XIII ENCONTRO ANUAL DE TECNOLOGIA DA INFORMAÇÃO
Anais do XIII Encontro Anual de Tecnologia da Informação - EATI
| Ano 11, n. 1 | novembro/ 2022

Classificação de Requisitos de Software com Processamento de Linguagem Natural utilizando BERT

Rafael Rodrigues Pedrosa, André Luiz Alves

Escola Politécnica
Pontifícia Universidade Católica de Goiás (PUC Goiás) – Goiânia - GO – Brasil

orafaelpedrosa@outlook.com, andre.luiz@pucgoias.edu.br

Abstract. *The task of classifying software requirements can be an extensive and expensive task, since it demands time and effort when performed manually, especially in large projects, since requirements written in natural language can have different interpretations, that is, their classification depends on from the interpretation of who performs the task, this can lead to failures that can be reflected in the software life cycle. Faced with the need to classify requirements in an automated way, this work demonstrates how Natural Language Processing (NLP) through the BERT language model can support the intelligent classification of requirements. The method is applied to the PROMISE dataset, reaching an accuracy of 92.8% for classifying functional requirements and non-functional requirements.*

Resumo. A tarefa de classificação de requisitos de *software* pode ser uma tarefa extensa e cara, uma vez que demanda tempo e esforço quando realizada manualmente, em especial em grandes projetos, pois requisitos escritos em linguagem natural podem ter diferentes interpretações, isto é, sua classificação depende da interpretação de quem realiza a tarefa, isso pode levar a falhas que poderão ser refletidas ao ciclo de vida do *software*. Diante da necessidade de classificar os requisitos de forma automatizada, este trabalho demonstra como o Processamento de Linguagem Natural (PNL) por meio do modelo de linguagem BERT pode apoiar na classificação inteligente dos requisitos. O método é aplicado ao conjunto dados PROMISE, atinge uma precisão de 92,8% para classificar os requisitos funcionais e requisitos não funcionais.

1. Introdução

Os requisitos de *software* podem ser compreendidos como necessidades que o sistema deve atender, sendo que os maiores interesses do projeto se encontram nos requisitos [Wiegers e Beatty 2013].

De acordo com [Sommerville 2011] um requisito pode ser classificado entre Requisito Funcional (RF) ou Requisito Não Funcional (RNF), sendo que RFs expressam o que o sistema deve fazer e como se comportar, enquanto os RNFs contêm restrições e estão diretamente ligados aos serviços oferecidos.

Segundo [Wazlawick 2013] requisitos de *software* são frequentemente escritos em linguagem natural, portanto possibilita diferentes interpretações pelas partes envolvidas no projeto.

Os requisitos ainda que sejam conhecidos e bem definidos, podem apresentar desafios quanto a classificação, especialmente quando escritos em linguagem natural pois engenheiros de requisitos usam diferentes estruturas de frases para descrever o mesmo requisito [Dias e Cordeiro 2020].

Como apresentado por [Cleland-Huang et al. 2007] os requisitos quando classificados manualmente, estão propensos a maior ocorrência de erros. Visto que a classificação manual, é considerada uma tarefa longa quando realizada em projetos que contêm grandes números de requisitos [Navarro-Almanza 2017]. Desta maneira, a classificação dos requisitos quando realizada de forma manual, demanda maior tempo, se automatizada garantiria economia de esforços e recursos [Quba et al. 2021].

As técnicas de Inteligência Artificial (IA) tem apoiado em atividades de requisitos, entre elas na classificação dos requisitos [Dreiseitl E Ohno-Machado 2022]. Na visão de Bellman, 1978] a inteligência artificial é a automatização de atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, e aprendizado". A IA tem potencial para uma grande mudança na qualidade de *software*, podendo otimizar a produtividade e aumento das chances de sucesso do projeto [Dam 2019].

O Processamento de Linguagem Natural (PLN) possui capacidade para a tarefa de classificação, uma vez que pesquisadores demostram interesse no estudo de técnicas de PLN para o processamento de requisitos expressos de forma textual [Zhao 2021].

A partir da constatação que requisitos de *software* quando descritos em linguagem natural pode ser um desafio devido estar propensos a inconsistências, e que quando classificados manualmente há maiores chances de erros, mostra-se a necessidade de mitigar este problema.

Neste contexto este trabalho apresenta uma forma de classificar os requisitos de *software* com o processamento de linguagem natural utilizando o modelo de linguagem *Bidirectional Encoder Representations from Transformers* (BERT) em abordagem feita no conjunto de dados do repositório *Software Engineering Repository* PROMISE composto de 625 requisitos.

2. Trabalhos Relacionados

[Kici et al. 2021] considera uma abordagem em transferência de aprendizado na classificação de requisitos dado que é examinado o modelo de linguagem pré-treinado DistilBERT, o submetendo a comparação de desempenho com outros modelos na classificação multiclasse (*multi-class*) como o LSTM e BiLSTM em conjunto de dados de especificação de requisitos.

Em [Hey et al. 2020] é proposto o NoRBERT, modelo para classificação de requisitos. Os autores exploram técnicas para aperfeiçoamento do BERT diante da má generalização com modelos que possuem menor treinamento. Seus estudos incluem abordagens em projetos que antes não haviam sido vistos.

No trabalho de [Sainani et al. 2020], os autores realizam um estudo exploratório para automatizar a extração e classificação de requisitos em contratos de engenharia de *software*. No estudo é feito a comparação entre o BERT e diferentes algoritmos para classificação, entre eles, algoritmos tradicionais como máquina de vetores de suporte (SVM, do inglês: *support-vector machine*), floresta aleatória (*random forest*) e *Naive*

Bayes. O estudo obteve melhor resultado utilizando BERT, chegando a mais de 84% de acurácia na classificação dos requisitos classificados.

3. Aplicação do BERT na Classificação de Requisitos

A ideia de que a classificação dos requisitos fosse uma tarefa automatizada é vista ao longo do tempo, conforme afirmado por [Hey et al. 2020].

Neste contexto, o BERT é aplicado na classificação dos requisitos, pois oferece recursos para realizar tal tarefa, dado que seu diferencial em classificação de textos é a sua capacidade de definir a importância da palavra bidireccionalmente. Desse modo, permite que cada palavra do requisito, tenha sua relevância determinada através do contexto em que está inserida, consequentemente, garante resultados ainda mais precisos [Koroteev 2021].

Em tarefas de classificação, utiliza-se o ajuste fino (*fine tune)* que consiste em ajustar os parâmetros do modelo para que possa ser realizado uma tarefa específica [González-Carvajal 2020]

3.1.BERT

Estudos têm demostrado quantidades significativas de pesquisas, quanto da análise de requisitos por uso de PLN, tanto como a priorização e classificação. Porém, a maioria das técnicas utilizadas são propostas técnicas de aprendizagem supervisionada [Kici et al. 2021].

Contudo, essas técnicas ignoram o valor semântico das palavras ou geralmente avaliam somente os elementos textuais (*tokens*) da esquerda para direita, de tal maneira que as representações dos *tokens* sejam considerados somente os *tokens* a esquerda.

Com o objetivo de solucionar este problema, é apresentado o BERT, que vêm do acrônimo *Bidirectional Encoder Representations from Transformers* e que pode ser traduzido como ("representações de codificador bidirecional de transformadores"). É um modelo de linguagem pré-treinado, que inova quando avalia bidireccionalmente o contexto que os *tokens* estão inseridos, isto é, permite uma avaliação tanto da esquerda para a direita, quanto da direita para a esquerda.

O modelo BERT foi projetado para pré-treinar representações de textos não rotulados, e depois para que pudesse ser ajustado e utilizado em diferentes tarefas do PNL [González-Carvajal 2020].

O BERT utiliza dos modelos de linguagem mascarada, ou seja, a função alvo do aprendizado de um *token,* é a representação e a atividade de prever uma palavra aleatoriamente escolhida e mascarada na sentença, considerando apenas o contexto em que está inserida [Koroteev 2021].

BERT é uma importante ferramenta no PLN, e tem se destacado para auxiliar em diversas tarefas complexas, sua aplicação linguagem natural trazido a automatização de tarefas antes realizadas manualmente [Gweon e Schonlau 2022].

BERT é apresentado em duas versões, a versão base, no qual o BERT possui 12 camadas de codificadores (*enconders),* e a versão mais ampla, composta de 24 camadas.

O modelo permite um treino adicional na classificação de textos. Dessa forma, o uso do BERT se mostra ter o melhor desempenho para solução de problemas envolvendo classificações textuais [Koroteev, 2021].

3.2. Conjunto de Dados

É analisado o conjunto de dados do repositório *Software Engineering Repository* PROMISE, composto de 625 requisitos de *software* declarados no idioma inglês, rotulados em 12 diferentes subclasses, sendo 255 requisitos funcionais e 370 não funcionais, contendo 11 diferentes subcategorias, conforme mostra a tabela 1.

Tabela 1 – Quantidade de requisitos em relação as classes

Classe de Requisito	**Quantidade**
Funcional (F)	255
Usabilidade (US)	67
Segurança (SE)	66
Operacional (O)	62
Desempenho (PE)	54
Look & Feel (LF)	38
Disponibilidade (A)	21
Escalabilidade (SC)	21
Manutenibilidade (MN)	17
Jurídico (L)	13
Tolerância a Falhas (FT)	10
Portabilidade (PO)	1
Requisitos Funcionais	255
Requisitos Não Funcionais	370
Total	625

3.3. Pré-processamento

Conforme é mostrado na figura 1, o conjunto de dados possui um desbalanceamento em relação as classes. Portanto, para melhor análise e aproveitamento dos dados é realizado o agrupamento de todas as subcategorias de RNFs em uma única classe, de tal maneira que pudesse identificar o requisito apenas como não funcional e não mais em sua subcategoria de RNF.

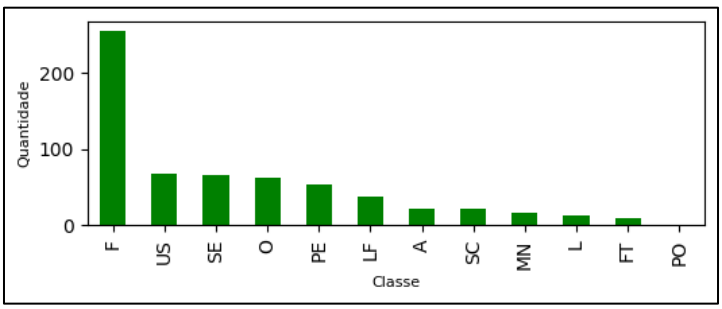

Figura 1 - Distribuição das classes de requisitos no conjunto de dados.

Na sequência são rotulados os requisitos funcionais e não funcionais em uma classificação binária, sendo proposta a rotulação de 1 para RF e 0 para RNF.

3.4. Classificação

Neste estudo é utilizado a linguagem de programação *python* e as bibliotecas *transformers* e *tensorflow.* O conjunto de dados é divido em 80% para treino, 20% do conjunto para teste, e para validação, utilizou-se de 20% do conjunto de treino.

Na classificação dos requisitos é preciso aplicar a *tokenização* para gerar os vetores das representações dos requisitos. Desta maneira, momento é feito a concatenação do requisito com os *tokens* especiais [CLS] e [SEP]. O *token* CLS é um *token* classificador a qual é inserido no início do requisito de modo que ao final, após passar pelas 12 camadas de codificadores do BERT, o *token* contenha um vetor de classificação, enquanto o SEP define o final do requisito.

Posteriormente, após a concatenação dos *tokens* especiais, é necessário que o tamanho dos requisitos seja igual para todos do conjunto de dados, de modo que o modelo possa ser treinado corretamente. Para isso, é necessário que o tamanho do requisito seja igual ao tamanho máximo, definido pelo parâmetro *max_len.*

O *max_len* consiste no tamanho máximo que o modelo pode receber de *tokens,* assim é definido este parâmetro com o valor de 160 uma vez que, a maioria dos requisitos possuem o tamanho inferior a 160. Logo, se o requisito for maior que 160 *tokens* (palavras) é necessário truncar o requisito para 160. Se o texto for menor, é preciso preencher o requisito com o *token* [PAD] até que o requisito tenha 160 *tokens*.

As palavras que compõem o requisito são transformadas em representações numéricas geradas pelo *tokenizador* para servir como entrada ao BERT. O modelo recebe como parâmetros de entrada um dicionário com as chaves *input_ids*, *attention_mask* e *token_type_ids*.

- ***input_ids***: são os *ids* dos requisitos, de modo que cada palavra do requisito recebe uma representação numérica.
- ***attention_mask***: é uma máscara que indica quais *tokens* estão preenchidos e quais são *padding,* onde o número 1 é utilizado para representar *tokens* preenchidos, enquanto 0 para os *tokens* PAD.
- ***token_type_ids***: são os segmentos de entrada, que são 0 para a primeira sentença e 1 para a segunda sentença. Nesse caso é gerado somente os segmentos 0 pois o trabalho utiliza uma única sentença como entrada.

Após os dados de entrada já estarem formatados, o *fine tune* é essencial para que o BERT forneça os dados para a classificação dos requisitos. Assim, o modelo definido para *fune-tuning* é o *BertForSequenceClassification* modelo para classificação de sequência, que adiciona uma camada linear no topo do BERT. Em seguida é gerado como saída um vetor de *logits*, constituído de valores que representam a probabilidade de cada classe para aquela palavra do requisito de entrada. O vetor de *logits* final é a soma dos vetores de cada palavra do requisito analisado.

Para treinar o modelo, é utilizado como parâmetros 5 iterações, *learning rate* de 0,0001 e *batch size* de 32. Já para validação, é utilizado *batch size* de 8.

Ao final, é utilizado a função de ativação *softmax* para converter os *logits* em probabilidades, valores entre 0 e 1 determinam qual a classe com maior probabilidade do requisito pertencer.

Os processos descritos anteriormente são ilustrados na figura2, que apresenta a arquitetura utilizada neste trabalho com o BERT.

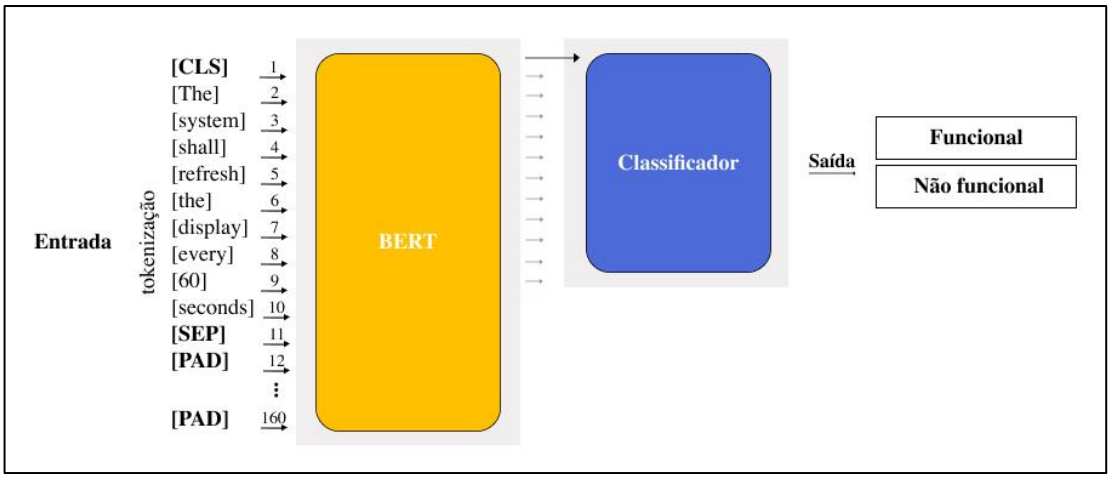

Figura 2: Arquitetura usada para a classificação.

4. Resultados

Este experimento obteve resultados satisfatórios. Na primeira iteraçã alcançou uma precisão de mais de 76,8% na validação, e já na segunda iteração pode ser observado um aumento significativo, mantendo estabilidade aos 92,8% de precisão, conforme observado na figura 3.

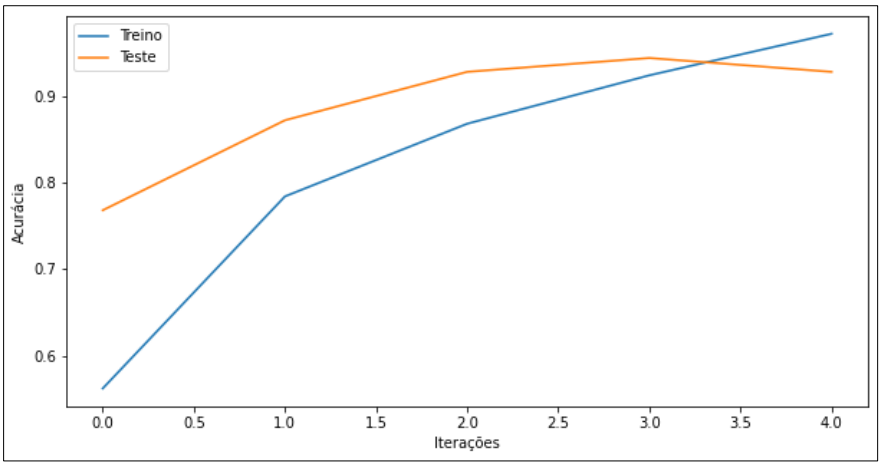

Figura 3: Gráfico de acurácia do treino e teste

A acurácia obtida atesta a taxa de acerto em classificar corretamente os requisitos entre funcionais e não funcionais. Demonstra-se ser mais preciso em relação a classificação dos requisitos não funcionais, tendo em vista que estes requisitos compõe a maior parte do conjunto de dados, este fator foi significante para o modelo que obteve uma acurácia de 96% para requisitos não funcionais, enquanto os requisitos funcionais tiveram 85%.

5. Conclusões

A classificação de requisitos quando realizada de forma manual pode ocasionar maior ocorrência de erros e demanda de esforços. Considerando os recursos de IA para automatização de tarefas complexas e a capacidade do processamento de linguagem natural com elementos textuais, evidencia-se a necessidade de automatizar a classificação de requisitos de *software*.

Deste modo, este trabalho demonstrou uma maneira de como automatizar a classificação dos requisitos, com o uso de processamento de linguagem natural utilizando o modelo de linguagem BERT. Ao realizar o *fine tune,* o BERT se mostra eficiente e desempenha resultados satisfatórios na classificação de requisitos funcionais e não funcionais. Quando aplicado em grandes coleções de requisitos, proporciona economia de tempo, esforço e recursos, garantindo uma melhor precisão dos requisitos e evitando futuros retrabalhos que poderiam ocorrer, se classificados de forma manual.

Posteriormente, pretende-se estudar a classificação entre as diferentes subclasses de requisitos não funcionais, no contexto de avaliar o desempenho do BERT em tarefas de classificação multiclasses de texto, como também analisar o modelo em um grande conjunto de dados.

Referências

Bellman, R. (1978). An introduction to artificial intelligence: can computers think?. Thomson Course Technology.

Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. Requirements engineering, 12(2), 103-120.

Davis, A. M. (1993). Software requirements: objects, functions, and states. Prentice-Hall, Inc..

De Araújo, A. F., & Marcacini, R. M. (2021, March). Re-bert: automatic extraction of software requirements from app reviews using Bert language model. In Proceedings of the 36th Annual ACM Symposium on Applied Computing (pp. 1321-1327).

Dias Canedo, E., & Cordeiro Mendes, B. (2020). Software requirements classification using machine learning algorithms. Entropy, 22(9), 1057.

Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. Journal of biomedical informatics, 35(5-6), 352-359.

González-Carvajal, S., & Garrido-Merchán, E. C. (2020). Comparing BERT against traditional machine learning text classification. arXiv preprint arXiv:2005.13012.

Gweon, H., & Schonlau, M. (2022). Automated classification for open-ended questions with BERT. arXiv preprint arXiv:2209.06178.

Hey, T., Keim, J., Koziolek, A., & Tichy, W. F. (2020, August). NoRBERT: Transfer learning for requirements classification. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 169-179). IEEE.

Kici, D., Malik, G., Cevik, M., Parikh, D., & Basar, A. (2021). A BERT-based transfer learning approach to text classification on software requirements specifications. In Canadian Conference on AI.

Koroteev, M. V. (2021). BERT: A review of applications in natural language processing and understanding. arXiv preprint arXiv:2103.11943.

Navarro-Almanza, R., Juarez-Ramirez, R., & Licea, G. (2017, October). Towards supporting software engineering using deep learning: A case of software requirements classification. In 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT) (pp. 116-120). IEEE.

Quba, G. Y., Al Qaisi, H., Althunibat, A., & AlZu'bi, S. (2021, July). Software requirements classification using machine learning algorithm's. In 2021 International Conference on Information Technology (ICIT) (pp. 685-690). IEEE.

Ramamurthy, R., Pielka, M., Stenzel, R., Bauckhage, C., Sifa, R., Khameneh, T. D., ... & Loitz, R. (2021, August). ALiBERT: improved automated list inspection (ALI) with BERT. In Proceedings of the 21st ACM Symposium on Document Engineering (pp. 1-4).

Sainani, A., Anish, P. R., Joshi, V., & Ghaisas, S. (2020, August). Extracting and classifying requirements from software engineering contracts. In 2020 IEEE 28th International Requirements Engineering Conference (RE) (pp. 147-157). IEEE.

Shirabad, J. S., & Menzies, T. J. (2005). The PROMISE repository of software engineering databases. School of Information Technology and Engineering, University of Ottawa, Canada. URL http://promise.site.uottawa.ca/SERepository.

Sommerville, I. (2011). Engenharia de Software / Ian Sommerville ; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves ; revisão técnica Kechi Hirama. — 9. ed. — São Paulo: Pearson Prentice Hall.

Wazlawick, R. (2019). Engenharia de software: conceitos e práticas. Elsevier Editora Ltda..

Wiegers, K., & Beatty, J. (2013). Software requirements. Pearson Education.

Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E. V., & Batista-Navarro, R. T. (2021). Natural language processing for requirements engineering: a systematic mapping study. ACM Computing Surveys (CSUR), 54(3), 1-41.