Análise e Avaliação de Performance de Técnicas de Machine Learning para Product Matching

Eduardo F. Valim[1], **Renato P. Ishii**[1]

[1]Faculdade de Computação — Universidade Federal de Mato Grosso do Sul (UFMS)
Campo Grande, MS — Brasil

{eduardo.valim,renato.ishii}@ufms.br

Abstract. *Product Matching is the task of identifying pairs that refer to the same product from distinct sources. Traditional rule-based approaches exhibit limitations when handling incomplete, non-standardized data or subtle variations. To address these constraints, Machine Learning methods have emerged as more effective alternatives. This study investigates these techniques through a Systematic Literature Review to identify the most promising current methods. Additionally, a series of experiments are conducted to evaluate the performance of the proposed solutions using product datasets from both The WDC Product Data Corpus (LSPM) and The Magellan Data Repository. The findings support the evidence presented in the selected sample of studies from the review. Strategies based on fine-tuning pre-trained Language Models, such as BERT and its variants, demonstrate their superiority in the task of distinguishing identical products. Multi-objective Learning and Supervised Contrastive Learning (SupCon) approaches stand out, achieving F1 scores exceeding 95%.*

Resumo. *Product Matching é a tarefa de identificar pares que se referem a um mesmo produto, a partir de fontes distintas. Abordagens tradicionais baseadas em regras fixas possuem limitações ao lidar com dados incompletos, não padronizados, ou com diferenças sutis. Para superar essas limitações, métodos que se beneficiam do uso de Aprendizado de Máquina se destacam como alternativas mais eficazes. Este trabalho investiga essas técnicas por meio de uma Revisão Sistemática da Literatura em busca dos métodos mais promissores da atualidade. Além disso, é realizada uma série de experimentos que avaliam a performance das soluções propostas, utilizando os datasets de produtos do WDC Product Data Corpus (LSPM) e do Magellan Data Repository. Os achados suportam as evidências apresentadas na amostra de trabalhos selecionados na revisão. As estratégias baseadas em ajuste fino de Modelos de Linguagem pré-treinados, como o BERT e suas variantes, demonstram sua superioridade na tarefa de distinção de produtos idênticos. Destacando-se os processos com Aprendizado Multiobjetivo e Aprendizado Contrastivo Supervisionado (SupCon), alcançando F1 scores superiores a 95%.*

1. Introdução

Product Matching (PM, ou Correspondência de Produtos) é um subcampo de *Entity Matching* (Correspondência de Entidades). Trata-se do processo de identificar pares de registros que se referem a um mesmo produto físico a partir de descrições potencialmente diferentes.

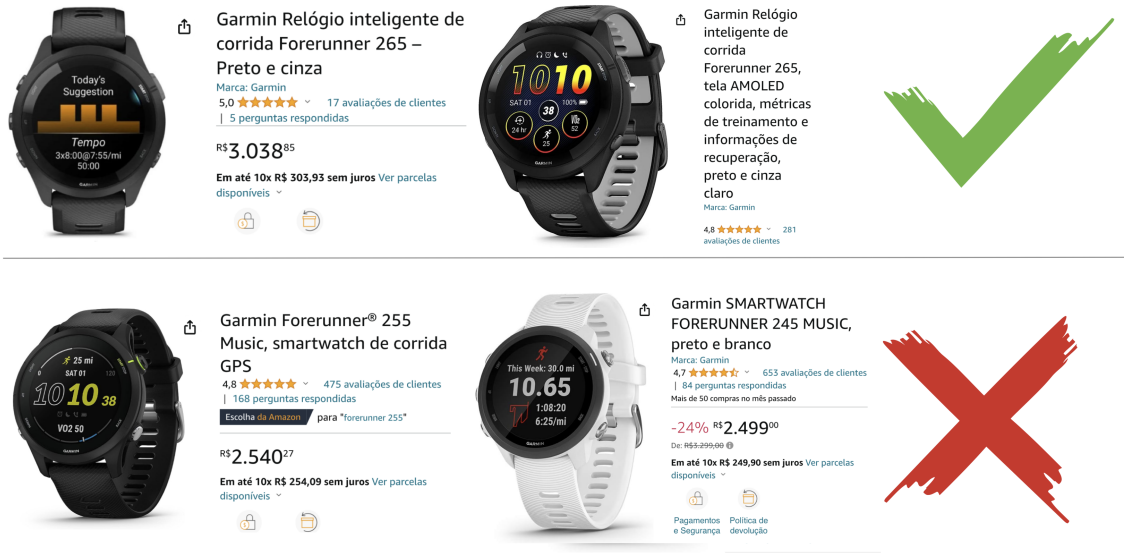

Figura 1. Exemplo de duas ofertas que correspondem e que não correspondem a um mesmo produto. Artigos similares, mas não idênticos, podem ser difíceis de serem classificados corretamente.

Essa técnica é essencial na otimização de diversas tarefas relacionadas à manutenção de extensas bases de dados. No setor privado, como no comércio eletrônico, onde empresas frequentemente gerenciam catálogos com milhares de produtos, o PM é vital, seja para a consolidação de dados provenientes de fornecedores diversos ou para a integração de várias lojas em marketplaces. Além disso, o PM é uma ferramenta valiosa em pesquisas de mercado, onde a comparação automatizada de preços de várias fontes se faz necessária, como na elaboração e auditoria de licitações públicas, contribuindo para a transparência e eficiência dos processos de aquisição governamentais.

O desafio da tarefa está no fato de estas informações serem comumente armazenadas de maneira não padronizada, com dados faltantes ou de baixa qualidade. Com isso, descrições de um mesmo produto acabam sendo listadas de maneiras ligeiramente distintas, como mostra a Figura 1. Na imagem, duas ofertas de relógios diferentes, porém muito semelhantes, poderiam ser erroneamente classificadas como um mesmo item. Além disso, pode-se notar que no par de anúncios correspondentes ao mesmo relógio, os títulos não são idênticos. Em casos como estes, números identificadores únicos, como o GTIN (*Global Trade Item Number*), e o EAN (*European Article Number*), poderiam auxiliar na correta classificação. No entanto, mesmo quando estes códigos estão presentes, é comum produtos similares, mas não idênticos, compartilharem um mesmo identificador. Obstáculos como esses tornam inviáveis as estratégias tradicionais de seleção por regras fixas.

Neste contexto, os algoritmos de *Machine Learning* (ML, ou Aprendizado de Máquina), emergem como uma alternativa valiosa. Esses algoritmos destacam-se por sua capacidade de adaptar-se às informações com as quais foram treinados, tornando-os mais flexíveis e precisos. Abordagens mais avançadas de ML, como *Deep Learning* (DP, ou Aprendizado Profundo), aplicado no *DeepMatcher* [Mudgal et al. 2018], e os recentes

modelos de linguagem (*Language Models – LMs*) baseados em mecanismos de atenção, como o BERT [Devlin et al. 2018], capazes de compreender contexto, têm se mostrado eficientes para tarefas como de PM [Brunner and Stockinger 2020] [Foxcroft et al. 2021] [Paganelli et al. 2023].

Por tratar-se de um tema relativamente novo, a fim de identificar quais são as técnicas de ML mais eficientes para a tarefa de PM, este trabalho explora as mais recentes abordagens, em busca do atual estado da arte. Com esse propósito, foi realizada uma Revisão Sistematizada da Literatura para seleção dos principais trabalhos recentes que tratam do tema. Estes foram analisados no detalhe, e seus resultados reproduzidos mediante uma extensa série de experimentos.

As técnicas apresentadas lidam com uma grande quantidade de dados e fazem uso de modernos LMs, demandando uma quantidade significativa de processamento de GPUs individualmente. A execução de várias destas, mesmo com o emprego de múltiplas placas gráficas, resultou em um período de avaliação substancial. Outro desafio foi a replicação dos ambientes de desenvolvimento originais, devido a implementações pouco documentadas e com grande número de dependências desatualizadas. Não obstante, foi possível reproduzir com sucesso os resultados dos artigos que reportavam os melhores resultados.

O trabalho é dividido em cinco partes. Na Seção 2, é estabelecida uma base teórica com os principais temas abordados. Em seguida, na Seção 3, é definido o protocolo utilizado na seleção dos artigos, sendo então analisados na Seção 4, onde também são reportados os resultados dos experimentos. Posteriormente, a Seção 5 apresenta uma discussão dos achados, comparando os métodos. Por fim, na última Seção, 6 são apresentadas as conclusões e sugestões para trabalhos futuros.

Os resultados encontrados corroboram com o relatado na amostra observada. Evidenciando a eficácia das técnicas baseadas no ajuste fino do BERT e suas variantes, principalmente o RoBERTa, alcançando F1 *scores* superiores a 95%.

2. Fundamentação Teórica

Com o intuito de auxiliar na compreensão dos temas explorados neste trabalho, esta seção apresenta um breve panorama dos conceitos teóricos que são base para as técnicas utilizadas pelos autores. É importante destacar que as áreas abordadas da Inteligência Artificial (IA) se inter-relacionam, conforme ilustra a Figura 2. O principal ponto de interesse são os Modelos de Linguagem (LMs) que estão presentes na interseção entre Processamento de Linguagem Natural (NLP) e Machine Learning (ML), sendo majoritariamente implementados com o uso de *Deep Learning* (DP). A seguir, procede-se à análise detalhada de cada um destes tópicos.

2.1. Machine Learning

Machine Learning, ou Aprendizado de Máquina, é uma área da IA, a qual explora sistemas dinâmicos capazes de se adaptar a uma determinada entrada, melhorando sua performance. A ideia central é ter, ao invés de códigos fixos, rigidamente programados, estruturas dinâmicas, capazes de se adaptar. Este "aprendizado" acontece por meio de um processo de treinamento de modelos, onde o algoritmo identifica padrões em grandes

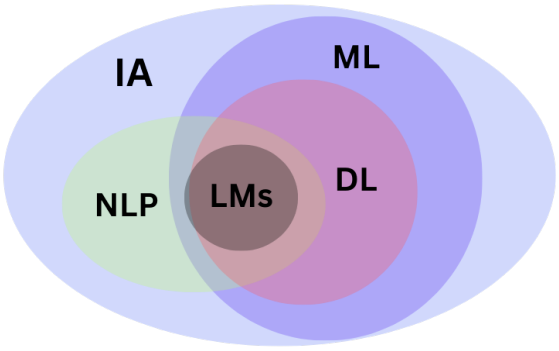

Figura 2. Relação entre diferentes áreas da Inteligência Artificial.

conjuntos de dados (*datasets*), ajustando seus parâmetros internos para então conseguir realizar previsões em informações até então desconhecidas [Goodfellow et al. 2016].

No treinamento, as entradas são mapeadas com as saídas, podendo estas serem conhecidas (Aprendizado Supervisionado) ou não (Aprendizado Não Supervisionado). Neste último, o objetivo do modelo é identificar padrões ocultos nos dados. Há ainda o Aprendizado por Reforço, no qual o algoritmo recompensa ou pune agentes, a medida em que se aproximam do objetivo do treinamento [Russell and Norvig 2010].

2.2. Deep Learning

Deep Learning (DP, ou Aprendizado Profundo) é o campo do ML que estuda as Redes Neurais (RNs) de múltiplas camadas. Estas redes são estruturas capazes de modelar padrões complexos. Inspiradas no funcionamento do cérebro, são compostas por neurônios artificias interligados, denominados nós. Estes, por sua vez, são divididos em camadas de entrada, internas (ocultas) e de saída, como ilustra a Figura 3 [Russell and Norvig 2010].

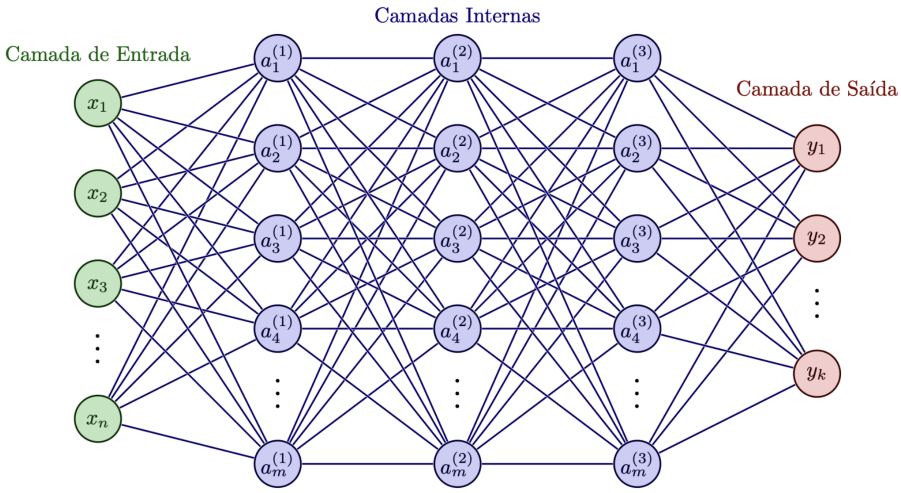

Figura 3. Topologia de uma Rede Neural Multicamadas com n entradas, k saídas, e camadas internas com m nós.

RNs são treinadas por um processo iterativo ao longo de vários exemplos provenientes de um *dataset*, como descrições de um produto. Usualmente, cada exemplo é

composto de várias *features* (características), como nome, cor e tamanho. No caso de Aprendizado Supervisionado, uma dessas *features* é denominada *label* (rótulo). Trata-se da variável de interesse, a que o modelo tentará predizer o valor.

Além disso, é comum que o modelo treine por todos os exemplos do *dataset* mais de uma vez. Para cada passagem completa, denomina-se uma *epoch* (época). Adicionalmente, em casos onde a quantidade de dados é muito grande, o treinamento pode ser divididos em *batches* (lotes) menores.

O processo de treinamento de uma RN simples, como as FFNNs (*Feedforward Neural Networks*) começa convertendo as *features* em uma representação numérica, então cada uma é associada a um nó da primeira camada da rede. Os nós de cada camada são conectados com os nós das anteriores e seguintes. Além disso, essas conexões possuem um valor numérico associado, denominado peso (*weight*). O valor de cada nó das camadas posteriores é o resultado da função de ativação, sobre o resultado de uma soma dos valores dos nós anteriores mais um viés (*bias*), multiplicados por seus respectivos pesos. O *bias* pode ser visto como um termo de ajuste, e a função de ativação introduz não linearidade ao modelo que, de outra forma, seria puramente linear [Haykin 2009].

Há uma variedade de funções de ativação. Em camadas de saída, um exemplo amplamente utilizado para problemas de classificação é a *Softmax* [Bridle 1990], descrita na Equação 1. Sua vantagem está na capacidade de, além de normalizar a saída, permitir que a soma das probabilidades seja igual a 1.

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{1}$$

onde:

- x_i é a i-ésima entrada do vetor de entrada;
- N é o número total de entradas;
- $\sum_{j=1}^{N} e^{x_j}$ é a soma de todas as exponenciais das entradas do vetor.

Após esta primeira passagem dos nós pela entrada, denominada *Forwardpropagation*, os resultados são então comparados com um valor de referência, conhecido como verdadeiro. Então os pesos do modelo w são atualizados por meio de uma Função de Custo (*Loss Function*), tal como a de Entropia Cruzada (*Cross Entropy*) [Ho and Wookey 2020], a qual pode ser definida como na Equação 2.

$$J(w) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \tag{2}$$

onde:

- w são os pesos do modelo;
- m é o número de exemplos de treinamento;
- C é o número de classes, sendo igual a 2 para classificação binária;
- $y_{i,c}$ é o valor verdadeiro para a classe c do exemplo i;
- $\hat{y}_{i,c}$ é a probabilidade prevista para a classe c do exemplo i.

Inicia-se então um processo de otimização denominado Gradiente Descendente (*Gradient Descent*), descrito na Equação 3 [Rumelhart et al. 1986], que visa minimizar a Função de Custo. Para isso, os pesos são atualizados, subtraindo de seu valor a multiplicação de uma taxa de aprendizagem pelo seu gradiente, o qual é a derivada parcial da Função de Custo em relação ao peso em questão. A este processo de retorno e ajuste dos pesos, denomina-se *Backpropagation*.

$$w = w - \eta \nabla J(w) \tag{3}$$

onde:

- w são os pesos do modelo;
- η é a taxa de aprendizagem;
- $\nabla J(w)$ é o gradiente da função de custo J em relação aos pesos w.

RNs podem ser modeladas para diversos fins, mas o de maior interesse no contexto deste trabalho é o de classificação binária, onde todos os nós da última camada são ligados a um último nó, que informa se duas ofertas correspondem ou não a um mesmo produto. A avaliação destes modelos adota métricas como precisão (*precision*), revocação (*recall*) e *F1 score*. As duas primeiras medem a proporção de casos corretamente identificados pelo modelo como positivos. Em relação a todos os previstos como positivos, na precisão, e em relação a todos que realmente são positivos, no caso da revocação [Manning 2008].

Já o *F1 score*, mede um balanço entre precisão e revocação, mediante uma média harmônica entre ambas, como mostra a Equação 4. Assim, é possível selecionar o modelo que minimiza as classificações erroneamente preditas, tanto para positivos como falsos. Por este motivo, esta é a métrica mais utilizada para comparação de modelos de classificação.

$$\textit{F1 score} = 2 \times \frac{\textit{Precisão} \times \textit{Revocação}}{\textit{Precisão} + \textit{Revocação}} \tag{4}$$

2.3. Aprendizado Contrastivo

Aprendizado Contrastivo é uma técnica utilizada para criar representações que aproximam amostras similares e afastam as dissimilares, no espaço vetorial de *embeddings* (detalhado a seguir). No aprendizado Contrastivo auto-supervisionado não há rótulos. Assim, as amostras positivas (similares) da entidade são obtidas por meio de exemplos aumentados (*augmentation*). Já as negativas (dissimilares), são escolhidas aleatoriamente. Esta técnica tem se mostrado eficaz em diversas aplicações, como Visão Computacional, mas depende de um processo complexo de modelagem, além de grandes quantidades de dados para a obtenção de exemplos negativos difíceis. Isto é, aqueles similares, mas ligeiramente distintos dos originais, tipo de informação necessária para um treinamento efetivo, capaz capturar diferenças sutis [Chen et al. 2020].

Já o Aprendizado Contrastivo Supervisionado (SupCon) estende o Aprendizado Contrastivo, utilizando informações de rótulos de classe para identificar amostras similares. Diferente do método tradicional, o SupCon usa várias amostras positivas da mesma classe para cada ponto de ancoragem, melhorando a compactação dos grupos de classes no espaço de *embeddings* e eliminando a necessidade de difícil mineração de negativos.

Essa abordagem resulta em uma melhor qualidade de representação dos dados e maior robustez em *benchmarks* de classificação, superando a tradicional perda de entropia cruzada. Além disso, o SupCon é menos sensível a mudanças nos parâmetros de treinamento, tornando-se uma técnica mais estável e eficaz para treinar modelos de classificação em grande escala [Khosla et al. 2020].

2.4. Processamento de Linguagem Natural

O Processamento de Linguagem Natural (NLP) é uma subárea do ML que estuda algoritmos capazes de compreender linguagem natural humana. O principal objetivo do NLP é possibilitar que os computadores interpretem e gerem texto de maneira eficiente. Sendo útil em uma variedade de tarefas, como análise de sentimento, resumo de texto, tradução, entre outras [Eisenstein 2018].

2.4.1. Representação de Texto

De modo que esses algoritmos conseguirem treinar de maneira eficiente, os textos de entrada devem ser convertidos em unidades menores, denominadas *tokens*, em um processo conhecido como *tokenização*. Os *token*s podem ser palavras, uma parte delas, ou até mesmo caracteres únicos. Após isso, os *tokens* são convertidos em uma forma numérica, tipicamente usando uma representação vetorial. Para tanto, são usadas técnicas como (TF-IDF) (*Term Frequency-Inverse Document Frequency*) [Bafna et al. 2016], que avalia a relevância de uma palavra em um documento, ponderando sua frequência, em relação à frequência inversa em um conjunto, e as *Word Embeddings*, com métodos como *Word2Vec* [Mikolov et al. 2013], *Glove* [Pennington et al. 2014] e *FastText* [Joulin et al. 2016] que criam representações vetoriais de alta dimensionalidade (*embeddings*), capturando o significado semântico das palavras, em que as dimensões modelam características específicas do *token*. A título de exemplo, considere as palavras "meia" e "casaco". Estas estariam próximas em uma dimensão que codificasse o significado de peça de roupa, mas distantes de outra que representasse o sentido de tamanho.

2.4.2. Transformadores

Os modelos baseados em Arquiteturas de Transformadores, (do inglês *Transformer*) [Vaswani et al. 2017], tem revolucionado não apenas o campo do NLP, mas várias áreas do ML, graças a seu mecanismo de atenção, capaz de codificar contexto. Em seu formato original, os transformadores são divididos em módulos denominados *encoder* (codificador) e *decoder* (decodificador).

Quando uma entrada é recebida por um transformador, esta é convertida em *tokens* que subsequentemente são codificados em *embeddings*, empregando alguma técnica como as mencionadas anteriormente. Estes, então, seguem para o *encoder* que, por sua vez, é composto por uma série de blocos de atenção e FFNNs. Nestes blocos, a atenção é geralmente implementada através do mecanismo de atenção autorregressiva, que permite às *embeddings* compartilhar informações entre si, sendo descrito matematicamente pela Equação (5).

$$\textit{Attention}(Q, K, V) = \textit{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

onde:

- Q é a matriz de *queries* (consultas);
- K^T é a matriz de *keys* (chaves) transposta;
- V é a matriz de *values* (valores);
- d_k é a dimensão das *keys*.

As matrizes Q, K, e V são resultado da multiplicação de *weights* específicos do modelo pela matriz X, que por sua vez é formada pelas *embeddings* dos *tokens* nos quais a atenção está sendo calculada.

A interação entre *query* e *key* determina como a atenção é distribuída na sequência de entrada. Cada *query* pode ser imaginada como uma pergunta feita ao modelo, que verifica as *keys* para atribuir uma relevância contextual a cada *token*. Além disso, a divisão pela raiz do termo d_k é usada para normalização, visando a estabilidade numérica das entradas do *Softmax*. Ao combinar este resultado com os *values*, chega-se aos valores que, adicionados às *embeddings* correspondentes, incorporaram contexto em suas representações.

Na prática, este processo é realizado várias vezes, em paralelo, em outras *heads* (cabeças), cada uma com uma *query* diferente que analisa aspectos contextuais distintos. O resultado deste procedimento é uma representação vetorial altamente contextualizada. Em uma técnica de vetorização comum, voltando à ilustração das peças de roupa, a palavra "meia" sempre terá a mesma *embedding*. Todavia, ao passar pelo *encoder*, "meia" nas frases "corri uma meia maratona" e "guardei a meia na gaveta" recebe representações distintas.

Seguindo do *encoder*, o *decoder* realiza o processo inverso partindo das *embeddings*, de volta para linguagem natural. No entanto, nem todos os modelos possuem esse componente, como nos utilizados no escopo deste trabalho, os quais são detalhados a seguir.

2.4.3. Modelos de Linguagem

Os Modelos de Linguagem (LMs) são estruturas projetadas para compreender e, em alguns casos, gerar texto [Jurafsky and Martin 2023]. Atualmente, a grande maioria destes modelos é baseada em transformadores. São treinados com grandes quantidades de texto, em processos que demandam grande capacidade computacional. Por este motivo, é comum que o uso destas arquiteturas se dê pela adaptação de LMs pré-treinados, empregando um método conhecido como ajuste fino (*fine-tuning*). Este processo permite que os modelos sejam otimizados para tarefas específicas com menor quantidade de dados e recursos computacionais.

Destacando-se como o modelo mais popular em tarefas de PM, o BERT (*Bidirectional Encoder Representations from Transformers*) [Devlin et al. 2018], criado por pesquisadores da *Google AI*, em 2018, é um modelo de linguagem pré-treinado não causal (que captura o contexto de uma palavra em ambas as direções), tornando-o eficaz para

tarefas de compreensão de linguagem, como análise de sentimentos e resposta a perguntas. Composto por 12 camadas em sua versão base, em seu treinamento utiliza o *Masked Language Modeling* (MLM), em que o modelo deve prever palavras aleatoriamente mascaradas no texto. Além disso, emprega uma tarefa adicional de *Next Sentence Prediction* (NSP), no qual é treinado para predizer se uma frase segue outra no texto original.

No BERT, é necessário formatar as entradas de forma específica, serializada e com *tokens* especiais. Entre eles, o **[SEP]**, o qual é inserido entre frases para indicar separação, como na concatenação de dois títulos de ofertas, e no final da sequência de palavras. Há também o **[CLS]**, inserido no início da entrada, cuja codificação final representa a informação da sequência inteira no fim do treinamento. Usualmente, a tarefa de classificação é realizada utilizando as informações deste *token*.

Há ainda diversos LMs baseados no BERT, como o DistilBERT (*Distilled BERT*) [Sanh et al. 2019], uma versão menor e mais leve, assim denominado por ter sido desenvolvido por meio uma técnica nominada destilação de conhecimento, na qual o modelo menor (DistilBERT) aprende a imitar o comportamento de um modelo maior (BERT). Assim, apesar de ter seu tamanho reduzido em cerca de 40%, ainda consegue precisão similar ao original.

Outra variação é o RoBERTa (*A Robustly Optimized BERT Pretraining Approach*) [Liu et al. 2019], desenvolvido pela *Facebook AI*. Em relação a sua performance, foi treinado com uma corpora dez vezes maior, além de outras otimizações, tal como a remoção da etapa de NSP, que se mostrou dispensável. Desta forma, apesar de maior e mais lento, alcança uma precisão superior ao BERT original.

3. Metodologia da Revisão

A abordagem metodológica adotada neste trabalho é baseada nas diretrizes de [Kitchenham and Charters 2007], que conceitua Revisão Sistemática da Literatura (RSL) como:

> "Uma forma de estudo secundário que utiliza uma metodologia bem definida para identificar, analisar e interpretar todas as evidências disponíveis relacionadas a uma questão de pesquisa específica de maneira imparcial e, até certo ponto, reprodutível".

Nesta seção, são detalhados todos os passos tomados no processo de revisão, definindo um protocolo que garante sua replicabilidade. Primeiro são descritas a expressão de pesquisa e as bases de dados utilizadas, seguido pelos critérios de seleção e extração dos dados, finalizando com a apresentação dos artigos encontrados.

3.1. Condução da Pesquisa

A presente revisão tem em vista identificar qual é a técnica de Machine Learning mais eficaz para Product Matching, esta é a questão de pesquisa da revisão. Desta forma, o processo iniciou-se pela definição da expressão de busca. A fim de extrair uma amostra inicial ampla, mas relevante, foi utilizada uma que bem sintetizasse o escopo da pesquisa, sem que para isso fosse muito restritiva:

> ("product matching" OR "product entity matching") AND ("machine learning" OR "deep learning" OR "neural network*" OR "*Transformer**")

Em seguida, foram selecionadas fontes de dados com alta credibilidade e relevância na área de Ciência da Computação. As utilizadas foram: *IEEE Xplore*, *Scopus*, *ACM Digital Library*, *Web of Science*, *Engineering Village (Ei Compendex)* e *SpringerLink*. A mesma expressão de busca foi aplicada em todas as plataformas, limitando-se aos artigos publicados nos últimos seis anos. A Tabela 1 lista o número de resultados encontrados em cada base, na pesquisa realizada em 13/02/2024.

BASE DE DADOS	RESULTADOS
IEEE Xplore	12
Scopus	39
ACM Digital Library	53
Web of Science	19
Engineering Village (Ei Compendex)	48
Springer Link	54

Tabela 1. Número de resultados por base de dados.

3.2. Critérios de Inclusão e Exclusão

Posteriormente, seguiu-se com a seleção dos trabalhos conforme os critérios de inclusão e exclusão apresentados na Tabela 2. Foram selecionados apenas aqueles publicados em inglês ou português, no período entre 01/01/2018 e 31/12/2023. Além disso, como critério de qualidade, foi utilizado o sistema QUALIS (Capes), sendo selecionadas apenas as publicações com estrato maior ou igual a B2. As publicações que não possuíam estratificação foram pontuadas com base na equivalência indicada pelo QUALIS, comparando com o índice H5 do Google Scholar para eventos, e com o percentil do Journal Impact Factor (JIF) do Scopus ou Web of Science, para periódicos, como apresentado na Tabela 3.

Critérios de Inclusão	**Critérios de Exclusão**
Estritos em Inglês ou Português	Duplicados
Publicados entre 01/01/2018 e 31/12/2023	Escopo fora do contexto da questão de pesquisa
Estrato Qualis (ou JIF equivalente) igual ou superior a B2	Não disponibilizavam *links* para implementação
Periódicos com JIF igual ou superior a 25	Apresentavam implementação incompleta ou com ambiente de execução incompatível
Eventos com Índice H5 igual ou superior a 9	Não utilizavam os datasets WDC ou Magellan
	Não utilizavam *F1 score* como métrica

Tabela 2. Critérios de Inclusão e Exclusão da RSL.

Adicionalmente, foram descartados os documentos duplicados e os que não pertenciam ao escopo desta revisão, bem como os que impossibilitavam a reprodução

Periódicos		Eventos	
JIF	**Estrato**	**Índice H5**	**Estrato**
$\geq 87,5$	A1	≥ 35	A1
$\geq 75,0$	A2	≥ 25	A2
$\geq 62,5$	A3	≥ 20	A3
$\geq 50,0$	A4	≥ 15	A4
$\geq 37,5$	B1	≥ 12	B1
$\geq 25,0$	B2	≥ 9	B2
$\geq 12,5$	B3	≥ 6	B3
$\leq 12,5$	B4	> 0	B4

Tabela 3. Equivalência de estrados do QUALIS.

e comparação com os demais. Isto é, os que não disponibilizavam *links* para a implementação completa do código, os que não utilizavam os *datasets Magellan* ou *WDC*, e os que não recorriam à métrica *F1 score* para avaliação.

Após esta seleção, chegou-se a 6 trabalhos acadêmicos, listados na Tabela 4. Estes constituem o escopo de estudo desta revisão.

Ano	Autores	Título
2020	Li; et al.	Deep Entity Matching with Pre-Trained Language Models
2020	Peeters; et al.	Intermediate Training of BERT for Product Matching
2021	Peeters; et al.	Dual-Objective Fine-Tuning of BERT for Entity Matching
2022	Peeters; et al.	Supervised Contrastive Learning for Product Matching
2022	Yao; et al.	Entity Resolution with Hierarchical Graph Attention Networks
2022	Możdżonek; et al.	Multilingual Transformers for Product Matching – Experiments and a New Benchmark in Polish

Tabela 4. Artigos Selecionados na RSL.

4. Análise dos Trabalhos e Experimentos

Esta seção examina os trabalhos selecionados na seção anterior. Além disso, são descritos os experimentos das reproduções e reportados os resultados encontrados. Na amostra de estudos observada, nota-se o uso predominante de modelos de linguagem como o BERT e suas variantes para tratar da tarefa de PM. Os autores exploram técnicas que vão desde o processamento dos dados, até o ajuste fino.

Os experimentos foram conduzidos em um ambiente computacional de alta performance, utilizando instâncias em nuvem[1], equipadas com 128GB de RAM e GPUs NVIDIA RTX 4090, com 24GB de VRAM. Isoladamente, os experimentos podem ser reproduzidos com apenas uma GPU, e apenas o pré-processamento para o Treinamento

[1] `https://vast.ai/`

Intermediário (Seção 4.2), Contrastivo (Seção 4.4) e JointBERT (Seção 4.3) necessitam de mais de 32GB de memória principal. Todos os códigos utilizados, bem como eventuais adaptações, estão disponíveis no repositório[2] deste trabalho.

Nesta análise, a fim de permitir uma melhor comparação entre os achados, foi dada maior atenção àqueles que utilizavam os *datasets* (conjuntos de dados) majoritariamente adotados como *benchmarks* pela comunidade. Estes tem seus subconjuntos de treinamento, de validação e de testes padronizados, e são descritos a seguir.

Os primeiros são os do *WDC Product Data Corpus and Gold Standard for Large-Scale Product Matching* [Primpeli et al. 2019]. Criado, especificamente, para avaliar técnicas PM em larga escala. Antes do WDC, os conjuntos de dados disponíveis para a tarefa não forneciam uma quantidade suficiente de dados para o treinamento adequado de modelos de DP. O WDC[3] resolve este problema com dados de 26 milhões de ofertas, extraídos de 79 mil *websites*, por meio do projeto *CommonCrawl*[4]. O resultado é um conjunto de treinamento com mais de 200 mil pares de produtos, e um de teste com 2.200 pares de produtos rotulados manualmente. A coleção é dividia em 20 *datasets*, agrupados em cinco categorias: câmeras, computadores, sapatos, relógios e uma quinta com todas as anteriores. Cada uma possuindo quatro variações de tamanho: pequeno (P), médio (M), grande (G) e extra-grande (XG). Os artigos revisados neste trabalho empregam um subconjunto dos *datasets* do WDC, que contém apenas ofertas em inglês.

Outra coleção vastamente adotada, são os *datasets Magellan*, originalmente disponibilizados no *Magellan Data Repository* [Das et al. 2024]. Deste grupo, são utilizados os relacionados a comércio eletrônico.

Os *datasets* são majoritariamente estruturados com atributos tais como títulos, descrições, preços, entre outros. Com exceção de uma versão "suja" do *Walmart-Amazon* que contém dados faltantes ou informações incorretas, e do *Abt-Buy*, classificado como "textual" por ter exemplos mais textualmente densos.

A Tabela 5 compila as informações de todos os *datasets* de treinamento com seus respectivos tamanhos e porcentagem de pares positivos (que correspondem ao mesmo produto).

[2] `https://github.com/edfvalim/ml-product-matching`
[3] `https://webdatacommons.org/largescaleproductcorpus/v2/`
[4] `https://commoncrawl.org/`

	Variação	Tamanho	Pares Positivos
Câmeras	**P**	1.886	25,77%
	M	5.255	21,08%
	G	20.036	19,18%
	XG	42.277	16,98%
Computadores	**P**	2.834	25,48%
	M	8.094	21,77%
	G	33.359	18,42%
	XG	68.461	14,15%
Relógios	**P**	2.255	25,72%
	M	6.413	22,11%
	G	27.027	19,10%
	XG	61.569	15,05%
Sapatos	**P**	2.063	25,69%
	M	5.805	20,91%
	G	22.989	15,15%
	XG	42.429	9,76%
Todos	**P**	9.038	25,65%
	M	25.567	21,52%
	G	103.397	18,01%
	XG	214.661	14,07%
Amazon-Google	**Estr.**	11.460	10,18%
Cervejas	**Estr.**	450	15,1%
Walmart-Amazon	**Estr.**	10.242	9,39%
Walmart-Amazon	**Sujo**	10.242	9,39%
Abt-Buy	**Textual**	9.575	10,74%

Tabela 5. *Datasets* das coleções WDC e Magellan utilizados nos experimentos.

4.1. Ditto

Ditto, apresentado em [Li et al. 2020], é um sistema baseado em LMs pré-treinados que trata o PM como um problema de classificação de pares de sentenças. No trabalho, são utilizados variantes do BERT como o RoBERTa e o DistillBERT, porém a metodologia é independente do modelo utilizado. Além do ajuste fino, também são introduzidas três otimizações que melhoram a performance do treinamento: sumarização, injeção de conhecimento de domínio (CD) e *data augmentation* (DA), processo em que o número de exemplos de testes é aumentado.

O fluxo de processamento, ilustrado na Figura 4, inicia-se combinando dois conjuntos de entidades em um único conjunto de pares. Esta etapa é dispensável nos *datasets* aqui utilizados, uma vez que estes pares já se encontram combinados. Posteriormente, na fase de *blocking*, o número de pares é reduzido, descartando aqueles onde uma correspondência é muito improvável. Foi desenvolvida uma técnica mais avançada, que acelera este estágio. Os exemplos são transformados em *embeddings*, com o *SenteceBERT*, e os pares com maior probabilidade de correspondência, selecionados com uma busca matricial. Na sequência, os pares candidatos passam por um processo de serialização, conca-

tenando as duas ofertas do par em uma única *string* e adicionando os *tokens* especiais do BERT.

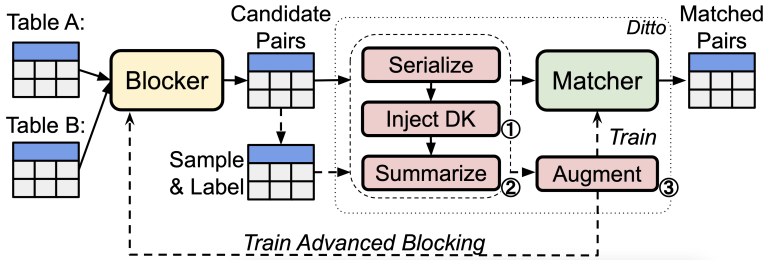

Figura 4. Diagrama do Ditto. Retirado de [Li et al. 2020].

Em seguida, os dados podem ou não passar pelas otimizações. Na injeção de conhecimento de domínio, partes específicas da *string* são destacadas conforme o contexto. Outra etapa é o resumo, que pode ser muito útil para pares muito longos que extrapolem a janela de contexto do modelo. Além disso, há a *augmentation*, que visa aumentar o número de exemplos, onde uma série de operações podem ser aplicadas a um par original. Na implementação do autor, estão disponíveis cinco possibilidades: *append_col*, *drop_col*, *del*, *swap* e *all*. As duas primeiras operam sobre atributos, adicionando e removendo aleatoriamente. As duas seguintes sobre tokens, excluindo ou trocando suas posições. A última operação escolhe aleatoriamente uma das anteriores.

Finalmente, é realizado o ajuste fino em uma versão do modelo adaptada para classificação binária com camadas lineares e outra *Softmax*, conforme ilustrado na Figura 5.

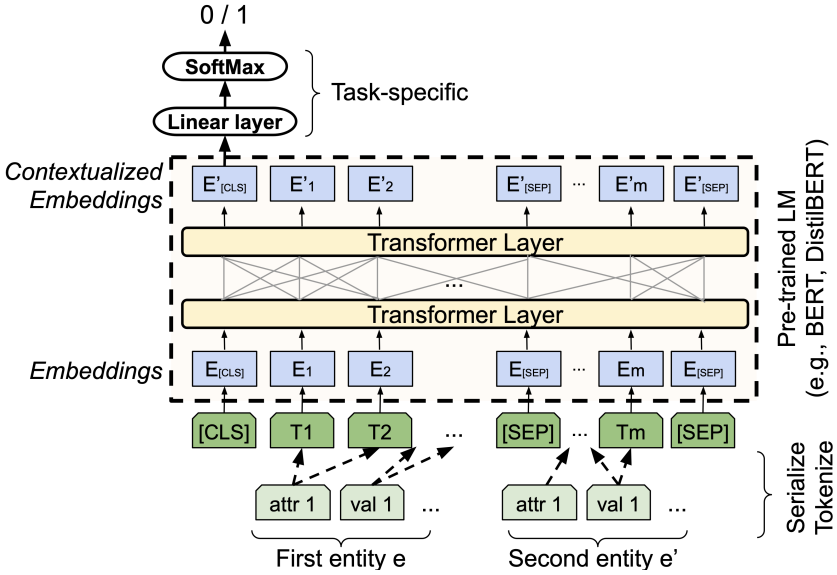

Figura 5. Arquitetura do modelo de linguagem. Retirado de [Li et al. 2020].

Os resultados apresentados pelos autores foram reproduzidos usando os mesmos parâmetros, taxa de aprendizagem de $3e^{-5}$, por 15 *epochs*, com tamanho máximo da sequência de *tokens* em 256, *batch* de 32 quando houve uso de *augmentation* (DA) e 64, caso contrário. Nestes experimentos, foram utilizados apenas o atributo título.

Todos os testes da reprodução foram executados três vezes, utilizando o RoBERTa como LM, inclusive para o *baseline*.

	DA	Ditto
Abt-Buy	swap	append_col
Amazon-Google	drop_col	drop_col
Beer	del	del
Walmart-Amazon	swap	del
Walmart-Amazon (S)	all	swap

Tabela 6. Parâmetros de *augmentation* utilizados nos experimentos do Ditto.

Nos *datasets* Magellan, foram testadas quatro variações, sendo a primeira apenas o ajuste fino, sem otimizações. A segunda com sumarização e injeção de conhecimento de domínio (CD), a terceira com sumarização e DA, e a final (Ditto) com todas as técnicas aplicadas. Já nos *datasets* do WDC, foi avaliada apenas a versão completa do Ditto. O texto não lista os parâmetros de DA utilizados para todos os *datasets*, e não discrimina em qual teste (Ditto ou DA) os parâmetros listados obtiveram melhor performance. Por este motivo, foi necessário realizar o mesmo procedimento do artigo, avaliando todas as possibilidades, para cada *dataset*, e reportando o resultado com melhor performance na média das três execuções. Os selecionados são listados na Tabela 6.

	P	M	G	XG
Todos	83,06	88,40	93,59	94,52
	84,36*	88,61*	93,05*	94,08*
Câmeras	79,46	86,74	92,70	94,32
	80,89*	88,09*	91,23*	93,78*
Computadores	81,70	88,04	91,62	96,05
	80,76*	88,62*	91,70*	95,45*
Sapatos	74,32	82,23	88,37	89,29
	75,89*	82,66*	88,07*	90,11*
Relógios	85,16	90,59	96,71	96,24
	85,12*	91,12*	95,69*	96,53*

Tabela 7. Desempenho do DITTO (com as três otimizações) nos *datasets* do WDC. Resultados com asterisco foram reportados em [Li et al. 2020].

Os números do Ditto nos testes do WDC se alinham em grande medida com os reportados, apresentando performance satisfatória, especialmente nos subgrupos maiores, como pode ser observado na Tabela 7. No entanto, os testes nos Magellan (Tabela 8), apesar de muito próximos, mostram que os usos isolados de DA ou CD superam a combinação dos dois, ao contrário do exposto no artigo, o qual mostrava o Ditto completo como o mais eficaz.

4.2. Treinamento Intermediário

Em [Peeters et al. 2020], os autores exploram o treinamento intermediário do BERT para PM. É assim designado por trata-se de uma etapa anterior ao ajuste fino. Como o LM

	W-A	W-A(S)	A-B	A-G	Cervejas
RoBERTa	85,72	84,42	89,52	75,70	83,23
CD	82,31	83,28	**91,13**	75,16	85,84
	83,73*	80,67*	81,69*	74,67*	90,46*
DA	**85,95**	**85,75**	90,46	**76,59**	**91,33**
	85,50*	85,49*	89,79*	75,08*	87,21*
DITTO	85,34	85,24	90,26	75,34	86,84
	86,76*	85,69*	89,33*	75,58*	94,37*

Tabela 8. Comparação das diferentes otimizações do DITTO nos *datasets* Magellan. S representa a variação suja do *dataset*. Valores com asterisco foram reportados em [Li et al. 2020].

foi pré-treinado com dados de propósito geral, a ideia é adicionar contexto específico de ofertas, antes do ajuste de pareamento dos produtos.

O método é avaliado utilizando as ofertas únicas do *dataset* Computadores do WDC de três formas. Primeiro é realizado o treinamento apenas da mesma categoria (computadores), depois com as quatro categorias do WDC, adicionando Câmeras, Relógios e Sapatos. Por fim, é testado com *Masked Language Modeling* (MLM) como um segundo objetivo, adicionando o *token* especial [MASK] às palavras mascaradas.

A reprodução seguiu os parâmetros relatados no artigo, com taxa de aprendizagem em $5e^{-5}$ com decaimento linear e *batch* de tamanho 256. Sequências de no máximo 128 *tokens* nas primeiras 36 *epochs*, e 512 nas 4 últimas. Já na etapa de ajuste fino, 10 *epochs* com *batches* de 32. No experimento original, foram avaliadas seis taxas de aprendizagem ($5e^{-6}$, $1e^{-5}$, $3e^{-5}$, $5e^{-5}$, $8e^{-5}$, $1e^{-4}$) por três vezes consecutivas. Por este motivo, e devido ao tempo excessivo de treinamento, a reprodução foi realizada com apenas uma execução, para as seis taxas, reportando a melhor para cada caso.

	Var.	CP-C	CP-4cat	CP+MLM
Computadores	**P**	92,68	93,27	**95,17**
		93,73*	90,64*	96,53*
	M	93,53	94,44	**96,31**
		94,88*	93,59*	96,58*
	G	94,09	**94,74**	94,53
		–	93,61*	95,82*
	XG	–	95,62	**95,77**
		94,61*	95,45*	97,37*
Computadores	**XG**	*92,15**	*91,49**	*94,16**

Tabela 9. F1 *scores* de métodos de treinamento intermediário do BERT. Valores em itálico não utilizaram ajuste fino, os com asterisco são reportados em [Peeters et al. 2020].

Os resultados são compilados na Tabela 9. Mesmo com a condução única, o experimento alcançou números muito próximos dos relatados e confirmou o objetivo dual como melhor abordagem, com exceção da variação G.

4.3. JointBERT

[Peeters and Bizer 2021] apresenta o JointBERT, um método que combina dois objetivos durante o ajuste fino do BERT. Além do objetivo de Classificação Binária, é adicionado um de Classificação Multiclasse. Neste segundo, a meta do modelo é tentar prever o número identificador único de cada produto, presente como um dos atributos nos conjuntos do WDC e no *Abt-Buy* dos Magellan.

		DistillBERT	BERT	RoBERTa	JointBERT
Câmeras	**P**	67,81	79,95	**85,25**	79,45
		77,47*	85,74*	78,30*	78,30*
	M	76,99	87,32	**92,21**	86,58
		87,02*	90,20*	87,91*	87,91*
	G	86,36	89,63	93,32	**95,17**
		91,02*	93,91*	96,51*	96,51*
	XG	89,50	91,15	94,65	**96,57**
		91,42*	94,39*	98,02*	98,02*
Computadores	**P**	75,00	77,25	**87,98**	79,48
		80,46*	86,37*	77,55*	77,55*
	M	81,93	87,30	**91,63**	90,20
		89,31*	91,90*	88,82*	88,82*
	G	90,27	91,83	94,57	**96,37**
		92,11*	94,68*	96,90*	96,90*
	XG	93,18	94,37	95,11	**97,84**
		94,57*	94,73*	97,49*	97,49*
Sapatos	**P**	67,76	73,52	**77,03**	70,65
		74,49*	80,29*	73,13*	73,13*
	M	74,07	80,11	**83,64**	82,64
		79,82*	81,12*	82,61*	82,61*
	G	83,53	87,09	86,08	**96,00**
		87,37*	86,60*	95,16*	95,16*
	XG	86,42	88,89	91,25	**97,18**
		87,44*	88,88*	97,88*	97,88*
Relógios	**P**	70,22	80,65	**86,85**	76,26
		78,73*	87,16*	75,83*	75,83*
	M	80,45	89,14	**91,15**	87,54
		89,00*	92,28*	87,46*	87,46*
	G	91,65	95,24	95,10	**97,61**
		95,23*	93,93*	98,46*	98,46*
	XG	91,68	95,76	95,90	**98,67**
		95,76*	94,87*	97,09*	97,09*
Abt–Buy	–	85,26	**90,91**	82,90	–
	–	84,64*	91,05*	83,44*	83,44*

Tabela 10. Comparação de desempenho de diferentes modelos em relação ao JointBERT. Valores com asterisco foram reportados em [Peeters and Bizer 2021].

Sua arquitetura é muito semelhante a apresentada por [Li et al. 2020], ilustrada na Figura 5, porém aplicando *Sigmoid* como função de ativação na classificação binária e adicionando outras duas RNs, que usam *Softmax* para a classificação multiclasse, uma para cada entidade. A função de custo da instância é calculada somando as respectivas funções das três RNs descritas anteriormente.

Os experimentos foram replicados usando *batches* de 32, entradas com no máximo 512 *tokens*, por no máximo 50 *epochs*, parando caso não houvesse melhora por 10 *epochs* consecutivas. A Tabela 10 apresenta os resultados. Todos os modelos foram treinados duas vezes, utilizando cinco taxas de aprendizagem ($1e^{-5}$, $3e^{-5}$, $5e^{-5}$, $8e^{-5}$, $1e^{-4}$), sendo reportado aquele com melhor média nas duas execuções. Além do JointBERT, foram avaliadas as performances do BERT, DistillBERT e RoBERTa como *baselines*, treinados apenas com fine-tuning de classificação binária. Os resultados se alinham com os reportados pelo autor. O JointBERT alcança os melhores F1 *scores* em todos os *datasets* grandes (G e XG), ficando atrás do RoBERTa nos menores. Isso se deve, muito provavelmente, ao número reduzido de pares positivos com identificadores únicos.

4.4. Aprendizado Contrastivo Supervisionado

O pôster [Peeters and Bizer 2022] é pioneiro na aplicação do Aprendizado Contrastivo Supervisionado (SupCon) para PM. A técnica se divide em duas partes, primeiro é realizado um pré-treinamento do *encoder* no RoBERTa, depois é efetuado um ajuste fino para o objetivo de classificação.

São testados dois métodos de Aprendizado Contrastivo, o já mencionado SupCon, e o SimCLR, o qual usa aprendizado auto-supervisionado. Neste último, cada par recebe um único identificador, e o pré-treinamento é conduzido com versões aumentadas deste. Já no Supcon, os autores tiram proveito do fato de que diferente de outros tipos de dados, os *datasets* de treinamento de PM já costumam trazer identificadores únicos (como o GTIN). Estes são utilizados para aumentar a quantidade de exemplos de uma mesma oferta em cada lote do pré-treinamento contrastivo.

Para os produtos que não possuem tais etiquetas, busca-se obtê-las de outros pares correspondentes que as possuem, utilizando um grafo de correspondência. No entanto, quando estes pares se originam da mesma fonte, é possível que um mesmo produto acabe tendo identificadores diferentes, gerando demasiado ruído no treinamento. Mitiga-se este problema separando ofertas de origens distintas em *batches* separados. Além disso, os dados são aumentados utilizando o pacote *nlpaug* [Khosla et al. 2020], aplicando seis técnicas escolhidas aleatoriamente.

Finalmente, é realizado o ajuste-fino, adicionando-se uma camada linear e de *dropout* para retornar um rótulo binário (correspondente ou não correspondente). As representações combinadas das duas ofertas são usadas como entrada para a camada de classificação final. Aqui o autor também explora congelar ou não os parâmetros do *encoder*.

Foi possível reproduzir todos os testes do trabalho, utilizando os mesmos hiperparâmetros. O autor utiliza os *datasets* Abt-buy, Amazon-Google e Computadores, do WDC. No pré-treinamento contrastivo, utiliza-se *batches* de tamanho 1.024, por 200 *epochs*, com taxa de aprendizagem de $5e^{-5}$ e proporção de aquecimento de $0,05$. No ajuste fino, o tamanho do *batch* é reduzido para 64 e treinado por até 50 *epochs*, parando antes

caso a perda no conjunto de validação não melhorasse em 10 *epochs* consecutivas. Todos os resultados foram repetidos três vezes, sendo reportados na Tabela 11.

	Com-P	**Com-M**	**Com-G**	**Com-XG**	**Abt-Buy**	**Amazon-Google**
R-SupCon(C)	93,27	97,60	98,33	98,33	93,25 (32,55)	**78,87** (41,54)
	93,18*	97,66*	98,16*	98,33*	93,70*(38,24*)	79,28* (42,44*)
R-SupCon(C)+a	**94,62**	**97,60**	**98,33**	**98,50**	**93,74**	72,85
	95,21*	98,50*	98,50*	98,33*	94,29*	76,14*
R-SupCon(NC)	80,41	87,42	94,45	95,88	78,02(73,68)	71,59(64,16)
	79,52*	87,32*	94,59*	96,16*	79,99*(71,47*)	71,81*(61,06*)
R-SupCon(NC)+a	80,09	87,27	93,89	95,48	77,20	64,55
	80,69*	89,12*	94,56*	96,13*	77,84*	68,37*
R-SimCLR(C)	55,51	56,04	59,14	59,19	57,13	55,10
	53,98*	55,25*	58,97*	60,66*	56,63*	56,16*
R-SimCLR(C)+a	52,91	51,96	53,19	53,41	51,83	43,76
	53,36*	54,97*	58,34*	62,19*	53,67*	54,29*
R-SimCLR(NC)	68,29	82,41	93,14	96,23	79,89	61,91
	65,75*	82,72*	92,20*	95,25*	79,99*	64,87*
R-SimCLR(NC)+a	69,97	79,96	91,63	94,37	78,43	55,23
	66,73*	82,24*	91,89*	95,75*	79,28*	63,71*

Tabela 11. F1 *scores* dos métodos de Aprendizado Contrastivo Supervisionado aplicados ao RoBERTa. Valores do com asterisco foram reportados em [Peeters and Bizer 2022]. C e NC diferenciam os modelos que tiveram os parâmetros do *encoder* congelados. Valores entre parentes são aqueles onde as duas fontes não foram separadas no pré-treinamento e +a sinaliza os conjuntos aumentados.

A vantagem do SupCon, em relação SimCLR é clara, principalmente nos *datasets* menores. Neste último, o congelamento dos parâmetros resultou em melhor performance, mas o contrário aconteceu com o Supcon. O aumento dos dados não gerou resultados conclusivos, mas a separação das fontes no Abt-Buy e Amazon-Google ocasionou em uma melhora significativa. Com exceção do Amazon-Google, a melhor técnica apresentada foi utilizando o SupCon, com aumento de dados, e parâmetros congelados.

4.5. LMs Multilíngues

[Mozdzonek et al. 2022] explora a tarefa de PM utilizando XLM-RoBERTa e o mBERT, duas variações multilíngues do BERT, treinadas em vastos corporas com mais de 100 idiomas. Em contraste, o modelo original foi treinado exclusivamente em dados em inglês e, portanto, tem dificuldades de identificar nuances em outras línguas.

No artigo, são reportados experimentos no WDC e em um novo *dataset*, em polonês, criado pelos autores especialmente para avaliação de métodos de PM. Os testes reportados neste último obtiveram F1 *scores* em torno de 90%. No entanto, por não serem comparáveis com os outros resultados desta revisão, foram reproduzidos apenas os testes no WDC, apresentados na Tabela 12. Ambos os modelos apresentaram desempenhos semelhantes, com uma leve vantagem para o mBERT.

Os LMs foram treinados por quatro vezes consecutivas por 10 *epochs* com *batches*

de tamanho 16 e taxa de aprendizagem em $5e^{-5}$. Assim como descrito no trabalho, o mBERT se mostra mais eficiente na maioria dos conjuntos de teste.

	Var.	mBERT	XLM-RoBERTa
Câmeras	**P**	80,72	**81,46**
		82,13*	81,96*
	M	87,68	**88,25**
		87,86*	88,11*
	G	**92,13**	91,91
		90,88*	92,36*
Computadores	**P**	84,32	**82,47**
		86,43*	81,10*
	M	**90,23**	88,97
		90,13*	88,69*
	G	**93,43**	92,96
		92,48*	93,71*
Sapatos	**P**	**76,08**	73,63
		79,20*	74,98*
	M	**84,35**	83,86
		84,11*	81,30*
	G	**89,40**	88,70
		90,28*	91,26*
Relógios	**P**	**84,99**	77,71
		87,31*	83,78*
	M	**91,30**	89,63
		91,17*	89,50*
	G	93,34	**93,57**
		93,52*	93,62*

Tabela 12. F1 *scores* de LMs multilíngua nos *datasets* do WDC. Resultados com asterisco foram reportados em [Mozdzonek et al. 2022].

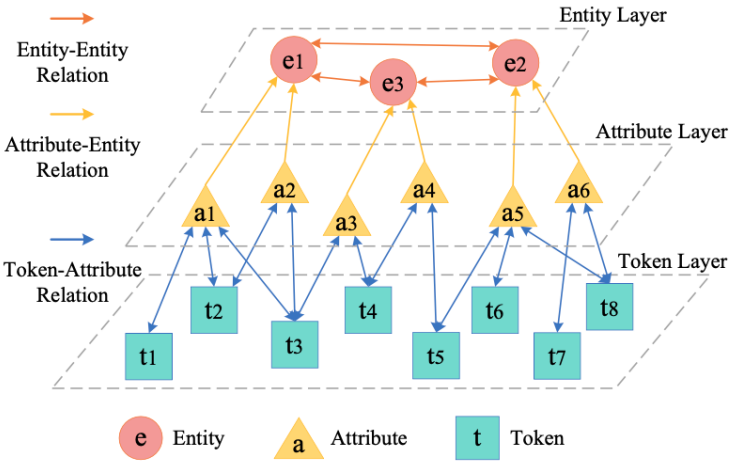

Figura 6. Estrutura do Grafo Hierárquico Heterogêneo, proposto em [Yao et al. 2022].

4.6. HierGAT

LMs pré-treinados são construídos em substanciais *corpora* de texto. Ainda assim, é possível que palavras específicas não apareçam em seu pré-treinamento, como o nome de uma marca ou modelo de produto específico. Ademais, nas técnicas apresentadas anteriormente, não se considera a relevância relativa de um atributo específico da entidade, situação onde pode haver informação contextual valiosa.

Para endereçar estes problemas, [Yao et al. 2022] apresenta o HierGAT (Rede Hierárquica Grafos de Atenção – *Hierarchical Graph Attention Network*), uma solução de PM que além de recorrer aos já mencionados LMs, adiciona ao modelo um novo tipo de rede, com mecanismo de atenção, baseado em grafos, denominado Grafo Hierárquico Heterogêneo (HGG). Neste, os nós são divididos em grupos, em três camadas, os de *tokens* individuais, (v^t) os quais são conectados com os de atributos (V^a), que por sua vez são ligados com os de entidades completas (V^e), como ilustra a Figura 6.

O *pipeline* da técnica, ilustrado na Figura 7, começa pela já citada etapa de *blocking*. Em seguida, usando a Auto Atenção de um LM, neste caso RoBERTa, são calculados as *embeddings* em nível de *token*, onde cada um é atualizado com o contexto de seus vizinhos. Em seguida, passando pelo HGG, são computados as *embeddings* de atributo, onde os *tokens* são refinados de acordo com seus atributos. A soma do resultado dessas operações resulta em uma *embedding* denominado *Palavra+Contexto* (Word+Context – W+C).

Na sequência, o W+C passa por uma camada de agregação, em que valores são combinados com atributos, e atributos com entidades. O resultado ainda passa por uma camada de comparação hierárquica, onde a semelhança entre duas entidades é medida conforme a similaridade dos atributos com maior peso calculado nas etapas anteriores.

Por fim, a *embedding* final do token [CLS] passa por uma rede classificadora com entropia cruzada. Os autores ainda apresentam um segundo método, o HierGAT+, no qual a comparação é realizada entre um produto e vários candidatos a correspondência dentro de um mesmo grafo, em um procedimento denominado Correspondência de Entidades Coletiva. Nesta abordagem, na etapa de agregação é realizada a concatenação das

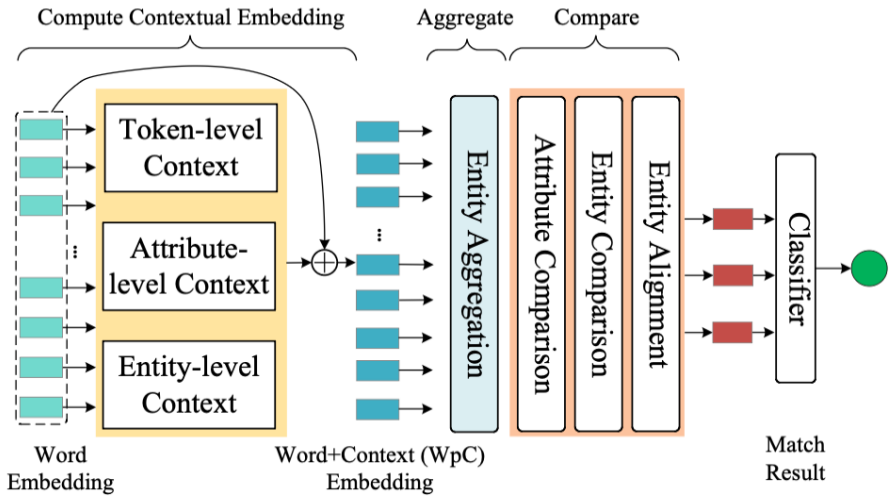

Figura 7. *Pipeline* do HierGAT. Figura extraída de [Yao et al. 2022].

representações de entidades com as dos atributos equivalentes.

Os experimentos foram conduzidos seguindo as instruções do artigo, realizando o treinamento por 10 *epochs*, com *batches* de tamanho 16, taxa de aprendizagem em $1e^{-5}$ sequência de texto limitada a 512. Todos os testes foram executados por três vezes consecutivas. Apesar dos autores terem mencionado o teste nos *datasets* do WDC, os resultados não foram reportados no texto. Portanto, foram realizadas as reproduções no Magellan, com exceção do conjunto Cervejas que apresentou erro durante o treinamento.

	HierGAT	**HierGAT+**
Abt-Buy	90,58	93,58
	89,80*	92,90*
Amazon-Google	73,54	85,19
	76,00*	83,00*
Cervejas	–	–
	92,30*	–
Walmart-Amazon	82,51	–
	86,30*	92,30*

Tabela 13. F1 *scores* do HierGAT e HierGAT+, utilizando o RoBERTa. Resultados com asterisco são reportados em [Yao et al. 2022].

Os resultados encontrados do HierGAT se alinham com o reportado, com exceção do Walmart-Amazon que apresentou uma diferença de quase quatro pontos. O HierGAT+ apresenta melhor performance. Porém, como apontado pelo autor, a comparação desses números com os outros métodos é inválida, já que a tarefa de CE Coletiva usa um conjunto de testes alternativos.

5. Discussão

Os resultados dos trabalhos demonstram o claro potencial da aplicação de técnicas baseadas em variações do BERT para a tarefa de PM. Nesta seção, estas técnicas são comparadas diretamente usando apenas os resultados obtidos nas reproduções. Ainda que

todos os artigos tenham sido testados nas mesmas coleções de *datasets*, alguns métodos foram avaliados em apenas um subconjunto desses. Por este motivo, a análise é feita faseadamente.

No contexto dos *datasets* Magellan, é interessante notar que não houve diferença significativa, em nenhum método, entre os testes nas variações estruturada e suja do *Walmart-Amazon*, evidenciando a capacidade destes modelos de conseguir ignorar ruído sem contexto. Isso aconteceu mesmo com a simples aplicação de ajuste fino ao RoBERTa, que inclusive apresentou performance muito similar às abordagens mais elaboradas, como pode ser visto na Tabela 14. A única exceção se dá no subconjunto Cervejas, onde o uso *augmentation* ocasionou em expressiva melhora, corrigindo sua provável falta de exemplos suficientes. Nos Magellan, a abordagem de [Peeters and Bizer 2022] provou-se como a melhor delas. No entanto, foi avaliado em apenas 2 dos 5 *datasets*.

	A-B	**A-G**	**Cervejas**	**W-A**	**W-A (S)**
RoBERTa	90,91	75,7	83,23	85,72	84,42
Ditto	90,26	75,34	86,84	85,34	85,24
Ditto (DA)	90,46	76,59	**91,33**	**85,95**	**85,75**
HierGAT	90,58	73,54	–	82,51	–
R-SupCon(C)	93,25	**78,87**	–	–	–
R-SupCon(C)+a	**93,74**	72,85	–	–	–

Tabela 14. Comparativo de F1 *scores* entre diferentes métodos nos *Magellan*.

O mesmo acontece nos *datasets* WDC, o SupCon alcança os maiores F1 *scores*, porém apenas foi replicado (e reportado) no subconjunto Computadores. Por este motivo, as comparações do WDC é feita primeiro apenas neste domínio (Tabela 15), e em seguida com os restantes.

Em Computadores, o Aprendizado Contrastivo Supervisionado também apresenta melhor performance, seguido pelo aprendizado intermediário com objetivo adicional de MLM. Este último, inclusive, se mostrou mais eficaz, por uma pequena margem, no menor subconjunto, mesmo quando o Supcon foi usado com *augmentation*. Nos *datasets* maiores, todos os modelos obtiveram pontuações superiores a 90. Isso acontece inclusive com o DistillBERT, evidenciando que com uma quantidade de dados de treinamento suficiente, é uma alternativa viável a alguns modelos mais lentos como o BERT e XLM-RoBERTa.

Na Tabela 16, são comparados os subconjuntos restantes do WDC. Neste agrupamento de modelos, o objetivo duplo do JointBERT alcança F1 *scores* próximos ou superiores a 97% nos *dataset* maiores, onde são disponibilizados exemplos com identificadores únicos suficientes. Nos conjuntos menores, o RoBERTa alcança os melhores resultados, seguido pelos modelos multilíngues.

É relevante apontar que, em média, há um grande salto de performance entre os *datasets* pequenos e médios. Fato esperado, devido à falta de exemplos nos conjuntos de treinamento menores. Esta melhora, no entanto, não é linear. Rendimentos decrescentes são observados conforme os *datasets* aumentam e pouquíssima vantagem é adquirida entre as variações G e XG. No caso específico de Relógios há, inclusive, uma queda na

	P	M	G	XG	Média
DistillBERT	75,00	81,93	90,27	93,18	85,10
BERT	77,25	87,30	91,83	94,37	87,69
XLM-ROBERTA	82,47	88,97	92,96	94,67	89,77
Ditto	81,70	88,04	91,62	96,05	89,35
mBERT	84,32	90,23	93,43	93,96	90,49
JointBERT	79,48	90,20	96,37	97,84	90,97
RoBERTa	87,98	91,63	94,57	95,11	92,32
CP-4cat	93,27	94,44	94,74	95,62	94,52
CP+MLM	**95,17**	96,31	94,53	95,77	95,45
R-SupCon(C)	93,27	**97,60**	**98,33**	98,33	96,88
R-SupCon(C)+a	94,62	**97,60**	**98,33**	**98,50**	97,26
Média	85,87	91,30	94,27	95,76	

Tabela 15. Comparativo de F1 *scores* entre diferentes métodos no *dataset* WDC Computadores.

média dos F1 *scores*.

	Var.	D.BERT	BERT	XLM-R.	mBERT	Ditto	RoBERTa	J.BERT	Média
Câmeras	**P**	67,81	79,95	81,46	80,72	79,46	**85,25**	79,45	79,16
	M	76,99	87,32	88,25	87,68	86,74	**92,21**	86,58	86,54
	G	86,36	89,63	91,91	92,13	92,70	93,32	**95,17**	91,60
	XG	89,50	91,15	91,75	90,30	94,32	94,65	**96,57**	92,61
Relógios	**P**	70,22	80,65	77,71	84,99	85,16	**86,85**	76,26	80,26
	M	80,45	89,14	89,63	**91,30**	90,59	91,15	87,54	88,54
	G	91,65	95,24	93,57	93,34	96,71	95,10	**97,61**	94,75
	XG	91,68	95,76	92,06	91,24	96,24	95,90	**98,67**	94,51
Sapatos	**P**	67,76	73,52	73,63	76,08	74,32	**77,03**	70,65	73,28
	M	74,07	80,11	83,86	**84,35**	82,23	83,64	82,64	81,56
	G	83,53	87,09	88,70	89,40	88,37	86,08	**96,00**	88,45
	XG	86,42	88,89	91,25	85,61	89,29	91,25	**97,18**	89,98
Média		80,54	86,54	86,98	87,26	88,01	89,37	88,69	

Tabela 16. Comparativo de F1 *scores* entre diferentes métodos nos *datasets* WDC.

6. Conclusão e Trabalhos Futuros

Neste trabalho, foi explorado o uso de técnicas de Machine Learning, com foco em LMs, para Product Matching. Mediante uma Revisão Sistematizada da Literatura, foi selecionada uma amostra relevante de trabalhos sobre o tema. Por meio da análise dos artigos e de uma extensa série de experimentos, foi possível confirmar os achados apontados pelos autores, notadamente a eficácia de técnicas que fazem uso de LMs pré-treinados, com destaque para o RoBERTa.

Valendo-se destes modelos, abordagens mais elaboradas, como Aprendizado Contrastivo Supervisionado e Aprendizado Multiobjetivo, alcançam F1 *scores* superiores a

94%, mesmo em *datasets* menores. A associação entre o RoBERTa com o SupCon, apresentada em [Peeters and Bizer 2022] alcançou os melhores resultados em todas as tarefas em que foi avaliada. Apesar disso, é importante notar que tal técnica necessita de identificadores únicos em seu conjunto de treinamento.

Trabalhos futuros poderiam explorar a elaboração de um *benchmark* em português para a avaliação de PM, onde hoje existe uma lacuna. Os dados brutos do WDC já contém uma seleção de mais de 500 mil páginas brasileiras, indicando um possível ponto de partida. Além disso, é necessário investigar o uso LLMs (*Large Scale Language Models*, ou Modelos de Linguagem de Grande Escala) para Product Matching. Publicações posteriores à realização desta revisão, que os utilizam, alcançam resultados superiores aos aqui expostos [Peeters and Bizer 2023]. Contudo, estes modelos requerem ambientes consideravelmente mais exigentes, em termos de processamento gráfico. Portanto, como as técnicas baseadas no BERT possuem performance próxima, ainda se destacam como as alternativas mais viáveis para a tarefa de Product Matching.

Referências

[Bafna et al. 2016] Bafna, P., Pramod, D., and Vaidya, A. (2016). Document clustering: Tf-idf approach. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 61–66. IEEE.

[Bridle 1990] Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, pages 227–236. Springer.

[Brunner and Stockinger 2020] Brunner, U. and Stockinger, K. (2020). Entity matching with transformer architectures-a step forward in data integration. In *23rd International Conference on Extending Database Technology, Copenhagen, 30 March-2 April 2020*, pages 463–473. OpenProceedings.

[Chen et al. 2020] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

[Das et al. 2024] Das, S., Doan, A., G. C., P. S., Gokhale, C., Konda, P., Govind, Y., and Paulsen, D. (2024). The magellan data repository. `https://sites.google.com/site/anhaidgroup/useful-stuff/the-magellan-data-repository`, Último acesso em: 27/07/2024.

[Devlin et al. 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Eisenstein 2018] Eisenstein, J. (2018). Natural language processing. *Jacob Eisenstein*.

[Foxcroft et al. 2021] Foxcroft, J., Chen, T., Padmanabhan, K., Keng, B., and Antonie, L. (2021). Product matching lessons and recommendations from a real world application. *Proceedings of the Canadian Conference on Artificial Intelligence*.

[Goodfellow et al. 2016] Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. `http://www.deeplearningbook.org`.

[Haykin 2009] Haykin, S. (2009). *Neural networks and learning machines, 3/E*. Pearson Education India.

[Ho and Wookey 2020] Ho, Y. and Wookey, S. (2020). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813.

[Joulin et al. 2016] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

[Jurafsky and Martin 2023] Jurafsky, D. and Martin, J. H. (2023). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.

[Khosla et al. 2020] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673.

[Kitchenham and Charters 2007] Kitchenham, B. A. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.

[Li et al. 2020] Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1):50–60.

[Liu et al. 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[Manning 2008] Manning, C. D. (2008). *Introduction to information retrieval*. Syngress Publishing.

[Mikolov et al. 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mozdzonek et al. 2022] Mozdzonek, M., Wroblewska, A., Tkachuk, S., and Lukasik, S. (2022). Multilingual transformers for product matching – experiments and a new benchmark in polish. In *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, volume 2022-July, pages 1–8. IEEE.

[Mudgal et al. 2018] Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 19–34, New York, NY, USA. Association for Computing Machinery.

[Paganelli et al. 2023] Paganelli, M., Tiano, D., and Guerra, F. (2023). A multi-facet analysis of bert-based entity matching models. *The VLDB Journal*, pages 1–26.

[Peeters and Bizer 2021] Peeters, R. and Bizer, C. (2021). Dual-objective fine-tuning of bert for entity matching. *Proc. VLDB Endow.*, 14(10):1913–1921.

[Peeters and Bizer 2022] Peeters, R. and Bizer, C. (2022). Supervised contrastive learning for product matching. In *Companion Proceedings of the Web Conference 2022*, pages 248–251. ACM.

[Peeters and Bizer 2023] Peeters, R. and Bizer, C. (2023). Entity matching using large language models. *arXiv preprint arXiv:2310.11244*.

[Peeters et al. 2020] Peeters, R., Bizer, C., and Glavas, G. (2020). Intermediate training of bert for product matching. In *DI2KG@VLDB*.

[Pennington et al. 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Primpeli et al. 2019] Primpeli, A., Peeters, R., and Bizer, C. (2019). The wdc training dataset and gold standard for large-scale product matching. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 381–386, New York, NY, USA. Association for Computing Machinery.

[Rumelhart et al. 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Russell and Norvig 2010] Russell, S. and Norvig, P. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition.

[Sanh et al. 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[Vaswani et al. 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Yao et al. 2022] Yao, D., Gu, Y., Cong, G., Jin, H., and Lv, X. (2022). Entity resolution with hierarchical graph attention networks. In *Proceedings of the 2022 International Conference on Management of Data*, SIGMOD '22, page 429–442, New York, NY, USA. Association for Computing Machinery.