Tasks

The code part consist of three ".py" files:

- "extract" to prepare the data and save it to folder "extracted"
- "estimator" with all the algorithms
- "main" to run the experiments. The result of the experiment is printed in console (optionally) and saved in "result.txt"

Training and testing sets splitting provided as 1:1.

The task is to classify a sections of the documents.

1. The leave-one-out method is implemented as optimization of μ. The scipy minimization method was used to optimize the leave-one-out likelihood.
   The optimal μ was obtained as approx. 14300, which is unlikely to be the true optimal μ (Often it's about 2000).
2. In the case of using the KL-divergence, we look for minimal divergence between query model and document models.

$$KL(q,d) = \sum_{w \in V} p(w|q) log(\frac{p(w,q)}{p(w,d)})$$

This divergence is asymmetric – if we exchange *q* and *d* we get different result.

In the implemented algorithm we use in $p(w,d)$ smoothing in order to get non-zero values and use the global frequency of word.

Algorithm:

1. Calculate KL for each pair (q, d)
2. Estimate, for which document d the KL-divergence is the least. That document is the class by which we label our document.

The obtained result contains Accuracy, Precision, Recall and F1-score for every section (summary, education and work_experience). The methods under consideration are Maximum Likelihood Estimator with Jelinec-Mercer (J_M), Dirichlet and No smoothing, and Kullback–Leibler divergence with No smoothing. To save place in this report, only f1-score results is places. Other information can be found in "report.txt".

Method	Summary	Education	Work_experience	Avg.
MLE with J_M	0.84	0.95	0.91	0.9
MLE with Dirichlet	0.74	1.0	0.92	0.89
Pure MLE	0.64	0.99	0.89	0.84
Pure KLD	0.62	0.99	0.9	0.83

Methods with smoothing perform better.

MLE with Jelinec-Mercer performs best ($\lambda = 0.9$), though MLE with Dirichlet is not much worse.This is consistent with the data that on long queries Jelinec-Mercer performs better than everyone else and the Dirichlet is just slightly behind it.

It can be seen that Pure MLE performs almost as Pure KLD. It strictly follows from their formulas:

$$\sum_{w \in V} c(w, q) log(p(w|d)) \rightarrow max\ _d(value)$$

$$\sum_{w \in V} p(w|q) log(\frac{p(w,q)}{p(w,d)}) \rightarrow min\ _d(value)$$

Questions

1. *Explain the idea behind Jelinec-Mercer smoothing. How can you interpret λ?*

   The intuition of the method is to find a compromise between a standard ML estimate (document-specific) and the probability of the word in all the documents for the calculation of the probability of a word in a particular document.

   $$p(w|d) = (1 - \lambda)\frac{c(w, d)}{|d|} + \lambda p(w|REF)$$

   As follows from the formula, more λ means more smoothing. λ belongs to [0, 1]. with increasing of λ (from 0 t 1), calculated probability is more and more affected by the probability of the word in a reference dictionary.

2. *Explain the idea behind Dirichlet smoothing. How is it essentially different from Jelinec-Mercer smoothing?*

   Dirichlet prior smoothing uses Dirichlet distribution.

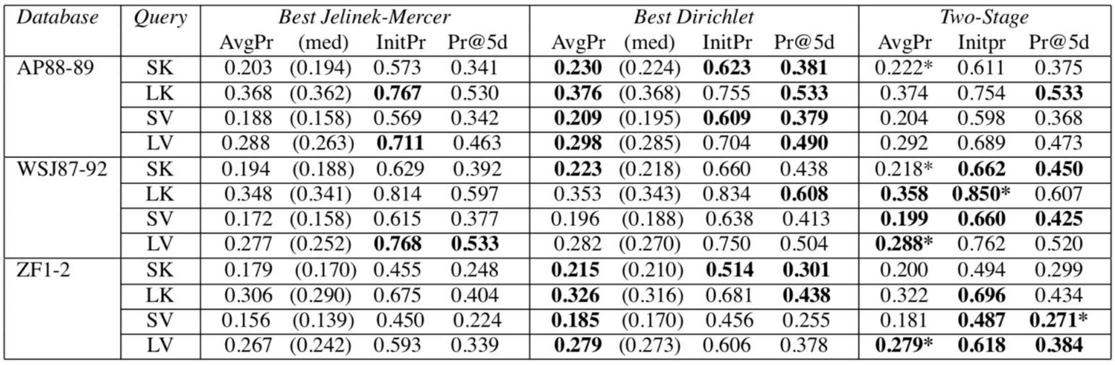
   $$p(w|d) = \frac{c(w, d) + \mu p(w|REF)}{|d| + \mu} = \frac{|d|}{|d| + \mu}\frac{c(w, d)}{|d|} + \frac{\mu}{|d| + \mu} p(w|REF)$$

   In the Jelinec-Mercer method, the term weight has a document length normalization implicit in $p_s(q_i|C)$, but in the Dirichlet smoothing the term weight is affected by only the raw counts of a term, not the length of the document. Further, $\frac{|d|}{\mu}$ is playing the same role as $\frac{(1-\lambda)}{\lambda}$ is playing in Jelinec-Mercer. For both of the methods, when the parameter approaches zero, the scoring formula is dominated by the count of matched terms. But the Dirichlet is more complicated for the large parameter.

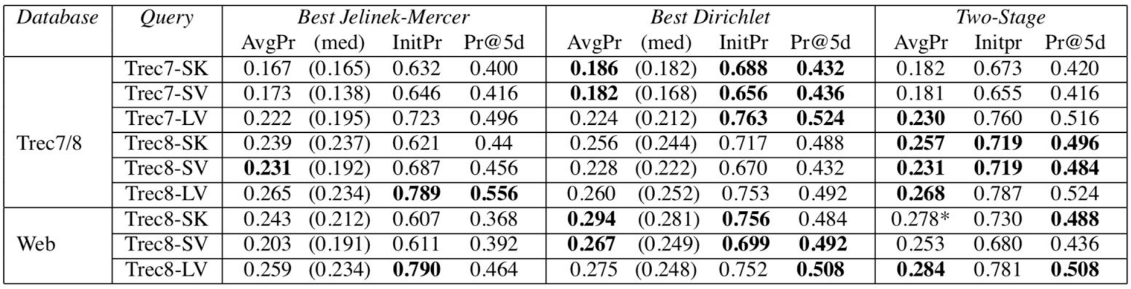
3. *Find information about how these two smoothing approaches are related to standard tf-idf and document length normalization heuristics. Cite your sources.*

   Smoothing approaches and TF_IDF have the similar form (according their formulas). $P_s(w|\theta)$ would be larger for words with high TF (≈ TF heuristic). Frequent items in collection would have high $P(w|C)$ and thus smaller overall weight (≈ IDF heuristic). Source.

4. *Find information about the performance of these two smoothing methods in practice.*



Database	*Query*	*Best Jelinek-Mercer*				*Best Dirichlet*				*Two-Stage*		
		AvgPr	(med)	InitPr	Pr@5d	AvgPr	(med)	InitPr	Pr@5d	AvgPr	Initpr	Pr@5d
AP88-89	SK	0.203	(0.194)	0.573	0.341	**0.230**	(0.224)	**0.623**	**0.381**	0.222*	0.611	0.375
	LK	0.368	(0.362)	**0.767**	0.530	**0.376**	(0.368)	0.755	**0.533**	0.374	0.754	**0.533**
	SV	0.188	(0.158)	0.569	0.342	**0.209**	(0.195)	**0.609**	**0.379**	0.204	0.598	0.368
	LV	0.288	(0.263)	**0.711**	0.463	**0.298**	(0.285)	0.704	**0.490**	0.292	0.689	0.473
WSJ87-92	SK	0.194	(0.188)	0.629	0.392	**0.223**	(0.218)	0.660	0.438	0.218*	**0.662**	**0.450**
	LK	0.348	(0.341)	0.814	0.597	0.353	(0.343)	0.834	**0.608**	**0.358**	**0.850***	0.607
	SV	0.172	(0.158)	0.615	0.377	0.196	(0.188)	0.638	0.413	**0.199**	**0.660**	**0.425**
	LV	0.277	(0.252)	**0.768**	**0.533**	0.282	(0.270)	0.750	0.504	**0.288***	0.762	0.520
ZF1-2	SK	0.179	(0.170)	0.455	0.248	**0.215**	(0.210)	**0.514**	**0.301**	0.200	0.494	0.299
	LK	0.306	(0.290)	0.675	0.404	**0.326**	(0.316)	0.681	**0.438**	0.322	**0.696**	0.434
	SV	0.156	(0.139)	0.450	0.224	**0.185**	(0.170)	0.456	0.255	0.181	**0.487**	**0.271***
	LV	0.267	(0.242)	0.593	0.339	**0.279**	(0.273)	0.606	0.378	**0.279***	**0.618**	**0.384**



Database	*Query*	*Best Jelinek-Mercer*				*Best Dirichlet*				*Two-Stage*		
		AvgPr	(med)	InitPr	Pr@5d	AvgPr	(med)	InitPr	Pr@5d	AvgPr	Initpr	Pr@5d
Trec7/8	Trec7-SK	0.167	(0.165)	0.632	0.400	**0.186**	(0.182)	**0.688**	**0.432**	0.182	0.673	0.420
	Trec7-SV	0.173	(0.138)	0.646	0.416	**0.182**	(0.168)	**0.656**	**0.436**	0.181	0.655	0.416
	Trec7-LV	0.222	(0.195)	0.723	0.496	0.224	(0.212)	**0.763**	**0.524**	**0.230**	0.760	0.516
	Trec8-SK	0.239	(0.237)	0.621	0.44	0.256	(0.244)	0.717	0.488	**0.257**	**0.719**	**0.496**
	Trec8-SV	**0.231**	(0.192)	0.687	0.456	0.228	(0.222)	0.670	0.432	**0.231**	**0.719**	**0.484**
	Trec8-LV	0.265	(0.234)	**0.789**	**0.556**	0.260	(0.252)	0.753	0.492	**0.268**	0.787	0.524
Web	Trec8-SK	0.243	(0.212)	0.607	0.368	**0.294**	(0.281)	**0.756**	0.484	0.278*	0.730	**0.488**
	Trec8-SV	0.203	(0.191)	0.611	0.392	**0.267**	(0.249)	**0.699**	**0.492**	0.253	0.680	0.436
	Trec8-LV	0.259	(0.234)	**0.790**	0.464	0.275	(0.248)	0.752	**0.508**	**0.284**	0.781	**0.508**

Jelinec-Mercer smoothing performs better on small training sets. Both methods works better than backoff smoothing methods. Sources: here and here. Also a lot of information can be found here.

5. *Explain the steps necessary to incorporate user language model into query processing using Mixture Model (Zhai & Laerty (2002)).*

   At the first stage, a document language model is smoothed using a Dirichlet prior, and in the second stage it is further smoothed using Jelinek-Mercer. The combined smoothing function is given by $p_{\lambda,\mu}(w|d) = (1-\lambda)\frac{c(w,d)+\mu p(w|C)}{|d|+\mu} + \lambda p(w|U)$, where $p(\cdot\,|U)$ is the user's query background language model, λ is the Jelinek-Mercer smoothing parameter, and μ is the Dirichlet prior parameter.