Universidade Do Minho

Mestrado Integrado Em Engenharia Biomédica

Bibliotecas Digitais

TERRIER

Braga, Junho 2013

Universidade Do Minho

Mestrado Integrado Em Engenharia Biomédica

Bibliotecas Digitais

TERRIER

Docentes: Joaquim Macedo

Discentes: Hugo Gomes (58536)
Marta Paes Moreira (54459)
Telma Veloso (58521)

Conteúdo

1 Configuração do Terrier

1.1 Configuração Geral

a) Em que ficheiros se colocam os parâmetros que determinam o modo de funcionamento do Terrier?

Relativamente à configuração geral do Terrier, o modo de funcionamento do programa é sobretudo determinado pelos parâmetros introduzidos em dois ficheiros centrais: `terrier.properties` e `terrier-log.xml`, ambos alojados na directoria `/etc`. Por defeito, o primeiro ficheiro apresenta a configuração abaixo declarada, podendo ser editado no sentido de adaptar o *framework* às necessidades específicas de diferentes aplicações [1].

```
#default controls for query expansion
querying.postprocesses.order=QueryExpansion
querying.postprocesses.controls=qe:QueryExpansion

#default and allowed controls
querying.default.controls=
querying.allowed.controls=qe,start,end,qemodel

#document tags specification
#for processing the contents of
#the documents, ignoring DOCHDR
TrecDocTags.doctag=DOC
TrecDocTags.idtag=DOCNO
TrecDocTags.skip=DOCHDR

#query tags specification
TrecQueryTags.doctag=TOP
TrecQueryTags.idtag=NUM
TrecQueryTags.process=TOP,NUM,TITLE
TrecQueryTags.skip=DESC,NARR

#stop-words file
stopwords.filename=stopword-list.txt

#the processing stages a term goes through
termpipelines=Stopwords,PorterStemmer
```

b) Indique como se declara a utilização de expansão de interrogações, a lista de stopwords e o stemming.

Relativamente à lista de *stopwords*, a sua integração no decorrer da indexação é definida através da propriedade `stopwords.intern.terms`, que pode, então, assumir os valores lógicos `true` ou `false`. A posterior aplicação desta lista e do processo de *stemming* às *queries* consideradas passa pelas definições ao nível da propriedade `termpipelines`, que deve contemplar, neste sentido, as entidades `Stopwords` e `PorterStemmer`. Por último, a utilização da expansão de interrogações depende directamente das definições inerentes à propriedade lógica `parameter.free.expansion`, sendo que existem, adicionalmente, propriedades que permitem configurar outras vertentes deste processo, como sejam os modelos de expansão a utilizar (`trec.qe.model`), o número de termos a utilizar na expansão (`expansion.terms`) e o número de documentos *top-ranked* a partir dos quais estes termos são extraídos (`expansion.documents`). No caso específico de se pretender utilizar como abordagem o mecanismo de expansão de interrogações de Rocchio, a propriedade `parameter.free.expansion` deve assumir o valor `false` [1].

c) Em termos de processamento das interrogações e dos documentos, que variantes estão disponíveis?

Ao nível do processamento de interrogações, as propriedades passíveis de configuração permitem controlar o tratamento a aplicar a interrogações vazias (`match.empty.query`), os controlos cuja especificação é permitida nas interrogações (`querying.allowed.controls`) e aqueles que são considerados por defeito (`querying.default.controls`), a ordenação dos pós-processos e pós-filtros (`querying.postprocesses.order`, `querying.postfilter.orders`), e os controlos inerentes à respectiva chamada (`querying.postprocesses.controls`, `querying.postfilters.controls`) [1].

d) Explique outras possibilidades de configuração que não foram explicitadas, embora estejam disponíveis.

No contexto da configuração geral, é ainda possível atender às propriedades de sistema providenciadas pela plataforma Java, que descrevem a configuração do ambiente de trabalho actual, nomeadamente ao nível da informação do utilizador, da versão do *runtime* e do caracter utilizado para a separação dos componentes na especificação de *paths*. Por último, é ainda possível configurar o *logging* utilizado pela aplicação — `Log4j` —, no sentido de controlar a quantidade de informação de *logging* debitada [1].

1.2 Configuração de Indexação

a) O primeiro aspecto a parametrizar é o componente TermPipeline. Existe algum stemmer para o Português, no Terrier? E lista de stop-words?

O `Terrier` inclui todos os *stemmers* do projecto `Snowball` e, entre eles, encontra-se o `PortugueseSnowballStemmer`. No ficheiro `terrier.properties`, deve colocar-se [1]:

```
#termpipelines=Stopwords,PortugueseSnowballStemmer.
```

Para as *stopwords*, é necessário fornecer ao Terrier um ficheiro de texto com a lista de palavras que são consideradas como *stopwords*. Para a realização deste trabalho, foi utilizado o ficheiro fornecido, devidamente alojado na directoria `terrier-3.5/share`.

b) Para manter a meta-informação dos documentos, existe o metaindex. O que se pode configurar aí?

Como referido no enunciado, o `MetaIndex` é uma estrutura que permite registar meta-informação dos documentos a indexar. Esta estrutura pode ser configurada de modo a guardar vários atributos dos documentos durante a indexação, como, por exemplo, o `DOCNO` e `URL` de cada documento. As seguintes propriedades permitem a configuração do `MetaIndex` [1]

- `indexer.meta.forward.keys`: lista de atributos a registar no `MetaIndex`, separada por vírgulas;
- `indexer.meta.forward.keylens`: lista do tamanho máximo de atributos a registar, separada por vírgulas — 20, por defeito);
- `indexer.meta.reverse.keys`: lista de atributos que identificam de forma única um determinado documento.

c) Relativamente ao processo de indexação, existem duas alternativas: uma indexação num único passo e uma indexação em dois passos. Quais são as diferenças fundamentais? Apresente também vantagens e desvantagens.

A indexação de passo único guarda em memória os termos das *posting lists* e, quando a memória está cheia, estes são escritos em disco como 'run'. De seguida, os termos em memória e em disco são usados para formar o léxico e o ficheiro invertido. Neste tipo de indexação, o ficheiro directo nunca é criado, daí esta alternativa ser mais rápida do que a indexação em dois passos. A maior parte das configurações desta indexação estão relacionadas com o consumo de memória e os parâmetros que definem quando se encontra cheia.

Por sua vez, a indexação em dois passos cria o ficheiro `InvertedIndex` através do `InvertedIndexBuilder`, depois de indexados todos os documentos. O `InvertedIndexBuilder` faz várias iterações porque o ficheiro com indexação directa não pode ser invertido todo de uma vez em

memória. Se usar muitos termos em cada iteração, o Terrier pode ficar sem memória, pelo que se deve diminuir a quantidade da propriedade `invertedfile.processpointers`.

Como referido, a indexação num passo é significativamente mais rápida que a indexação em dois passos, no entanto a primeira ocupa mais memória para reduzir a escrita em disco. Para além disso, quando usada para índices muito grandes, necessita de abrir muitos ficheiros, podendo atingir o limite do sistema operativo [1].

d) O que é o BlockIndexer? Podemos indexar com informação de posição? Como?

É possível indexar com informação de posição, usando o `BlockIndexer` em vez do `BasicIndexer` através da propriedade `block.indexing`. Com o `BlockIndexer`, os ficheiros *DirectIndex* e *InvertedIndex* são gerados num formato diferente que inclui os *blockids* de cada postagem. Quando se faz indexação com blocos, o `Terrier` guarda informação sobre a posição de cada termo. Se o bloco — unidade de texto num documento — for de tamanho 1, é possível determinar a posição exacta de cada termo [1].

e) Crédito Extra: Para a indexação de colecções muito grandes, está disponível o Hadoop Map-Reduce indexing. Explique sucintamente.

Hadoop Map-Reduce indexing é uma solução distribuída para a indexação de colecções de grande dimensão. Esta implementação é usada pelo `Terrier` desde a sua versão 2.2 e usa a indexação num passo para indexar secções de cada documento como *map tasks*. O resultado surge em três formas: termos e pequenas listas de postagem, índices de documentos de cada tarefa e informação sobre o número de documentos guardados em cada execução [1].

f) Uma vez feita a indexação, estamos em condições de começar a responder a interrogações.

(Considerou-se não se tratar de uma pergunta, mas apenas de uma declaração.)

1.3 Configuração de Recuperação da Informação

a) No modo batch, temos de indicar onde o sistema deve ir buscar as interrogações. Explique como.

A indicação da localização dos ficheiros que contêm as interrogações a utilizar é efectuada através da propriedade `trec.topics` [1], que deve ter como valor o *path* até ao ficheiro pretendido.

b) Como se indica que parte do ficheiro do tópico usar para construir a interrogação?

Uma vez definido o ficheiro do tópico, a identificação das partes de interesse é mediada pela alteração conforme das propriedades `TrecQueryTags.process`, `TrecQueryTags.idtag` e `TrecQueryTags.skip`, que denotam, respectivamente, as etiquetas a processar, as etiquetas que contêm os identificadores das *queries* consideradas e as etiquetas a ignorar [1].

c) Apresente quatro métodos usados para ponderação dos termos. Inclua, pelo menos, um método da família DFR. Explique, sucintamente, os métodos seleccionados.

A título de exemplo, descrevem-se os modelos de ponderação DLH, PL2, DFI0 e BM25, pertencendo os dois primeiros à classe DFR [1]:

1. **DLH**: consiste numa generalização do modelo hipergeométrico (e livre de parâmetros) do modelo DFR na vertente binomial. Contrariamente a modelos como o BM25, o modelo hipergeométrico assume que as ocorrências de um termo da interrogação num documento são amostras da totalidade da colecção e não apenas daquele documento em particular. Uma vez que não inclui uma componente de normalização da frequência de termos ou quaisquer parâmetros que requeiram afinações de relevância, o modelo DLH dispensa a necessidade de treino extensivo com juízos de relevância. Por outras palavras, todas as variáveis da fórmula de peso do documento são automaticamente definidas a partir das estatísticas da colecção [2].

2. **PL2**: um dos modelos DFR mais populares, revela-se particularmente eficiente em tarefas que requeiram elevados valores de precisão. O seu funcionamento assenta sobre o modelo aleatório de Poisson, que assume, por sua vez, que as ocorrências de um termo são distribuídas de acordo com um modelo binomial, baseando-se, à semelhança da modelação linguística, na frequência dos termos ao longo de toda a colecção para fornecer uma segregação entre os termos da interrogação semelhante àquela atingível através da *Inverse Document Frequency* (IDF) [3].

3. **DFI0**: assenta sobre o princípio de que o conteúdo em palavras de um documento pode ser identificado através da medida da divergência da independência, considerando que, se o rácio de frequências de duas palavras diferentes for constante para todos os documentos, as ocorrências dessas palavras em documentos são ditas independentes desses mesmos documentos. Ao substituir a aleatoriedade pela independência, este modelo pode ser encarado

como a contraparte do modelo DFR, segundo o qual os termos importantes de um documento são os termos cujas frequências divergem da frequência sugerida por um modelo aleatório básico — como sejam os de Poisson e Bose-Einstein. De forma constrastante, o modelo DFI0 assume que a importância dos termos diverge da frequência sugerida pelo modelo de independência [4].

4. **BM25**: baseado no modelo de independência binária de Stephen Robertson e Karen Jones, consiste numa função utilizada por motores de busca para pontuar documentos de acordo com a sua relevância face a uma dada interrogação, ignorando a proximidade relativa dos termos da interrogação que figuram em cada documento [5].

d) Explique os modelos de ponderação dependentes do campo.

O referido tipo de modelo de ponderação considera não só a presença de um termo num dado campo, mas a frequência com que este ocorre nesse mesmo campo. Assim, dado um documento D representado por n vectores, cada um dos quais correspondente a um campo, o modelo atribui uma pontuação a D e a uma dada interrogação Q, estabelecendo a distinção entre a ocorrência dos termos da interrogação nos diferentes campos e pesando a contribuição de cada um deles de forma apropriada [6]. A título de exemplo, considere-se a situação em que, num dado documento, o termo da interrogação ocorre apenas uma vez no corpo do texto. Desta forma, a probabilidade de o documento estar realmente relacionado com o termo especificado é muito baixa, situação que é invertida caso o termo ocorra, por exemplo, no título do documento [6, 1].

e) Pode-se inflacionar a pontuação dos documentos baseados na proximidade dos termos de interrogação? Como?

A inflacção da pontuação dos documentos com base na proximidade dos termos da interrogação é possível, por intermédio da utilização dos denominados Modelos de Proximidade. O funcionamento destes modelos baseia-se, então, na atribuição de pontuações elevadas a documentos nos quais os termos da interrogação se encontrem próximos, sendo que a sua utilização implica o recurso à indexação por blocos [1].

f) Como se pode alterar, no modo batch, o formato de entrada das interrogações e saídas dos resultados?

O formato de entrada das interrogações pode ser alterado por intermédio da propriedade `trec.topics.parser`, na qual se define a QuerySource a implementar. Embora a aplicação suporte apenas dois formatos de entrada (`TREC tagged` e `one query per line`), é possível proceder à implementação de outras instâncias que sejam adequadas à especificidade dos casos considerados. Por sua vez, os resultados são, por defeito, exportados no formato `TREC`, sendo possível optar por um formato alternativo através da implementação de um `OutputFormat` diferente (`TRECQuerying.NullOutputFormat`, `TRECQuerying.TRECDocidOutputFormat` ou `TRECQuerying.TRECDocnoOutputFormat`), sendo posteriormente necessário configurar o TRECQuerying de forma a utilizar o formato seleccionado, atendendo à propriedade `trec.querying.outputformat` [1].

g) Como incluiria na pontuação dos documentos um valor independente na interrogação, como o PageRank? Explique.

Torna-se possível integrar um valor independente da interrogação na pontuação dos documentos devido à existência da classe `SimpleStaticScoreModifier`. Este processo passa pela criação de um dado valor para todos os documentos da colecção e pela consequente exportação do ficheiro resultante num dos formatos suportados ('`oos`', '`docno2score`', '`docno2score_seq`' ou '`listofscores`'). Posteriormente, a adição do valor à pontuação dos documentos é efectuada através de um comando que contenha a definição das propriedades requeridas, como seja [1]:

```
bin/trec_terrier.sh -r -Dmatching.dsms=SimpleStaticScoreModifier -Dssa.input.
file=/path/to/feature -Dssa.input.type=listofscores -Dssa.w=0.5
```

No comando anterior, as propriedades `ssa.input.file`, `ssa.input.type` e `ssa.w` fazem referência, respectivamente, à definição do *path* e do tipo do ficheiro de *input*, e ao peso do valor considerado [1].

h) Que outro tipo de configuração pode ser feita que não foi apresentada?

Adicionalmente, é possível configurar aspectos como a designação dos ficheiros de resultados — que, por defeito, compreendem o nome do modelo utilizado, o valor do parâmetro de normalização da frequência de termos e um valor de contador —, através da propriedade `trec.results.file`, sendo ainda desejável, em casos de funcionamento competitivo intensivo, definir-se como aleatório o contador, de forma a evitar falhas (`trec.querycounter.type=random`). Dois outros aspectos passíveis de configuração são o número máximo de documentos recuperados por interrogação — 1000, por defeito —, que pode ser alternativamente controlado através das propriedades `matching.retrieved_set_size` ou trec.output.format.length, e o tratamento dado aos termos com uma IDF muito elevada, associado à propriedade binária (`ignore.low.idf.terms`). A um nível mais avançado, torna-se ainda possível implementar modelos de ponderação e estratégias de *matching* desenvolvidas pelo utilizador [1].

1.4 Linguagem de Interrogação

a) Explique sucintamente a linguagem de interrogação disponível no Terrier.

O *Terrier* oferece uma poderosa e flexível linguagem de interrogação que permite especificar operações adicionais às interrogações consideradas básicas. Assim, não só é possível a procura por termos mas a procura em campos particulares, frases, adicionar restrições de distância entre termos, controlo de opções e atribuição de pesos a termos específicos [1]. Através de combinações das operações básicas é possível construir interrogações mais complexas. Algumas dessas combinações estão expressas na Tabela 1. Desta forma, o *Terrier* apresenta uma flexibilidade vantajosa na construção de interrogações que consegue satisfazer as necessidades especiais na procura de documentos. Importante referir que uma interrogação é desconstruída para obter os termos únicos. A estes termos únicos podem ser atribuídas características de interrogação, algumas já mostradas na tabela 1 concluindo assim que a linguagem de interrogação do *Terrier* consegue interpretar uma interrogação não sendo apenas de termos textuais mas de termos que podem estar associados a parâmetros e/ou restrições.

Tabela 1: Exemplos da construção de interrogações no Terrier através de operações básicas

Interrogação	**Funcionalidade**
termo1	Recupera os documentos em que o termo aparece
termo1 termo2	Recupera os documentos em que pelo menos apareça um dos termos
$termo1^2.3$	O peso do termo1 é multiplicado por 2.3
$+termo1\ +termo2$	Recupera os documentos em que apareçam simultaneamente os termos
$+termo1\ -termo2$	Recupera apenas os documentos em que apareça o termo1 e não apareça o termo2
title:termo1	Recupera os documentos que contenham o termo num determinado campo (neste caso no campo *title*, porém teve que ocorrer a definição desse campo aquando a indexação)
"termo1 termo2"	Recupera os documentos em que ambos os termos estejam na mesma frase
$"termo1\ termo2"\sim n$	Recupera os documentos em que os termos apareçam pelo a uma distância *n* um do outro
control:on/off	Activa ou desactiva uma dada funcionalidade, por exemplo: *qe:on*, activa a expansão de interrogações.

1.5 Expansão de Interrogações

Crédito Extra: Explique o que é a expansão de interrogações e como está implementada no Terrier.

O Terrier dispõe também de uma funcionalidade de expansão de interrogações. Este método funciona através do acréscimo dos termos mais relevantes dos documentos mais relevantes da interrogação original a uma nova interrogação com os termos anteriores e estes mesmos novos termos. Esta nova interrogação é implementada resultando num conjunto mais rico de documentos recuperados [1].

De acordo com a literatura, o modelo usado para a ponderação de termos escolhidos está especificado no ficheiro **etc/qemodels**. Este ficheiro contém os nomes da classe dos modelos de ponderação a serem usados na expansão de interrogações. O conteúdo pré-definido é:

```
uk.ac.gla.terrier.matching.models.queryexpansion.Bo1
```

Para além disso, existem dois parâmetros que podem ser definidos para a expansão de interrogações. O primeiro é o número de termos da expansão de interrogações pela propriedade **expansion.terms** cujo valor padrão é 10. O segundo é o número de documentos com melhor classificação e a partir dos quais são extraídos os termos, sendo este parâmetro especificado por **expansion.documents** e 3 é o seu valor por defeito [1].

Para obter uma colecção teste indexada, usando a expansão de interrogações, com parâmetro de normalização de frequência igual a 1.0 pode se usar o seguinte comando:

```
bin/trec_terrier.sh -r -q -c 1.0
```

O mecanismo de expansão de interrogação extrai os termos mais relevantes dos documentos *top* retornados como os termos da expansão da interrogação [1].

Neste processo de expansão, os termos são ponderados utilizando um modelo DFR. Actualmente o *Terrier* implementa modelos como o Bo1 (Bose-Einstein 1), Bo2 (Bose.Einstein 2) e KL (Kullback-Leibler) sendo o modelo por defeito o Bo1 [1].

1.6 Avaliação

a) Que ferramentas estão disponíveis no Terrier para avaliação?

Depois de efectuado o "retrieve" de informação, o `Terrier` é capaz de avaliar os resultados presentes na directoria `/var/results`, utilizando os juízos de relevância fornecidos, neste caso, pelo ficheiro `AH-PORTUGUESE-CLEF2006.txt`. Para tal, deve ser utilizada a opção `-e`. O resultados da avaliação são guardados num ficheiro `.eval`.

Caso o objectivo não seja avaliar todos os ficheiros de resultados, mas apenas um ficheiro em particular, é possível indicar o nome do ficheiro `.res` ou o caminho para o ficheiro que pretende avaliar. Quando se utiliza a opção `-n` existem ainda mais duas opções adicionais [1, 7]:

- `-p`: guarda a precisão média de cada query em vez da precisão média de conjuntos/grupos/batch de queries;
- `-n`: avalia a tarefa de encontrar pelo nome da página; é utilizado quando o objectivo é relacionar os termos da query com o nome de uma página e não com o documento inteiro.

b) Os Runs criados pelo Terrier são compatíveis com a ferramenta trec_eval. Descreva sucintamente as funcionalidades desta ferramenta. Que resultados nos permite obter?

TREC_EVAL é uma ferramenta que permite avaliar vários sistemas de Recuperação de Informação (IR - Information Retrieval), calculando várias medidas de recuperação de informação em apenas alguns segundos. Retorna o número total de documentos para todas as *queries* - recuperados, relevantes, recuperados e relevantes - assim como, os valores médios de *recall* e precisão interpolados, a precisão média não interpolada, a precisão aos X documentos e o valor de *R-precision* - o valor da precisão quando o número de documentos recuperados coincide com o número total de documentos relevantes na coleção. É ainda capaz de retornar os valores de MAP para diferentes documentos recuperados [8].

2 Experiências com a Colecção Chave

2.1 Indexação

a) Quanto tempo demora a indexar com indexação de um/dois passo(s), com ou sem informação de posição?

Os resultados obtidos para as variantes de indexação referidas encontram-se patentes na Tabela 2 e, conforme esperado, verificam que a indexação de um passo é significativamente mais rápida do que a indexação de dois passos e, ainda, que a indexação com informação de posição é, naturalmente, mais demorada. As condições requeridas foram testadas em dois sistemas operativos distintos — Windows 7 e Mac OS X 10.7 — e, uma vez que se verificaram diferenças ao nível dos tempos obtidos, incluem-se os resultados obtidos em ambas as situações. Não obstante, o resultado comparativo globalmente obtido, e anteriormente mencionado, é semelhante.

Tabela 2: Tempos de indexação da colecção, para os tipos de indexação de um e dois passos, com e sem informação de posição

Sistema Operativo	Tipo de Indexação	Informação de Posição	Tempo (s)
Windows 7	Um Passo	Sim	282.637
Windows 7	Um Passo	Não	179.381
Windows 7	Dois Passos	Sim	734.365
Windows 7	Dois Passos	Não	239.962
Mac OS X 10.7	Um Passo	Sim	205.451
Mac OS X 10.7	Um Passo	Não	127.191
Mac OS X 10.7	Dois Passos	Sim	411.640
Mac OS X 10.7	Dois Passos	Não	196.249

2.2 Medidas

a) Vamos fazer um gráfico de precisão e recall, r-precision, MAP aos 10, 20 e 30.

De forma a obter os gráficos com as medidas requeridas, recorreu-se à ferramenta Trec_Eval, que, conforme explicitado anteriormente, é compatível com os ficheiros de resultados gerados através da aplicação Terrier, para obter uma descrição extensiva das medidas resultantes da avaliação. Assim, utilizando o ficheiro de resultados (neste caso, `PL2c1.0_0.res`) e o ficheiro QRELS (`AH-PORTUGUESE-CLEF2006.txt`), partiu-se da execução do comando:

```
/trec_eval test2/AH-PORTUGUESE-CLEF2006.txt test2/PL2c1.0_0.res -m all\_trec
```

Assim, obtiveram-se todas as medidas possíveis (`-m all_trec`), de entre as quais se utilizaram os valores de precisão interpolada (`iprec_at_recall_i`), os valores de r-precisão (`Rprec_mult_i`) e os valores de MAP (`map_cut_i`) para construir, respectivamente, os gráficos de Precision/Recall, R-Precision e MAP aos 10, 20 e 30. A Tabela 3 resume os valores das medidas consideradas mais importantes para a análise em questão, utilizadas para construir, posteriormente, os gráficos patentes na Figura 1.

Tabela 3: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `PL2`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades originalmente fornecido.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.6915	5	0.4920	0.1095	0.0913
0.10	0.5610	10	0.4480	0.1618	0.1299
0.20	0.5084	15	0.4280	0.2067	0.1594
0.30	0.4502	20	0.4090	0.2422	0.1810
0.40	0.3981	30	0.3547	0.2899	0.2099
0.50	0.3318	100	0.2192	0.4952	0.2888
0.60	0.2849	200	0.1401	0.5956	0.3119
0.70	0.2320	500	0.0706	0.7080	0.3267
0.80	0.1743	1000	0.0391	0.7623	0.3304
0.90	0.1121	-	-	-	-
1.00	0.0586	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5018	-	-
-	-	0.40	0.4791	-	-
-	-	0.60	0.4058	-	-
-	-	0.80	0.3768	-	-
-	-	1.00	0.3480	-	-

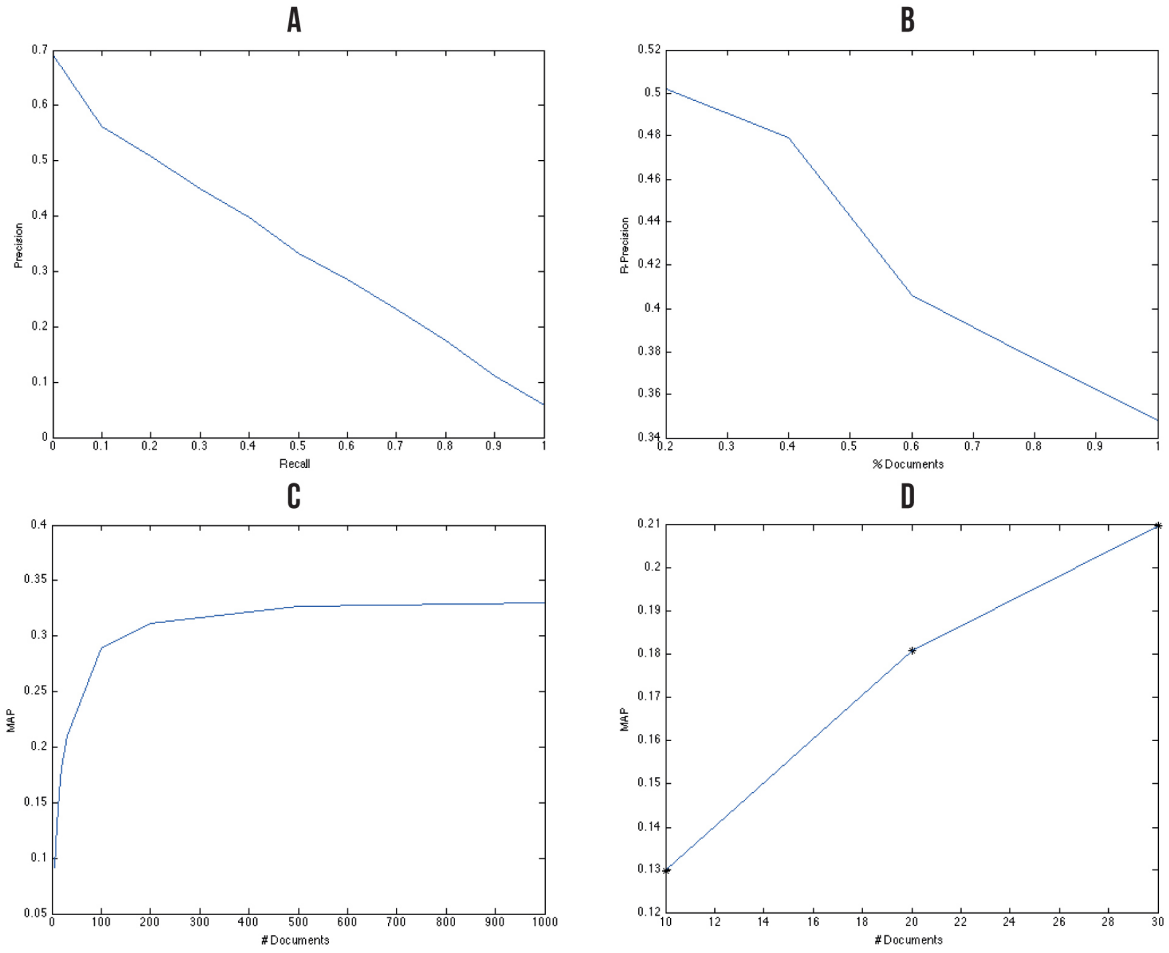

Figura 1: Gráficos de Precision/Recall (A), R-Precion (B), MAP (C) e MAP aos 10, 20 e 30 (D), para o modelo `PL2`.

2.3 Métodos de Ponderação

a) Vamos comparar dois métodos de ponderação disponibilizados no Terrier (por exemplo, TF-IDF e BM25) e, em particular, um baseado no DFR.

Em termos de utilização de métodos de ponderação, nomeadamente o `TF-IDF`, `BM25` e a versão DFR do `BM25`, consegue retirar-se da observação destes algumas ilações acerca do seu mecanismo de funcionamento. Em primeiro lugar, comparando a utlização de um modelo baseado no DFR e outro não (`BM25` e `DFR_BM25`) os resultados não apontam para diferenças significativas. No entanto os três métodos usados, em comparação com o `PL2`, apresentam melhores resultados em termos de Precision/Recall, R-Precision e valores de MAP. Estes dados podem ser consultados, respectivamente, nas Figuras 2-4. Os quatro métodos apresentam também a mesma tendência de resultados, isto é, os gráficos seguem relações aproximadas de resultados obtidos, não variando muito a forma do gráfico para qualquer um dos métodos.

Finalizando, após a utilização e avaliação dos 4 modelos utilizados, é possível deduzir que o método que apresenta melhores resultados gerais é o `DFR_BM25`.

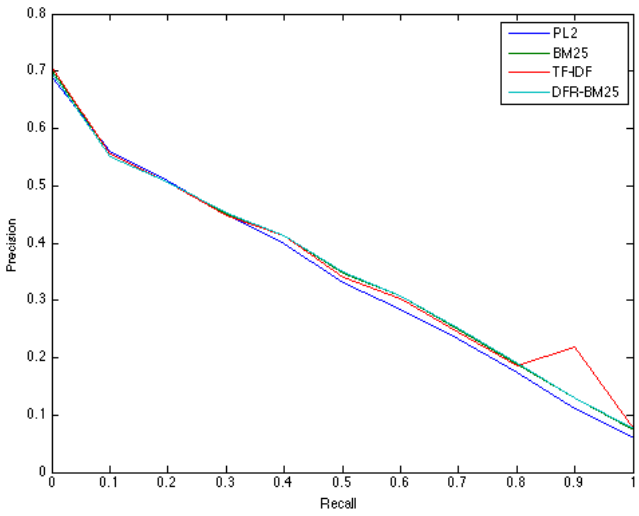

Figura 2: Gráficos de Precision/Recall obtidos para os métodos PL2, BM25, TF-IDF e DFR-BM25

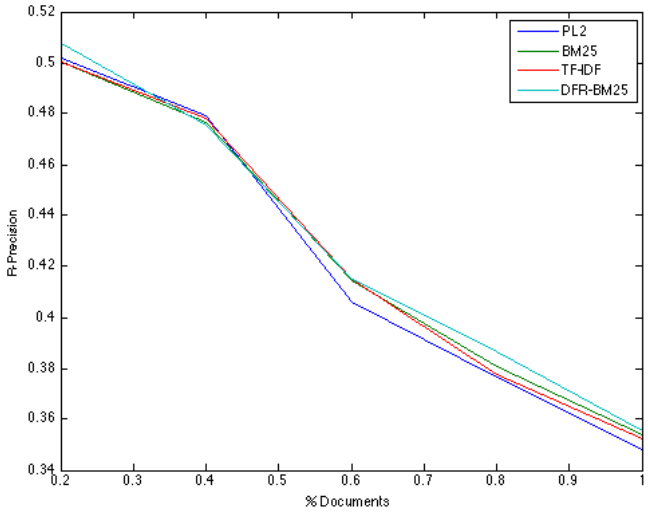

Figura 3: Gráficos de R-Precision obtidos para os métodos PL2, BM25, TF-IDF e DFR-BM25.

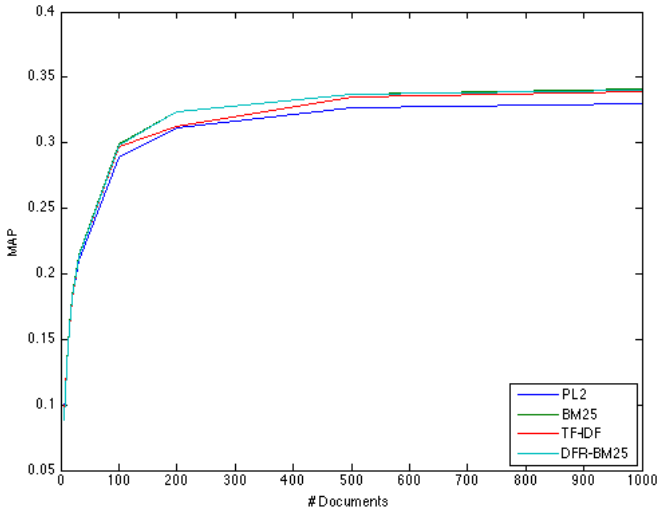

Figura 4: Gráficos de MAP obtidos para os métodos PL2, BM25, TF-IDF e DFR-BM25.

2.4 Expansão de Interrogações

a) Qual a melhoria nos resultados se usarmos expansão de interrogações?

De acordo com os resultados obtidos para a expansão de interrogações e em comparação com os obtidos na Tabela 3, correspondentes a uma interrogação normal, verifica-se que, na aplicação da expansão de interrogações, à medida que o *recall* aumenta, os valores obtidos são, de um modo geral, superiores aos equivalentes de numa interrogação normal. Tal como refere a literatura [1], a expansão de interrogações fornece um conjunto mais rico de documentos recuperados e tal facto é comprovado pelos melhores valores obtidos quando comparados a uma interrogação normal. A título de exemplo, consegue-se um valor de precisão de 0.0422 para 1000 documentos enquanto que na interrogação normal o valor é inferior, 0.0391. No entanto, esta diferença não é significativa e quando se olha para resultados de MAP esta distância entre os resultados torna-se ainda mais evidente, mais uma vez para 1000 documentos, o valor de MAP é de 0.3729 enquanto que para uma interrogação normal o valor é de 0.3304.

Em suma, a expansão de interrogações traduz, na sua maioria,uma melhoria nos resultados de precisão interpolada, precisão, *recall* em relação a uma interrogação normal.

Tabela 4: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo de expansão de interrogação `Bo1`, utilizando a ferramenta `Trec_Eval 9.0`

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.6720	5	0.4920	0.1020	0.0825
0.10	0.5858	10	0.5000	0.1822	0.1375
0.20	0.5360	15	0.4680	0.2264	0.1700
0.30	0.4911	20	0.4370	0.2650	0.1962
0.40	0.4383	30	0.3893	0.3268	0.2300
0.50	0.3827	100	0.2436	0.5447	0.3260
0.60	0.3292	200	0.1565	0.6567	0.3545
0.70	0.2917	500	0.0765	0.7612	0.3692
0.80	0.2456	1000	0.0422	0.8095	0.3729
0.90	0.1708	-	-	-	-
1.00	0.0923	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5143	-	-
-	-	0.40	0.4855	-	-
-	-	0.60	0.4419	-	-
-	-	0.80	0.4108	-	-
-	-	1.00	0.3768	-	-

2.5 Stop-Words e Stemming

a) O facto de se usar stop-words e stemming melhora os resultados?

De forma a abordar esta questão, foram testados os resultados de 4 casos de indexação diferentes:

1. Com stopwords e sem stemmer;
2. Sem stopwords e com stemmer;
3. Com stopwords e com stemmer;
4. Sem stopwords e sem stemmer.

Quando se utiliza o ficheiro de *stopwords* — lista de palavras que não devem ser indexadas porque têm alta frequência e baixo peso semântico — a indexação é mais rápida, porque a quantidade de palavras é menor. Foi possível reparar que ao fazer *retrieve* de uma indexação feita sem *stopwords* apareceram avisos informando que algumas palavras foram ignoradas do *score* por terem um IDF muito pequeno. No caso 4, aparecem 28 avisos e no caso 2 aparecem 51. Quando é utilizado o ficheiro de *stopwords*, estes avisos não aparecem porque essas palavras já foram ignoradas antes da indexação.

O uso de *stemmer* também prevê uma redução da quantidade de palavras a indexar, assim como melhores valores de *recall*. O caso 3 corresponde à situação onde o léxico tem o menor número de entradas, pelo que seria expectável que apresentasse melhores resultados. No entanto, esta conclusão não é sustentada pelos gráficos obtidos.

Ao contrário do esperado, o caso 4 não foi o que demorou mais tempo, nem o que teve mais entradas no léxico. Tal facto pode ser justificado por possíveis erros na definição das propriedades.

Pela observação dos gráficos obtidos, o caso 1 (apenas *stopwords*) apresenta resultados bastante piores do que aqueles esperados, no entanto não foi possível encontrar justificação para tal acontecimento [9].

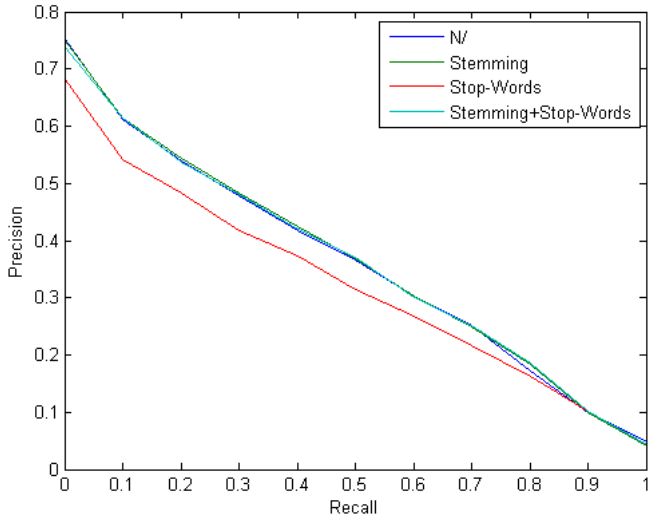

Figura 5: Gráficos de Precision/Recall obtidos para as situações de não utilização de *stop-words* ou *stemming*, para a utilização apenas de *stemming*, apenas de *stop-words* e para a utilização conjunta de *stop-words* e *stemming*

.

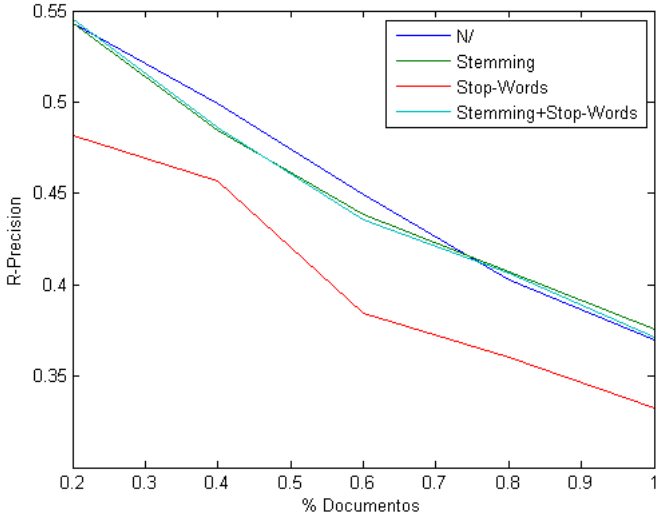

Figura 6: Gráficos de R-Precision obtidos para as situações de não utilização de *stop-words* ou *stemming*, para a utilização apenas de *stemming*, apenas de *stop-words* e para a utilização conjunta de *stop-words* e *stemming*

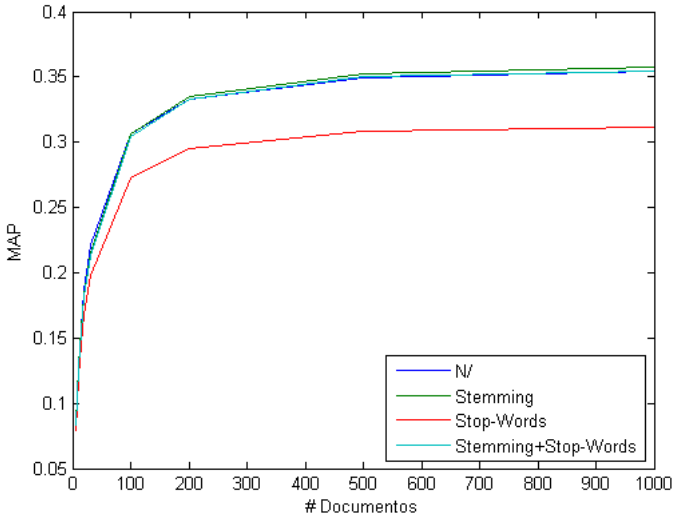

Figura 7: Gráficos de MAP obtidos para as situações de não utilização de *stop-words* ou *stemming*, para a utilização apenas de *stemming*, apenas de *stop-words* e para a utilização conjunta de *stop-words* e *stemming*

3 Referências

[1] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, 2006.

[2] B. Hen and I. Ounis. Combining Fields for Query Expansion and Adaptive Query Expansion. *Information Processing and Management*, 43:1294–1307, 2007.

[3] C. Macdonald and I. Ounis. Global Statistics in Proximity Weightinh Models. SIGIR 2010 WEB N-GRAM Workshop, 2010.

[4] B. Dinçer, Í. Kocabas, and B. Karaoğlan. Index Term Weighting by Divergence From Independence Model. IRRA at TREC 2010, 2010.

[5] G. Bennett, F. Scholer, and A. Uitdenbogerd. A Comparative Study of Probabilistic and Language Models for Information Retrieval. In *Proceedings of the Nineteenth Conference on Australasian Database - Volume 75*, pages 65–74. Australian Computer Society, Inc., 2007.

[6] R. Santos, R. McCreadie, and V. Plachouras. Large-scale Information Retrieval Experimentation with Terrier. In *CIKM*, pages 2601–2602, 2011.

[7] K. Collins-Thompson, P. Ogilvie, Y. Zhan, and J. Callan. Information Filtering, Novelty Detection and Named-Page Finding.

[8] F. Scholer. Trec_Eval, 2012.

[9] AmareshKumar Pandey and TanvverJ Siddiqui. Evaluating Effect of Stemming and Stop-word Removal on Hindi Text Retrieval, 2009.

ANEXOS

Anexos

I Métodos de Ponderação

Os valores obtidos para as medidas consideradas mais relevantes relativas aos métodos de ponderação `BM25`, `TF-IDF` e `DFR_BM25` encontram-se respectivamente representadas nas Tabelas 5-7.

Tabela 5: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `BM25`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades originalmente fornecido.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.7023	5	0.4720	0.1009	0.0893
0.10	0.5565	10	0.4540	0.1623	0.1304
0.20	0.5068	15	0.4293	0.2078	0.1598
0.30	0.4497	20	0.4100	0.2474	0.1825
0.40	0.4121	30	0.3653	0.3023	0.2140
0.50	0.3471	100	0.2250	0.5058	0.2989
0.60	0.3071	200	0.1443	0.6129	0.3237
0.70	0.2501	500	0.0712	0.7135	0.3370
0.80	0.1892	1000	0.0395	0.7716	0.3408
0.90	0.1287	-	-	-	-
1.00	0.0748	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5005	-	-
-	-	0.40	0.4765	-	-
-	-	0.60	0.4145	-	-
-	-	0.80	0.3808	-	-
-	-	1.00	0.3538	-	-

Tabela 6: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `TF-IDF`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades originalmente fornecido.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.7066	5	0.4800	0.1035	0.0901
0.10	0.5566	10	0.4560	0.1624	0.1307
0.20	0.5054	15	0.4240	0.2047	0.1582
0.30	0.4491	20	0.4080	0.2403	0.1805
0.40	0.4130	30	0.3613	0.3005	0.2128
0.50	0.3413	100	0.2246	0.5031	0.2976
0.60	0.3026	200	0.1436	0.6060	0.3219
0.70	0.2441	500	0.0707	0.7101	0.3349
0.80	0.1858	1000	0.0393	0.7706	0.3389
0.90	0.2187	-	-	-	-
1.00	0.0755	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5005	-	-
-	-	0.40	0.4782	-	-
-	-	0.60	0.4149	-	-
-	-	0.80	0.3776	-	-
-	-	1.00	0.3524	-	-

Tabela 7: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `DFR_BM25`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades originalmente fornecido.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.6971	5	0.4800	0.1015	0.0885
0.10	0.5500	10	0.4520	0.1631	0.1298
0.20	0.5060	15	0.4280	0.2074	0.1585
0.30	0.4519	20	0.4080	0.2463	0.1810
0.40	0.4120	30	0.3647	0.3022	0.2129
0.50	0.3495	100	0.2254	0.5070	0.2983
0.60	0.3083	200	0.1449	0.6169	0.3237
0.70	0.2511	500	0.0714	0.7151	0.3368
0.80	0.1898	1000	0.0395	0.7725	0.3405
0.90	0.1305	-	-	-	-
1.00	0.0764	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5077	-	-
-	-	0.40	0.4753	-	-
-	-	0.60	0.4148	-	-
-	-	0.80	0.3868	-	-
-	-	1.00	0.3555	-	-

II Stop-Words e Stemming

Os valores obtidos para as medidas consideradas mais relevantes relativas às diferentes situações de experimentação com a utilização e a ausência de *stop-words* e *stemming* encontram-se representadas nas Tabelas 8-11.

Tabela 8: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `PL2`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades alterado no sentido da não utilização explícita de *stemming* e *stop-words*.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.7535	5	0.5440	0.0977	0.0844
0.10	0.6126	10	0.5060	0.0.1647	0.1294
0.20	0.5386	15	0.4693	0.2076	0.1578
0.30	0.4789	20	0.4500	0.2651	0.1862
0.40	0.4179	30	0.4047	0.3255	0.2214
0.50	0.3665	100	0.2368	0.5213	0.3063
0.60	0.3037	200	0.1544	0.6258	0.3331
0.70	0.2514	500	0.0780	0.7437	0.3491
0.80	0.1724	1000	0.0442	0.8109	0.3546
0.90	0.0996	-	-	-	-
1.00	0.0470	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5432	-	-
-	-	0.40	0.4995	-	-
-	-	0.60	0.4420	-	-
-	-	0.80	0.4026	-	-
-	-	1.00	0.3696	-	-

Tabela 9: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `PL2`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades alterado no sentido da utilização de *stemming*.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.7508	5	0.5480	0.0966	0.0841
0.10	0.6130	10	0.4940	0.1699	0.1293
0.20	0.5444	15	0.4627	0.2133	0.1576
0.30	0.4831	20	0.4450	0.2608	0.1828
0.40	0.4255	30	0.3953	0.3182	0.2154
0.50	0.3694	100	0.2428	0.5276	0.3061
0.60	0.3035	200	0.1600	0.6362	0.3353
0.70	0.2481	500	0.0807	0.7542	0.3523
0.80	0.1834	1000	0.0447	0.8130	0.3571
0.90	0.0994	-	-	-	-
1.00	0.0410	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5436	-	-
-	-	0.40	0.4847	-	-
-	-	0.60	0.4385	-	-
-	-	0.80	0.4068	-	-
-	-	1.00	0.3757	-	-

Tabela 10: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `PL2`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades alterado no sentido da utilização de *stop-words*.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.6839	5	0.4640	0.0968	0.0787
0.10	0.5413	10	0.4300	0.1624	0.1196
0.20	0.4837	15	0.4053	0.2030	0.1460
0.30	0.4166	20	0.3850	0.2448	0.1683
0.40	0.3728	30	0.3407	0.2992	0.1981
0.50	0.3143	100	0.2026	0.4858	0.2726
0.60	0.2680	200	0.1312	0.5971	0.2956
0.70	0.2173	500	0.0657	0.7053	0.3083
0.80	0.1628	1000	0.0364	0.7624	0.3118
0.90	0.1009	-	-	-	-
1.00	0.0430	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.4817	-	-
-	-	0.40	0.4565	-	-
-	-	0.60	0.3846	-	-
-	-	0.80	0.3605	-	-
-	-	1.00	0.3326	-	-

Tabela 11: Medidas de Recall, Precision, MAP e R-Precision obtidas para o modelo `PL2`, utilizando a ferramenta `Trec_Eval 9.0` e o ficheiro de propriedades alterado no sentido da utilização conjunta de *stemming* e *stop-words*.

Recall	**Interpolated Precision**	**# Documents**	**Precision**	**Recall**	**MAP**
0.00	0.7399	5	0.5320	0.0955	0.0828
0.10	0.6150	10	0.4960	0.1696	0.1283
0.20	0.5364	15	0.4587	0.2099	0.1548
0.30	0.4810	20	0.4420	0.2597	0.1813
0.40	0.4205	30	0.3933	0.3176	0.2137
0.50	0.3698	100	0.2438	0.5302	0.3040
0.60	0.3017	200	0.1606	0.6376	0.3332
0.70	0.2519	500	0.0806	0.7533	0.3501
0.80	0.1860	1000	0.0447	0.8128	0.3549
0.90	0.1009	-	-	-	-
1.00	0.0432	-	-	-	-
-	-	**% Documents**	**R-Precision**	-	-
-	-	0.20	0.5458	-	-
-	-	0.40	0.4961	-	-
-	-	0.60	0.4357	-	-
-	-	0.80	0.4062	-	-
-	-	1.00	0.3714	-	-