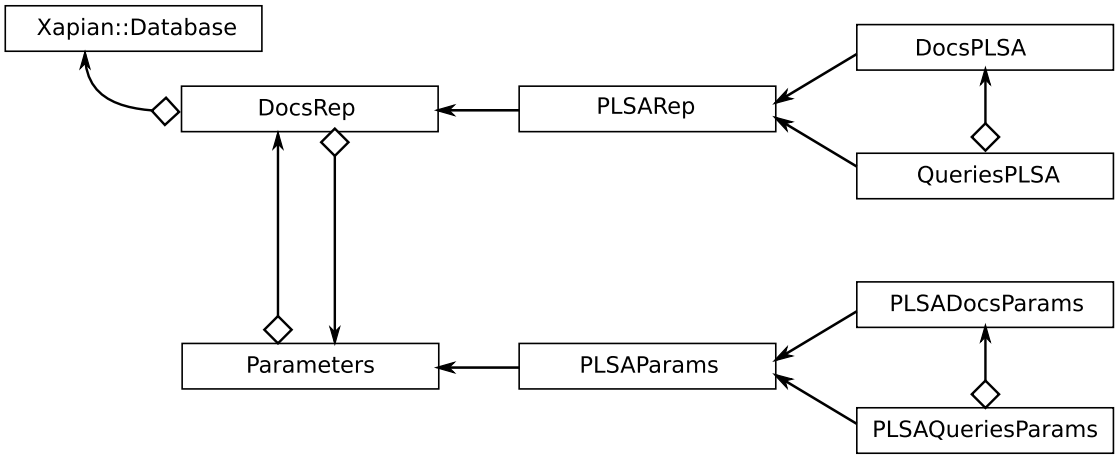


Figure A.2: Conception of the Topian layer for PLSI

`TREC::DocId`	foobar, foo, bar, foo2
`Xapian::DocId`	1, 2, 23, 42
`Topian::DocId`	0, 1, 2, 3

Figure A.3: Example of document identifiers

classes `PLSARep` and `Parameters` have further children to take specificities of queries versus collection documents into account.

A.2 Document identifiers

Document identifiers in NLP toolchains can take several types: `int`, `string`, etc. In the concrete case of using Xapian and Topian to process TREC data, we find ourselves with three flavours of identifiers. TREC identifiers are arbitrary strings; Xapian identifiers, typed `Xapian::DocId` are non-continuous integers starting from 1, `0` being reserved for out-of-collection documents; and Topian uses continuous integers starting with `0`. We will name these three types of identifiers `TREC::DocId`, `Xapian::DocId` and `Topian::DocId` (see figure A.3).

Xapian documents have a string container whose content can be accessed from `Xapian::Document::get_data()` member. The `Xapian::DocId` of a document can be retrieved either from the document itself with the `Xapian::Document::get_docid()` member, or by dereferencing an iterator over the `Xapian::Documents` with the `operator*() const`.

Over the course of the processing, we switch from the TREC identifiers to internal identifiers, and back again. This allows abstracting the TREC format if necessary (for instance when using the software on another format), and using more efficient algorithms that maps of strings.

Two standards containers of the `std` library could be used to map different types of docids: maps and vectors. Vectors map continuous integer indices to a type. They have an access time in $O(1)$. Maps allow using any type as indice; in particular, they allow using strings or non-continuous integers as indices. Their access time is in $O(\log n)$. Using an iterator, going through a whole container (map or vector) is $O(n)$. Hence it is advantageous to use vectors for punctual accesses, but performance is not

lost if a whole map is processed linearly.

We propose to implement the translations between the formats as follow:

- Xapian to Topian: the corresponding `Topian::DocId` is stored in a `std::map` contained in `DocsRep`. This data *must not* be contained in the `data` attribute of the `Xapian::Documents` because we need it to contain the `TREC::DocId`, which is set there at indexing time. Furthermore, `data` is of the `std::string` type, and it is unclean to set Topian things in the Xapian database.
- Topian to Xapian: a `std::vector` of `Xapian::DocId` is created.

`Topian::DocID`	0	1	2	3	...
`Xapian::DocID`	1	2	23	42	...

  Access being is $O(n)$, this structure can be used for punctual access (by opposition to skimming through the whole collection)
- Topian to TREC: a `std::vector` of `TREC::DocId` is created. This vector can simply be constructed at indexing time, since the `Xapian::DocId` indexing scheme is trivial.

`Topian::DocID`	0	1	2	3	...
`TREC::DocID`	foobar	foo	bar	foo2	...

  This makes it possible to access the `TREC::DocId` of any document in $O(1)$.

It is straightforwards that Xapian to TREC is obtained through Topian, also in $O(1)$. With the orientation of the data flow, it is not necessary to translate `TREC::DocId` into the internal indices `Xapian::DocId` and `Topian::DocId` (which should stay transparent TREC-wise anyway). Hence, we have a whole set of translators.

A.3 User's manuals

The user's manuals for the tools developed during the thesis are provided here, detailing the arguments and options that can be used.

A.3.1 bayespian-learn

`bayespian-learn` is a machine learning programme for Naive Bayes (mixture of multinomials) inference on `Xapian` databases. Given the name of a `Xapian` database, it performs Expectation-Maximisation evaluation of the $P(z)$ and $P(w|z)$ parameters; these parameters are written in two text files, `Pz.data` and `PwKz.data`, at the end of the process.

By default, `bayespian-learn` performs EM with 32 categories, from a random seed. It stops after convergence or after a fixed number of iteration have been performed.

Usage: `bayespian-learn [OPTION...] input_database`
Example: `bayespian-learn cacm.base.xapian`

`-c`, `--categories=CATS` : fixes the number of categories.

`-d`, `--debug` : Outputs values of P(z), P(d|z) and P(w|z) at every iteration.

`-e`, `--eta=ETA` : fixes eta.

`-i, --iterations=ITER` : fixes the maximum number of iterations. Set to `0` to allow unlimited number of iterations. Default is 128.

`-m, --displayMemoryUsage` : Reports on memory allocation during initialisation.

`-r, --random seed=SEED` : fixes the seed for randomisation. Default is timer.

`-s, --silent` : runs silently.

`-t, --tracking` : Outputs values of Beta, error and number of iteration while learning

`-v, --verbose` : tells you more.

`-?, --help` : Give a help list

`--usage` Give a short usage message

`-V, --version` : Print programme version

A.3.2 bayespian-compare

`bayespian-compare` performs Information Retrieval on document collections using Fisher Kernels on Naive Bayes (mixture of multinomials) latent topics representation spaces. Given two `Xapian` databases, it will use the first one as a document collection and the second one as a query collection, iteratively performing IR for each query over all the documents. At the end of the process, a standard TREC topic file is produced (by default to `output.top`).

Usage: `bayespian-compare [OPTION...] documents queries`

Example: `bayespian-compare cacm.base.xapian cacm.queries.xapian`

`-f, --Fisher` : utilises the Fisher kernel comparison method (original version).

`-g, --GFisher` : utilises the Fisher kernel comparison method (modified version).

`-h, --Hofmann` : utilises the Hofmann kernel.

`-i, --IID` : utilises the IID kernel.

`-I, --DFIM-IID` : utilises the IID kernel.

`-j, --Fisherbis` : utilises the Fisher kernel comparison method (original version).

`-s, --sampleSize=SAMPLE` : set sample size for PLS-IR to SAMPLE.

`-v, --verbose` : tells you more.

`-?, --help` : Give this help list

`--usage` : Give a short usage message

`-V, --version` : Print programme version

A.3.3 displayPLSAcats

`displayPLSAcats` is a visualisation tool that allows quick and intuitive review of PLSI parameters produced by `plsapian-learn` on a document collection.

Usage: `displayPLSAcats [OPTION...] documents`
Examples: `testPLSA cacm.base.xapian cacm.query.xapian`

`-c, --categories=NUMBER` : display the NUMBER most probable categories

`-o, --output=FILE` : sets output to FILE.

`-v, --verbose` : tells you more.

`-w, --words=NUMBER` : display the NUMBER most probable words for each displayed category

`-?, --help` : Give this help list

`--usage` : Give a short usage message

`-V, --version` : Print programme version

A.3.4 plsapian-learn

`plsapian-learn` is a machine learning programme for Probabilistic Latent Semantic Indexing (PLSI) inference on `Xapian` databases. Given the name of a `Xapian` database, it performs Expectation-Maximisation evaluation of the $P(z)$, $P(d|z)$ and $P(w|z)$ parameters; these parameters are written in two text files, `Pz.data` `PdKz.data`, and `PwKz.data`, at the end of the process.

By default, `plsapian-learn` performs EM with 32 categories, from a random seed. It stops after convergence or after a fixed number of iteration have been performed.

Usage: `plsapian-learn [OPTION...] input_database`
Examples: `plsapian-learn cacm.base.xapian`

`-c, --categories=CATS` : fixes the number of categories.

`-d, --debug` : Outputs values of P(z), P(d|z) and P(w|z) at every iteration.

`-e, --eta=ETA` : fixes eta.

`-i, --iterations=ITER` : fixes the maximum number of iterations. Set to `0` to allow unlimited number of iterations. Default is 128.

`-m, --displayMemoryUsage` : Reports on memory allocation during initialisation.

`-r, --random seed=SEED` : fixes the seed for randomisation. Default is timer.

`-s, --silent` : runs silently.

`-t, --tracking` : Outputs values of Beta, error and number of current iteration while learning

`-v, --verbose` : tells you more.

`-?, --help` : Give this help list

`--usage` : Give a short usage message

`-V, --version` : Print programme version

A.3.5 plsapian-split

`plsapian-split` splits a Xapian document collection on which PLSI learning has been performed using `plsapian-learn` into two document collections. The daughter collections have the same $P(z)$ and $P(w|z)$ parameters than the mother collection, and the $P(d|z)$ parameters are dispatched accordingly. This is useful to dispense with "folding in" parameters of the queries, by transforming a text clustering collection into a set of collections fit for Information Retrieval.

Usage: `plsapian-split [OPTION...] NUMBER input_database`

Examples: `plsapian-split 64 cacm.base.xapian`: splits `cacm.xapian` into `cacm.xapian.base` and `cacm.xapian.queries`, `with cacm.xapian.queries` containing the 64 last documents of `cacm.xapian`.

`-s`, `--silent` : runs silently.

`-v`, `--verbose` : tells you more.

`-?`, `--help` : Give this help list

`--usage` : Give a short usage message

`-V`, `--version` : Print programme version

A.3.6 plsapian-compare

`plsapian-compare` performs Information Retrieval on document collections using Fisher Kernels on PLSI latent topics representation spaces. Given two `Xapian` databases, it will use the first one as a document collection and the second one as a query collection, iteratively performing IR for each query over all the documents. At the end of the process, a standard TREC topic file is produced (by default to `output.top`).

Usage: `plsapian-compare [OPTION...] documents queries`

Examples: `plsapian-compare cacm.base.xapian cacm.queries.xapian`

`-a`, `--HzOnly` : use the $K_z^{\text{IFIM-H}}$ kernel only

`-A`, `--DFIMHzOnly` : use the $K_z^{\text{DFIM-H}}$ kernel only

`-b`, `--baseline` : baseline (random answers)

`-f`, `--Fisher` : utilises the unnormalised $K^{\text{IFIM-U}}$ kernel.

`-F`, `--DFIMF` : utilises the unnormalised $K^{\text{DFIM-U}}$ kernel.

`-g`, `--GFisher` : utilises the Fisher kernel comparison method (modified version).

`-G`, `--DFIMG` : utilises the DFIMG kernel ($= K_z^{\text{F}} + |C| * K_w^{\text{F}}$).

`-h`, `--Hofmann` : utilises Hofmann's kernel $K^{\text{IFIM-H}}$.

`-H`, `--DFIMH` : utilises the DFIMH kernel $K^{\text{DFIM-H}}$.

`-i`, `--iid` : use the $K_z^{\text{IFIM-IID}}$ kernel

`-I`, `--DFIMiid` : use the $K_z^{\text{DFIM-IID}}$ kernel with FIM normalisation

`-j`, `--Fisherbis` : utilises the $K^{\text{IFIM-U}}$ Fisher kernel (different approximations of term probabilities).

`-l`, `--lamdba=LAMBDA` : Smoothing factor. Default is `0.0` (not smoothed)

`-m`, `--Hofmannbis` : utilises the Hofmann kernel $K^{\text{IFIM-H}}$ (bis).

`-M`, `--DFIMHbis` : utilises the $K^{\text{DFIM-H}}$ kernel (bis).

`-o`, `--output=FILE` : sets output to FILE.

`-q`, `--plsaQ` : utilises the PLSA-Q comparison method (cosine-based).

`-r`, `--plsaR` : utilises the Language Model comparison method.

`-s`, `--sampleSize=SAMPLE` : set sample size for pseudo-feedback to SAMPLE.

`-t`, `--lowerbound=TRIGGER` : in DFIM kernels, sets lower bound for k_w to TRIGGER.

`-T`, `--upperbound=TRIGGER` : in DFIM kernels, sets upper bound for k_w to TRIGGER.

`-u`, `--plsaU` : utilises the PLSA-U comparison method (cosine-based).

`-v`, `--verbose` : tells you more.

`-w`, `--VS` : utilises the Vector Space kernel $K_z^{\text{IFIM-VS}}$.

`-W`, `--DFIMVS` : utilises the $K_z^{\text{DFIM-VS}}$ kernel ($= K_z^{\text{F}} + K_w^{\text{H}}$).

`-x`, `--HwOnly` : use the $K_w^{\text{IFIM-H}}$ kernel only

`-X`, `--DFIMHwOnly` : use the $K_w^{\text{DFIM-H}}$ kernel only

`-y`, `--FwOnly` : use the $K_w^{\text{IFIM-F}}$ kernel only

`-Y`, `--DFIMFwOnly` : use the $K_w^{\text{DFIM-F}}$ kernel only

`-z`, `--FzOnly` : use the $K_z^{\text{IFIM-F}}$ kernel only

`-Z`, `--DFIMFzOnly` : use the $K_z^{\text{DFIM-F}}$ kernel only

`-?`, `--help` : Give this help list

`--usage` : Give a short usage message

`-V`, `--version` : Print programme version

A.3.7 plsapian-info

`plsapian-info` displays information on the latent-based features of a document collection in which PLSI has been performed using `plsapian-learn`. This may be used for debugging purposes or for intuitive illustrations of the effects of PLSI. Since the parameter files generated by `plsapian-learn` are in plain text format, the functionalities of `plsapian-info` can be obtained from `awk`, `grep` and similar tools.

Usage: `plsapian-info [OPTION...] documents queries`
Examples: `plsapian-info cacm.base.xapian`

`-d`, `--docs` : displays $P(d) = \sum_{z \in Z} P(z)P(d|z)$.

`-D`, `--docsKz` : displays $P(d|z)$ and $\sum_{w \in V} n(d,w)P(w|z)$.

`-w`, `--words` : displays $P(w) = \sum_{z \in Z} P(z)P(w|z)$.

`-?`, `--help` : Give this help list

`--usage` : Give a short usage message

`-V`, `--version` : Print programme version

A.3.8 sdpian-compare

`sdpian-compare` performs Information Retrieval on document collections using the Smoothed Dirichlet similarity. Given two `Xapian` databases, it will use the first one as a document collection and the second one as a query collection, iteratively performing IR for each query over all the documents. At the end of the process, a standard TREC topic file is produced (by default to `output.top`).

Usage: `sdpian-compare [OPTION...] documents queries`
Examples: `sdpian-compare cacm.base.xapian cacm.queries.xapian`

`-b`, `--baseline` : baseline (random answers)

`-l`, `--lambda=LAMBDA` : sets lambda to given value. Default is 0.9

`-o`, `--output=FILE` : sets output to FILE.

`-s`, `--samplesize=LAMBDA` : sets sample size to given value. Default is 5

`-v`, `--verbose` : tells you more.

`-?`, `--help` : Give this help list

`--usage` : Give a short usage message

`-V`, `--version` : Print programme version

A.3.9 trecindex

`trecindex` is an indexer that indexes TREC files into the Xapian format.

Usage: `trecindex [OPTION...] output_database TREC_corpus_input_file stoplist`
Examples: `trecindex cacm.base.xapian cacm.base.trec --stoplist=english.stopword`

`-e`, `--empty` : indexes documents with empty text field.

`-l`, `--language=LANGUAGE` : sets language to LANGUAGE for stemming. Default is English (en). See below for details.

`-m`, `--min_word_size=SIZE` : do not index words smaller than SIZE characters. Default is 2. Set to 0 to skip this.

`-n`, `--min_occurrences=NUMBER` : do not index words occurring less than NUMBER times. Default is 2.

`-q`, `--quiet` : runs silently.

`-s`, `--stoplist=FILE` : read stoplist in FILE

`-S`, `--stems` : use only stems for indexing

`-t`, `--titleweight=WEIGHT` : Weights occurrences in title with WEIGHT. Default is 1.

`-v`, `--verbose` : tells you more.

`-?`, `--help` : Give this help list

`--usage` : Give a short usage message

`-V`, `--version` : Print programme version

The stemmer uses the snowball algorithm. The language can be specified either by its name in English, or by the two-letter ISO639 code. Supported languages are:

none - don't stem terms;

danish (`da`);

dutch (`nl`);

english (`en`) - Martin Porter's 2002 revision of his stemmer;

english_lovins (`lovins`) - Lovin's stemmer;

english_porter (`porter`) - Porter's stemmer as described in his 1980 paper [85];

finnish (`fi`);

french (`fr`);

german (`de`);

italian (`it`);

norwegian (`no`);

portuguese (`pt`);

russian (`ru`);

spanish (`es`)

swedish (`sv`).

A.3.10 xapiandbinfo

`xapiandbinfo` displays statistical information on Xapian-indexed document collections. This programme can also be used to examine document collections after Naive Bayes learning or PLSI learning has been performed.

Usage: `xapiandbinfo [OPTION...] <path to database>` xapiandbinfo, a tool for dumping Xapian Database directories.

See also `delve(1)`, the Xapian tool that provides some similar functionalities.

`-d`, `--doc=DOC` : Lists the terms and Term Frequencies in document DOC (Xapian::docid)

`-D`, `--docnames` : Lists the document names

`-n`, `--numwithocc=NUMBER` : Gives the number of terms with number of occurrences equal or higher than NUMBER

`-s`, `--stems` : Lists stems

`-t`, `--term=STRING` : Lists the Document Frequencies for terms beginning with STRING

`-v`, `--verbose` : Increases level of output

`-V`, `--voc` : Lists the indexing terms

`-?`, `--help` : Give this help list

`--usage` : Give a short usage message

`--version` : Print programme version

A.3.11 xapiandbfilter

`xapiandbfilter` allows filtering a document collection indexed by Xapian. It allows removing unnecessary terms, or removing non-stem indexing terms.

Usage: `xapiandbfilter [OPTION...] input_database output_database`

`-i`, `--use_index=FILE` : keeps only those index terms that are also used to index FILE (a Xapian Database).

`-s`, `--silent` : runs silently.

`-S`, `--stems_only` : keeps only stems.

`-v`, `--verbose` : tells you more.

`-?`, `--help` : Give this help list

`--usage` : Give a short usage message

`-V`, `--version` : Print programme version

Appendix B

Evaluation tool: Treceval

Obtaining, compiling and using Treceval : TrecEval can be obtained from Michel Beigbeder's page `http://www.emse.fr/~mbeig/IR/tools.html`. It compiles quite straightforwardly with a simple `make` in the source directory (further instructions are given the in the `README` file).

TrecEval is used via the command line in the following way:

```
./treceval reference_file IR_programme_output_file
```

For instance, with a TrecEval compiled as above

```
./trec_eval test/qrels.test test/results.test
```

is a valid command in the source directory

Reference file: The reference file `reference_file`, which contains the "correct answers" for queries, is relative to a particular document file/queries file couple (a TrecEval user will not need editing these files). A sample of such a file is given as figure B.1. In such a file, each line holds a tuple

`query_id iter doc_id rank`

Spaces are used as delimiters.

`query_id` : query identification number, a three-digit integer. Tuples are sorted by increasing `query_id`.

`iter` : iteration constant, required, yet ignored, by TrecEval .

`doc_id` : document identification "number" (in fact a string). It is given by the element found between the "DOCNO" XML tags in the corpus (document) file.

`rank` : the relevance of query `query_id` toward document `docno`.

```
1       0       511     1
1       0       513     1
2       0       80      1
2       0       90      1
(...)
```

Figure B.1: Extract of a TrecEval reference file

Result file: The result files `IR_programme_output_file` produced by the IR programme being tested contain the rank and similarity of every possible document/query pair obtained by combining the contents of the document and queries files (see figure B.1 for example). Each line of the file is of the form

`query_id iter doc_id rank sim run_id`

Spaces are used as delimiters.

`query_id` : query identification number, a three-digit integer. The results are to be sorted by increasing `query_id`.

`iter` : iteration constant, required, yet ignored, by TrecEval .

`doc_id` : document identification "number" (in fact a string). It is given by the element found between the "DOCNO" XML tags in the corpus (document) file.

`rank` : rank, an integer between `0` and `1000`. Like `iter`, this value is required in the file format, but ignored by TrecEval .

`sim` : similarity, a "float" floating-point value which gives the numerical value of the mathematical similarity computer for the couple (query, document)

`run_id` : arbitrary name for the run (execution of the programme). This string is printed in the output at runtime, but does not have any influence otherwise.

```
1   0   15   20   0.0197334   0
1   0   1    21   0           0
2   0   71   1    0.213504    0
2   0   68   2    0.158238    0
(...)
```

Figure B.2: Example of a TrecEval answer file (answers given by the programme being tested): queries 1 and 2 are compared to documents 15 and 1, and 71 and 68 respectively. The matching of the couple (doc=71, query=2) ranks 1st, with a similarity of `0.213504`; that of (doc=15, query=1) is 15th with a similarity of `0.0197334`; the couple (1, 1) comes last, since this query and document are orthogonal (similarity `0`).

Output: TrecEval output is given in a terminal, as seen in figure B.3. The various
Useful references:

- `http://www.ir.iit.edu/~dagr/cs529/files/project_files/trec_eval_desc.htm`
- `http://www.cs.colorado.edu/~martin/Csci7000/using_trec_eval.txt`

```
$ ./trec_eval test/qrels.test test/results.test
num_q             all     3
num_ret           all     1500
num_rel           all     561
num_rel_ret       all     131
map               all     0.1785
gm_ap             all     0.1051
R-prec            all     0.2174
bpref             all     0.1981
recip_rank        all     0.4064
ircl_prn.0.00     all     0.4665
ircl_prn.0.10     all     0.3884
ircl_prn.0.20     all     0.3186
ircl_prn.0.30     all     0.2732
ircl_prn.0.40     all     0.2666
ircl_prn.0.50     all     0.2184
ircl_prn.0.60     all     0.0822
ircl_prn.0.70     all     0.0348
ircl_prn.0.80     all     0.0312
ircl_prn.0.90     all     0.0312
ircl_prn.1.00     all     0.0312
P5                all     0.2667
P10               all     0.3000
P15               all     0.3111
P20               all     0.3667
P30               all     0.3333
P100              all     0.2467
P200              all     0.1600
P500              all     0.0873
P1000             all     0.0437
```

Figure B.3: Example of a TrecEval terminal output. `R-Prec` gives the R-Precision, and `map` gives the Mean Averable Precision. The series of `ircl_prn` are the interpolation precisions at different values of recall; they can be used to draw a Precision-Recall graph. `P5`, `P10` ...give the precision at 5, 10 etc. retrieved documents.

Index

Bibliography

[1] Carmen Alvarez, Philippe Langlais, and Jian-Yun Nie. Word pairs in language modeling for information retrieval, 2004.

[2] S.-I. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical Monographs*. American Mathematical Society, 2000.

[3] Leif Azzopardi. *Incorporating Context within the Language Modeling Approach for ad hoc Information Retrieval*. PhD thesis, University of the West of Scotland, Paisley, Scotland, 2005.

[4] Leif Azzopardi, Mark Girolami, and Keith van Risjbergen. Topic based language models for ad hoc information retrieval. In *IJCNN 2004: Proceedings of the International Joint Conference in Neural Networks*, 2004.

[5] Holger Billhardt, Daniel Borrajo, and Victor Maojo. A context vector model for information retrieval. *J. Am. Soc. Inf. Sci. Technol.*, 53(3):236–249, 2002.

[6] D. Blei and J. Lafferty. A correlated topic model of *Science*. *Annals of Applied Statistics*, 1(1):17–35, 2007.

[7] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation, 2003.

[8] P. Bollmann. Two axioms for evaluation measures in information retrieval. In *SIGIR '84: Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 233–245, Swinton, UK, UK, 1984. British Computer Society.

[9] P. Bollmann and V. S. Cherniavsky. Measurement-theoretical investigation of the mz-metric. In *SIGIR '80: Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, pages 256–267, Kent, UK, UK, 1981. Butterworth & Co.

[10] Wray Buntine and Aleks Jakulin. Discrete component analysis. In *Subspace, Latent Structure and Feature Selection Techniques*. Springer-Verlag, 2006.

[11] Igor V. Cadez, Scott Gaffney, and Padhraic Smyth. A general probabilistic framework for clustering individuals and objects. In *Proc. of 6th Int. Conf. on Knowledge Discovery and Data mining*, pages 140–149, 2000.

[12] Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. pages 78–102, 2001.

[13] C. Clarke, N. Craswell, and I. Soboroff. *Proceedings of TREC 2004*, 2004.

[14] W.W. Cohen. Learning to classify English text with ILP methods. In L. De Raedt, editor, *ILP95*, pages 3–24. DEPTCW, 1995.

[15] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. 1990.

[16] Gabriel Doyle and Charles Elkan. Accounting for burstiness in topic models. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 281–288, New York, NY, USA, 2009. ACM.

[17] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM Press.

[18] Georges Dupret. Latent concepts and the number orthogonal factors in latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 221–226, New York, NY, USA, 2003. ACM Press.

[19] Hui Fang, Tao Tao, and ChengXiang Zhai. A formal study of information retrieval heuristics. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 49–56, New York, NY, USA, 2004. ACM Press.

[20] Edward A. Fox. Lexical relations: enhancing effectiveness of information retrieval systems. *SIGIR Forum*, 15(3):5–36, 1980.

[21] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Trans. Inf. Syst.*, 9(3):223–248, 1991.

[22] Luigi Galavotti, Fabrizio Sebastiani, and Maria Simi. Experiments on the use of feature selection and negative evidence in automated text categorization. In *ECDL '00: Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries*, pages 59–68, London, UK, 2000. Springer-Verlag.

[23] Mark Girolami and Ata Kabán. On an equivalence between plsi and lda. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 433–434, New York, NY, USA, 2003. ACM.

[24] Donna Harman. Overview of the fourth Text REtrieval Conference (TREC–4). *Proc. of the 4th Text REtrieval Conf.*, pages 1–23, 1995.

[25] Xiaofei He, Deng Cai, Haifeng Liu, and Wei-Ying Ma. Locality preserving indexing for document representation. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 96–103, New York, NY, USA, 2004. ACM Press.

[26] Alexander Hinneburg, Hans-Henning Gabriel, and André Gohr. Bayesian folding-in with Dirichlet kernels for PLSI. In *Proc. of the 7th IEEE Int. Conf. on Data Mining*, pages 499–504, 2007.

[27] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. of 22th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 50–57, 1999.

[28] T. Hofmann. Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. In *Advances in Neural Information Processing Systems*, volume 12, pages 914–920, 2000.

[29] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, volume 11, pages 487–493. MIT Press, 1999.

[30] T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In *Proceedings of the 1999 Conference on AI and Statistics*. Morgan Kaufmann, 1999.

[31] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 487–493, Cambridge, MA, USA, 1999. MIT Press.

[32] F. Jelinek and R.L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice.*, pages 381–397. North-Holland, 1980.

[33] E. Kokiopoulou and Y. Saad. Polynomial filtering in latent semantic indexing for information retrieval. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 104–111, New York, NY, USA, 2004. ACM Press.

[34] Wessel Kraaij. *Variations on Language Modeling for Information Retrieval*. PhD thesis, University of Twente, June 2004.

[35] Solomon Kullback. *Information theory and statistics*. John Wiley and Sons, 1959.

[36] Oren Kurland and Lillian Lee. Clusters, language models, and ad hoc information retrieval. *ACM Trans. Inf. Syst.*, 27(3):1–39, 2009.

[37] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proc. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2001.

[38] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proc. of 24th Annual Int. Conference on Research and Development in Information Retrieval (SIGIR)*, pages 111–119, 2001.

[39] Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–89, New York, NY, USA, 1998. ACM Press.

[40] Leah S. Larkey. Automatic essay grading using text categorization techniques. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 90–95, New York, NY, USA, 1998. ACM Press.

[41] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 289–297, New York, NY, USA, 1996. ACM Press.

[42] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proc. of 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 120–127, 2001.

[43] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50, New York, NY, USA, 1992. ACM Press.

[44] David D. Lewis. Feature selection and feature extraction for text categorization. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 212–217, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[45] David D. Lewis. Text representation for intelligent text retrieval: a classification-oriented view. pages 179–197, 1992.

[46] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, Las Vegas, US, 1994.

[47] Y. H. Li and Anil K. Jain. Classification of text documents. *Comput. J.*, 41(8):537–546, 1998.

[48] Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, New York, NY, USA, 2004. ACM.

[49] Yuanhua Lv and ChengXiang Zhai. Adaptive relevance feedback in information retrieval. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 255–264, New York, NY, USA, 2009. ACM.

[50] Yuanhua Lv and ChengXiang Zhai. Positional language models for information retrieval. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306, New York, NY, USA, 2009. ACM.

[51] Rasmus E. Madsen, David Kauchak, and Charles Elkan. Modeling word burstiness using the dirichlet distribution. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 545–552, New York, NY, USA, 2005. ACM.

[52] Geoffrey McLachlan and David Peel. *Finite Mixture Models*. Wiley, 2000.

[53] Donald Metzler and W. Bruce Croft. Modeling query term dependencies in information retrieval with markov random fields, 2005.

[54] Donald Metzler and W. Bruce Croft. Beyond bags of words: Modeling implicit user preferences in information retrieval, 2006.

[55] Dunja Mladenic. Feature subset selection in text-learning. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 95–100, London, UK, 1998. Springer-Verlag.

[56] Isabelle Moulinier and Jean-Gabriel Ganascia. Applying an existing machine learning algorithm to text categorization. In *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 343–354, London, UK, 1996. Springer-Verlag.

[57] Ramesh Nallapati. Discriminative models for information retrieval. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 64–71, New York, NY, USA, 2004. ACM Press.

[58] Ramesh Nallapati. *The Smoothed Dirichlet Distribution: Understanding Cross-Entropy Ranking in Information Retrieval*. PhD thesis, University of Massachusetts, Amherst, MA, USA, 2006.

[59] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perception learning, and a usability case study for text categorization. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 67–73, New York, NY, USA, 1997. ACM Press.

[60] M. Nyffenegger. Similarités textuelles à base de noyaux de Fisher et "Latent Semantic Analysis". Master's thesis, Ecole Polytechnique Fédérale de Lausanne, Switerland, march 2005.

[61] M. Nyffenegger, J.-C. Chappelier, and E. Gaussier. Revisiting Fisher kernels for document similarities. In *Proc. of 17th European Conf. on Machine Learning*, pages 727–734, 2006.

[62] Laurence A. F. Park and Kotagiri Ramamohanarao. The sensitivity of latent dirichlet allocation for information retrieval. In Wray Buntine, Marko Grobelnik, Dunja Mladenič, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD Proceedings, Part II*, Lecture Notes in Artificial Intelligence, pages 176–188. Springer, Bled, Slovenia, September 2009.

[63] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *21st SIGIR Conf. on Research and Development in Information Retrieval*, pages 275–281, 1998.

[64] Joaquín Pérez-Iglesias, José R Pérez-Agüera, Víctor Fresno, and Yuval Z Feinstein. Integrating the probabilistic models bm25/bm25f into lucene. Technical Report arXiv:0911.5046, Nov 2009. Comments: Software can be downloaded from: http://nlp.uned.es/ jperezi/Lucene-BM25/.

[65] Loïs Rigouste. *Méthodes probabilistes pour l'analyse exploratoire de données textuelles*. PhD thesis, École Nationale Supérieure des Télécommunications, 2006.

[66] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC–3. *Proc. of the 3rd Text REtrieval Conf.*, 1994.

[67] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing.*, pages 313–323. Prentice Hall, 1971.

[68] Miguel E. Ruiz and Padmini Srinivasan. Hierarchical neural networks for text categorization (poster abstract). In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 281–282, New York, NY, USA, 1999. ACM Press.

[69] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

[70] Hinrich Schétze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 229–237, New York, NY, USA, 1995. ACM Press.

[71] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[72] Fabrizio Sebastiani, Alessandro Sperduti, and Nicola Valdambrini. An improved boosting algorithm and its application to automated text categorization. Technical report, Paris, France, France, 2000.

[73] Florian Seydoux and Jean-Cédric Chappelier. Semantic Indexing using Minimum Redundancy Cut in Ontologies. In Galia Angelova, Kalina Bontcheva, Ruslan Mitkov, and Nikolai Nicolov, editors, *Proc. of International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, pages 486–492, 2005.

[74] Florian Seydoux, Martin Rajman, and Jean-Cédric Chappelier. *Exploitation de connaissances sémantiques externes dans les représentations vectorielles en recherche documentaire*. PhD thesis, 2006.

[75] Georges Siolas. *Modèles Probabilistes et noyaux pour l'extraction d'informations à partir de documents*. PhD thesis, Paris 6, 2003.

[76] Georges Siolas and Florence d'Alché Buc. Support vector machines based on a semantic kernel for text categorization. In *IJCNN '00: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 5*, page 5205, Washington, DC, USA, 2000. IEEE Computer Society.

[77] Ian Soboroff, Ellen Voorhees, and Nick Craswell. Summary of the sigir 2003 workshop on defining evaluation methodologies for terabyte-scale test collections. *SIGIR Forum*, 37(2):55–58, 2003.

[78] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *CIKM '99: Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321, New York, NY, USA, 1999. ACM.

[79] Pascal Soucy and Guy W. Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *IJCAI*, pages 1130–1135, 2005.

[80] Karen Sparck Jones and Peter Willett, editors. *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[81] Hirotoshi Taira and Masahiko Haruno. Feature selection in svm text categorization. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 480–486, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[82] Chunqiang Tang, Sandhya Dwarkadas, and Zhichen Xu. On scaling latent semantic indexing for large peer-to-peer systems. In *SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval*, pages 112–121, New York, NY, USA, 2004. ACM Press.

[83] Yee Whye Teh. A hierarchical bayesian language model based on pitman-yor processes. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[84] C.J. van Rijsbergen, editor. *Information Retrieval*. Butterworth-Heinemann; 2nd edition, London, 1979.

[85] C.J. van Rijsbergen, S.E. Robertson, and M.F. Porter. New models in probabilistic information retrieval. In *British Library Research and Development Report, no. 5587*. British Library, 1980.

[86] Ellen M. Voorhees. *Information Extraction*, volume 1714 of *Lecture Notes in Computer Science*.

[87] Ellen M. Voorhees and Donna Harman. The text retrieval conference (trec): history and plans for trec-9. *SIGIR Forum*, 33(2):12–15, 1999.

[88] Xing Wei and W. Bruce Croft. LDA-based document models for ad-hoc retrieval. In *Proc. of 29th Int. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 178–185, 2006.

[89] Max Welling, Chaitanya Chemudugunta, and Nathan Sutter. Deterministic latent variable models and their pitfalls. *SIAM Conference on Data Mining SDM 2008*, 2008.

[90] Erik D. Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, US, 1995.

[91] Yiming Yang and Christopher G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Trans. Inf. Syst.*, 12(3):252–277, 1994.

[92] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA, 1999. ACM Press.

[93] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[94] Sarah Zelikovitz and Haym Hirsh. Improving short-text classification using unlabeled background knowledge to assess document similarity. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1183–1190, 2000.

[95] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proc. of 10th Int. Conf. on Information and Knowledge Management (CIKM)*, pages 403–410, 2001.

[96] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2(3):137–213, 2008.

[97] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56, New York, NY, USA, 2002. ACM Press.

[98] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

Emmanuel ECKARD

Artificial Intelligence Laboratory
EPFL, Bâtiment IN
1015 Lausanne
Email: emmanuel.eckard@epfl.ch
Phone: (+41) 21 693 66 97

French national.
No military obligations.
Single, no children.

CURRENT POSITION AND RESEARCH INTERESTS

Ph.D. student at the Artificial Intelligence Laboratory of the Swiss Institute of Technology (EPFL), to be defended in March 2010.
Machine learning. Statistical models. Latent semantics. Information retrieval.

EDUCATION

2006 – 2009 **PhD Thesis**: *Fisher kernels and probabilistic latent semantics models*
Swiss Institute of Technology (EPFL), Lausanne, Switzerland.
Defence planned for automn 2009.

2003 **Master in Physics**: *Étude de la sélection inclusive de désintégration $B \to X_u l \nu$ dans l'expérience BELLE* ("Study of the inclusive selection of $B \to X_u l \nu$ decay in BELLE experiment")
Swiss Institute of Technology (EPFL), Lausanne, Switzerland

1996 **Maturité fédérale scientifique.** (High school diploma specialised in sciences)
Gymnase cantonal de Neuchâtel, Switzerland.
With honours.

RESEARCH EXPERIENCE

2006 – 2009 PhD Thesis
Supervisors: Dr. Martin RAJMAN – LIA/EPFL (Switzerland)
Dr. Jean-Cédric CHAPPELIER – LIA/EPFL (Switzerland)
Information retrieval; latent topics and semantics; Fisher kernels; machine learning; modelling.
Work supported by grants #200021-111817 and #200020-119745 of the Swiss Foundation of Science.

2003 Study of the inclusive selection of the semi-leptonic decay channel of B mesons
Adviser: Dr. O. Schneider – Laboratoire de Physique des Hautes Énergies, UNIL
Analysis of simulated data as acquired by the BELLE detector.

2001 Haptic device for Atomic Force Microscope (AFM) control
Adviser: Dr. C. Baur – VRAI Group (EPFL)
Control software for an Atomic Force Microscope (AFM). Simulation of nanotubes and AFMs.

Peer-reviewed publications

- *An Ad Hoc Information Retrieval Perspective on PLSI through language model identification*, J.-C. Chappelier, E. Eckard, Proc. of ICTIR 2009.
- *PLSI: the true Fisher kernel and beyond – IID processes, information matrix and model identification in PLSI*, J.-C. Chappelier, E. Eckard, Proc. of ECML-PKDD 2009.
- *Utilisation de PLSI en recherche d'information*, J.-C. Chappelier, E. Eckard, Proc. of TALN 2009.
- *Rôle de la matrice d'information et pondération des composantes dans les noyaux de Fisher pour PLSI*, J.-C. Chappelier, E. Eckard, Proc. of CORIA 2009.

Other publications

- *Free Software for research in Information Retrieval and Textual Clustering*, E. Eckard and J.-C. Chappelier, technical report, 2007.
- *Topian 0.1 reference Manual*, E. Eckard and J.-C. Chappelier, Technical report, 2007.
- *Inclusion de sens dans la représentation de documents textuels : état de l'art*, E. Eckard and J.-C. Chappelier. Technical report, 2007.

Conferences attended

2009	International Conference on the Theory of Information Retrieval (ICTIR), Cambridge, UK.
2009	European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Bled, Slovenia.
2009	Traitement Automatique des Langues Naturelles (TALN – "Natural Language Automated Processing"), Senlis, France.
2009	Conférence en Recherche d'Information et Applications (CORIA – "Conference in Information Retrieval and Applications"), peninsula of Giens, France.
2006	École d'Automne en Recherche d'Information et Applications (EARIA – "Autumn School for Information Retrieval and Applications"), Autrans, France.
2006	European Summer School in Logic, Language and Information (ESSLLI), Málaga, Spain.
2004	International Conference on Computational Linguistics (CORIA), Geneva, Switzerland.
2003	World Student Community on Sustainable Development (WSCSD), Tōkyō, Japan.

Teaching experience

- Yearly lecture on Information Retrieval in a Master-level ex cathedra course on Natural Language Processing.
- Instructorship in C++ for first year students, given during three years.
- Numerous install parties at the Linux User Group of the EPFL (and co-founder thereof).

Awards and honours

- Research grant of the Canon Foundation, 2006.
- Presentation of the Gnuwin Free software project at the 2003 WSCSD (World Student Community on Sustainable Development) meeting in Tōkyō, Japan.
- Price for supplementary mémoire: *Malraux et Hemingway, témoins la Guerre d'Espagne* ("Malraux and Hemingway, witnesses of the Spanish Civil War").

Languages

- French: native speaker.
- English: fluent.
- German: good.

Referees

Dr. Martin Rajman
EPFL / IC / IIF / LIA
Bâtiment IN, Station 14
CH - 1015 Lausanne (Switzerland)
martin.rajman@epfl.ch

Dr. Jean-Cédric Chappelier
EPFL / IC / IIF / LIA
Bâtiment IN, Station 14
CH - 1015 Lausanne (Switzerland)
jean-cedric.chappelier@epfl.ch