Análise artigo: RE-BERT: Automatic Extraction of Software Requirements from App Reviews using BERT Language Model

https://dl.acm.org/doi/10.1145/3412841.3442006

RESEARCH-ARTICLE

RE-BERT: automatic extraction of software requirements from app reviews using BERT language model

Authors: Adailton Ferreira de Araújo, Ricardo Marcondes Marcacini | Authors Info & Claims

SAC '21: Proceedings of the 36th Annual ACM Symposium on Applied Computing • Pages 1321 - 1327
https://doi.org/10.1145/3412841.3442006

Published: 22 April 2021 Publication History

Problema

- Métodos tradicionais (entrevistas, pesquisas) não conseguem acompanhar o grande volume de feedback de usuários encontrado nas lojas de aplicativos.

- As avaliações de aplicativos frequentemente contêm linguagem informal, erros e dados irrelevantes, o que dificulta a extração automática de requisitos de software.

Solução proposta: RE-BERT

RE-BERT (Requirements Engineering with BERT) utiliza um modelo BERT pré-treinado ajustado para detectar requisitos de software em avaliações de usuários.

What is BERT ?

BERT (Representações de Codificadores Bidirecionais de Transformadores) é um poderoso modelo de linguagem desenvolvido pelo Google que compreende o contexto das palavras com base no texto ao seu redor. Diferente dos modelos tradicionais que leem o texto da esquerda para a direita ou da direita para a esquerda, o BERT lê nas duas direções ao mesmo tempo, permitindo capturar significados mais profundos e relações entre palavras.

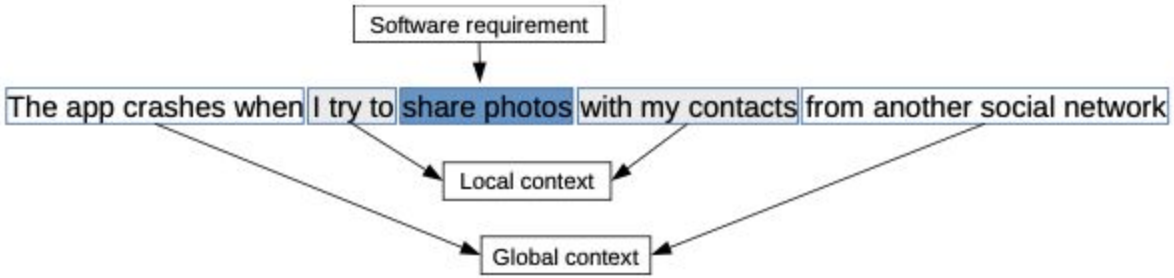


Figure 2: Example of an app review organized in three parts: (1) software requirement; (2) global context; and (3) local context.

Resultados experimentais

- Testado em 8 conjuntos de dados reais de aplicativos (por exemplo, WhatsApp, Netflix, Twitter).

Table 1: The overview of the datasets used in the experimental evaluation.

App	reviews	labeled reviews	sentences	features	distinct features	single-word features	multi-word features
eBay	1,962	125	294	206	167	78	128
Evernote	4,832	125	367	295	259	82	213
Facebook	8,293	125	327	242	204	80	162
Netflix	14,310	125	341	262	201	94	168
Photo editor	7,690	125	154	96	80	39	57
Spotify	14,487	125	227	180	145	69	111
Twitter	63,628	125	183	122	99	39	83
WhatsApp	248,641	125	169	118	100	49	69

- O RE-BERT obteve melhorias no F1-score de 80% a 560% em comparação com métodos de ponta (SAFE, GuMa, ReUS).
- Destacou-se tanto em cenários de correspondência exata quanto de correspondência parcial na extração de requisitos.

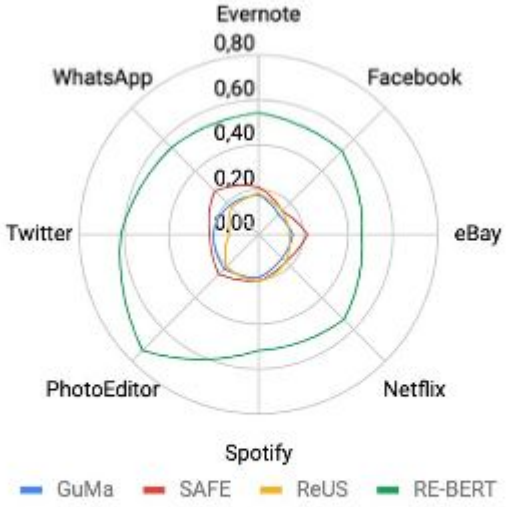
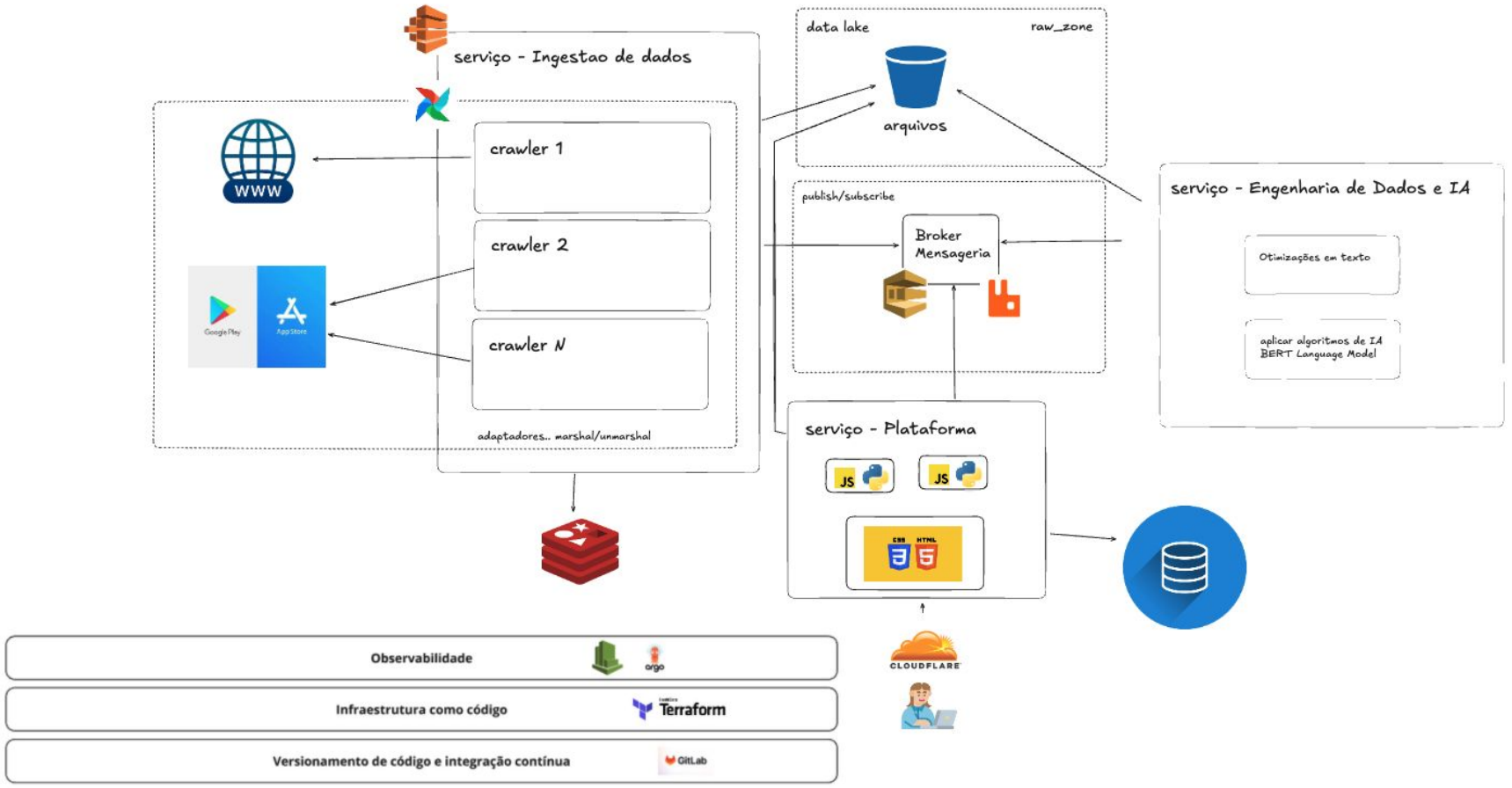


Figure 7: Overall methods comparison using the average F_1 measure calculated from all matching scenarios.

Proposta de uma arquitetura de sistemas distribuídos



Agora vamos falar um pouco sobre Replicação

Replicação é a manutenção de múltiplas cópias (réplicas) de dados ou serviços em diferentes nós de um sistema distribuído.

Objetivos principais: melhorar disponibilidade, tolerância a falhas, desempenho e escalabilidade do sistema.

Motivações para Replicação

- Disponibilidade: Se um nó falhar, outros nós com réplicas podem continuar atendendo solicitações.
- Tolerância a falhas: Reduz o risco de perda de dados.
- Desempenho: Réplicas próximas ao usuário reduzem latência.
- Escalabilidade: Permite distribuir a carga de trabalho entre vários servidores.

Exemplo 1: CDN

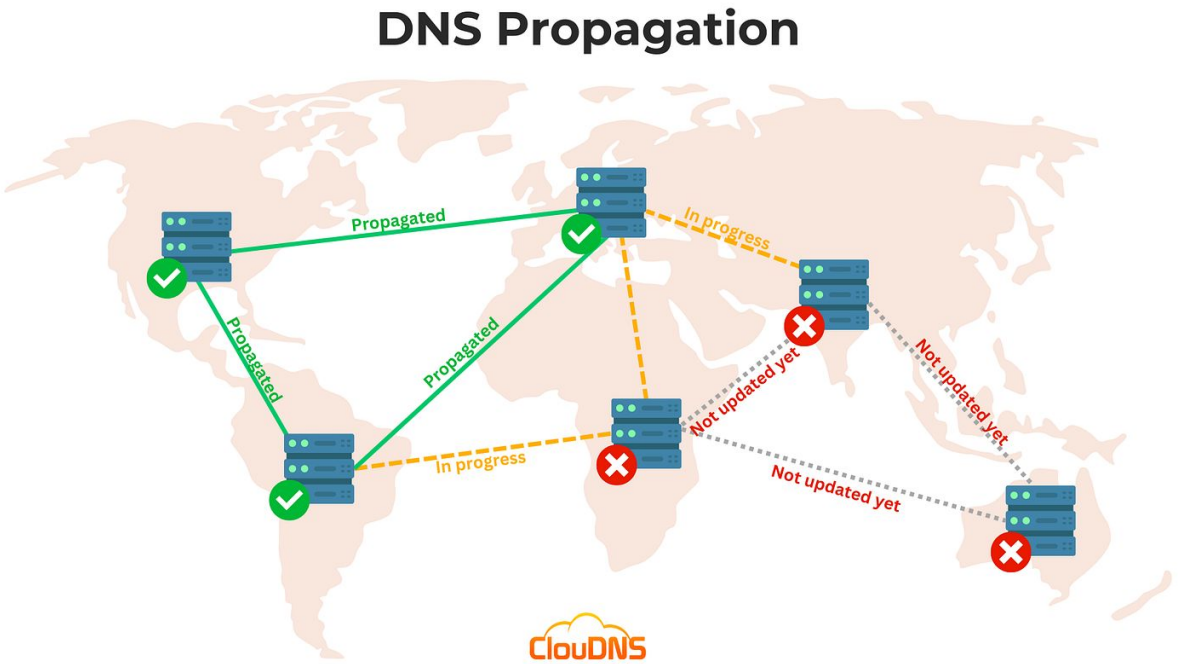
Uma CDN (como Cloudflare, AWS CloudFront) replica conteúdo estático (imagens, CSS, JS, vídeos, etc.) em servidores geograficamente distribuídos chamados Edge Nodes.

Usuários acessam o conteúdo a partir do servidor mais próximo, reduzindo o tempo de carregamento.

Exemplo 2: DNS

O DNS traduz nomes de domínio (como google.com) em endereços IP.

É um sistema distribuído globalmente que depende de servidores replicados para garantir escalabilidade e disponibilidade.



Exemplo 3: Arquitetura microsserviços

- Cada microserviço pode ter múltiplas réplicas, geralmente stateless (sem estado local).
- O balanceador de carga distribui requisições entre essas réplicas.
- Isso melhora resiliência, throughput e evita sobrecarga em uma única instância.

Onde entra a replicação de dados

- Como os serviços são stateless, o estado e os dados ficam em sistemas externos: bancos de dados, caches, filas.
- A replicação de dados acontece principalmente nesses sistemas compartilhados, e não entre as réplicas dos serviços.

Exemplo 4: Bancos de dados

É o processo de copiar dados de um banco primário para um ou mais bancos secundários (réplicas).

Modelos arquiteturais comuns

Primário → Secundários (read replicas)

- Escrita só no banco principal.
- Réplicas só para leitura → alivia carga.
- Comum em PostgreSQL, MySQL, etc.

Multi-master (multi-primary)

- Vários nós aceitam escrita.
- Mais complexo: exige resolução de conflitos, sincronização entre nós.
- Usado em bancos como Cassandra, CockroachDB.

Desafios

- Consistência
  - Consistência centrada em dados
    - Consistência estrita, sequencial, causal, eventual, entre outros.
  - Consistência centrada no cliente
    - Leituras monotônicas, Escritas monotônicas, Read-your-writes e Writes-follow-reads
- Conflitos de escrita em arquitetura Multi Master
- Gerenciamento de réplicas
- Propagação lenta
- Failover e recuperação