

Projeto Final de Programação

Marcelo Costalonga - 2120593
Orientador: Marcos Kalinowski

Dezembro 2022

Contexto

O objetivo dessa pesquisa é investigar mais a fundo formas de apoiar pesquisadores da área de Engenharia de Software (ES) durante a atualização de uma revisão sistemática da literatura acadêmica (*Systematic Literature Review* – SLR) através do uso de Machine Learning (ML) para a detecção de textos relevantes para uma atualização. Visto que atualização de SLR muitas vezes são feitas manualmente, por apenas um pesquisador e acabam sendo uma tarefa bem extensa.

Esse busca trazer benefícios através do uso de ML e técnicas de Natural Language Processing (NLP) para reduzir o esforço manual de pesquisadores durante a atualização de uma SLR e também se seria possível usar ML para verificar a concordância entre os algoritmos de ML e os demais revisores.

1. Especificação do Programa

Escopo

Nesse trabalho, realizamos um experimento comparando o desempenho de algoritmos de dois algoritmos de ML, Decision Tree (DT) e Support Vector Machine (SVM), para seleção de textos da atualização de uma SLR. Foram usadas técnicas de NLP para processamento e análise de títulos e abstracts de múltiplos artigos.

Utilizamos os dados obtidos pelo resultado de uma atualização de uma SLR realizada por três especialistas na área de ES. Dessa forma, os modelos de ML foram treinados com todos os artigos relacionados (obtidos pela estratégia de backward-snowballing) antes do ano de sua atualização e depois foram feitas previsões de quais artigos deveriam ser incluídos ou excluídos pelos algoritmos, utilizando os artigos posteriores ao ano de publicação inicial da SLR. Todos os dados foram fornecidos pelos revisores dessa atualização, para simularmos exatamente o mesmo comportamento.

Requisitos

- O programa deve ser capaz validar o formato de arquivos .bib passados como entrada (é necessário que cada artigo de um arquivo .bib, contenha pelo menos as chaves 'title', 'author', 'year' e 'abstract') e informar erros como falta chaves, ou artigos duplicados. É necessário especificar quais artigos foram excluídos/incluídos e assim como quais devem ser usados para treinamento e para testes dos algoritmos de ML (por isso são esperados quatro arquivos .bib distintos).
- O programa deve ser capaz de informar erros de formatação dos arquivos .bib de entrada e casos de arquivos duplicados
- O programa deve ser capaz de processar arquivos .bib válidos, filtrar textos das entradas através do uso de NLP ,
- O programa deve ser capaz vectorizar os textos filtrados para determinar o número de “features” (palavras de maior peso) de cada artigo dos arquivos .bib, a partir de um valor informado pelo usuário (representando o número de features daquela execução)
- O programa deve ser capaz de treinar os algoritmos de ML (DT e SVM) considerando o número de features informado pelo usuário e gerar arquivos .csv contendo as respostas de cada algoritmo para que o usuário possa analisá-las posteriormente.

2. Código Fonte

Todo código desenvolvido para esse experimento foi feito em python. Inicialmente, para facilitar a visualização e exploração dos dados, foi criado um notebook utilizando a plataforma do Google Colab (<https://colab.research.google.com/>). Posteriormente, criamos um projeto python a partir do notebook para simular o funcionamento de um pipeline (com o mesmo propósito do notebook, mas facilitando a reprodutibilidade do experimento).

Vale ressaltar que parte do código utilizado em nosso experimento foi adaptado do repositório: <https://github.com/watinha/automatic-selection-slr>. Esse repositório é referente ao experimento de outros pesquisadores, que também investigaram o uso de ML para automatização de seleção de SLR (artigo: [Reducing efforts of software engineering systematic literature reviews updates using text classification](#)).

Nosso projeto python e o notebook encontram-se disponível em no repositório: https://github.com/bmnapoleao/SLR-Automated_selection_of_studies/tree/projeto-final-programacao.

Em cada módulo do projeto python, na primeira linha há um comentário indicando os autores de cada arquivo e em caso de adaptação, exatamente quais partes do código foram adaptadas a partir desse outro repositório.

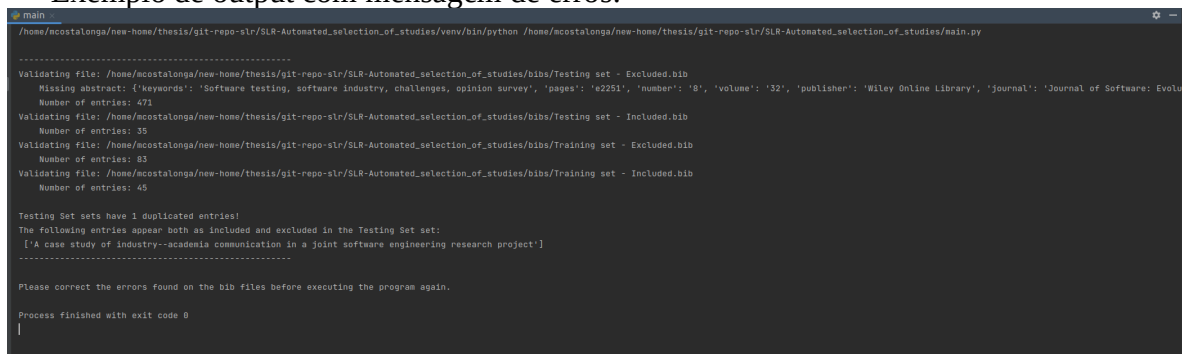
3. Projetos do programa

O [projeto](#) simula um a execução de um pipeline composto por seis etapas, na raiz do projeto há um arquivo README.md com instruções de como executá-lo. No diretório [/bibs](#) encontram-se os artigos utilizados nos experimentos:

Há um arquivo “main.py” que instancia e executa cada uma das etapas do pipeline, representado por seis arquivos python distintos.

1. Inicialmente o módulo “[InputValidator](#)” irá ler e validar os arquivos .bib de entrada. Caso seja encontrado algum erro, será indicado como output para usuário e o programa será abortado.

Exemplo de output com mensagem de erros:



```
main
/home/mcostalonga/new-home/thesis/git-repo-slr/SLR-Automated_selection_of_studies/venv/bin/python /home/mcostalonga/new-home/thesis/git-repo-slr/SLR-Automated_selection_of_studies/main.py

-----
Validating file: /home/mcostalonga/new-home/thesis/git-repo-slr/SLR-Automated_selection_of_studies/bibs/Testing set - Excluded.bib
Missing abstract: ['Keywords': 'Software testing, software industry, challenges, opinion survey', 'pages': '62251', 'number': '8', 'volume': '32', 'publisher': 'Wiley Online Library', 'journal': 'Journal of Software: Evolu
Number of entries: 471
Validating file: /home/mcostalonga/new-home/thesis/git-repo-slr/SLR-Automated_selection_of_studies/bibs/Testing set - Included.bib
Number of entries: 35
Validating file: /home/mcostalonga/new-home/thesis/git-repo-slr/SLR-Automated_selection_of_studies/bibs/Training set - Excluded.bib
Number of entries: 83
Validating file: /home/mcostalonga/new-home/thesis/git-repo-slr/SLR-Automated_selection_of_studies/bibs/Training set - Included.bib
Number of entries: 45

Testing Set sets have 1 duplicated entries!
The following entries appear both as included and excluded in the Testing Set set:
['A case study of industry-academia communication in a joint software engineering research project']
-----

Please correct the errors found on the bib files before executing the program again.

Process finished with exit code 0
```

2. Caso os arquivos de entrada estejam válidos, será executado o módulo de “[TextFiltering](#)”, onde serão aplicadas técnicas de NLP para filtrar o conteúdo dos títulos e asbract de cada artigo, removendo alguns caracteres desnecessários.
3. Posteriormente, será executado o módulo “[DatasetGenerator](#)”, que irá receber listas com os conteúdos dos arquivos .bib e irá aplicar técnicas de vectorização de texto, para gerar “feautres” do conteúdo de cada artigo. Ao final retorna dicionários python representando os datasets com o conteúdo de todos os artigos e suas features.
4. Em seguida, será executado o módulo de “[Seleção das melhores N Features](#)” (correspondente ao valor passado pelo usuário no começo do programa), para determinar as palavras de maior peso. (OBS: Esse módulo pode demorar bastante tempo para ser

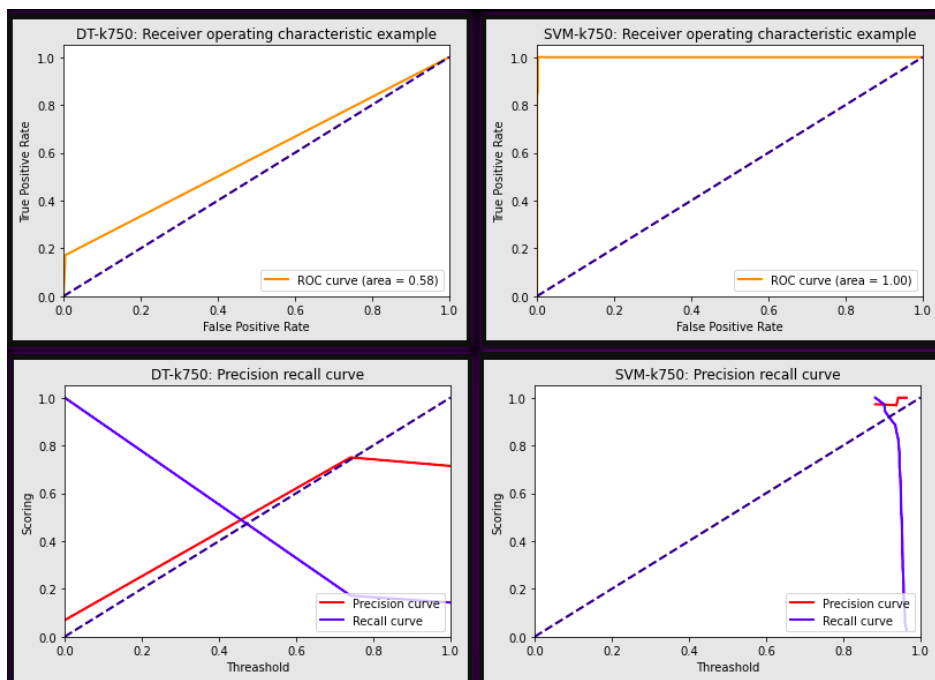
processado, devido a complexidade do método de “feature selection”, nossos testes levaram de 20min à 45min para processarem essa etapa).

5. Será executado o [módulo de classificadores de ML](#), cada classificador será treinado e depois irá prever quais artigos deveriam ser incluídos e quais deveriam ser excluídos. Informando na tela o número de falso positivos, falso negativos, precision e recall de cada classificador.
6. Ao final será executado o módulo de [Report](#) que irá analisar os resultados de cada classificador, informando número de falsos positivos e negativos, assim como os resultados corretos. Também será gerado um arquivo .csv ao final de cada execução que teve sucesso, contendo informações linha por linha de quais artigos foram considerados selecionados ou não pelos classificadores.

4. Testes

Realizamos diversas execuções comparando seus resultados, variando os valores dos números de features em cada execução, assim como alguns parâmetros de configuração dos classificadores de ML para verificar seu desempenho. Foram reparadas pequenas diferenças em algumas execuções feitos no notebook e no pipeline (possivelmente devido a pequenas diferenças no ambiente python do notebook e de suas dependências). Mas em ambos os casos conseguimos avaliar que o SVM obteve um desempenho melhor que a DT.

Segue abaixo, alguns exemplos de métricas avaliadas ao final de algumas de nossas execuções, nos notebooks:



Decision Tree vs SVM

In [45]:

```
df_testing.head(20)
```

Out[45]:

	texts	features	categories	years	DT_pred	SVM_pred
0	planning unknown : lessons learned ten months ...	(0, 21)w0.14958509341845988ln (0, 24)w0.0...	1	2015	1	1
1	exploratory study technology transfer software...	(0, 17)w0.14232047367077214ln (0, 21)w0.2...	1	2015	0	1
2	fast feedback cycles empirical software engine...	(0, 13)w0.19531677287385588ln (0, 14)w0.2...	1	2015	1	1
3	integration se research industry : reflections...	(0, 24)w0.0934486830681684ln (0, 35)w0.09...	1	2015	0	1
4	researcher ' experiences supporting industrial...	(0, 23)w0.166421288531331ln (0, 38)w0.088...	1	2016	0	0
5	softcoder approach : promoting software engine...	(0, 0)w0.0932000917621854ln (0, 17)w0.085...	1	2016	0	0
6	research framework build spi proposal small or...	(0, 35)w0.09027327981061076	1	2016	0	0
7	academic industrial software testing conferenc...	(0, 23)w0.17069480186168143ln (0, 38)w0.0...	0	2016	0	0
8	applying data analytics towards optimized issu...	(0, 23)w0.1362476225050109	0	2016	0	0
9	meeting industry-academia research collaborati...	(0, 0)w0.12545450748786866ln (0, 4)w0.229...	1	2017	0	1
10	industry-academia collaborations software engi...	(0, 0)w0.08963012942684441ln (0, 5)w0.163...	1	2017	0	0
11	industry -- academia collaboration software te...	(0, 0)w0.11151179612439246ln (0, 1)w0.148...	1	2017	0	1
12	serp-test : taxonomy support industry -- acade...	(0, 0)w0.13118237054669415ln (0, 3)w0.222...	1	2017	0	0
13	model-based testing digital tvs : industry-as-...	(0, 24)w0.08952160159982998ln (0, 77)w0.2...	1	2017	0	0
14	structuring automotive product line feature mo...	(0, 21)w0.16877554754502608	0	2017	0	0
15	case studies industry-academia research collab...	(0, 0)w0.11950713640089392ln (0, 24)w0.09...	0	2017	0	0
16	wide band patch antenna structures cognitive r...		0	2017	0	0
17	concept analysis programming language research...	(0, 35)w0.0958660112782843	0	2017	0	0
18	exploring interconnectivity risk management , ...	(0, 35)w0.08165843280212062	0	2017	0	0
19	inventorying systems : action research .	(0, 35)w0.15913523609531297	0	2017	0	0

In [46]: