DeDup2

Definição

DeDup2 é uma versão melhorada do software de deduplicação de documentos implementada pela BIREME/OPAS/OMS onde a principal diferença em relação a primeira versão é que o software apresenta uma lista dos documentos duplicados enquanto que na segunda, o software apresenta uma lista de documentos em que um determinado campo está duplicado (geralmente o campo titulo) e a comparação se os demais campos estão duplicados ou não deixando para o usuário a decisão se os dois documentos estão duplicados.
Com esta estratégia, o software permite que o usuário utilize seu próprio conjunto de regras baseadas nas similaridades dos campos para decidir se dois documentos estão duplicados ou não. Somente o usuário tem o conhecimento de quais campos de dois documentos têm de ser idênticos ou com qual grau de similaridade para que os mesmos sejam considerados duplicados.

Funcionamento geral

Sempre que buscamos documentos duplicados neste software, estamos buscando, na realidade documentos pertencentes a um conjunto de documentos de entrada (na atual implementação, documentos pertencentes a um arquivo tipo csv) em relação aos documentos que estão armazenados em um índice Lucene. Desnecessário dizer que todos os documentos precisam ter um conjunto de campos em comum para poderem ser comparados.
O primeiro passo então é criar um índice Lucene com os documentos que serão comparados com os outros que serão entrados por meio de arquivo de entrada. Todos os documentos da coleção serão indexados por um determinado campo para que o software procure para cada documento do arquivo de entrada, todos documentos que tenham esse campo muito similar e a partir daí gere um relatório de saída contendo o identificador de cada documento de entrada considerado duplicado assim como a similaridade de alguns de seus campos com os campos de documentos indexados considerados como possíveis similares. De posse deste relatório, o usuário pegará cada par de documentos possivelmente duplicados (baseado na similaridade do campo indexado) e aplicará suas regras baseadas na similaridade dos demais campos para concluir se os dois documentos são duplicados ou não.

Resumindo

1) Para cada documento do arquivo de entrada (geralmente arquivo csv) pega um campo pre-definido e procura no indice todos os documentos cujo o campo tem uma similaridade muito grande.
2) Compara a similaridade dos demais campos (definidos conforme um arquivo de configuracao) entre os dois documentos e apresenta tais similaridades em um relatório de saida.
3) O usuario de posse de tal relatório executa seu algoritmo onde compara as similaridades de vários campos dos dois documentos para finalmente decidir se os documentos são similares/duplicados ou não.

Utilizando os aplicativos

O primeiro passo para o processo de deduplicação é a geração de um índice Lucene contendo um conjunto de documentos a serem comparados com os outros documentos de um arquivo de entrada.

Uma das maneiras de se fazer isso é utilizando o aplicativo CSV2Lucene presente no diretorio bin deste projeto. Neste aplicativo que será explicado posteriormente com mais detalhes, fornecemos um arquivo de entrada no formato csv contendo o identificador de cada documento e seus os campos que serão utilizados para a comparação de similaridade além do nome do índice Lucene a ser criado. Vale a pena dizer que para cada documento que entra na criação do índice, apenas um de seus campos é indexado (este é que vai ser utilizado para a recuperação de todos os documentos do índice que tenham tal campo muito similar em relação a ao campo de um documento do arquivo de entrada), os demais campos serão apenas armazenados no índice.

O segundo passo é a criação de um arquivo csv contendo o outro conjunto de documentos que serão comparados com os armazenados no índice criado no primeiro passo. Tal arquivo terá estrutura semelhante ao primeiro, isto é, um campo contendo o identificador e os outros contendo os campos que serão utilizados para a comparação entre documentos.

O terceiro passo é a comparação propriamente dita entre os documentos do arquivo csv de entrada e os documentos armazenados no índice. Para tanto, utilizaremos outro aplicativo denominado SimilarDocs cuja chamada será explicada mais a frente, mas que simplificadamente tem como parâmetros de entrada o arquivo csv com os documentos, o caminho para o índice Lucene, e um arquivo de configuração onde se indicará como e quais campos serão comparados e o local do relatório de saída.

O passo final é a utilização do relatório para a aplicação das regras criadas pelo usuário para decidir se os documentos presentes em cada para presente no relatório são duplicados ou não.

Arquivo de configuração

O aplicativo SimilarDocs descrito a seguir tem com um de seus parâmetros de entrada um arquivo de configuração que em termos gerais indica a localização do índice Lucene, os comparadores de campos entre os documentos e a localização do arquivo de relatório de saída.
Os comparadores de campos são utilizados para descrever a similaridade de um mesmo campo entre dois documentos diferentes, cada comparador utiliza um método diferente para o cálculo de similaridade sendo que alguns são do tipo binário (0 – campos não similares e 1 – campos similares) enquanto outros utilizam um valor real entre 0 e 1 para indicar o grau de similaridade. Cabe ao usuário escolher o tipo de comparador que mais adéqua para um determinado campo, por exemplo, para o campo ano de publicação o comparador deve ser binário (ou os anos são iguais ou não) enquanto que para o campo revista de publicação o comparador deve ser não binário para poder lidar com casos de erros de digitação.

A parte do arquivo de configuração que especifica os comparadores segue a seguinte estrutura:

```
"comparators" : [
 "exact" : {
  "fieldName" : "",
  "normalize" : true
 },
 "dice" : {
  "fieldName" : "",
  "normalize" : true,
  "minSimilarity" : 80.5
 },
 "ngram" : {
```

```
  "fieldName" : "",
  "normalize" : true,
  "minSimilarity" : 80.5
 },
 "regex" : {
  "fieldName" : "",
  "normalize" : true,
  "regex" : "",
  "compString": ""
 },
 "authors" : {
  "fieldName" : ""
 }
],
```

Faremos a seguir uma descrição de cada um dos comparadores implementados até a atual versão do software.

a) exact – comparador binário que compara se duas strings são iguais ou não. Possui dois parâmetros: ‘fieldName’ – indica o nome do campo a ser comparado e ‘normalize’ cujos valores podem ser ‘true’ or ‘false’ e que indicam se o conteúdo dos campos deve ser normalizado antes da comparação ou não. A normalização do campo implica na mudança de todas as letras maiúsculas para minúsculas e também na eliminação de todos os caracteres que não sejam letras e números.

b) dice – comparador não binário que compara duas string utilizando-se do algoritmo dice score (https://vickumar1981.github.io/stringdistance/api/com/github/vickumar1981/stringdistance/implicits/ScoreDefinitions$DiceCoefficientScore$.html). Possui três parâmetros: ‘fieldName’ – indica o nome do campo a ser comparado, ‘normalize’ cujos valores podem ser ‘true’ or ‘false’ que indicam se o conteúdo dos campos deve ser normalizado antes da comparação ou não e ‘minSimilarity’ que especifica um valor real entre 0 e 100 indicando qual a menor similaridade aceita para os campos serem considerados similares.

c) ngram - comparador não binário que compara duas string utilizando-se do algoritmo ngram (https://vickumar1981.github.io/stringdistance/api/com/github/vickumar1981/stringdistance/implicits/ScoreDefinitions$NGramScore$.html). Possui três parâmetros: ‘fieldName’ – indica o nome do campo a ser comparado, ‘normalize’ cujos valores podem ser ‘true’ or ‘false’ que indicam se o conteúdo dos campos deve ser normalizado antes da comparação ou não e ‘minSimilarity’ que especifica um valor real entre 0 e 100 indicando qual a menor similaridade aceita para os campos serem considerados similares.

d) regex – comparador binário que utiliza expressão regular para extrair o conteúdo dos campos e depois realizar uma comparação de strings. Possui quatro parâmetros: ‘fieldName’ – indica o nome do campo a ser comparado, ‘normalize’ cujos valores podem ser ‘true’ or ‘false’ que indicam se o conteúdo dos campos deve ser normalizado antes da comparação ou não, ‘regex’ que especifica a expressão regular a ser utilizada. Note que tal expressão pode conter ‘capture groups’ para se extrair uma ou mais substrings desejadas do campo. O último parâmetro é o ‘compString’ que indica o formato final da string a ser comparada que pode conter $<number> para se referir a um determinado ‘capture group’ ou não quando este parâmetro substituirá a string recuperada pela expressão regular

e) authors – comparador não binário especializado na comparação de autores. Possui apenas um parâmetro que é o ‘fieldName’ – indica o nome do campo a ser comparado.

Além dos comparadores (comparators) o arquivo de configuração requer a especificação da localização do índice Lucene, o campo a ser indexado e da localização do relatório de saída.

A especificação da localização do índice Lucene é feita no campo ‘index’ e o campo a ser indexado no campo ‘searchField’ seguindo a estrutura abaixo.

```
"finder" : {
 "lucene" : {
  "index": ""
  "searchField": "",
 }
}
```

A localização do relatório de saída é indicada no campo ‘file’. Outros parâmetros devem também ser especificados: ‘encoding’ indicando a codificação de caracteres do arquivo de saída, ‘record separator’ indicando o carácter a ser utilizado para a separação dos campos, ‘putHeader’ indicando se um cabeçalho deva ser incluído no arquivo (true) ou não (false) e o ‘minTrue’ indicando o número mínimo de campos similares para um par de documentos ser considerado como candidato a duplicado e portanto ser incluído no arquivo de saída. A estrutura dos parâmetros é indicada abaixo:

```
"reporters" : [
 "pipe" : {
  "file" : "",
  "encoding"; "",
  "recordSeparator" : "",
  "putHeader" : true,
  "minTrue" : 5
 }
]
```

A estrutura final do arquivo de configuração pode ser vista na figura abaixo:

```
{
 "finder" : {
  "lucene" : {
   "index": ""
   "searchField": "",
  }
 }

 "comparators" : [
  "exact" : {
   "fieldName" : "",
   "normalize" : true
  },
  "dice" : {
   "fieldName" : "",
   "normalize" : true,
   "minSimilarity" : 80.5
  },
```

```
  "ngram" : {
    "fieldName" : "",
    "normalize" : true,
    "minSimilarity" : 80.5
  },
  "regex" : {
    "fieldName" : "",
    "normalize" : true,
    "regex" : "",
    "compString": ""
  },
  "authors" : {
    "fieldName" : ""
  }
 ],

 "reporters" : [
  "pipe" : {
    "file" : "",
    "encoding"; "",
    "recordSeparator" : "",
    "putHeader" : true,
    "minTrue" : 5
  }
 ]
}
```

CSV2Lucene

O aplicativo CSV2Lucene utiliza como entrada um arquivo do tipo CSV (comma separated value) contendo os campos dos documentos para criar um índice Lucene correspondente. Está presente no diretório bin (arquivo CVS2Lucene.sh) e tem como parâmetros:

csvFile – caminho/nome do arquivo csv contendo os campos dos documentos separados por vírgula ou por outro carácter separador de campo (veja parâmetro fieldSeparator)

index – caminho para o índice Lucene a ser criado

schema – associa posições dos campos no arquivo csv com o nome dos campos a serem criados no índice. Tem o seguinte formato:

<pos>=<nome do campo>,<pos>=<nome do campo>,…,<pos>=<nome do campo>

onde <pos> é a posição do campo no csv (primeira posição é zero) e <nome do campo> indica o nome do campo do documento no índice Lucene. As ocorrências de <pos>=<nome do campo> são separadas por vírgulas. Observe que somente as posições especificadas serão exportadas para o índice.
Uma outra forma de entrar os dados seria escrevê-los em um arquivo e depois passar o caminho do mesmo utilizando o seguinte formato no parâmetro:

-schema=file=<path to schema file>

onde cada linha do arquivo teria o seguinte formato:
<*pos*>=<*nome do campo*>

fieldToIndex – nome do campo (conforme descrito no parâmetro schema) que será indexado no índice e utilizado para se recuperar documentos que contém este campo similar. Somente um campo podera ser indexado.

fieldSeparator – parâmetro opcional utilizado para especificar o carácter a ser utilizado para se separar os campos no arquivo de entrada csv. Se ausente, o separador padrão será a vírgula.

encoding – parâmetro opcional utilizado para especificar a codificação de caracteres utilizada no arquivo csv de entrada. Se ausente, a codificação utilizada será a utf-8.

hasHeader - parâmetro opcional utilizado para especificar se o arquivo csv de entrada tem como primeira linha o nome dos campos. Se presente, a linha de cabeçalho será desconsiderada. Se ausente, a primeira linha do arquivo será considerada como sendo a do primeiro documento.

Exemplo de chamada:

bin/CSV2Lucene.sh -csvFile=/home/data/documents.csv -index=./indexes/myIndex -schema=2=title,3=date,5=authors,0=id -fieldToIndex=title -fieldSeparator=; -encoding=iso-8859-1 --hasHeader

SimilarDocs

O aplicativo SimilarDocs é o responsável por dado um arquivo de entrada com documentos e um índice Lucene também contendo documentos, comparar cada par de documentos pertencente aos dois conjuntos e produzir um relatório de saída contendo os possíveis pares de documentos duplicados. Está presente no diretório bin do projeto (arquivo SimilarDocs.sh) e tem como parâmetros:

inputPipeFile – caminho/nome do arquivo de entrada no formato csv contendo os campos de cada documento que se buscará possíveis duplicados.

schema – associa posições dos campos no arquivo csv com o nome dos campos presentes no índice. Tem o seguinte formato:
<pos>=<nome do campo>,<pos>=<nome do campo>,…,<pos>=<nome do campo> onde <pos> é a posição do campo no csv (primeira posição é zero) e <nome do campo> indica o nome do campo do documento no índice Lucene. As ocorrências de <pos>=<nome do campo> são separadas por vírgulas.

confFile – caminho/nome do arquivo de configuração descrito na seção *Arquivo de configuração*

otherFields – nome de campos do documento separados por vírgulas que não estão descritos no parâmetro schema mas que devem aparecer no relatório final (sem a aplicação de um comparador)

maxDocs – parâmetro opcional que descreve o número máximo de candidatos a documentos duplicados do índice para cada documento do conjunto de entrada. Este parâmetro existe para efeitos de desempenho do software e tem como valor padrão o número 100.

encoding – parâmetro opcional que indica a codificação de caracteres que deve ser utilizada para a leitura do arquivo de entrada. Valor padrão é o utf-8.

fieldSep - parâmetro opcional que indica o carácter a ser utilizado como separador de campos no arquivo de saída. Valor padrão é o carácter pipe '|'.

hasHeader - parâmetro opcional que indica se uma linha cabeçalho contendo o nome dos campos deva ser colocada no arquivo de saída. Se presente, o parâmetro indica que o cabeçalho será acrescentado, se ausente indica que não.

Exemplo de chamada:

bin/SimilarDocs.sh -inputPipeFile=/home/in/input.csv
-schema=2=title:3=date:5=authors:0=id -confFile=/home/conf/configuration.json
-otherFields=doi,language -fieldSep=; --hasHeader

Determinação de documentos duplicados em um conjunto de documentos

Este é caso particular onde não se está procurando documentos duplicados dentre os armazenados em um arquivo de entrada e os armazenados em um índice Lucene, mas sim os documentos duplicados dentre um conjunto específico.
Para acharmos os documentos duplicados neste caso em particular, precisamos transformá-lo no caso geral para podermos encontrar as duplicações.

Para tanto:
- Criamos um arquivo csv de entrada contendo os campos dos documentos e seus identificadores.
- Utilizamos o arquivo para gerar o índice Lucene
- Utilizamos o mesmo arquivo como entrada no aplicativo SimilarDocs
- Retiramos do arquivo de saida os documentos duplicados contendo os mesmos ids.

Conclusões

O DeDup2 é uma ferramenta para checagem de deduplicação de documentos que partiu da versão anterior (DeDup) e que busca dar mais poder ao usuário permitindo ao mesmo a inclusão de suas próprias regras heurísticas na decisão sobre as possiveis duplicações entre documentos.
Por estar ainda em fase de desenvolvimento, a ferramenta ainda requer um grupo interno de testes que deve validar a correção na execução da ferramenta, desenvolver suas regras heurísticas e ainda fornecer sugestões para aprimoramento do software.
Espera-se em pouco tempo que a versão 2 possa superar a versão 1 em termos de qualidade de resposta e que possa se equiparar com ferramentas análogas de terceiros tais como o Deduklick.