NLP From Scratch Without Large-Scale Pretraining: A Simple and Efficient Framework

Xingcheng Yao [*1] **Yanan Zheng** [*2] **Xiaocong Yang** [3 4] **Zhilin Yang** [1 5 4]

Abstract

Pretrained language models have become the standard approach for many NLP tasks due to strong performance, but they are very expensive to train. We propose a simple and efficient learning framework TLM that does not rely on large-scale pretraining[1]. Given some labeled task data and a large general corpus, TLM uses task data as queries to retrieve a tiny subset of the general corpus and jointly optimizes the task objective and the language modeling objective from scratch. On eight classification datasets in four domains, TLM achieves results better than or similar to pretrained language models (e.g., RoBERTa-Large) while reducing the training FLOPs by two orders of magnitude. With high accuracy and efficiency, we hope TLM will contribute to democratizing NLP and expediting its development [2].

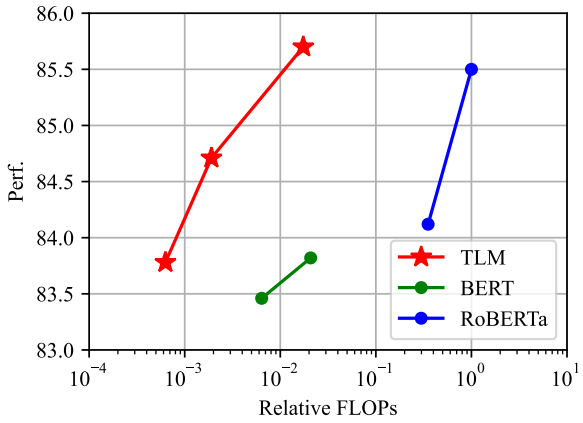


Figure 1. Average performance on eight tasks v.s. relative FLOPs w.r.t. RoBERTa-Large (Liu et al., 2019). TLM slightly outperforms RoBERTa-Large while reducing FLOPs by two orders of magnitude.

1. Introduction

Pretrained language models (PLMs) have drawn much attention from the natural language processing (NLP) community. Neural networks based on the Transformer architecture (Vaswani et al., 2017) are trained on large general corpora for self-supervised language modeling tasks such as masked language modeling (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019), autoregressive language modeling (Radford et al., 2018; Brown et al., 2020), permutation language modeling (Yang et al., 2019), etc, and then are finetuned on a small amount of labeled data for downstream tasks. This pretraining-finetuning framework has significantly improved the performance of many NLP tasks.

However, while considered effective, large-scale pretraining is usually computationally expensive. For example, RoBERTa-Large (Liu et al., 2019), a widely-used PLM, consumes a computational cost of 4.36×10^{21} FLOPs[3]. Larger PLMs such as GPT-3 (Brown et al., 2020) consume 50 times more FLOPs for training than RoBERTa-Large. The expensiveness of large-scale pretraining prevents many research groups with limited budgets from pretraining customized language models, exploring new neural architectures, or improving pretraining loss functions. In contrast, a large number of NLP researchers resort to improving the finetuning algorithms, whose performance is largely upper-bounded by the pretraining procedure. This creates a high barrier of NLP research and might not be ideal for the long-term development of the field.

[*]Equal contribution [1]Institute for Interdisciplinary Information Sciences, Tsinghua University [2]Department of Computer Science and Technology, Tsinghua University [3]School of Economics and Management, Tsinghua University [4]Recurrent AI, Inc [5]Shanghai Qi Zhi Institute. Correspondence to: Zhilin Yang <zhiliny@tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]In the broadest sense, pretraining means training on some objectives before optimizing the target tasks. In contrast, throughout the paper, we use "pretraining" to only refer to task-agnostic training of language models on a large general corpus, such as BERT (Devlin et al., 2019).

[2]Our code, model checkpoints and datasets are publicly available at: https://github.com/yaoxingcheng/TLM

[3]It was pretrained with 1,000 V100 GPUs each with 32GB memory for approximately one day.

Even though there have been efforts devoted to studying and improving the efficiency of language model pretraining (Clark et al., 2020; So et al., 2021; Tay et al., 2021; Chen et al., 2021), most of them focus on designing sample-efficient self-supervised tasks or discovering efficient Transformer architectures suitable for pretraining. Their improvements are limited, with a reduction of computational costs (in terms of FLOPs) less than one order of magnitude. Another line of works target reducing the sizes of PLMs using distillation (Sanh et al., 2019; Jiao et al., 2020) to improve the efficiency of inference, but these methods rely on pretraining a large PLM before distillation. Moreover, distilled models often do not perform as well as some of the best non-distilled PLMs such as RoBERTa-Large (Sanh et al., 2019; Jiao et al., 2020).

This work explores alternatives to the standard pretraining-finetuning paradigm, aiming at more drastic efficiency improvement without performance drop. We propose a simple, efficient, pretraining-free framework, **T**ask-driven **L**anguage **M**odeling (**TLM**). Given a large general corpus and some labeled task data, TLM directly trains a model from scratch without relying on PLMs. TLM is motivated by two key ideas. First, humans master a task by using only a small portion of world knowledge (e.g., students only need to review a few chapters, among all books in the world, to cram for an exam). We hypothesize that there is much redundancy in the large corpus for a specific task. Second, training on supervised labeled data is much more data efficient for downstream performance than optimizing the language modeling objective on unlabeled data. Based on these motivations, TLM uses the task data as queries to retrieve a tiny subset of the general corpus. This is followed by jointly optimizing a supervised task objective and a language modeling objective using both the retrieved data and the task data.

We evaluate TLM on eight different tasks covering the domains of news, review, computer science, and biomedical science, following the setting of Gururangan et al. (2020). TLM achieves results better than or similar to BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) while reducing the training FLOPs by **two orders of magnitude**[4].

2. Related work

Pretrained Language Models Pretrained language models have become the de-facto solution to many of the NLP tasks (Radford et al., 2018; Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2019; Brown et al., 2020; Yang et al., 2019). Those models are usually pretrained on a large-scale corpus in a self-supervised manner to learn a contextualized representation of tokens in natural language, and then are fine-tuned with labeled data for specific tasks. BERT (Devlin et al., 2019), one of the most popular PLMs, is pretrained on a 16GB English corpus using a masked language modeling objective (i.e. predicting randomly masked tokens). RoBERTa (Liu et al., 2019) inherits the training objective of BERT, but is pretrained on a larger corpus consisting of 160GB English texts with larger batch size and dynamic token masking. In this work, we take both BERT and RoBERTa as our major baselines.

Efficient Pretraining for NLP There is a line of work dedicated to improving the efficiency of pretraining language models. You et al. (2020) and Shoeybi et al. (2019) utilized the data and model parallelism across different computational devices to accelerate the pretraining process. However, accelerating through parallelism does not actually reduce computational costs in terms of FLOPs for training models at large scale. Chen et al. (2021) and So et al. (2021) tried to identify efficient neural network architectures for language model pretraining, based on the lottery ticket hypothesis and neural architecture search. Such modifications on architecture can bring about $50\% \sim 70\%$ reduction in computational costs. Clark et al. (2020) and He et al. (2021) incorporated manually designed mechanisms into language model pretraining, such as adversarial training and disentangled representation of content and position, which brings about $50\% \sim 75\%$ reduction in computational costs. Gu et al. (2020) proposed to use task-guided pre-training with selective masking, which reduces the computation cost by around 50%. In this work, orthogonal to the aforementioned works, we investigate improving efficiency by reducing training data redundancy. Our approach also results in more drastic improvements.

Efficient Inference of Pretrained Models Another line of work aims at improving inference efficiency of PLMs. Some works improve inference efficiency by distilling large PLMs into small-sized models and using the distilled models for inference, such as DistilBERT (Sanh et al., 2019), TinyBERT (Jiao et al., 2020), MobileBERT (Sun et al., 2020), FastBERT (Liu et al., 2020), BORT (de Wynter & Perry, 2020), and BERT-of-Theseus (Xu et al., 2020). Other works speed up inference by quantizing PLMs with low-precision representations during inference, such as Q8-BERT (Zafrir et al., 2019), Q-BERT (Shen et al., 2020), and I-BERT (Kim et al., 2021). Another type of works, such as (Michel et al., 2019; Wang et al., 2020; Gordon et al., 2020), adopt pruning by removing parts of PLMs to make it smaller and faster. However, these methods rely on large PLMs, and the performance after distillation, pruning, or quantization often decreases to a certain extent compared with some of the best PLMs (e.g., RoBERTa-Large). In contrast, our approach doesn't rely on large-scale pre-training and achieves better or at least comparable performance.

[4]This effectively reduces the cost from training on 1,000 GPUs for one day to training on 8 GPUs for 42 hours.

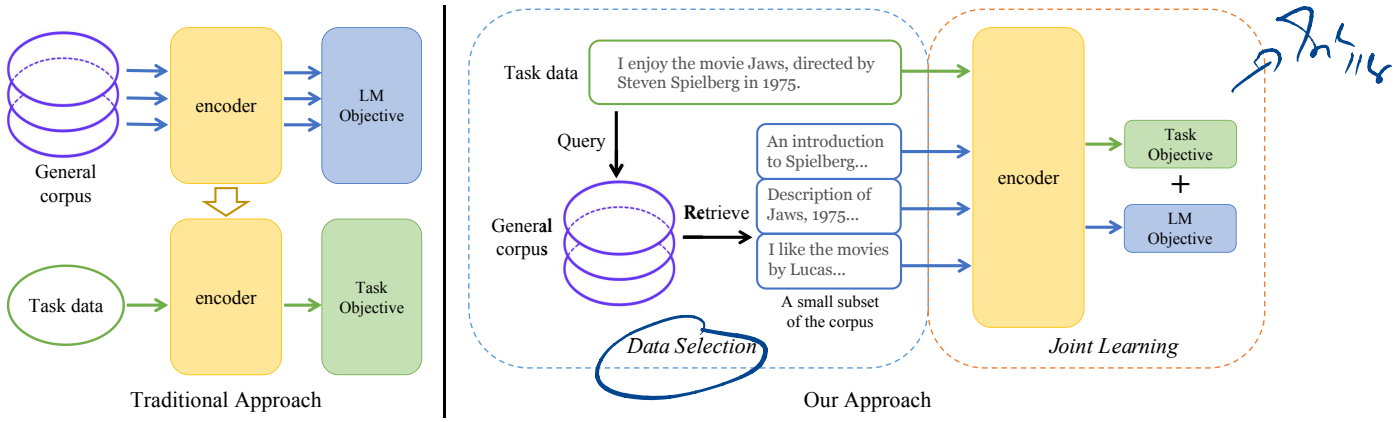


Figure 2. Comparison between the traditional pretraining-finetuning approach and our proposed framework TLM: instead of training a language model over the entire general corpus and then finetuning it on task data, we first use task data as queries to retrieve a tiny subset of the general corpus, and then perform joint learning on both the task objective and self-supervised language modeling objective.

Domain and Task Adaptation for Pretrained Models Domain-adaptive finetuning is a method that finetunes a pretrained model on in-domain data using a language modeling objective. It has been shown to be effective for domain and task adaptation (Zhang et al., 2019; Gururangan et al., 2020; Li et al., 2020; Lee et al., 2020). There are a few crucial differences between domain-adaptive finetuning and TLM. First, TLM is a general method to improve training efficiency that does not use any additional domain data. It only utilizes the general corpus as in BERT and RoBERTa. In comparison, domain-adaptive finetuning uses domain data to improve domain adaptation. Second, while previous works on domain-adaptive finetuning are built upon a model pretrained on the general corpus, TLM learns from scratch without large-scale pretraining to substantially save computation costs.

Co-training for Semi-supervised Learning and Data-Density-Based Active Learning Additionally, we observe two techniques related to TLM. They are Co-Training (CT) (Qiao et al., 2018; Yang et al., 2021) and Data-Density-Based Active Learning (DAL) (Zhu et al., 2010; Wang et al., 2017) respectively. Both CT and TLM utilize unlabeled data to aid the learning on a certain task. The difference between TLM and CT is 2-fold: First, CT requires training distinct models from multiple views of unlabeled data, yet TLM only trains a single model through pre-text tasks such as MLM. Second, TLM takes the selection process of unlabeled data into account, which is little discussed in CT. TLM and DAL share the same flavor of finding representative instances in a pool of unlabeled data. However, DAL makes the assumption that every unlabeled sample can be effectively labeled by the definition of the task, which is not required by TLM. Also, DAL tries to find critical instances iteratively from the whole pool of unlabeled data, yet TLM only tries to find relevant instances in a one-shot way with respect to labeled data, which makes TLM more efficient than classic DAL algorithms.

3. Method

3.1. TLM: Task-Driven Language Modeling

It is an interesting phenomenon that humans are able to quickly master a certain task with limited time and effort by focusing only on pieces of relevant knowledge. For example, when students cram for exams, they review a few chapters instead of going through all books in the world. Following this observation, we conjecture that one of the key aspects of learning a task is to quickly and precisely locate task-relevant information. To this end, we develop TLM that first automatically retrieves relevant training data from a general corpus and then learns on the retrieved data and task data combined.

Formally, given a general corpus $\mathcal{D} = \{d_i\}_i$ where d_i is a document, and labeled task data $\mathcal{T} = \{(x_i, y_i)\}_i$ where x_i is text and $y_i \in \mathcal{Y}$ is a label[5], our goal is to train a model f to estimate the conditional probability for classification $f(x) = \hat{p}(y|x)$.

TLM consists of two steps as shown in Figure 2.

1. Retrieve data from a general corpus using task data as queries.
2. Train a model from scratch by jointly optimizing the task objective and the language modeling objective on the retrieved data and task data.

Retrieval From General Corpus For each example in the task data $x_i \in \mathcal{T}$, we retrieve a set of documents

[5]While it is straightforward to extend our framework to generation tasks, we focus on classification tasks in this work.

$\mathcal{S}_i = \{\tilde{d}_{i,1}, \tilde{d}_{i,2}, \cdots\}$ from the given general corpus $\mathcal{D}$. The set $\mathcal{S}_i$ represents the top-K similar documents to x_i in $\mathcal{D}$. Retrieved data for all examples x_i are combined $\mathcal{S} = \cup_i S_i$. Retrieved data $\mathcal{S}$ is a tiny subset of the general corpus $\mathcal{D}$.

We use BM25 (Robertson & Zaragoza, 2009) for retrieval due to its efficiency. While using embedding-based dense retrievers (Karpukhin et al., 2020) might lead to better retrieval results, we do not consider these methods to keep our approach as simple as possible. Moreover, dense retrievers rely on pretraining, which might bring additional computational costs. The exploration of achieving a better tradeoff between efficiency and retrieval performance is left to future work. Moreover, for tasks with extremely long texts (e.g., Helpfulness (McAuley et al., 2015)), we find it more efficient to extract keywords (e.g., using the RAKE algorithm (Rose et al., 2010)) to form the queries for retrieval instead of using the entire input sequence. We call the retrieved data $\mathcal{S}$ external data and the task data $\mathcal{T}$ internal data.

Note that our data retrieval method is task-agnostic—it only depends on text x without dependency on y. Moreover, the retrieval procedure does not assume the availability of domain-specific data. It operates on a general corpus and has the same input as the pretraining-finetuning paradigm.

Joint Training Given both the internal and external data, we train a language model f from scratch. Let $\mathcal{L}_{\text{mlm}}(x)$ be the masked language modeling loss as in BERT (Devlin et al., 2019), and let $\mathcal{L}_{\text{task}}(f(x), y)$ be the task loss function (e.g., cross entropy for classification). TLM optimizes the following loss function:

$$\rho_1 \mathbb{E}_{x\sim\mathcal{S}}[\mathcal{L}_{\text{mlm}}(x)] + \mathbb{E}_{x,y\sim\mathcal{T}}\left[\rho_2 \mathcal{L}_{\text{mlm}}(x) + \mathcal{L}_{\text{task}}(f(x), y)\right]$$

where ρ_1 and ρ_2 are hyperparameters. The network architecture we employ is identical to BERT, where we use a CLS head for classification and an LM head for masked language modeling. TLM can also be extended to other architectures for non-classification tasks. Our implementation involves a two-stage training procedure. In the first stage, we interleave one batch of internal data with ρ_1 batches of external data for mini-batch stochastic gradient descent, where ρ_1 is set as an integer. In the second stage, we set both ρ_1 and ρ_2 as zero to only finetune the model on internal data with the task objective.

3.2. Comparison Between TLM and PLMs

Both TLM and pretraining-finetuning have two stages. In fact, the second stage of TLM equals the traditional finetuning stage. The main difference between the first stage of TLM and pretraining (PLMs) is shown in Table 1. Unlike PLMs which learn as much task-agnostic knowledge as possible at an extremely high cost, TLM learns task-related knowledge for each task with very low costs.

Table 1. Comparison between TLM and PLMs. Here we provide qualitative comparison, while quantitative comparison in terms of training data size, FLOPs, and the number of parameters is available in Table 2.

	TLM	**PLMs**
Loss Function	$\mathcal{L}_{task}$ and $\mathcal{L}_{mlm}$	$\mathcal{L}_{mlm}$
Training Data	A tiny subset of D and task data $\mathcal{T}$	The entire D
Compute Cost	8 GPUs 42 hours	1,000 GPUs one day
Generality	Task-Driven	Task-Agnostic

Given the above difference between TLM and PLMs, we will discuss the pros and cons of TLM in detail.

Democratizing NLP In pretraining-finetuning paradigm, the finetuning performance is largely upper bounded by the pretrained model. However, due to the constraints of computational resources, the majority of NLP researchers cannot afford training large-scale language models and resort to studying the finetuning algorithms. Since only a small portion of researchers are working on the architectures, loss functions, and other design choices of PLMs, there is a risk that the development of the field might be slowing down. On the other hand, TLM is efficient and highly performant. As a result, TLM has the potential of democratizing NLP and expediting its development by allowing most researchers to freely explore the architectures, loss functions, algorithms, and other design choices in the neighborhood of a state-of-the-art solution.

Efficiency TLM improves over PLMs in terms of per-task FLOPs. In many cases when there are only a few target tasks, TLM is favorable. For example, a researcher might be interested in solving four textual entailment datasets, or an industrial team might want to improve a recommender system which can be viewed as one task. However, if the goal is to solve 1,000 tasks at once (e.g., building an NLP platform to serve multiple business units within a corporate), PLMs might still be preferred.

Flexibility Since TLM is task-driven, there is a larger degree of flexibility. Researchers can use custom strategies for tokenization, sequence length, data representations, hyperparameter tuning, etc, which might improve performance and/or efficiency.

Generality PLMs learn task-agnostic general representations and can be used for few-shot and zero-shot learning (Brown et al., 2020). In comparison, TLM trades generality for efficiency by learning only task-specific representations. How to further improve TLM in terms of learning more gen-

eral representations poses a challenge for future work. We believe multi-task learning might alleviate this issue given recent observations (Wei et al., 2021; Zhong et al., 2021), especially for in-domain zero-shot generalization. It might also be possible to combine pretraining with TLM, e.g., using a small PLM with TLM to match a larger PLM, to achieve a better tradeoff between generality and efficiency.

4. Experiments

4.1. Setup

Datasets Following (Gururangan et al., 2020), we conduct experiments on eight tasks over four domains, including biomedical science, computer science, news, and reviews (two tasks in each domain). The tasks can be categorized into high-resource and low-resource tasks. High-resource tasks has more than 5K task data, including AGNews (Zhang et al., 2015), IMDB (Maas et al., 2011), RCT (Dernoncourt & Lee, 2017), and Helpfulness (McAuley et al., 2015), while low-resource tasks include ChemProt (Kringelum et al., 2016), ACL-ARC (Jurgens et al., 2018), SciERC (Luan et al., 2018), and HyperPartisan (Kiesel et al., 2019). For the general training corpus, we collected two corpora that respectively match the original training corpora of BERT and RoBERTa. We name them respectively Corpus-BERT ($\mathcal{C}_{\text{BERT}}$) and Corpus-RoBERTa ($\mathcal{C}_{\text{RoBERTa}}$). The size of $\mathcal{C}_{\text{RoBERTa}}$ is 10 times larger than $\mathcal{C}_{\text{BERT}}$.

Baselines Our experiments focus on comparison with general PLMs. We finetuned both BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) of base and large scales as the baselines. Although TLM is a general method without using addition in-domain data, it even performs close to domain-adaptive finetuning methods (Gururangan et al., 2020) (see Appendix A for detailed comparison).

Evaluation Strategy We report the average performance across three random seeds, together with the standard deviation. We follow Beltagy et al. (2019) and Gururangan et al. (2020) to report the test micro-F1 for ChemProt and RCT, and macro-F1 for the rest of the datasets.

For fair comparison, we evaluate TLM of different *training scales*. The training scale is defined by three factors, including the number of parameters, the size of the general corpus, and the number of total training tokens. The number of total training tokens is calculated as the product of training steps, batch size, and sequence length. We report TLM at three training scales as shown in Table B.1, namely *small*, *medium*, and *large* scales. Each scale of TLM is accordingly compared to the PLM baselines with an increasing computational cost.

Training Details For each experiment of TLM, while fixing the training scale hyper-parameters (i.e., training steps, batch size and sequence length), we perform a grid search over ρ_1 and ρ_2. We listed the hyper-parameters used in Table B.1 in Appendix.

4.2. Main Results

Table 2 shows the main results that compare TLM of three different scales and the according PLM baselines. In conclusion, TLM can achieve results that are better than or comparable to the baselines with substantial reduction in FLOPs and the size of training data. Specifically, at a small scale, TLM achieves comparable results to BERT-Large with an average of 1/33 of FLOPs and 1/16 of the training corpus. At the medium and large scales, TLM improves the performance by 0.59 and 0.24 points on average respectively, while significantly reducing both FLOPs and the training data size by two orders of magnitude or more. These results confirm that TLM is highly accurate and much more efficient than PLMs. Moreover, TLM gains more advantages in efficiency at a larger scale. This indicates that larger-scale PLMs might have been trained to store more general knowledge that is not useful for a specific task.

4.3. Ablation Study

4.3.1. Data Retrieval

Table 3 shows the comparison between different retrieval methods (i.e., BM25 and random retrieval) and different sizes of the general corpus. We find that given the same general corpus, the results of BM25 significantly outperform those of random retrieval by a large margin on all tasks, showing that using task-relevant data for joint training is crucial for the best performance. Specifically, BM25 shows an advantage of almost 1 point against random retrieval on high-resource tasks such as IMDB, and more significant advantages on low-resource tasks such as SciERC and ChemProt by around 3-4 points. This is aligned with our intuition that low-resource tasks rely more on external data.

By comparing the results of $\mathcal{C}_{\text{BERT}}$ and $\mathcal{C}_{\text{RoBERTa}}$ with BM25, we observe that increasing the size of the general corpus improves performance (by 0.5, 1.34, and 1.35 points on IMDB, SciREC, and ChemProt respectively). The gains of using 10 times more data are similar to the ones observed in PLMs (Yang et al., 2019; Liu et al., 2019). This indicates that although TLM only uses a small amount of data, it is able to scale when a larger general corpus is available while maintaining efficiency. On the other hand, the gains of using a larger corpus diminish with random retrieval, showing that random retrieval, as a task-agnostic method, is not very sensitive to the general corpus size.

Data retrieval selects the top-K similar documents from

Table 2. Evaluation results for TLM at three different training scales. For each task, we report the average F1 score across three random seeds with standard deviations as subscripts. We also list the number of parameters, the total training compute (FLOPs), and the size of training corpus for comparison.

Model	#Param	FLOPs[1]	Data[2]	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT	Avg.
BERT-Base[3]	109M	2.79E19	16GB	93.50 $\pm$0.15	91.93 $\pm$1.74	69.11 $\pm$0.17	93.77 $\pm$0.22	69.45 $\pm$2.90	80.98 $\pm$1.07	81.94 $\pm$0.38	87.00 $\pm$0.06	83.46
BERT-Large[3]	355M	9.07E19	16GB	93.51 $\pm$0.40	91.62 $\pm$0.69	69.39 $\pm$1.14	**94.76** $\pm$0.09	69.13 $\pm$2.93	**81.37** $\pm$1.35	**83.64** $\pm$0.41	**87.13** $\pm$0.09	**83.82**
TLM *(small-scale)*	109M	2.74E18	0.91GB	**93.74** $\pm$0.20	**93.53** $\pm$1.61	**70.54** $\pm$0.39	93.08 $\pm$0.17	**69.84** $\pm$3.69	80.51 $\pm$1.53	81.99 $\pm$0.42	86.99 $\pm$0.03	83.78
RoBERTa-Base[3]	125M	1.54E21	160GB	**94.02** $\pm$0.15	93.53 $\pm$1.61	70.45 $\pm$0.24	**95.43** $\pm$0.16	68.34 $\pm$7.27	81.35 $\pm$0.63	82.60 $\pm$0.53	87.23 $\pm$0.09	84.12
TLM *(medium-scale)*	109M	8.30E18	1.21GB	93.96 $\pm$0.18	**94.05** $\pm$0.96	**70.90** $\pm$0.73	93.97 $\pm$0.10	**72.37** $\pm$2.11	**81.88** $\pm$1.92	**83.24** $\pm$0.36	**87.28** $\pm$0.10	**84.71**
RoBERTa-Large[3]	355M	4.36E21	160GB	94.30 $\pm$0.23	**95.16** $\pm$0.00	70.73 $\pm$0.62	**96.20** $\pm$0.19	**72.80** $\pm$0.62	82.62 $\pm$0.68	84.62 $\pm$0.50	**87.53** $\pm$0.13	85.50
TLM *(large-scale)*	355M	7.59E19	3.64GB	**94.34** $\pm$0.12	**95.16** $\pm$0.00	**72.49** $\pm$0.33	95.77 $\pm$0.24	72.19 $\pm$1.72	**83.29** $\pm$0.95	**85.12** $\pm$0.85	87.50 $\pm$0.12	**85.74**

[1] The total training compute (FLOPs) is calculated by $(6 \times \text{Total\_Training\_Tokens} \times \text{Parameter\_Size})$ as in (Brown et al., 2020). For TLM, FLOPs are reported as the averaged result over eight tasks.
[2] The size of data selected from general corpus that are actually used in training. For TLM, it is reported by averaging over eight tasks.
[3] The BERT-Base and BERT-Large are pretrained by (Devlin et al., 2019) and RoBERTa-Base and RoBERTa-Large are pretrained by (Liu et al., 2019). We finetuned them to obtain the results over the eight tasks.

Table 3. Results on the development set using different retrieval methods and different general corpora on each task. We compared two data retrieval methods: random retrieval and the BM25 algorithm. We compare two source general corpora: the corpus used in BERT ($\mathcal{C}_{\text{BERT}}$) and the corpus used in RoBERTa ($\mathcal{C}_{\text{RoBERTa}}$). The size of $\mathcal{C}_{\text{RoBERTa}}$ is 10 times larger than $\mathcal{C}_{\text{BERT}}$.

	IMDB	SciERC	ChemProt
Random			
w/ $\mathcal{C}_{\text{BERT}}$	93.65$\pm$0.09	83.80$\pm$0.62	80.65$\pm$0.48
w/ $\mathcal{C}_{\text{RoBERTa}}$	94.04$\pm$0.22	83.10$\pm$1.54	80.73$\pm$0.46
BM25			
w/ $\mathcal{C}_{\text{BERT}}$	94.40$\pm$0.09	86.07$\pm$0.48	83.64$\pm$0.26
w/ $\mathcal{C}_{\text{RoBERTa}}$	**94.90**$\pm$0.06	**87.41**$\pm$0.36	**84.99**$\pm$0.72

Table 4. Results on the development set with different values of K. The value K is the number of retrieved documents per task example. AGNews is a high-resource task, while SciREC and ChemProt are low-resource ones. Here we use $\rho_2 = 20$ for all tasks. When there are external data available, we use $\rho_1 = 4$ for AGNews and $\rho_1 = 1000$ for SciERC and ChemProt.

	AGNews	SciERC	ChemProt
Only Task Data	93.41$\pm$0.10	51.23$\pm$1.13	55.05$\pm$0.18
Top-50	**94.51**$\pm$0.15	77.61$\pm$1.75	77.21$\pm$0.47
Top-500	94.32$\pm$0.05	82.39$\pm$0.55	81.44$\pm$0.50
Top-5000	94.42$\pm$0.10	**86.07**$\pm$0.48	**83.64**$\pm$0.26

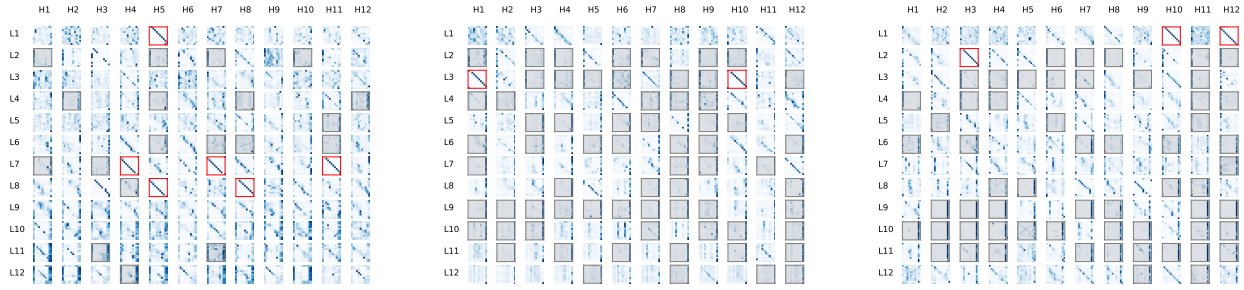
the general corpus. Table 4 shows the results of different K values. We observe that high-resource tasks such as AGNews only need a small K value, while low-resource tasks such as SciREC and ChemProt require a large K to obtain the best performance. The observation is consistent with the above analysis that low-resource tasks rely more on external data to improve from joint training.

4.3.2. Language Modeling Weights ρ_1 and ρ_2

The hyperparameters ρ_1 and ρ_2 are the weights for the LM loss on external and internal data respectively. We conduct sensitivity analysis over ρ_1 and ρ_2. Results are shown in Table 5 and Table 6.

For ρ_1, we find that high-resource tasks such as Helpfulness perform better with a smaller ρ_1 (i.e., Helpfulness achieves best when $\rho_1 = 1$) while low-resource tasks such as SciERC and ChemProt achieve their best when ρ_1 is large (i.e., both tasks use $\rho_1 = 999$). This is in line with conclusions in Section 4.3.1 that low-resource tasks rely more on external data. In addition, removing task data and only using external data for training (i.e., $\rho_1 = \#\mathcal{C}_{\text{BERT}}$), it performs worse than when incorporating the task data, proving the indispensability of small task data.

Results in Table 6 show that language modeling on internal data is necessary: consistently better results are achieved when ρ_2 is non-zero. Based on our observations, competitive performance can be achieved when ρ_2 is set to a proper value between 20 and 1000.



(a) TLM (*Medium scale*) (b) BERT-Base (c) RoBERTa-Base

Figure 3. Attention visualization of TLM and pretraining-finetuning baselines, with "[CLS] crystallographic comparison with the structurally related. [SEP]" from ChemProt as the input. The positional heads (Voita et al., 2019) are highlighted in red boxes and vertical heads (Kovaleva et al., 2019) are masked in gray.

Table 5. Results on the development set with different weights on external data (i.e., ρ_1). We assign different values for ρ_1 for the first stage, and report the final performance after two-stage joint learning. "Ext only" means using only external data for training (i.e., $\rho_1 = \infty$). Helpfulness is a high-resource task, and the others are low-resource ones. For all tasks, we fix $\rho_2 = 20$.

	Helpfulness	SciERC	ChemProt
ρ_1 =1	**71.02**±0.51	80.72±3.32	73.27±0.30
ρ_1 =3	70.41±0.52	80.01±0.72	79.43±1.03
ρ_1 =99	69.56±0.23	84.95±0.57	83.30±0.30
ρ_1 =999	69.35±0.72	**86.07**±0.48	**83.64**±0.26
Ext only	69.76±0.50	85.66±1.58	82.50±0.27

Table 6. Results on the development set with different language modeling weights on internal data (i.e., ρ_2). Here we set $\rho_1 = 1000$ for SciERC and ChemProt, and $\rho_1 = 4$ for RCT

	RCT	SciERC	ChemProt
ρ_2 =0	85.75±0.11	83.31±0.88	83.41±0.33
ρ_2 =20	88.08±0.02	**86.07**±0.48	83.64±0.26
ρ_2 =100	**88.16**±0.15	85.48±1.01	**83.77**±0.77
ρ_2 =1000	88.02±0.04	85.29±1.86	83.63±0.90

4.3.3. Second Stage of Training

TLM contains two training stages—first training on all three terms combined and then finetuning using only the task objective. To validate the effectiveness of the second stage of TLM, we compare the performance of two-stage training against using only stage one. Results are shown in Table 7. We find that removing the second stage hurts the ultimate performance consistently, proving its indispensability. Particularly, the second stage has much more influence on low-resource tasks (with a huge decrease of 19.37 points on ACL-ARC and 14.34 points on ChemProt) than on high-resource tasks (with a performance decrease of 0.53 points on AGNews and 2.17 points on IMDB).

Table 7. Results on the development set of two-stage training and one-stage training (removing stage 2).

	AGNews	IMDB	ChemProt	ACL-ARC
two-stage	**94.51**	**94.40**	**83.64**	**76.37**
wo/ stage-2	93.98 ↓	92.23↓	69.30↓	57.00↓

Table 8. Results of adding MLM loss on task data into PLM. Results are based on RoBERTa-base.

Model	AGNews	Hyp.	Help.	IMDB	ACL.	SciERC	Chem.	RCT	Avg.
PLM	94.02	93.53	70.45	95.43	68.34	81.35	82.60	87.23	84.12
PLM+MLM	93.83	93.50	71.12	95.54	70.94	80.90	82.53	87.09	84.43
TLM	93.96	94.05	70.90	93.97	72.37	81.88	83.24	87.28	84.71

4.3.4. MLM Loss on Task Data

During the first training stage, TLM uses masked language loss on task data. To examine whether the trick attains the main improvements, we compare results on PLM, PLM with additional MLM loss on task data (PLM+MLM) and TLM. Results in Table 8 show that adding MLM loss on task data into PLM has only marginal gains and does not affect the main conclusion of the paper. In addition, results in Table 3 and Table 4 show that retrieving appropriate relevant data is also essential for the performance of TLM.

4.4. Analysis

4.4.1. Attention Weight Visualization

We also study the difference between the model behaviors of TLM and pretraining-finetuning by visualizing their attention weights. Voita et al. (2019) found that a specific kind of heads, referred to as "positional head" in which at least 90% of the maximum attention weights are assigned to adjacent tokens, have vital contributions to final predictions of the model. Another sort of heads we are interested in are those in which most maximum attention weights are assigned to [CLS],[SEP] or the period token("."), which potentially encode less semantic or syntactic information (Kovaleva et al., 2019). In our experiments, if more than 90% maximum weights are assigned to [CLS], [SEP] or the period token, we categorize this head as a "vertical head". Results in Figure 3 show that on the task ChemProt, more

Table 9. Examples of retrieved data. The overlap between queries and retrieved data are highlighted in blue in italics.

Task	Task Data as Query	Retrieved General Data
Hyp.	"A Republican student association at *San Diego State University (SDSU)* is facing backlash for sending a letter demanding Muslim students condemn last week's terror attacks in Barcelona. ... "	**Example 1:** "...The *SDSU* Aztecs intercollegiate water polo, swimming and diving teams are based at the Aztec Aquaplex..." **Example 2:** The Daily Aztec is a not-for-profit, independent student newspaper serving *San Diego State University (SDSU)* and the surrounding College Area in San Diego, California. ...
Help.	*Poor Quality*. The case broke after dropping it on the tile floor. ...	**Example 1:** ...a collaborative algorithm will be able to recommend it, the *quality* of those recommendations will be *poor*. ... **Example 2:** ... Books that're of *poor quality* will quickly cease to sell. ...
ChemProt	FCEO significantly inhibited nitric oxide (NO) and prostaglandin E2 (PGE2) by suppressing the protein expression of *inducible nitric oxide synthase (iNOS)* and *cyclooxygenase (COX)-2*, respectively.	**Example 1:** ... They regulate the development of sperm by controlling their cell division and survival. Other immune factors found in the testis include the enzyme *inducible nitric oxide synthase (iNOS)* ... **Example 2:** These compounds have been shown "in vivo" to reduce two proteins that mediate inflammation, *cyclooxygenase-2 (COX-2)* and *inducible nitric oxide synthase (iNOS)*. ...
SciERC	*Image* sequence *processing* techniques are used to study exchange , growth , and transport processes and to tackle key questions in environmental physics and biology.	**Example 1:** ... Driving forces in signal *processing* for data parallelism are video encoding, *image* and graphics *processing*, wireless communications to name a few. **Example 2:** They have applications in many disciplines such as biology, chemistry, ecology, neuroscience, physics, *image processing*, ...

Table 10. Evaluation results on the GLUE benchmark. Model size, data, and FLOPs are similar to Table 2.

Method	CoLA	RTE	STS-B	MRPC	QQP	SST-2	QNLI	MNLI	Avg.
BERT-Base	59.3	68.2	89.8/89.4	86.0/90.5	91.1/88.1	92.5	91.8	84.5/84.5	82.97
TLM *(small-scale)*	59.8	67.1	89.0/88.7	86.8/90.4	91.1/88.1	92.2	91.0	83.3/83.9	82.60

positional heads and less vertical heads are observed in TLM than in PLMs. We also observe similar patterns across various tasks (see Appendix C). These phenomena suggest that TLM learns different (probably more informative) attention patterns compared to PLMs.

4.4.2. Case Study of Retrieved Data

We have shown several casess of retrieved data in Table 9. TLM retrieves relevant data from a general corpus using BM25 (Robertson & Zaragoza, 2009). Since BM25 is based on sparse features, it focuses more on lexical similarity instead of semantic similarity. This might be specifically beneficial for professional domains, e.g., SciERC for computer science and ChemProt for biomedical science), since there are a large number of proper nouns in these domains. For other domains, it seems BM25 also performs reasonably well for retrieving related documents.

4.5. Results on More Datasets

So far we have followed the setting of Gururangan et al. (2020) and adopted the datasets therein. In this section, we additionally experiment with the GLUE benchmark (Wang et al., 2018) following the setting of BERT (Devlin et al., 2019) to examine the performance of TLM on a more diverse set of tasks including natural language understanding. We follow the *small-scale* setting in Section 4.2 in terms of model size, data, and FLOPs. Results in Table 10 show that given the advantages in efficiency, the average performance of TLM is comparable to BERT across 8 tasks, which is consistent with our previous findings and demonstrates the effectiveness of TLM.

5. Conclusions

In this paper, we have proposed a simple, efficient, pretraining-free framework, TLM. The core idea is to only use a tiny, task-relevant subset of the general corpus for language model training. Our experiments show that TLM achieves results similar to or even better than PLMs, with a reduction of training FLOPs by two orders of magnitude. TLM opens the possibility of reducing the heavy reliance on large-scale PLMs and training a model from scratch in an efficient manner, while not hurting the overall performance. We hope TLM will contribute to democratizing NLP and

expediting its development by allowing most researchers to freely explore the architectures, loss functions, algorithms, and other design choices in the neighborhood of a state-of-the-art solution.

As discussed in Section 3.2, there are several potential directions for future work. It will be interesting to study how to use TLM to match the performance even larger-scale PLMs. Moreover, further extending and improving TLM for few-shot and zero-shot learning is a crucial problem.

References

Beltagy, I., Lo, K., and Cohan, A. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3613–3618, Hongkong, China, 2019. Association for Computational Linguistics.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020.

Chen, X., Cheng, Y., Wang, S., Gan, Z., Wang, Z., and Liu, J. EarlyBERT: Efficient BERT training via early-bird lottery tickets. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

de Wynter, A. and Perry, D. J. Optimal subarchitecture extraction for BERT. *CoRR*, abs/2010.10499, 2020.

Dernoncourt, F. and Lee, J. Y. Pubmed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *IJCNLP(2)*, pp. 308–313. Asian Federation of Natural Language Processing, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

Gordon, M., Duh, K., and Andrews, N. Compressing bert: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pp. 143–155. Association for Computational Linguistics, 2020.

Gu, Y., Zhang, Z., Wang, X., Liu, Z., and Sun, M. Train no evil: Selective masking for task-guided pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6966–6974, Online, November 2020. Association for Computational Linguistics.

Gururangan, S., Marasovic, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, pp. 8342–8360. Association for Computational Linguistics, 2020.

He, P., Liu, X., Gao, J., and Chen, W. DeBERTa: Decoding-enhanced bert with disentangled attention. In *2021 International Conference on Learning Representations*, May 2021.

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., and Liu, Q. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174, Online, November 2020. Association for Computational Linguistics.

Jurgens, D., Kumar, S., Hoover, R., McFarland, D. A., and Jurafsky, D. Measuring the evolution of a scientific field through citation frames. *Trans. Assoc. Comput. Linguistics*, 6:391–406, 2018.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics.

Kiesel, J., Mestre, M., Shukla, R., Vincent, E., Adineh, P., Corney, D. P. A., Stein, B., and Potthast, M. Semeval-2019 task 4: Hyperpartisan news detection. In *SemEval@NAACL-HLT*, pp. 829–839. Association for Computational Linguistics, 2019.

Kim, S., Gholami, A., Yao, Z., Mahoney, M. W., and Keutzer, K. I-BERT: integer-only BERT quantization. *International Conference on Machine Learning*, 2021.

Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint*

Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 4365–4374, Hongkong, China, 2019. Association for Computational Linguistics.

Kringelum, J., Kjærulff, S. K., Brunak, S., Lund, O., Oprea, T. I., and Taboureau, O. Chemprot-3.0: a global chemical biology diseases mapping. *Database J. Biol. Databases Curation*, 2016, 2016.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Li, J., Zhang, Z., Zhao, H., Zhou, X., and Zhou, X. Task-specific objectives of pre-trained language models for dialogue adaptation. *arXiv preprint arXiv:2009.04984*, 2020.

Liu, W., Zhou, P., Wang, Z., Zhao, Z., Deng, H., and Ju, Q. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6035–6044, Online, July 2020. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized bert pretraining approach, 2019.

Luan, Y., He, L., Ostendorf, M., and Hajishirzi, H. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*, pp. 3219–3232. Association for Computational Linguistics, 2018.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *ACL*, pp. 142–150. The Association for Computer Linguistics, 2011.

McAuley, J. J., Targett, C., Shi, Q., and van den Hengel, A. Image-based recommendations on styles and substitutes. In *SIGIR*, pp. 43–52. ACM, 2015.

Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Qiao, S., Shen, W., Zhang, Z., Wang, B., and Yuille, A. Deep co-training for semi-supervised image recognition. *Lecture Notes in Computer Science*, pp. 142–159, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-01267-0_9. URL http://dx.doi.org/10.1007/978-3-030-01267-0_9.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2018. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.

Robertson, S. E. and Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

Rose, S., Engel, D., Cramer, N., and Cowley, W. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20, 2010.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Q-BERT: Hessian based ultra low precision quantization of BERT. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 8815–8821, Apr. 2020.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-LM: training multi-billion parameter language models using model parallelism, 2019.

So, D. R., Mańke, W., Liu, H., Dai, Z., Shazeer, N., and Le, Q. V. Primer: Searching for efficient transformers for language modeling, 2021.

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., and Zhou, D. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2158–2170, Online, July 2020. Association for Computational Linguistics.

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. Scale efficiently: Insights from pre-training and fine-tuning transformers, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads

do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics.

Wang, M., Min, F., Zhang, Z.-H., and Wu, Y.-X. Active learning through density clustering. *Expert Systems with Applications*, 85:305–317, 2017. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2017.05.046. URL https://www.sciencedirect.com/science/article/pii/S095741741730369X.

Wang, Z., Wohlwend, J., and Lei, T. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6151–6162, Online, November 2020. Association for Computational Linguistics.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Xu, C., Zhou, W., Ge, T., Wei, F., and Zhou, M. BERT-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7859–7869, Online, November 2020. Association for Computational Linguistics.

Yang, L., Wang, Y., Gao, M., Shrivastava, A., Weinberger, K. Q., Chao, W.-L., and Lim, S.-N. Deep co-training with task decomposition for semi-supervised domain adaptation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2021. doi: 10.1109/iccv48922.2021.00878. URL http://dx.doi.org/10.1109/iccv48922.2021.00878.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

You, Y., Li, J., Reddi, S. J., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations (ICLR 2020)*, Addis Ababa, Ethiopia, 2020. OpenReview.net.

Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. Q8BERT: quantized 8bit BERT. *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, Dec 2019.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pp. 649–657. Curran Associates, Inc., 2015.

Zhang, X., Shapiro, P., Kumar, G., McNamee, P., Carpuat, M., and Duh, K. Curriculum learning for domain adaptation in neural machine translation. *Proceedings of the 2019 Conference of the North*, 2019.

Zhong, R., Lee, K., Zhang, Z., and Klein, D. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 2856–2878, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Zhu, J., Wang, H., Tsou, B. K., and Ma, M. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1323–1331, 2010. doi: 10.1109/TASL.2009.2033421.

A. Comparison to Domain Adaptation

Our work is different from domain adaptation such as Gururangan et al. (2020). While domain adaptation aims to address how to effectively adapt a pretrained LM into one domain-specific task with sufficient domain data, this work targets to provide a method that is general enough to solve any task without domain data. Nevertheless, we still compare TLM with (Gururangan et al., 2020) as Table A.2 shows. We hope to figure out that, under the harsh but practical condition that no domain data is accessible, whether our proposed framework TLM can still match or even outperform the traditional domain adaptation methods with large pretrained language models as well as domain data.

From results in Table A.2, we have observations:

1. We reproduced the RoBERTa-Base results using the hyper-parameters reported by Gururangan et al. (2020) as well as our own hyper-parameters. Results show that the baseline RoBERTa-Base results are underestimated in the paper with a gap of around 3 points. We list our hyper-parameters for fine-tuning RoBERTa in Table A.1.

2. We also reproduced the DAPT+TAPT results using our own hyper-paraemters. Results show that DAPT+TAPT with new hyper-parameters also performs slightly better than it was reported by Gururangan et al. (2020).

3. From the perspective of total training computes (FLOPs), DAPT+TAPT consumes a comparable FLOPs with TLM (*large-scale*), and TLM (*large-scale*) achieved comparable results with DAPT+TAPT (i.e., 85.70 vs 85.57). However, from the perspective of data usage, DAPT+TAPT uses large amounts of domain data, the amount of which for each domain almost equals the amount of BERT total training corpus. TLM does not rely on it.

Table A.1. Comparison between the hyperparameters for fine-tuning from our implementation and from Gururangan et al. (2020).

Hyper-parameters	**Ours**	**Reported**
Epochs	-	3 or 10
Training steps	3e4	-
Patience	-	3
Learning rate	2e-5	2e-5
Batch size	32	16
Max. grad. norm	-	1
Weight decay	0	0.1

Table A.2. Comparison results of TLM and Gururangan et al. (2020).

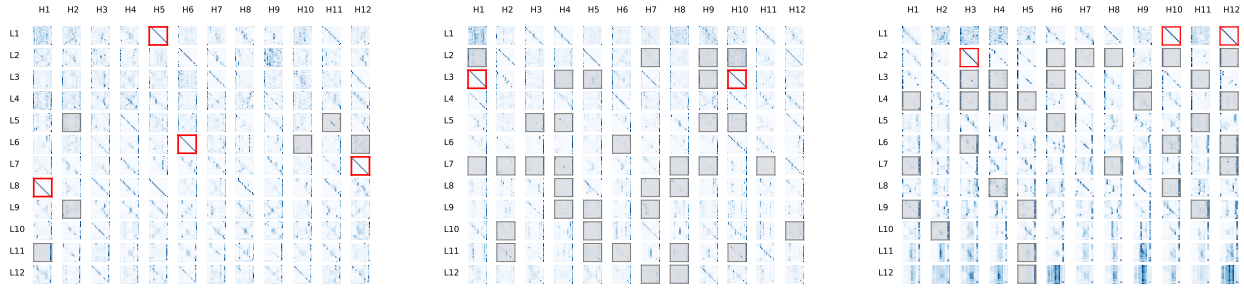
	AGNews	**Hyp.**	**Help.**	**IMDB**	**ACL.**	**SciERC**	**Chem.**	**RCT**	**Avg.**
RoBERTa-Base[1]	93.90±0.20	86.60±0.90	65.10±3.40	95.00±0.20	63.00±5.80	77.30±1.90	81.90±1.00	87.20±0.10	81.25
RoBERTa-Base[2]	93.97±0.13	88.50±4.18	67.45±0.49	95.43±0.07	63.87±1.24	79.97±1.29	81.50±0.94	87.26±0.08	82.24
RoBERTa-Base[3]	94.02±0.15	93.53±1.61	70.45±0.24	95.43±0.16	68.34±7.27	81.35±0.63	82.60±0.53	87.23±0.09	84.12
DAPT[1]	93.90±0.20	88.20±5.90	66.50±1.40	95.40±0.10	75.40±2.50	80.80±1.50	84.20±0.20	87.60±0.10	84.00
DAPT+TAPT[1]	94.60±0.10	90.00±6.60	68.70±1.80	95.60±0.10	75.60±3.80	81.30±1.80	84.40±0.40	87.80±0.10	84.75
DAPT+TAPT[3]	94.07±0.07	93.59±0.00	71.44±0.99	95.65±0.14	75.62±1.77	82.06±0.90	84.45±0.68	87.67±0.11	85.57
TLM *(large-scale)*	94.32±0.07	95.16±0.00	72.49±0.33	95.77±0.24	72.19±1.72	83.29±0.95	85.12±0.85	87.50±0.12	85.74

[1] Results reported by Gururangan et al. (2020)
[2] Our reproduced results with the hyper-parameters reported by Gururangan et al. (2020)
[3] Results obtained by our own hyper-parameters

Table B.1. Detailed hyper-parameters for TLM of different scales for each task.

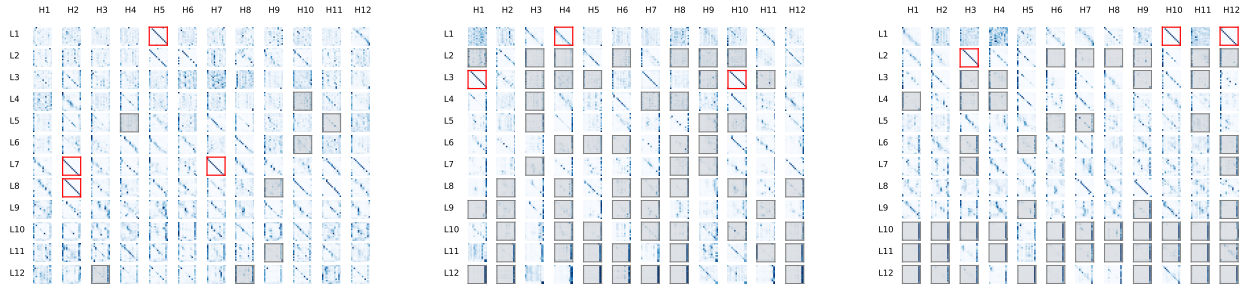
	Hyper-Parameters	**AGNews**	**Hyp.**	**Help.**	**IMDB**	**ACL.**	**SciERC**	**Chem.**	**RCT**
Small Scale	Top-K	50	5000	50	500	5000	5000	5000	50
	ρ_1	1	99	1	19	999	999	999	3
	ρ_2	100	20	100	100	100	20	20	20
	Source Corpus[2]	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$
	Training Data Size[3]	1.1GB	0.2GB	0.5GB	0.9GB	1.5GB	1.6GB	0.7GB	0.8GB
	Training Steps	1E5	5E4	1.5E5	1.5E5	1.5E5	1.5E5	1.5E5	1E5
	Batch Size	256	256	256	256	256	256	256	256
	Sequence Length	128	128	128	128[1]	128	128	128	128
Medium Scale	Top-K	50	5000	50	500	5000	5000	5000	50
	ρ_1	3	99	1	99	999	999	999	3
	ρ_2	100	100	1000	100	20	20	100	100
	Source Corpus[2]	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$	$\mathcal{C}_{\text{BERT}}$
	Training Data Size[3]	1.1GB	0.2GB	0.5GB	3.3GB	1.5GB	1.6GB	0.7GB	0.8GB
	Training Steps	3E5	1E5	3E5	3E5	3E5	3E5	3E5	3E5
	Batch Size	256	256	256	256	256	256	256	256
	Sequence Length	128	128	128	512	128	128	128	128
Large Scale	Top-K	100	10000	100	1000	10000	10000	10000	100
	ρ_1	3	499	7	99	1999	1999	1999	7
	ρ_2	100	20	100	1000	20	20	20	100
	Source Corpus[2]	$\mathcal{C}_{\text{RoBERTa}}$	$\mathcal{C}_{\text{RoBERTa}}$	$\mathcal{C}_{\text{RoBERTa}}$	$\mathcal{C}_{\text{RoBERTa}}$	$\mathcal{C}_{\text{RoBERTa}}$	$\mathcal{C}_{\text{RoBERTa}}$	$\mathcal{C}_{\text{RoBERTa}}$	$\mathcal{C}_{\text{RoBERTa}}$
	Training Data Size[3]	3.1GB	0.9GB	1.7GB	11GB	3.5GB	4.2GB	2.5GB	2.2GB
	Training Steps	5E5	3E5	5E5	5E5	5E5	3E5	5E5	5E5
	Batch Size	256	512	512	512	512	512	256	256
	Sequence Length	128	128	128	512	128	128	128	128

[1] At a small scale on IMDB, we use a sequence length of 512 for internal data and a sequence length of 128 for external data.
[2] $\mathcal{C}_{\text{BERT}}$ and $\mathcal{C}_{\text{RoBERTa}}$ are our collected corpus that respectively match the original training corpus of BERT and RoBERTa.
[3] TLM only uses a tiny subset of the source general corpus for training. We list the data size that are actually used for TLM training.



(a) TLM (*Medium scale*) (b) BERT-Base (c) RoBERTa-Base

Figure C.1. task: RCT ; input: "[CLS] twenty-eight individuals from outpatient physiotherapy departments were randomized. [SEP]"



(a) TLM (b) BERT-Base (c) RoBERTa-Base

Figure C.2. task: SciERC ; input: "[CLS] multi-view constraints associated with groups of patches are combined. [SEP]"

B. Detailed Experiment Settings

Table B.1 lists the detailed hyperparameters for TLM at stage 1 of different scales for each task. At small and medium scales, for tasks with less than 5K training examples (HyperPartisan, ChemProt, SciERC, ACL-ARC), we set $K = 5000$; for tasks with more than 100K training examples (RCT, AGNews, Helpfulness), we set $K = 50$, for the rest of the tasks (IMDB), we set $K = 500$. At the large scale, K is doubled for each task. At each scale on every task, we conduct grid search for $\rho_1 \in \{1, 3, 7, 19, 99, 499, 999, 1999\}$ and $\rho_2 \in \{20, 100, 1000\}$, and adjust training steps, batch size and sequence length to minimize the training cost while preserving competitive performance. We observe that for almost all the tasks, the larger the training scale, the more reliance on external data, indicated by the increasing trend of ρ_1 and ρ_2 as the total training tokens goes up.

C. Attention visualization on other tasks

Besides ChemProt (Figure 3), we also experimented on RCT (Figure C.1) and SciERC (Figure C.2) to get attention visualizations. We find TLM consistently contains more positional heads (in red box) and less vertical heads (in gray mask). These results reveal that the aforementioned pattern generally holds for TLM.

The Probabilistic Relevance Framework: BM25 and Beyond

Article *in* Foundations and Trends® in Information Retrieval · January 2009

DOI: 10.1561/1500000019 · Source: DBLP

2 authors:

Stephen E. Robertson
University College London
258 PUBLICATIONS **20,692** CITATIONS
SEE PROFILE

Hugo Zaragoza
Websays
109 PUBLICATIONS **5,070** CITATIONS
SEE PROFILE

Foundations and Trends® in
Information Retrieval
Vol. 3, No. 4 (2009) 333–389

DOI: 10.1561/1500000019

now
the essence of knowledge

The Probabilistic Relevance Framework: BM25 and Beyond

By Stephen Robertson and Hugo Zaragoza

Contents

Foundations and Trends® in
Information Retrieval
Vol. 3, No. 4 (2009) 333–389

DOI: 10.1561/1500000019

now
the essence of knowledge

The Probabilistic Relevance Framework: BM25 and Beyond

Stephen Robertson[1] and Hugo Zaragoza[2]

[1] *Microsoft Research, 7 J J Thomson Avenue, Cambridge CB3 0FB, UK*
ser@microsoft.com
[2] *Yahoo! Research, Av. Diagonal 177, Barcelona 08028, Spain*
hugoz@yahoo-inc.com

Abstract

The Probabilistic Relevance Framework (PRF) is a formal framework for document retrieval, grounded in work done in the 1970–1980s, which led to the development of one of the most successful text-retrieval algorithms, BM25. In recent years, research in the PRF has yielded new retrieval models capable of taking into account document meta-data (especially structure and link-graph information). Again, this has led to one of the most successful Web-search and corporate-search algorithms, BM25F. This work presents the PRF from a conceptual point of view, describing the probabilistic modelling assumptions behind the framework and the different ranking algorithms that result from its application: the binary independence model, relevance feedback models, BM25 and BM25F. It also discusses the relation between the PRF and other statistical models for IR, and covers some related topics, such as the use of non-textual features, and parameter optimisation for models with free parameters.

1

Introduction

This monograph addresses the *classical* probabilistic model of information retrieval. The model is characterised by including a specific notion of relevance, an explicit variable associated with a query–document pair, normally hidden in the sense of *not observable*. The model revolves around the notion of estimating a probability of relevance for each pair, and ranking documents in relation to a given query in descending order of probability of relevance. The best-known instantiation of the model is the BM25 term-weighting and document-scoring function.

The model has been developed in stages over a period of about 30 years, with a precursor in 1960. A few of the main references are as follows: [30, 44, 46, 50, 52, 53, 58]; other surveys of a range of probabilistic approaches include [14, 17]. Some more detailed references are given below.

There are a number of later developments of IR models which are also probabilistic but which differ considerably from the models developed here — specifically and notably the *language model* (LM) approach [24, 26, 33] and the *divergence from randomness* (DFR) models [2]. For this reason we refer to the family of models developed here as the *Probabilistic Relevance Framework* (PRF), emphasising the

importance of the relevance variable in the development of the models. We do not cover the development of other probabilistic models in the present survey, but some points of comparison are made.

This is not primarily an experimental survey; throughout, assertions will be made about techniques which are said to work well. In general such statements derive from experimental results, many experiments by many people over a long period, which will not in general be fully referenced. The emphasis is on the theoretical development of the methods, the logic and assumptions behind the models.

The survey is organised as follows. In Section 2 we develop the most generic retrieval model, which subsumes a number of specific instantiations developed in Section 3. In Section 4 we discuss the similarities and differences with other retrieval frameworks. Finally in Section 5 we give an overview of optimisation techniques we have used to tune the different parameters in the models and Section 6 concludes the survey.

2

Development of the Basic Model

2.1 Information Needs and Queries

We start from a notion of information need. We take a query to be a representation (not necessarily very good or very complete) of an individual user's information need or perhaps search intent. We take *relevance* to mean the relevance of a document to the information need, as judged by the user. We make no specific assumption about the conceptual nature of relevance; in particular, we do *not* assume relevance to be topical in nature.[1] We do, however, makes some assumptions about the formal status of the relevance variable, given below, which might be taken to imply some assumptions about its conceptual nature.

[1] The notion of *topic*, as used in TREC, is somewhat akin to *information need*, or at least to a more detailed and complete representation and specification of an information need. The use of the term 'topic' for this purpose is a little unfortunate, as it seems to imply some kind of topical nature of relevance. However, we also note that the widespread use of the term following TREC also includes some examples of non-topical 'topics', for example the home-page topics used in the TREC Web track [59].

2.2 Binary Relevance

The assumptions about relevance are as follows:

1. Relevance is assumed to be a property of the document given information need only, assessable without reference to other documents; and
2. The relevance property is assumed to be binary.

Either of these assumptions is at the least arguable. We might easily imagine situations in which one document's relevance can only be perceived by the user in the context of another document, for example. Regarding the binary property, many recent experimental studies have preferred a graded notion of relevance. One might go further and suggest that different documents may be relevant to a need in different ways, not just to different degrees. However, we emphasise that we consider the situation of a single information need (rather than the multiple needs or intents that might be represented by a single text query). If we take *relevant* to mean 'the user would like to be pointed to this document', the binary notion is at least moderately plausible.

2.3 The Probability Ranking Principle

Given that an information retrieval system cannot *know* the values of the relevance property of each document, we assume that the information available to the system is at best probabilistic. That is, the known (to the system) properties of the document and the query may provide probabilistic or statistical evidence as to the relevance of that document to the underlying need. Potentially these properties may be rich and include a variety of different kinds of evidence; the only information assumed to be absent is the actual relevance property itself. Given whatever information is available, the system may make some statistical statement about the possible value of the relevance property.

Given a binary document-by-document relevance property, then this statistical information may be completely encapsulated in a *probability* of relevance. The probability of relevance of a given document to a

given query plays a central role in the present theory. We can in fact make a general statement about this:

> If retrieved documents are ordered by decreasing probability of relevance on the data available, then the system's effectiveness is the best that can be obtained for the data.

This is a statement of the *Probability Ranking Principle* (PRP), taken from [52], an abbreviated version of the one in [40]. The PRP can be justified under some further assumptions, for a range of specific measures of effectiveness. It is not, however, completely general; one can also construct counter-examples. But these counter-examples depend on probabilities defined over a population of users. The PRP is safe for the case considered: an individual user. It will be assumed for the remainder of this work.

2.4 Some Notation

In this section, we introduce some of the notation that will be used throughout the survey. The notation assumes a single query q representing a single information need. The first symbol is used to indicate that two functions are equivalent as ranking functions.

$$\text{rank equivalence:} \quad \propto_q \quad \text{e.g. } g() \propto_q h()$$

In developing the model, from the probability of relevance of a document to a term-weighting and document-scoring function, we make frequent use of transformations which thus preserve rank order. Such a transformation (in a document-scoring function, say) may be linear or non-linear, but must be strictly monotonic, so that if documents are ranked by the transformed function, they will be in the same rank order as if they had been ranked by the original function.

The property of relevance is represented by a random variable *Rel* with two possible values:

$$\text{relevance } \mathit{Rel}: \quad \text{rel}, \overline{\text{rel}} \quad \text{(relevant or not)}$$

As discussed above, we assume that relevance is a binary property of a document (given an information need). We will use the short-hand notation $P(\text{rel}\,|\,d,q)$ to denote $P(Rel = \text{rel}\,|\,d,q)$.

For documents and queries, we generally assume a bag or set of words model. We have a vocabulary of terms indexed into the set $\mathbf{V}$, each of which may be present or absent (in the set of words model) or may be present with some frequency (bag of words model). In either case the objects (documents or queries) may be represented as vectors over the space defined by the vocabulary. Thus a document is:

$$\text{document:} \quad d := (tf_1, \ldots, tf_{|\mathbf{V}|}),$$

where tf_i normally represents the frequency of term t_i in the document. We will also need to distinguish between the random variable TF_i and its observed value in a document, tf_i. The random variable will usually refer to the term frequencies of a given term in the vocabulary. However, the formulation of the basic model is somewhat more general than this, and can accommodate any discrete variable as a feature (e.g., any discrete property or attribute of the document). Thus we can re-interpret tf as simply representing presence or absence of a term; this is the basis of Section 3.1 below. Continuous variables can also be accommodated by replacing probability mass functions TF_i (of discrete variables) by probability density functions (of continuous variables); although we do not present a formal development of the model with continuous variables, we will use this fact in Section 3.7 to introduce some non-textual continuous features into the model.

A query is represented in two different ways. In the first, it is treated similarly as a vector:

$$\text{query: } q \ := (qtf_1, \ldots, qtf_{|\mathbf{V}|})$$

Here again the components qtf_i may represent term frequencies in the query, or may represent a binary presence or absence feature.

Throughout this survey we will need to sum or multiply variables of terms present in the query (i.e., with $qtf_i > 0$). For this purpose we define the set of indices:

$$\text{query terms:} \quad \mathbf{q} := \{i \,|\, qtf_i > 0\} \quad \text{(indices of terms in the query)}$$

2.4.1 Preview of the Development

The following provides an overview of the development of the basic model; the individual steps are discussed below. The main point to note is that we start with the very general PRP, and end up with an explicit document scoring function, composed of a simple sum of weights for the individual query terms present in the document.

$$P(\text{rel}\,|\,d,q) \propto_q \frac{P(\text{rel}\,|\,d,q)}{P(\overline{\text{rel}}\,|\,d,q)} \tag{2.1}$$

$$= \frac{P(d\,|\,\text{rel},q)}{P(d\,|\,\overline{\text{rel}},q)} \frac{P(\text{rel}\,|\,q)}{P(\overline{\text{rel}}\,|\,q)} \tag{2.2}$$

$$\propto_q \frac{P(d\,|\,\text{rel},q)}{P(d\,|\,\overline{\text{rel}},q)} \tag{2.3}$$

$$\approx \prod_{i\in\mathbf{V}} \frac{P(TF_i = tf_i\,|\,\text{rel},q)}{P(TF_i = tf_i\,|\,\overline{\text{rel}},q)} \tag{2.4}$$

$$\approx \prod_{i\in\mathbf{q}} \frac{P(TF_i = tf_i\,|\,\text{rel})}{P(TF_i = tf_i\,|\,\overline{\text{rel}})} \tag{2.5}$$

$$\propto_q \sum_{\mathbf{q}} \log \frac{P(TF_i = tf_i\,|\,\text{rel})}{P(TF_i = tf_i\,|\,\overline{\text{rel}})} \tag{2.6}$$

$$= \sum_{\mathbf{q}} U_i(tf_i) \tag{2.7}$$

$$= \sum_{\mathbf{q},tf_i>0} U_i(tf_i) + \sum_{\mathbf{q},tf_i=0} U_i(0) - \sum_{\mathbf{q},tf_i>0} U_i(0) + \sum_{\mathbf{q},tf_i>0} U_i(0) \tag{2.8}$$

$$= \sum_{\mathbf{q},tf_i>0} (U_i(tf_i) - U_i(0)) + \sum_{\mathbf{q}} U_i(0) \tag{2.9}$$

$$\propto_q \sum_{\mathbf{q},tf_i>0} (U_i(tf_i) - U_i(0)) \tag{2.10}$$

$$= \sum_{\mathbf{q},tf_i>0} w_i \tag{2.11}$$

2.4.2 Details

We start with the probability of relevance of a given document and query.

The first three steps are simple algebraic transformations. In Step (2.1), we simply replace the probability by an odds-ratio.[2] In Step (2.2), we perform Bayesian inversions on both numerator and denominator. In Step (2.3), we drop the second component which is independent of the document, and therefore does not affect the ranking of the documents.

In Step (2.4) (term independence assumption), we expand each probability as a product over the terms of the vocabulary. This step depends on an assumption of statistical independence between terms — actually a pair of such assumptions, for the relevant and non-relevant cases, respectively. Note that we are *not* assuming unconditional term independence, but a weaker form, namely *conditional* independence.[3] This is clearly a much more arguable step: in general terms will not be statistically independent in this fashion. The obvious reason for taking this step is to lead us to a simple and tractable (even if approximate) scoring function. However, we may make some further arguments to support this step:

1. Models of this type in other domains, known as Naïve Bayes models, are well known to be remarkably good, simple and robust, despite significant deviations from independence. Experimental evidence in IR provides some support for this general statement.
2. The pair of assumptions is not in general equivalent to a blanket assumption of independence between terms over the whole collection. On the contrary, for a pair of query terms, both statistically correlated with relevance, the pair of assumptions predict a positive association between the two terms over the whole collection. In fact we often observe such a positive association. In effect the model says that this

[2] This transformation is order preserving; the odds-ratio of p is $\frac{p}{1-p}$, and this function is a monotonous increasing function of $p \in [0,1)$.

[3] $P(ab|c) = P(a|c)P(b|c)$.

positive association is induced by relevance to the query.[4] This is clearly an over-simplification, but perhaps not such a bad one.

3. Cooper [11] has demonstrated that we can arrive at the same transformation on the basis of a weaker assumption, called *linked dependence*.[5] This is essentially that the degree of statistical association between the terms is the same in the relevant as in the non-relevant subsets. Again, this theoretical result may help to explain the robustness of such a model.

We may represent the independence assumptions by means of a graphical model, as in Figure 2.1. This diagram shows how the term frequencies $(tf_i)_{i \in \mathbf{V}}$ are assumed to be generated as stochastic observations dependant only on the state of the relevance variable Rel (a full explanation of graphical model diagrams is outside the scope of this paper, we refer the reader to [6]).

In step (2.5) (query-terms assumption), we restrict ourselves to the query terms only: in effect, we assume that for any non-query term, the ratio of conditional probabilities is constant and equal to one.[6] This might seem a drastic assumption (that no non-query terms have any association with relevance); however, this is not quite the explanation of the step. We have to consider the question of what information we

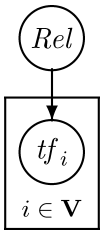


Fig. 2.1 Graphical model indicating basic independence assumptions.

[4] If $P(a\,|\,c) > P(a\,|\,\overline{c})$ and similarly for b, then it follows that $P(ab) > P(a)P(b)$ even under the conditional independence assumptions made.

[5] $\frac{P(a,b\,|\,c)}{P(a,b\,|\,\overline{c})} = \frac{P(a\,|\,c)}{P(a\,|\,\overline{c})} \frac{P(b\,|\,c)}{P(b\,|\,\overline{c})}$.

[6] Note that if the ratio is constant, then it must be equal to one; otherwise we would have one of the probability distribution with probabilities always higher than another one, which is impossible since they both need to sum to 1.

have on which to base an estimate of the probability of relevance. It is a reasonable prior assumption, and turns out to be a good one, that in general query terms are (positively) correlated with relevance. However, we can make no such assumption about non-query terms (a term relating to a completely different topic could well be *negatively* correlated with relevance). In the absence of any link to the query, the obvious vague prior assumption about a random vocabulary term must be that it is not correlated with relevance to this query. Later we will consider the issue of query expansion, adding new terms to the query, when we have evidence to link a non-query term with relevance to the query.

Again, we can illustrate the assumptions by means of a graphical model, as in Figure 2.2. This diagram shows that in this model the relevance variable only affects terms in the query.

Starting in Step (2.6) we use the following short-hand notation: under the summation operator we will write $\mathbf{q}$ (the starting set of values for i) followed by conditions that i should satisfy. For example: $\sum_{\mathbf{q}}$ should be read as $\sum_{i \in \mathbf{q}}$, and $\sum_{\mathbf{q}, tf_i > 0}$ should be read as $\sum_{\{i \,|\, i \in \mathbf{q}, tf_i > 0\}}$.

In Step (2.6), we make a common, order-preserving transformation, namely we take a log. This allows us to express the product of probabilities as a sum of log probabilities — actually log-odds because of the ratio in the product.

In Step (2.7), we rewrite the previous equation using a function $U_i(x)$:

$$U_i(x) := \log \frac{P(\,TF_i = x \,|\, \mathrm{rel})}{P(\,TF_i = x \,|\, \overline{\mathrm{rel}})} \tag{2.12}$$

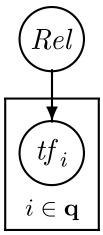


Fig. 2.2 Graphical model for restriction to query terms.

Note that this is not the log-odds function. Note also that in Step (2.7) each term frequency tf_i will be applied to a different function $U_i(tf_i)$. This is necessary since the weight of a term does not depend only on its frequency, but also in other factors such as the collection frequency, etc.

The following two steps use an arithmetic trick (sometimes referred to as *removing the zeros*) which eliminates from the equation terms that are not present in the document. This is crucial for the efficient implementation of PRF ranking functions using inverted indices.

In Step (2.8), we do two things. In the first line, the sum over query terms in Step (2.7) has been split into two groups: those terms that are present in the document (first sum) and those that are absent (second sum). In the second line, we subtract and add the same quantity (the sum of zero frequency weights for terms in the document) leaving the result unchanged. The reason for doing this will become clear in the next step.

In Step (2.9), we regroup the sums of Step (2.8). First we note that the first and third sums in Step (2.8) are over the same terms, and can be grouped. This forms the first sum in Step (2.9). Then we note that the second and fourth sums in Step (2.8) span all terms in the query. This forms the second sum in Step (2.9).

In Step (2.10) we drop the last sum since its value is document-independent. We note that by doing so we are left with a single sum over terms present both in the document and in the query. We have removed terms in the query with zero frequency in the document.

In Step (2.11), we again rewrite the equation using the short-hand notation:

$$W_i(x) := U_i(x) - U_i(0) \tag{2.13}$$

$$= \log \frac{P(TF_i = x \,|\, \mathrm{rel}) P(TF_i = 0 \,|\, \overline{\mathrm{rel}})}{P(TF_i = x \,|\, \overline{\mathrm{rel}}) P(TF_i = 0 \,|\, \mathrm{rel})} \tag{2.14}$$

$$w_i := W_i(tf_i) \tag{2.15}$$

Equation (2.14) is the formula for the basic weight of a query term in a document. Both U_i and W_i are term-weighting functions which can be precomputed and stored at indexing time. The difference is that in

order to use the U_i in a document, we would have to sum over all query terms, whether or not they are present in the document. With W_i, we need only to consider the weights w_i of terms present in the document (in effect the rewriting has defined the weight of absent terms as zero). This fits very well with the sparseness of the document-term matrix — we have no need to calculate scores of any document that contains no query terms. Indeed, this is a highly desirable property for a scoring function to be used in an inverted-file-based document retrieval system, since it means that we can easily calculate the score of all documents with non-zero scores by merging the inverted lists of the query terms.

As indicated above, the model is not restricted to terms and term frequencies — any property, attribute or feature of the document, or of the document–query pair, which we reasonably expect to provide some evidence as to relevance, may be included. Below we will consider static features of documents — properties that are not dependent on the query — for inclusion. Any discrete property with a *natural* zero can be dealt with using the W_i form of the weight — if we want to include a property without such a natural zero, we need to revert to the U_i form.

We note also that both forms are simple linear models — the combination of evidence from the different query terms is just by summation. This is not in itself an assumption — it arises naturally from the more basic assumptions of the model.

In the sections which follow, we define various instantiations of this basic sum-of-weights scoring model.

2.5 A Note on Probabilities and Rank Equivalence

One consequence of our reliance on the probability ranking principle is that we are enabled to make the very cavalier transformations discussed above, on the basis that the *only* property we wish to preserve is the rank order of documents. This might be a reasonable assumption for traditional ad hoc retrieval, but does not work for all retrieval situations. In some, for example in adaptive filtering [42], we find it desirable or necessary to arrive at an explicit estimate of the probability of relevance of each considered document. Unfortunately, while

the above development allows us to serve the ranking purpose well, it is not easily reversible to give us such an explicit estimate. In particular, some of the transformations involved dropping components which would not affect the ranking, but would be required for a good probability estimate. Often, as in the case of the component that we drop at Step (2.3), it would be very difficult to estimate. Thus the above model has to be considerably modified if it is to be used in a situation which requires an explicit probability. This issue is not discussed further in the present survey.

3

Derived Models

The models discussed in this section are all derived from the basic model presented in the previous section. We note again that the symbol TF in the basic weighting formula (2.14) can in general be any discrete property or attribute of the document. We start by interpreting it as a binary variable, indicating the presence or absence only of a query term in a document; in Section 3.4 we return to the more familiar term frequency.

3.1 The Binary Independence Model

Suppose that TF_i is a binary variable, having only the values zero and one. We can think of this, without loss of generality, as presence (one) or absence (zero) of a term: t_i will refer to the event that the term is present in the document. The absence event is simply the complement of the presence event; probability of absence is one minus probability of presence. Now Equation (2.14) reduces to:

$$w_i^{\mathrm{BIM}} = \log \frac{P(t_i \,|\, \mathrm{rel})(1 - P(t_i \,|\, \overline{\mathrm{rel}}))}{(1 - P(t_i \,|\, \mathrm{rel}))P(t_i \,|\, \overline{\mathrm{rel}})} \tag{3.1}$$

We now suppose that we do actually have some evidence on which to base estimates of these probabilities. Since they are conditional on

the relevance property, we are assuming that we have some judgements of relevance. We first imagine, unrealistically, that we have a random sample of the whole collection, which has been completely judged for relevance. We derive an estimator that will also be useful for more realistic cases.

We use the following notation:

N — Size of the judged sample
n_i — Number of documents in the judged sample containing t_i
R — Relevant set size (i.e., number of documents judged relevant)
r_i — Number of judged relevant docs containing t_i

Given this information, we would like to estimate in an appropriate fashion the four probabilities needed for the weights defined in Equation (3.1). The standard maximum likelihood estimate of a probability from trials is a simple ratio, e.g., $P(t_i\,|\,\mathrm{rel}) \approx \frac{r_i}{R}$. However, this is not a good estimator to plug into the weighting formula. A very obvious practical reason is that the combination of a ratio of probabilities and a log function may yield (positive or negative) infinities as estimates. In fact there are also good theoretical reasons to modify the simple ratio estimates somewhat, as discussed in [44], to obtain a more robust estimator which introduces a small *pseudo-count* of frequency 0.5. The resulting formula is the well-known Robertson/Sprck Jones weight:

$$w_i^{\mathrm{RSJ}} = \log \frac{(r_i + 0.5)(N - R - n_i + r_i + 0.5)}{(n_i - r_i + 0.5)(R - r_i + 0.5)} \tag{3.2}$$

We next suppose that some documents, probably a small number, have been retrieved and judged – this is the usual relevance feedback scenario. In this case we might reasonably estimate the probability conditioned on (positive) relevance in the same way, from the known relevant documents. Estimation of the probabilities conditioned on non-relevance is more tricky. The obvious way, which is what Equation (3.2) first suggests, would be to use the documents judged to be not relevant. However, we also know that (normally, for any reasonable query and any reasonable collection) the vast majority of documents are not relevant; those we have retrieved and judged are not only probably few in

number, they are also likely to be a very non-typical set. We can make use of this knowledge to get a better estimate (at the risk of introducing a little bias) by assuming that any document not yet known to be relevant is non-relevant. This is known as the *complement* method. Under this assumption, the RSJ weighting formula defined above still applies, with the following redefinitions of N and n_i.

N — Size of the whole collection
n_i — Number of documents in the collection containing t_i

Experiments suggest that using this complement method gives better estimates than using judged documents only.

Finally, we suppose that we have no relevance information at all (the more usual scenario). In this case, we can only assume that the relevance probability $P(t_i|\text{rel})$ is fixed, but we can continue to use the complement method for the non-relevance probability — now we assume for this estimate that the entire collection is non-relevant. All this can be achieved by setting $R = r_i = 0$ in the RSJ formula (3.2) — this is equivalent to setting $P(t_i|\text{rel}) = 0.5$ (other values can be used [15]).

The resulting formula is a close approximation to classical *idf* (it can be made closer still by a slight modification of the model [47]):

$$w_i^{\text{IDF}} = \log \frac{N - n_i + 0.5}{n_i + 0.5} \tag{3.3}$$

3.2 Relevance Feedback and Query Expansion

The model thus far clearly contains a natural mechanism for relevance feedback — that is, for modifying the query based on relevance information. If we start with no relevance information, then we would weight the terms using the inverse document frequency (IDF) formula. Once the user makes some judgements of relevance, we should clearly reweight the terms according to the RSJ formula. But term reweighting is not in general an effective method for improving search. Additionally, we have to consider expanding the query by adding new terms.

At an early stage in the development of the basic model, rather than considering the entire vocabulary of terms in the estimation of

probabilities, we restricted ourselves to query terms only. This was not because we assumed that non-query terms were incapable of giving us any useful information, but rather that in the absence of any evidence about either which terms might be useful, or how useful they might be, a reasonable neutral prior assumption was that all non-query terms had zero correlation with relevance.

However, in the relevance feedback scenario discussed above, we do indeed have some evidence for the inclusion of non-query terms. Potentially we can treat any term that occurs in any of the relevant documents as possibly useful. However, it is clear that many such terms will be noise, so we will probably need a conservative rule for adding new terms to the query.

For each candidate term (i.e., non-query term present in a document judged relevant) we can consider how useful it might be if added to the query. One measure of this is simply the RSJ weight as above. However, this will emphasise very rare terms (this is consistent with the IDF idea) — such terms may indeed be good evidence of relevance when they occur, but because they occur in so few documents, they will not have much overall effect on retrieval. As an alternative, we look for a measure of the possible overall effect of adding a term; we refer to this (following [53]) as an *offer weight*.

We could base an offer weight formula on a number of different models. One proposed in [41], to go with the binary independence model, is as follows. We look for terms that will maximally increase the *difference in average score* between relevant and non-relevant items. If we were to add term t_i to the query with weight w_i, then under the binary independence model (or indeed any additive term-weighting system based on term presence/absence only), it would increase the score of any document containing it by w_i. The scores of other documents would not be affected, so the increase in average score could be calculated from the probability of the presence of t_i. Thus the *difference* in average score between relevant and non-relevant documents would be

$$OW_i = (P(t_i \,|\, \mathrm{rel}) - P(t_i \,|\, \overline{\mathrm{rel}}))w_i \tag{3.4}$$

(note that this is *not* the same as w_i itself). Further, an appropriate estimate of this quantity can be derived as follows:

$$\begin{aligned} OW_i^{\mathrm{RSJ}} &\approx P(t_i \,|\, \mathrm{rel}) w_i \\ &\approx \frac{r_i}{R} w_i \\ &\propto_q r_i \; w_i^{\mathrm{RSJ}} \end{aligned} \tag{3.5}$$

The first step approximates by ignoring the second probability (usually much smaller than the first); the second replaces the probability by the obvious estimate; and the third multiplies by R, which is constant for a given query.

The usual approach is to extract all terms from the relevant documents, rank them in order of offer weight by formula (3.5), and add only the top x terms from this ranked list (x may be of the order of 10).

This approach to query expansion was intended for the binary independence model and RSJ weighting, but has also been used with some success for BM25 (see Section 3.4 below). But it clearly has some limitations. As we add more and more terms to the query, we are likely to introduce synonyms or closely related words (indeed, this is why we do query expansion in the first place). However, in [36, 37] authors argue that this query expansion may worsen the term independence assumption; they propose an extension of the PRF model which attempts to correct this by taking into account some of the semantic structure of the query.

3.3 Blind Feedback

The same principles may be used in what is now known as pseudo-relevance or *blind feedback*. Here we assume that we have no actual relevance judgements, but we run an initial search on an initial version of the query (using only original query terms), take the top-ranked y documents (say 5 or 10), assume them to be relevant, and then follow the above relevance feedback procedures.

We note, however, the following:

1. Blind feedback is generally known to be capable of improving search results on average, but tends to fail on some queries, particularly on queries that are difficult to start with, where the top-ranked documents from the initial search may be poor.
2. The (true) relevance feedback procedure described above is in some sense correct in terms of the probabilistic model, on the assumption that the relevance judgements are good. In the blind feedback case, we have a set of documents whose relevance is (at best) likely rather than sure. It would be better to take account of the *probability* of relevance of each of these top-ranked documents, rather than simply assuming relevance. Such a method could be devised if we had a calibrated probability of relevance for each of these documents. However, the fact that we do not normally have such a calibrated probability in the present model, as discussed in Section 2.5, makes it more difficult to see how to accomplish this.

3.4 The Eliteness Model and BM25

We now re-introduce term frequencies into our model; this requires a model of how different term frequencies might arise in a document (this model is originally due to Harter [19]). We suppose that for any document-term pair, there is a hidden property which we refer to as *eliteness*. This can be interpreted as a form of aboutness: if the term is elite in the document, in some sense the document is *about* the concept denoted by the term. Now we assume that actual occurrences of the term in the document depend on eliteness, and that there may be an association between eliteness (to the term) and relevance (to the query). But we further assume that these relations are enough to explain the association between term frequency tf and relevance to the query — that is, given the two assumed dependencies, tf is independent of Rel. As before, we can illustrate the assumptions by means of a graphical model, as in Figure 3.1.

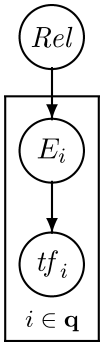


Fig. 3.1 Graphical model of eliteness (E).

We further assume that eliteness itself is binary.

The following development, using Harter's ideas in the context of the PRF, was originally proposed in part in [45] and in full (up to and including BM25) in [46].

3.4.1 The 2-Poisson Model

We introduce some new notation. The eliteness random variable E can take two values:

$$\text{Eliteness E:} \quad \text{elite}, \overline{\text{elite}} \quad \text{(elite, not elite)}$$

We now decompose all the probabilities we want using these two disjoint events, following this pattern:

$$P(\alpha\,|\,\beta) = P(\alpha\,|\,\text{elite})P(\text{elite}\,|\,\beta) + P(\alpha\,|\,\overline{\text{elite}})P(\overline{\text{elite}}\,|\,\beta)$$

The relationship between eliteness and relevance is denoted thus:

$$p_{i1} := P(E_i = \text{elite}\,|\,\text{rel}); \quad p_{i0} := P(E_i = \text{elite}\,|\,\overline{\text{rel}})$$

The relationship between term frequencies and eliteness is denoted thus:

$$E_{i1}(tf) := P(TF_i = tf\,|\,E_i = \text{elite}); \quad E_{i0}(tf) := P(TF_i = tf\,|\,E_i = \overline{\text{elite}})$$

Now, following the above pattern, we arrive at expressions for all the probabilities we are interested in relating the observed tfs to relevance

like the following:

$$P(TF_i = tf \,|\, \text{rel}) = p_{i1}E_{i1}(tf) + (1 - p_{i1})E_{i0}(tf), \quad \text{etc.}$$

This gives us an equation for our term-weights:

$$w_i^{\text{elite}} = \log \frac{(p_1 E_1(tf) + (1 - p_1)E_0(tf))(p_0 E_1(0) + (1 - p_0)E_0(0))}{(p_1 E_1(0) + (1 - p_1)E_0(0))(p_0 E_1(tf) + (1 - p_0)E_0(tf))} \tag{3.6}$$

(for readability, we dropped the i subscript from all the elements in the fraction).

More specifically, we make distributional assumptions about these events. Again following Harter, we assume that the distributions of term frequencies across documents, conditioned on eliteness, are Poisson:

$$E_{ie}(tf) \sim \mathcal{P}(\lambda_{ie}) \quad \text{(Poisson with mean } \lambda_{ie}\text{)}, \tag{3.7}$$

where $e \in \{0, 1\}$. In general, we expect $\lambda_{i1} > \lambda_{i0}$. Thus in the entire collection of documents, which is a mixture of elite and non-elite documents, tf is distributed as a mixture of two Poisson distributions — so that this model is known as the 2-Poisson model. The consequences of these distributional assumptions are analysed below.

The nature of the 2-Poisson model deserves further discussion. In Harter's original formulation, it was applied to abstracts rather than full-text documents, and indeed it can be said to assume that documents are of fixed length. We can interpret the model as follows. We assume that each document is generated by filling a certain number of word-positions (fixed length) from a vocabulary of words. Furthermore, we assume a simple multinomial distribution over words, so that for each position each word has a fixed (small) probability of being chosen, independent of what other words have been chosen for other positions. Then it follows that the distribution of tfs for a given word is binomial, which approximates to a Poisson under these conditions [16, VI.5].

Now the eliteness model can be seen as a simple topical model which causes variation in the unigram distributions. The author is assumed first to choose which topics to cover, i.e., which terms to treat as elite and which not. This defines specific probabilities for the unigram model,

and the author then fills the word-positions according to this chosen model.

This generative version of the 2-Poisson model (that is, a model for how documents are generated) ties it very closely with the language models and topical models discussed further in Sections 4.3 and 4.5.

We note the following characteristics of this model:

1. The model of topicality is a very simple one — one word one topic.
2. There is no attempt to normalise the probabilities across the full unigram model for the document.
3. The model depends fairly crucially on the notion that all documents are of the same (fixed) length.

We do not in the present survey attempt to do anything about the first two points — however, they do point forward to the more recent work on language models, discussed briefly below. Concerning the third point, the issue of document-length normalisation is critical to the present model, and is discussed in Section 3.4.5.

3.4.2 Saturation

If we plug the Poisson distributional assumptions into Equation (3.6), we can express the term weight as a function of the two means λ_e and the mixing proportion of elite and non-elite documents in the collection (as well as the observed tf). This is a somewhat messy formula, and furthermore we do not in general know the values of these three parameters, or have any easy way of estimating them. The next step in the development of the model was therefore to investigate the qualitative behaviour of the term-weighting function, under different conditions, in the hope of arriving at a much simpler expression which would capture its dominant behaviour [46].

Clearly its exact behaviour depends on the parameters, but some generalisations are possible. We note in particular that:

1. $w_i^{\text{elite}}(0) = 0$ (this is by design);
2. $w_i^{\text{elite}}(tf)$ increases monotonically with tf;

3. ... but asymptotically approaches a maximum value as $tf \to \infty$; and
4. the asymptotic limit being

$$\lim_{tf\to\infty} w_i^{\text{elite}}(tf) = \log \frac{p_1(1-p_0)}{(1-p_1)p_0} \tag{3.8}$$

$$= w_i^{\text{BIM}}. \tag{3.9}$$

This last formulation is the weight that the eliteness feature on its own would have. That is, if eliteness were observable, instead of being hidden, we could treat it like a simple binary attribute and weight it in exactly the same way as we weighted term presence in the binary independence model.

This asymptotic property makes perfect sense. Given (as we have assumed) that the only association between *tf* and relevance is via eliteness, the best information we can hope to get from a term is that the document is indeed elite for that term. In reality our information on this score is probabilistic, and thus the term weight is correspondingly reduced. Although this behaviour of the weighting function has been established only for the 2-Poisson case, it seems likely that *any* reasonable distributional assumptions would exhibit similar characteristics.

We refer to this behaviour as *saturation*. That is, any one term's contribution to the document score cannot exceed a saturation point (the asymptotic limit), however, frequently it occurs in the document. This turns out to be a very valuable property of the BM25 weighting function defined below.

3.4.3 A Special Case

There is one case in which the saturation limit does not apply. If we assume that the eliteness property for each query term coincides with relevance for the query/need, so that $p_{i1} = 1$ and $p_{i0} = 0$, then the limit is infinite, and the weight becomes linear in *tf*. Thus the commonly used term-weighting functions such as the traditional *tf*idf*, linear in *tf*, seem to fit with such a model. However, the non-linear, saturating function of *tf* developed below (also combined with an *idf* component) has frequently been shown to work better than traditional *tf*idf*.

3.4.4 BM25 Precursor

We investigate the shape of the saturation function a little more closely. It is clear that the properties listed above severely limit the possible functions; nevertheless, there remain many possibilities, as illustrated for example in the left-hand graph in Figure 3.2. However, the 2-Poisson model generates much smoother functions, as shown in the right-hand graph. For most realistic combinations of the parameters the curve is convex, as the top two lines; for some combinations it has an initial concavity, as the bottom line.

The next step in the development of BM25 is to approximate this shape. Lacking an appropriate generative corpus model from which to derive a convenient formula, the authors of BM25 decided to fit a simple parametric curve to this shape. The following one-parameter function was chosen:

$$\frac{tf}{k + tf} \quad \text{for some } k > 0 \tag{3.10}$$

This function satisfies the properties listed above, and fits well the possible convex curves. We show values of this function for three different values of k in Figure 3.3; the middle line is for $k = 1$, the upper line for lower k and the lower line for higher k. Note that because we apply this to all terms, the absolute height does not matter; what matters is the relative increments for different increments in tf. Thus for high k, increments in tf continue to contribute significantly to the score,

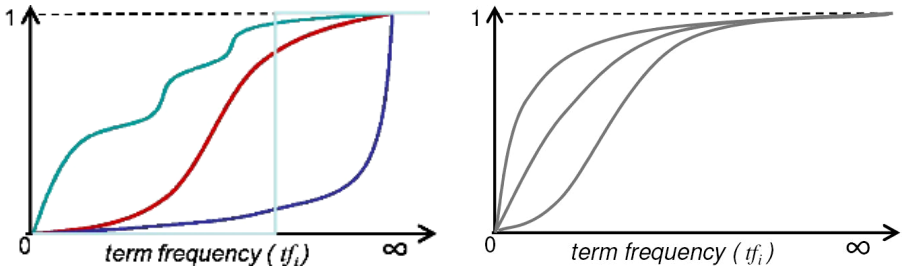


Fig. 3.2 *Left*: some possible saturation functions. *Right*: saturation functions generated by the 2-Poisson model.

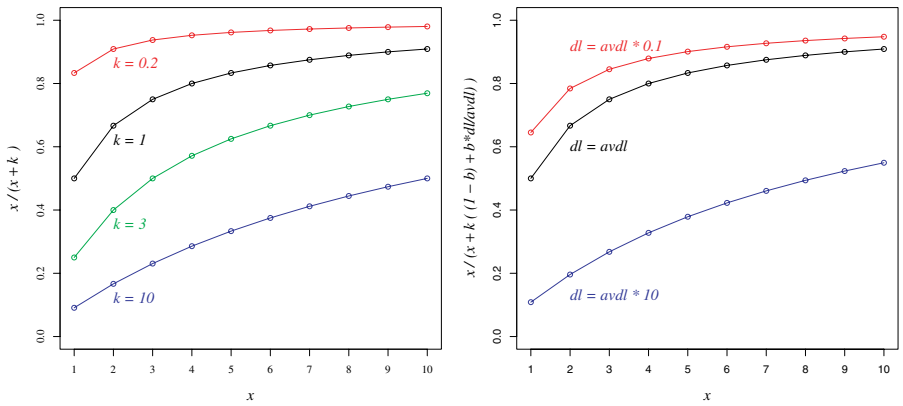


Fig. 3.3 Saturation functions generated by Equation (3.10) with raw frequencies (*left*) and with frequencies normalised using Equation (3.13) (*right*). Stronger saturation is obtained with lower values of k (*left*) and with shorter documents (*right*). For the plot on the right we used $k = 1$ and $b = 0.5$.

whereas for low k, the additional contribution of a newly observed occurrence tails off very rapidly.

We obtain an early version of BM25 by combining the saturation function of Equation (3.10) with an approximation to the asymptotic maximum of Equation (3.9). The latter is obtained by using the old RSJ presence–absence term weight of Equation (3.2):

$$w_i(tf) = \frac{tf}{k + tf}\, w_i^{\text{RSJ}} \tag{3.11}$$

(We will modify this below for the final version of BM25.)

The main thing missing so far from the analysis is the question of document length.

3.4.5 Document Length

The fixed-document-length assumption was made to allow a connection between a simple language model and BM25; we imagined an author filling a fixed number of slots with a fixed unigram model. Now we imagine instead that the author also chooses a document length.

We suppose that there is something like a *standard* length for a document, but that an author may decide to make a document longer

or shorter; we consider only the longer case. Why might an author so decide? We can postulate two extreme cases:

Verbosity: Some authors are simply more verbose than others, using more words to say the same thing.

Scope: Some authors have more to say: they may write a single document containing or covering more ground. An extreme version would have the author writing two or more documents and concatenating them.

The verbosity hypothesis suggests that we should simply normalise any observed tfs by dividing by document length. The scope hypothesis, on the other hand, at least in its extreme version, suggests the opposite. In a real collection of documents we will observe variations in length, which might be due to either effect, or to a combination. We suppose in general a combination: that each hypothesis represents some partial explanation for the observed variation. This in turn suggests that we should apply some kind of soft normalisation.

We define document length in an obvious way:

$$\text{document length:} \quad dl := \sum_{i \in \mathbf{V}} tf_i$$

and also an average document length for the collection:

$$\text{average doclength:} \quad avdl \quad \text{(average over collection)}$$

The length normalisation component will be defined in relation to the average; this ensures that the definition of document length used is not critical. In practice, we could take (for example) the number of characters in the document, or the number of words before parsing, or even the number of unique terms, and still get very similar results.

The soft length normalisation component is:

$$B := \left((1-b) + b\frac{dl}{avdl}\right), \quad 0 \le b \le 1 \tag{3.12}$$

Thus setting $b = 1$ will perform full document-length normalisation, while $b = 0$ will switch normalisation off. Now we use B to normalise

tf, before applying the saturation function, as follows:

$$tf' = \frac{tf}{B} \tag{3.13}$$

$$w_i^{\text{BM25}}(tf) = \frac{tf'}{k_1 + tf'}\, w_i^{\text{RSJ}} \tag{3.14}$$

$$= \frac{tf}{k_1\left((1-b) + b\frac{dl}{avdl}\right) + tf}\, w_i^{\text{RSJ}} \tag{3.15}$$

This is the classic BM25 term-weighting and document-scoring function. As with all term-document weights defined in this survey, the full document score is obtained by summing these term-weights over the (original or expanded) set of query terms.

3.5 Uses of BM25

In order to use BM25 as a ranking function for retrieval, we need to choose values for the internal parameters b and k_1, and also instantiate RSJ.

Concerning the RSJ weight (Equation (3.2)), all the previous comments apply. In particular, it can be used with or without relevance information. In the absence of relevance information, it reduces as before to a form of idf. In this case, the BM25 weight looks very much like a traditional $tf{*}idf$ weight — a product of two components, one based on tf and one on idf. However, there is one significant difference. The tf component involves the saturation function discussed, and is therefore somewhat unlike most other tf functions seen in the literature, where common choices are tf itself and $(1 + \log tf)$. The latter has a somewhat similar shape curve, but does not have an asymptotic maximum — it goes to infinity, even if somewhat slower than tf itself.

Concerning the internal parameters, the model provides no guidance on how these should be set. This may be regarded as a limitation of the model. However, it provides an opportunity for optimisation, given some evaluated set of queries and relevance judgements in the traditional retrieval experiment style. A significant number of such experiments have been done, and suggest that in general values such as $0.5 < b < 0.8$ and $1.2 < k_1 < 2$ are reasonably good in many

circumstances. However, there is also evidence that optimal values do depend on other factors (such as the type of documents or queries).

3.5.1 Some Variations on BM25

Published versions of BM25 can vary somewhat (the original BM25 [46] was a little more complicated than that of Equation (3.15), for example). Here we indicate some differences that might be encountered in different versions of the function in published sources.

- The original had a component for within-query term frequency qtf, for longer queries where a term might occur multiple times. In its full generality, this had a similar saturation function to that used for tf, but with its own k_3 constant. However, experiments suggested that the saturation effect for qtf was unimportant, leading to a formula which was linear in qtf. In other words, one could simply treat multiple occurrences of a term in the query as different terms.
- The original also had a further correction for document length, to the total document score. This correction was again found to be unimportant.
- A common variant is to add a $(k_1 + 1)$ component to the numerator of the saturation function. This is the same for all terms, and therefore does not affect the ranking produced. The reason for including it was to make the final formula more compatible with the RSJ weight used on its own. If it is included, then a single occurrence of a term would have the same weight in both schemes.
- Some published versions are based on specific values assigned to b and k_1. A common combination would be $b = 0.5$ and $k_1 = 2$. (However, many experiments suggest a somewhat lower value of k_1 and a somewhat higher value of b.)

3.6 Multiple Streams and BM25F

So far, all the arguments in this survey have been based on the idea that the document is a single body of text, unstructured and undifferentiated

in any way. However, it is commonplace in search systems to assume at least some minimal structure to documents. In this section we consider documents which are structured into a set of *fields* or *streams*. The assumption is that there is a single flat stream structure, common to all documents. That is, there is a global set of labelled streams, and the text of each document is split between these streams. An obvious example is a title/abstract/body structure such as one might see in scientific papers. In the Web context, with extensive hyperlinks, it is usual to enhance the original texts with the *anchor text* of incoming links.

This is clearly a very minimal kind of structure; one can imagine many document structures that do not fit into this framework. Nevertheless, such minimal structures have proved useful in search. The general idea, not at all confined to the present framework but implemented in many different ways in different systems, is that some streams may be more predictive of relevance than others. In the above examples, a query match on the title might be expected to provide stronger evidence of possible relevance than an equivalent match on the body text. It is now well known in the Web context that matching on anchor text is a very strong signal.

3.6.1 Basic Idea

Given a ranking algorithm or function that can be applied to a piece of undifferentiated text, an obvious practical approach to such a stream structure would be to apply the function separately to each stream, and then combine these in some linear combination (with stream weights) for the final document score. In terms of the eliteness model, this approach may be regarded as assuming a separate eliteness property for each term/stream pair. Thus for a given term, eliteness in title would be assumed to be a different property from eliteness in body. Actually, the assumption would be even stronger: we would have to apply the usual assumptions of independence (given relevance) between these distinct eliteness properties for the same term.

This seems a little unreasonable — a better assumption might be that eliteness is a term/document property, shared across the streams of the document. We would then postulate that the relationship of

eliteness to term frequency is stream-specific. In terms of the underlying language model discussed above, we might imagine that the author chooses a length for (say) the title and another for the body. Then, given eliteness (from the author's choice of topics to cover), the unigram probabilities for the language model for filling the term-positions would also be stream-specific. In particular, there would be a much stronger bias to the elite terms when choosing words for the title than for the body (we expect a title to be much denser in topic-specific terms than an average body sentence).

The consequence of this term-document eliteness property is that we should combine evidence across terms and streams in the opposite order to that suggested above: first streams, then terms. That is, for each term, we should accumulate evidence for eliteness across all the streams. The saturation function should be applied at this stage, to the total evidence for each term. Then the final document score should be derived by combination across the terms.

3.6.2 Notation

We have a set of S streams, and we wish to assign relative weights v_s to them. For a given document, each stream has its associated length (the total length of the document would normally be the sum of the stream lengths). Each term in the document may occur in any of the streams, with any frequency; the total across streams of these term–stream frequencies would be the usual term-document frequency. The entire document becomes a vector of vectors.

streams	$s = 1, \ldots, S$	
stream lengths	sl_s	
stream weights	v_s	
document	$(\mathbf{tf}_1, \ldots, \mathbf{tf}_{\lvert\mathbf{V}\rvert})$	vector of vectors
$\mathbf{tf}_i$ vector	$(tf_{1i}, \ldots, tf_{Si})$	

where tf_{si} is the frequency of term i in stream s.

3.6.3 BM25F

The simplest extension of BM25 to weighted streams is to calculate a weighted variant of the total term frequency. This also implies having

a similarly weighted variant of the total document length:

$$\widetilde{tf}_i = \sum_{s=1}^{S} v_s \, tf_{si} \tag{3.16}$$

$$\widetilde{dl} = \sum_{s=1}^{S} v_s \, sl_s \tag{3.17}$$

$$\widetilde{avdl} = \text{average of } \widetilde{dl} \text{ across documents}$$

$$w_i^{\text{simpleBM25F}} = \frac{\widetilde{tf}_i}{k_1 \left((1-b) + b \frac{\widetilde{dl}}{\widetilde{avdl}} \right) + \widetilde{tf}_i} \, w_i^{\text{RSJ}} \tag{3.18}$$

However, we may also want to allow the parameters of BM25 to vary between streams — it may be that the different streams have different characteristics, e.g., in relation to verbosity *v.* scope (as defined in Section 3.4.5). It is in fact possible to re-arrange formula (3.15) so as to include any of the following in the stream-specific part: k_1, b, w_i^{RSJ} or its non-feedback version w_i^{IDF}. We have in particular found it useful to allow b to be stream-specific; we present here the appropriate version of BM25F:

$$\widetilde{tf}_i = \sum_{s=1}^{S} v_s \frac{tf_{si}}{B_s} \tag{3.19}$$

$$B_s = \left((1-b_s) + b_s \frac{sl_s}{avsl_s} \right), \quad 0 \le b_s \le 1 \tag{3.20}$$

$$w_i^{\text{BM25F}} = \frac{\widetilde{tf}_i}{k_1 + \widetilde{tf}_i} \, w_i^{\text{RSJ}} \tag{3.21}$$

This version was used in [62]; the simple version in [50]. As usual, in the absence of relevance feedback information, w_i^{RSJ} should be replaced by w_i^{IDF}. In [50, 62] we computed IDFs on the entire collection disregarding streams. This worked well in practice, but it can lead to degenerate cases (e.g., when a stream is extremely verbose and contains most terms for most documents). The proper definition of IDF in this context requires further research (this is also discussed in [37] where the notion of an *expected idf* is introduced).

Table 3.1. BM25F parameters reported in [62] for topic distillation (TD) and name page (NP) search tasks

Parameter	TD'03	NP'03
k_1	27.5	4.9
b_{title}	0.95	0.6
b_{body}	0.7	0.5
b_{anchor}	0.6	0.6
v_{title}	38.4	13.5
v_{body}	1.0	1.0
v_{anchor}	35	11.5

As an illustration, we report in Table 3.1 the BM25Fweights reported in [62] for the 2003 TREC Web Search tasks.

3.6.4 Interpretation of the Simple Version

It is worth mentioning a very transparent interpretation of the simple version — although it does not apply directly to the version with variable b, it may give some insight. If the stream weights v_s are integers, then we can see the simple BM25F formula as an ordinary BM25 function applied to a document in which some of the streams have been replicated. For example, if the streams and weights are $\{v_{\text{title}} = 5$, $v_{\text{abstract}} = 2$, $v_{\text{body}} = 1\}$, then formula 3.18 is equivalent to 3.15 applied to a document in which the title has been replicated five times and the abstract twice.

3.7 Non-Textual Relevance Features

In many collections there are other sources of relevance information besides the text. Things like the age of a file, its type or its link in-degree may provide useful information about the relevance of a document. The development of the PRF is quite general and does not make explicit reference to terms or text; it is therefore possible, at least in principle, to take non-textual features into account.

We will make two assumptions here about non-textual features. First we make the usual assumption of feature independence with respect to relevance odds (as discussed in Section 2). Second, we will assume that all non-textual features provide relevance information

which is *independent of the query*. With these two assumptions we can integrate non-textual features very easily into the PRF and BM25 scoring frameworks. It is possible in principle to relax these assumptions and derive more sophisticated models of relevance for non-textual features.

Let us call $\mathbf{c} = (tf_1, \ldots, tf_{|\mathbf{V}|})$ the vector of term frequency counts of document d, and let us call $\mathbf{f}$ an extra vector of non-textual features $\mathbf{f} = (f_1, \ldots, f_F)$. We have that $d = (\mathbf{c}, \mathbf{f})$.

Note that the initial PRF development (Equations (2.1–2.4) in Section 2) applies also to this extended version of d. Equation (2.4) makes the assumption of feature independence, which carries over the non-textual features. Therefore the product in Equation 2.4 would include all non-textual features $\mathbf{f}$.

In Equation (2.5), where we drop all the non-query terms in $\mathbf{c}$, none of the terms in $\mathbf{f}$ would be dropped — non-textual features are assumed to be correlated with relevance for all queries. After taking the log in Equation (2.6) we see that the addition of non-textual features simply adds a new term to the sum. Removing the zeros in Equations (2.7–2.9) does not affect this term, so after (2.11) we may write:

$$P(\mathrm{rel}\,|\,d, q) \propto_q \sum_{\mathbf{q}, tf_i > 0} W_i(tf_i) + \sum_{i=1}^{F} \lambda_i V_i(f_i), \tag{3.22}$$

where

$$V_i(f_i) := \log \frac{P(F_i = f_i\,|\,\mathrm{rel})}{P(F_i = f_i\,|\,\overline{\mathrm{rel}})} \tag{3.23}$$

We have artificially added a free parameter λ_i to account for re-scalings in the approximation of W_i and V_i.

We note that features f_i may well be multi-valued or continuous, and this implies the need for care in the choice of function $V(f_i)$ (just as we paid attention to finding a good function of tf). This will depend on the nature of each non-textual feature f_i and our prior knowledge about it. Here are some V_i functions that we have used with success in the past for different features:

- Logarithmic: $\log(\lambda_i' + f_i)$

- Rational: $f_i/(\lambda'_i + f_i)$
- Sigmoid: $[\lambda'_i + \exp(-f_i\,\lambda''_i)]^{-1}$

This development can explain for example why simple scoring functions such as BM25F(q,d)+log$(PageRank(d))$ may work well in practice for Web Search retrieval. This can much improved by adding the scaling parameter λ, and it can be further improved (only slightly) by changing the log into a sigmoid and tuning the two extra parameters λ' and λ'' [12]. In our work we developed more sophisticated ranking functions integrating several forms of non-textual information and using over a dozen parameters [12, 13]. The optimisation of these parameters is discussed in Section 5.

3.8 Positional Information

For the most part the PRF ignores positional information: it cares only about the number of occurrences of a term, but not about their position. There are two important reasons that have held back the PRF from considering positional information: (i) it is extremely hard to develop a sound formal model of relevance which takes into account positional information without exploding the number of parameters, and (ii) position information has been shown to have surprisingly little effect on retrieval accuracy on average. In this section we only give an overview of the existing approaches and discuss the main difficulties. We also propose some intuitions of why position may not be as crucial as it seems at first sight.

Why is it so hard to model relevance with respect to the position of terms in the query and document? Several problems appear. First, we need to define an appropriate universe of events. In the traditional PRF this universe is simply $\mathbb{N}^{|q|}$, all possible term frequency vectors of terms in the query. The most natural way to consider positions would be to characterise all sequences of terms in the query separated by some number of non-query terms. This leads to an excessively high-dimensional space, and one that is very hard to factor it in an appropriate way. In the traditional model the natural unit over which we build (factorise) the required quantities is the occurrence of a term a given number of

times. What would be the natural unit over which to build (factorise) the probabilities required to model positions adequately?

There have been a number of attempts to deal with these issues and introduce positional information into BM25 (see for example [3, 54, 51] and references therein). The three main approaches used are the following:

1. Indexing phrases as individual terms. This approach is ideal for multi-term tokens (such as *White House*) for which a partial match would in fact be incorrect. Its implementation is extremely simple since one does not need to modify the ranking function in any way (each phrase would have its own *tf* and *idf*). There are three problems however: (i) it does not take into account partial matches; (ii) it does not take into account terms which are close but not exactly adjacent; and (iii) one needs to define the valid phrases [48].
2. Scoring *spans* of text instead of entire documents. This can be done explicitly, with a *passage retrieval* ranking function [3], or implicitly by constructing a ranking function that integrates scores computed over many document spans [51].
3. Deriving *position features* (such as the minimum and maximum length of the document spans containing all the terms in the query) which can then be integrated into the scoring function as non-textual features (as those in Section 3.7) [54].

In our opinion none of these attempts would qualify as a *natural* extension of the PRF, since it is not clear what the assumptions about relevance are. Metzler [32, 33] proposes a novel probabilistic retrieval model which makes clear assumptions about positions and relevance, and could perhaps be integrated within the PRF. The model estimates the probability of relevance of document and query jointly: $P(Q, D \,|\, Rel)$. This is done by a Markov Random Field (MRF) which can take into account term-positions in a natural way. The MRF can use any appropriately defined potentials: while the original work used LM-derived potentials, BM25-like potentials were used in [31]. However, even when using BM25-like potentials, we cannot call this model an *extension* of

the PRF, since it models a different distribution: $P(D,Q\,|\,Rel)$ instead of the posterior $P(Rel\,|\,D,Q)$.

We end this section with a brief discussion of why position information may not be as important as it may seem at first view. It is sobering to see how hard it has been in the past to effectively use proximity in IR experiments. All the works referenced in this section claim statistically significant improvements over non-positional baselines, but the improvements reported are small. We believe this is specially the case for small collections and high recall situations (typical of academic IR evaluations), since position information is a precision enhancement technique. But even in large collections with high-precision requirements (such as realistic Web Search evaluations) the gains observed are small. Why is this? We do not know of theoretical or empirical studies about this, but we propose here two hypotheses.

First, we believe that natural language, and queries in particular, are quite robust. We would argue that for many queries, a human could determine the relevance of a document to a query even after words in the document and query were scrambled. And for cases that this is not true, it is likely that the user would unconsciously *correct* the query by adding terms to it that disambiguate it. This does not mean that all queries and documents are position-proof, but the fraction that require positions is small. Second, it should be noted that taking into account the structure of a document (e.g., in BM25F) implicitly rewards proximity within important short streams, such as the title.

3.9 Open Source Implementations of BM25 and BM25F

We review here several implementations of BM25 and BM25F available as open source. This list is not exhaustive, there may be other search engines or extensions of them that implement BM25 and BM25F.

As far as we know only the MG4J [9, 34] fully implements BM25F (version 2.2 or later). BM25 is implemented by Lemur, MG4J, Okapi, PF/Tija, Terrier, Wumpus, Xapian and Zettair [22, 27, 34, 35, 39, 56, 57, 60, 61, 63]. All these search engines are quite modular and could in principle be modified to extend BM25 in a number of ways, in particular to implement BM25F. Lucene [29] does not implement BM25

but there exist a third party Lucene extensions that implement BM25 and BM25F [38]. We inspected the code of all these implementations and we believe they are correct.[1] (inspection was visual: we did not run tests ourselves; in two cases implementation was incorrect and we asked the authors to correct it, which they did). None of the systems above provide support for parameter optimisation, although it should not be difficult to extend them for this.

[1] There are some minor differences in BM25 implementations in these packages at the time of writing this survey. For example, PF/Tijah uses idf= $\log \frac{N+0.5}{n+0.5}$, Terrier does not allow modifying k programmatically and Xapian does not allow modifying any parameters programmatically.

4

Comparison with Other Models

The first probabilistic model for retrieval was proposed by Maron and Kuhns in 1960 [30]. It was similarly based on a notion of probability of relevance; however, there was an interesting discrepancy between the Maron and Kuhns approach and that of Robertson and Spärck Jones fifteen years later. The discrepancy was the subject of research in the early 1980s on a unification of the two. In order to explain both the discrepancy and the attempted unification, we first describe the Maron and Kuhns model.

4.1 Maron and Kuhns

The situation envisaged by Maron and Kuhns was that of a librarian indexing a book (document). The idea was that indexing should anticipate how people make requests in the first place. Ideal indexing should match the requests of just those people who would want to be pointed to this monograph — those people who would find it relevant to their needs. The *system* was assumed to be a system of subject headings, any of which might be assigned by the librarian to the book in question; a *user request* would take the form of a chosen subject heading to

look under. Thus the librarian would be invited to estimate, in respect of each candidate subject heading, the probability that a user looking there would find this particular book relevant.

Thus far, the basis for the model looks very like the PRF defined in Section 2. However, to better understand the nature of the probability of relevance as interpreted by Maron and Kuhns, we need to consider the event space in which it is defined. This will give us the basis for a comparison in the following section.

We have (at least in the mind of the librarian) a group of users (with information needs) who look under a particular subject heading. On the other hand, we have a single individual book in front of the librarian. Therefore the event space involved, in which the probability should be defined, is a space of users. If we were to attempt to use feedback to estimate such a probability, we would ideally count users — that is, of all the users who express their queries by means of a particular subject heading, what proportion would like this monograph.

This kind of approach has recently acquired new force, because the big Web Search engines typically see some queries (the so-called 'head' queries) repeated very many times by different users. Click log data looks a little like relevance feedback on individual documents from a population of users with similar queries. However, the relation between this and the PRF discussed in this monograph needs further analysis.

4.2 The Unified Model

We re-visit the original RSJ model, the foundation of the model presented in this survey, in order to define it in similar terms. In this case, we start with a single individual user, who puts a request using certain words. Now we ask the question, what is the probability that any arbitrary document matching one (or a combination) of these words is relevant to the user. Thus the event space here consists of documents, and if we want to use feedback to estimate the probability, we would count documents, as in Section 3.1.

It immediately becomes clear that although both models refer to *probability of relevance*, they define their respective versions of this

probability in different event spaces. In fact, the two probabilities of relevance are actually quite distinct.

The unification attempt [43] was based on the following argument. We imagine a matrix of users-with-information-needs against documents (individual users u, individual documents d). Ideally we would like to assign a meaningful probability of relevance to the specific combination of an individual user with an individual document: $P(\text{rel}\,|\,u,d)$. However, doing this directly looks difficult — at least if we are looking for direct evidence (feedback). If we show d to u and get a judgement, we no longer need to assign a probability! Indeed, it only makes sense to do so when we have classes of similar events.

Maron and Kuhns start by classifying users together, according to the subject headings they consult. Thus we are dealing with a class U of users, and ask about $P(\text{rel}\,|\,U,d)$. On the other hand, Robertson and Sprck Jones classify documents together in classes such as D, and ask about $P(\text{rel}\,|\,u,D)$.

The unified model proceeded to define *four* different probabilities of relevance. We might consider starting with $P(\text{rel}\,|\,U,D)$, which is a general model needing only feedback from similar users about similar documents (this is referred to as Model 0). If we have feedback about the particular document, we can improve this estimate by considering $P(\text{rel}\,|\,U,d)$ (Model 1). On the other hand, if we have feedback from the particular user, we should go for $P(\text{rel}\,|\,u,D)$ (Model 2). Finally, if we have both kinds of more specialist evidence simultaneously (Model 3), we might aim for an even better probability. However, its exact relationship to $P(\text{rel}\,|\,u,d)$ is not quite obvious, because while it is based on feedback of the above three kinds, it would not actually have feedback on the exact individual pair.

The unified model has been the subject of some more recent work [8], and we are now entering a domain in which many different kinds of feedback may be possible, given the kind of logging of Web searching behaviour that is now the norm. Other authors (for example [14, 17], and later [25] in the context of the language model discussed below) have approached the problem by in effect limiting themselves to Model 0, by considering only *representations* of documents and queries, rather than individual instances.

However, in the present survey we have restricted ourselves to the RSJ view of the probability of relevance.

4.3 The Simple Language Model

The language model (LM) of IR is a more recent innovation [24, 26, 33], also with a strong probabilistic motivation. We first describe the simple LM, and then some more sophisticated developments.

In the LM, we regard any piece of text (document, query, etc.) as having been generated by a statistical process. Outside of IR, such models have been very successful in various areas, particularly in speech recognition, and the IR application has borrowed from that domain. In this particular view of text, it is regarded as being generated word-by-word in sequence, each word being chosen from a vocabulary. The simplest statistical model, so-called *unigram*, has a fixed probability distribution over the vocabulary, applied to all word-positions (so actually the sequence is not important). n-gram models assume that the choice of the next word depends on the $n-1$ previous words chosen.

The simple LM approach to IR assumes that each document has its own model, and asks this question about each document: what is the probability that the query came from (was generated by) the same language model as the document (there is no separate query model in this approach). This probability is used to rank documents for a query. Thus there is no explicit notion of relevance; the implicit notion is that a document is relevant to a query if the query came from the same language model as the document. This approach also does not distinguish between individual users — a query is understood to be just a text, and each query–document pair is considered as an equivalent individual event. From an individual user point of view, the model implicitly assumes that there is just one relevant document (if this instance of a query was generated by the language model for document 1, it could not also have been generated by the different language model for document 2). However, since the approach does not distinguish individuals, in effect it represents a version of Model 0 (in the terms of the unified model, as discussed above). Different instances of the same query can be generated from different documents; in the (U, D) class, more than

one document can be relevant. But it follows that it makes no sense to consider individual feedback in the context of the simple LM.

In more detail, the document language model is usually built by mixing (*smoothing*) probabilities derived from the document itself with those taken from a general background language model. One purpose of this smoothing is to avoid a zero probability being assigned to any term that does not occur in the document. In general, a rich fund of models and estimation methods has been mined within the general framework of the LM. We further explore only two of these developments in the following sections.

4.4 The Relevance (Language) Model

Some of the limitations of the simple model are addressed in work on a relevance model for the LM framework [26, 33]. Here, by contrast, we assume that the query has (that is, is generated by) its own model, distinct from any particular document model. The initial source for this model is clearly the query itself; however, relevance feedback (from the individual user, for example) can provide additional evidence about it.

With this approach, the document–query matching process becomes much less obvious. Note that both in the simple LM, and in the traditional probabilistic relevance framework (PRF) described in this survey, the process of matching the query to the document is inherent in the model, entirely determined by the model itself. In this new context, no such matching process is defined; it is necessary to choose one from outside the model.

Given that both the document LM and the query (relevance) LM take the form of statistical distributions over a vocabulary, matching becomes a question of matching two statistical distributions. The most common way to do this is to use the Kullback–Leibler (KL) divergence measure. This has good empirical support.

4.5 Topic Models

There is perhaps potential for some kind of bridge between the LM approach and the PRF in work on implicit topic models. Most such

work has sought to discover the topics implicit in the statistical structure of the language of documents; examples are Latent Semantic Indexing (LSI) [18], Probabilistic LSI [21] and Latent Dirichlet Allocation [7]. The hidden *eliteness* variables postulated in the probabilistic relevance framework have some similarities (although much simplified by the assumption of one eliteness variable per term). A related approach is the *parsimonious* LM [20], which attempts to model in what ways a particular document or query is distinguished from the general background language. However, we appear to be some distance from a serious unification of the LM and the PRF which is the main subject of this survey.

4.6 Divergence from Randomness

The DFR models [2], like the language models, do not contain "relevance" as a primitive notion. Also like the language models, they concentrate on the statistical distribution of terms in documents. In general, they seek to identify the ways in which term distributions differ from what one might expect on a random basis — evidence of such divergence is taken implicitly as evidence about relevance.

It is possible to derive a variety of term-weighting and document ranking functions within this framework, including a formulation that is approximately the same as BM25.

5

Parameter Optimisation

Like most IR models, the models in the PRF have free parameters that need to be set to appropriate values. The BM25 and BM25F models are known to be quite robust with respect to their parameters, meaning that small changes in the parameter values (or in the collection) do not produce large changes in accuracy or relevance. Nevertheless significant gains in relevance can be obtained by properly optimising the parameters, specially when we deal with a new collection.

Parameter optimisation comes with considerable costs: it will require the human evaluation of many query results, which is expensive, and the optimised parameters will be specific to the collection evaluated and may not work well for other collections. Furthermore, the optimisation procedure can be computationally costly, requiring more computing power that the search engine itself. For these reasons this approach is only appropriate for specific collections which merit the cost needed to optimise the ranking function. Examples of such collections are the WWW, large corporate collections or high-value News or Help sites.

Let us call θ the vector of all free parameters of the ranking model being tuned. In the case of BM25 this vector would have two

components: $\theta = (k_1, b)$. In the case of BM25F it would have more: $\theta = (k_1, b_1, \ldots, b_S, v_1, \ldots, v_S)$. If we include non-textual features then we have even more parameters, the exact number depending on the transformation used for each non-textual feature.

'Tuning' the parameters of the model can be seen as an optimisation problem, where we seek to maximise the relevance of the ranking model with respect to the model parameters θ, for a given set of relevance judgements. More specifically, if we fix the document collection and a set of queries with their associated relevance judgements, then for any parameter setting θ we can compute the performance measure of our choice $M(\theta)$. What is left is to find the parameter setting that maximises $M(\theta)$. This is exactly an n-dimensional optimisation problem with M as the target function being optimised over the space of valid θ values.

Optimising standard IR measures,[1] however, is not easy: they are very expensive to evaluate, they have local maxima and *plateaus*, they are not smooth and they don't have gradients [49]. For these reasons, it is not easy to apply standard optimisation techniques. Even applying simple 0-th order optimisation techniques such as trusted region optimisation is difficult and expensive. In practice we use a number of ad hoc techniques to speed up exhaustive search. We will discuss these in the next subsection.

Another alternative is to change the function being optimised. This approach is specially useful when one is optimising very many features, and is discussed in the last subsection.

5.1 Greedy Optimisation

We discuss here a number of techniques (some heuristics, some implementation tricks) we used in the past to speedup the exhaustive search and find good parameter values.

Caching: Because function evaluations are so expensive, we cache the evaluations. Indeed we may often re-visit a parameter value in our

[1] Such as Average Precision, Precision@k, Mean Reciprocal Rank and Discounted Cumulative Gain.

search for the optimum. It would be wasteful to re-evaluate all the queries; instead, we store the resulting relevance in a hash table using the parameter values as the key: $H[\theta] \leftarrow M(\theta)$.

Grid: It is also useful to set a minimum resolution for every parameter, defining a *grid* of allowed parameters values. For example, in most of our experiments we did not allow parameters to change beyond the second decimal. This has negligible affect on the relevance performance and greatly increases the effect of caching and the speed of convergence. Furthermore it makes it easier to report and share parameter values.

5.1.1 Robust Line Search

We first consider methods for optimising over a single parameter. Most parameters being optimised are positive but unbounded above. Therefore we do not know the region of interest of the parameter being optimised, nor do we know the required resolution (the size of the intervals to be tested). For this reason we developed the following search algorithm, which we call Robust Line Search (RLS).

Call l and r the current left and right brackets of the 1−D search space. Split the region in m equal parts of size $(r - l)/m$. Compute the performance on the $m + 1$ region boundaries, and call c the boundary with the highest performance.

The idea is to move towards c by re-centering the search region around c and scaling it appropriately. If $l < c < r$ we need to *zoom in* into c, by scaling down the search region. Since the function being optimised has local maxima we cannot zoom too much, or we may completely lose the global maximum; for this reason we tend to zoom very conservatively. On the other hand, if c equals r or l, it is most likely that the maximum is outside the current search region, and possibly far away. Therefore we increase the size of the search region. We iterate this until l and r are within the some minimum distance, typically below the minimum resolution set. An example optimisation is illustrated in Figure 5.1.

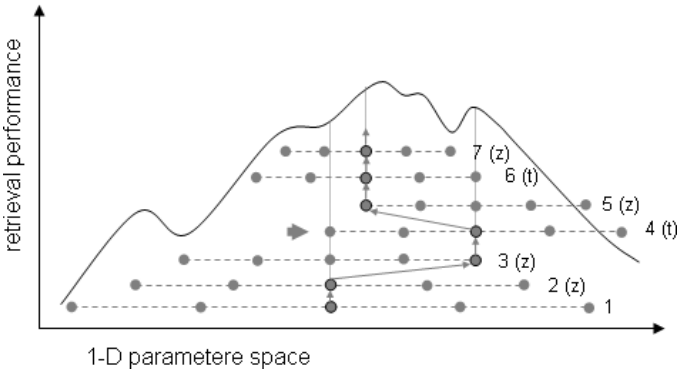


Fig. 5.1 Greedy Optimisation example: robust line search.

5.2 Multidimensional Optimisation

Any 1-D optimisation method can be generalised to n-D in several ways. We have used two methods to do this, both of which worked well in practice.

Greedy Robust Line Search: Imagine that we want to run the RLS method on a 2-D problem, with parameters $\theta = (\theta_1, \theta_2)$. Let's leave θ_2 as a subordinate variable and run RLS on θ_1. Every time that we need to compute the performance $M(\theta)$ at a given point $\theta_1 = z$, we would need to fix the subordinate θ_2 at its optimal value. To find out this value, we can run a local RLS to optimise θ_2 locally while keeping $\theta_1 = z$. Generalising this, every line optimisation with i parameters fires off an optimisation with $i - 1$ subordinate parameters and so on, recursively. Of course this is a greedy approximation to the exhaustive (exponential) exploration of the parameter space, because we are running RLS. However, this approach remains exponentially expensive with respect to n because of its recursive nature, and therefore it is not practicable for large n (e.g., $n > 3$).

Promising Directions: Another way we have used to carry out searches in n dimensions is the following.[2] Choose an initial point for

[2] This method can be seen as a linear version of *trusted region* optimisation methods [4, 5]; it has the advantage of requiring much fewer function evaluations, which are extremely expensive in our case.

each parameter (call the resulting the vector θ). Run n 1−D independent searches, to find the optimum value θ'_i of each parameter when all others are kept fixed at θ. Each of these optimal values found defines a *promising directions* in parameter space. Now consider the vector going from θ to $\theta' = (\theta'_1, \ldots, \theta'_n)$. We expect this vector to move through interesting regions of space if there is correlation between features. Therefore we do one more 1−D optimisation along this line. We do this by re-parameterising the system with a new variable that moves all parameters linearly from θ to θ'. Finally we choose the best parameter setting found so far and we re-start the process. An example optimisation is illustrated in Figure 5.2, where we show the first two iterations (noted 1 and 2), each consisting of three line searches (noted a, b and c). The total cost of an iteration is $(n + 1)$ 1−D optimisations. Therefore, it grows only linearly with n, but it may require very many iterations to complete.

K1 Scaling: Note that in general k_1 will depend on the weights assigned to streams in BM25F, even if it is kept as a stream-independent parameter. This is because the stream weights in effect rescale the *tf* values, and k_1 has to be adjusted accordingly. If we have a good k_1^{BM25} value for the regular BM25 function (no streams), we can propose a good initial value of k_1 for BM25F by scaling it according to the change

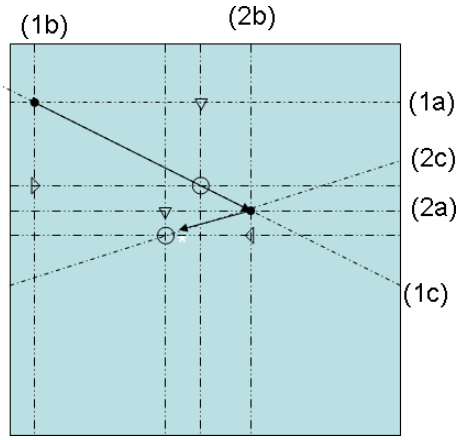


Fig. 5.2 Greedy optimisation example: promising directions.

in average document length from the unweighted to the weighted form:

$$k_1^{\text{BM25F}} \approx k_1^{\text{BM25}} \frac{\sum_s v_s avsl_s}{\sum_s avsl_s} \tag{5.1}$$

5.2.1 Factoring the Search

A very useful technique for speeding up the search in practice is to factor the search into *batches*, optimising together only the parameters that are known to be strongly dependent. When doing this it is important to choose the initial parameters and the order of the batches judiciously. A typical schedule for BM25F may be:

1. Compute the optimal k_1 and b (ignoring streams). This is equivalent to setting all $v_s = 1$ and all $b_s = b$. (Cost: 1×2D);
2. Optimise stream weights $\{w\}_{s=1..S}$ jointly. We use here the k_1 re-scaling trick in Step (5.1). (Cost: 1×SD); and
3. Optimise k_1 and $\{b\}_{s=1..S}$ independently (Cost: $(S + 1)$ ×1D).

This schedule may be repeated until no further increase in performance is obtained. When dealing with non-textual features, the optimisation above is usually interleaved with the optimisation of the non-textual features, which can also be done independently or jointly by batches.

5.3 Gradient Optimisation

A different possibility is to choose a performance function that can be optimised directly by machine learning techniques. This approach was pursued with some success a few years ago [1, 10, 55], and has now become an active area of research (see for example the recent NIPS and SIGIR workshops on this topic [23, 28]). The main idea is to approximate rank-dependant relevance functions such as NDCG by a function with known gradients. Then we can apply the battery of gradient-based optimisation methods. The speed of these methods does not grow exponentially with the number of dimensions optimised, so one can optimise very many parameters simultaneously. Furthermore, if BM25F is being combined with a larger number of other (non-textual)

features, methods can be used to optimise jointly all parameters. For example, this is the case in [55] where the overall model is an artificial neural network which takes as input many features, one of them being BM25F. It is out of the scope of this survey to detail gradient-based optimisation methods.

6

Conclusions

The classical probabilistic relevance framework has provided a series of well-founded scoring formula, as well as some significant insights into different aspects of search. One of the reasons of the success of the PRF, we believe, is the powerful combination of sound theoretical modelling and a pragmatic parameterisation that exploits our prior knowledge in IR. We do not believe that the PRF has reached the end of its useful life. When it is well understood, the PRF model can provide a solid ground on which to analyse new IR problems and derive new solutions.

References

[1] S. Agarwal, C. Cortes, and R. Herbrich, eds., *Proceedings of the NIPS 2005 Workshop on Learning to Rank*, 2005.

[2] G. Amati, C. J. van Rijsbergen, and C. Joost, "Probabilistic models of information retrieval based on measuring the divergence from randomness," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 357–389, 2002.

[3] M. M. Beaulieu, M. Gatford, X. Huang, S. E. Robertson, S. Walker, and P. Williams, "Okapi at TREC-5," *The Fifth Text Retrieval Conference (TREC-5). NIST Special Publication* 500-238, pp. 143–165, 1997.

[4] F. V. Berghen, "Trust Region Algorithms," Webpage, http://www.lemurproject.org.

[5] F. V. Berghen, "CONDOR: A constrained, non-linear, derivative-free parallel optimizer for continuous, high computing load, noisy objective functions," PhD thesis, Université Libre de Bruxelles, 2004.

[6] C. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[8] D. Bodoff and S. E. Robertson, "A new unified probabilistic model," *Journal of the American Society for Information Science and Technology*, vol. 55, pp. 471–487, 2004.

[9] P. Boldi and S. Vigna, "MG4J at TREC 2005," in *The Fourteenth Text Retrieval Conference (TREC 2005) Proceedings, NIST Special Publication* 500-266, 2005. http://mg4j.dsi.unimi.it/.

[10] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 22, p. 89, 2005.

[11] W. Cooper, "Some inconsistencies and misidentified modelling assumptions in probabilistic information retrieval," *ACM Transactions on Information Systems*, vol. 13, pp. 110–111, 1995.

[12] N. Craswell, S. E. Robertson, H. Zaragoza, and M. Taylor, "Relevance weighting for query independent evidence," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 472–479, ACM, 2005.

[13] N. Craswell, H. Zaragoza, and S. E. Robertson, "Microsoft Cambridge at TREC-14: Enterprise track," in *The Fourteenth Text Retrieval Conference (TREC 2005)*, 2005.

[14] F. Crestani, M. Lalmas, C. J. van Rijsbergen, and I. Campbell, ""Is this document relevant? ... probably": A survey of probabilistic models in information retrieval," *ACM Computing Surveys*, vol. 30, no. 4, 1998.

[15] W. B. Croft and D. J. Harper, "Using probabilistic models of document retrieval without relevance information," *Journal of Documentation*, vol. 35, pp. 285–295, 1979.

[16] W. Feller, *An Introduction to Probability Theory and Its Applications,* vol. 1. Wiley, 1968.

[17] N. Fuhr, "Probabilistic Models in Information Retrieval," *The Computer Journal*, vol. 35, no. 3, 1992.

[18] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum, "Information retrieval using a singular value decomposition model of latent semantic structure," in *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 465–480, ACM, 1988.

[19] S. P. Harter, "A probabilistic approach to automatic keyword indexing (parts 1 and 2)," *Journal of the American Society for Information Science*, vol. 26, pp. 197–206 and 280–289, 1975.

[20] D. Hiemstra, S. E. Robertson, and H. Zaragoza, "Parsimonious language models for information retrieval," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 178–185, ACM, 2004.

[21] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 50–57, ACM, 1999.

[22] Indri. Homepage. http://www. lemurproject.org/indri.

[23] T. Joachims, H. Li, T. Y. Liu, and C. Zhai, "Learning to rank for information retrieval (LR4IR 2007)," *SIGIR Forum*, vol. 41, no. 2, pp. 58–62, 2007.

[24] J. Lafferty and C. Zhai, "Document language models, query models, and risk minimization for information retrieval," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2001.

[25] J. Lafferty and C. Zhai, "Probabilistic relevance models based on document and query generation," in *Language Modelling for Information Retrieval*, (W. B. Croft and J. Lafferty, eds.), pp. 1–10, Kluwer, 2003.
[26] V. Lavrenko and W. B. Croft, "Relevance based language models," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 120–127, ACM, 2001.
[27] Lemur Toolkit. Homepage. http://www.lemurproject.org.
[28] H. Li, T. Y. Liu, and C. Zhai, "Learning to rank for information retrieval (LR4IR 2008)," *SIGIR Forum*, vol. 42, no. 2, pp. 76–79, 2008.
[29] Lucene. Homepage. http://lucene.apache.org/.
[30] M. E. Maron and J. L. Kuhns, "On relevance, probabilistic indexing and information retrieval," *Journal of the ACM*, vol. 7, no. 3, pp. 216–244, 1960.
[31] D. Metzler, "Automatic feature selection in the Markov random field model for information retrieval," in *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pp. 253–262, ACM New York, NY, USA, 2007.
[32] D. Metzler and W. B. Croft, "A Markov random field model for term dependencies," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 472–479, ACM, 2005.
[33] D. Metzler, T. Strohman, and B. Croft, *Information Retrieval in Practice*. Pearson Education (US), 2009.
[34] MG4J: Managing gigabytes for java. Homepage. http://mg4j.dsi.unimi.it/.
[35] Okapi-Pack. Homepage. http://www.soi.city.ac.uk/ andym/OKAPI-PACK.
[36] J. R. Pérez-Agüera and H. Zaragoza, "UCM-Y!R at CLEF 2008 Robust and WSD tasks," *CLEF 2008 Workshop*, 2008.
[37] J. R. Pérez-Agüera, H. Zaragoza, and L. Araujo, "Exploiting morphological query structure using genetic optimization," in *NLDB 2008 13th International Conference on Applications of Natural Language to Information Systems*, Lecture Notes in Computer Science (LNCS), Springer Verlag, 2008.
[38] J. Pérez-Iglesias, "BM25 and BM25F Implementation for Lucene," Webpage, http://nlp.uned.es/∼jperezi/Lucene-BM25.
[39] PF-Tijah. Homepage. http://dbappl.cs.utwente.nl/pftijah.
[40] S. E. Robertson, "The probability ranking principle in information retrieval," *Journal of Documentation*, vol. 33, pp. 294–304, 1977.
[41] S. E. Robertson, "On term selection for query expansion," *Journal of Documentation*, vol. 46, pp. 359–364, 1990.
[42] S. E. Robertson, "Threshold setting and performance optimization in adaptive filtering," *Information Retrieval*, vol. 5, pp. 239–256, 2002.
[43] S. E. Robertson, M. E. Maron, and W. S. Cooper, "The unified probabilistic model for IR," in *Proceedings of Research and Development in Information Retrieval*, (G. Salton and H.-J. Schneider, eds.), pp. 108–117, Berlin: Springer-Verlag, 1983.
[44] S. E. Robertson and K. Sparck Jones, "Relevance weighting of search terms," *Journal of the American Society for Information Science*, 1977.

[45] S. E. Robertson, C. J. van Rijsbergen, and M. F. Porter, "Probabilistic models of indexing and searching," in *Information Retrieval Research (Proceedings of Research and Development in Information Retrieval, Cambridge, 1980)*, (R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen, and P. W. Williams, eds.), pp. 35–56, London: Butterworths, 1981.

[46] S. E. Robertson and S. Walker, "Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval," in *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 232–241, ACM/Springer, 1994.

[47] S. E. Robertson and S. Walker, "On relevance weights with little relevance information," in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 16–24, ACM, 2007.

[48] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau, "Okapi at TREC," in *The First Text Retrieval Conference (TREC-1), NIST Special Publication* 500-207, pp. 21–30, 1992.

[49] S. E. Robertson and H. Zaragoza, "On rank-based effectiveness measures and optimization," *Information Retrieval*, vol. 10, no. 3, pp. 321–339, 2007.

[50] S. E. Robertson, H. Zaragoza, and M. Taylor, "Simple BM25 extension to multiple weighted fields," in *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management*, pp. 42–49, ACM, 2004.

[51] R. Song, M. J. Taylor, J. R. Wen, H. W. Hon, and Y. Yu, "Viewing term proximity from a different perspective," *Advances in Information Retrieval (ECIR 2008),* Springer LNCS 4956, pp. 346–357, 2008.

[52] K. Sparck Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: Development and comparative experiments. Part 1," in *Information Processing and Management*, pp. 779–808, 2000.

[53] K. Sparck Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: Development and comparative experiments. Part 2," in *Information Processing and Management*, pp. 809–840, 2000.

[54] T. Tao and C. Zhai, "An exploration of proximity measures in information retrieval," in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 295–302, ACM, 2007.

[55] M. Taylor, H. Zaragoza, N. Craswell, S. E. Robertson, and C. Burges, "Optimisation methods for ranking functions with multiple parameters," in *Fifteenth Conference on Information and Knowledge Management (ACM CIKM)*, 2006.

[56] Terrier. Homepage. http://ir.dcs.gla.ac.uk/terrier.

[57] R. van Os D. Hiemstra, H. Rode, and J. Flokstra, "PF/Tijah: Text search in an XML database system," *Proceedings of the 2nd International Workshop on Open Source Information Retrieval (OSIR)*, pp. 12–17, http://dbappl.cs.utwente.nl/pftijah, 2006.

[58] C. J. van Rijsbergen, *Information Retrieval*. Butterworth, 1979.

[59] E. M. Voorhees and D. K. Harman, "Overview of the eighth text retrieval conference (TREC-8)," *The Eighth Text Retrieval Conference (TREC-8), NIST Special Publication* 500-246, pp. 1–24, 2000.

[60] Wumpus. Homepage. http://www.wumpus-search.org/.
[61] Xapian. http://xapian.org.
[62] H. Zaragoza, N. Craswell, M. Taylor, S. Saria, and S. E. Robertson, "Microsoft Cambridge at TREC 2004: Web and HARD track," in *The Thirteenth Text Retrieval Conference (TREC 2004), NIST Special Publication,* 500-261, 2005.
[63] Zettair. Homepage. http://www.seg.rmit.edu.au/zettair.