Disciplina: Processamento de Linguagem Natural (2025.1)

Projeto Final - Comparação dos métodos Prediction by Partial Matching e Sequence-to-sequence para geração de texto)

Docente: Yuri de Almeida Malheiros Barbosa

Centro de Informática
Universidade Federal da Paraíba

Discente: Arthur Henrique da Silva
Matrícula: 20200077587
Discente: Ludmila Vinólia Guimarães Gomes
Matrícula: 20210025065

João Pessoa, 2025

Relatório - Comparação dos métodos Prediction by Partial Matching e Sequence-to-sequence para geração de texto

Relatório apresentado à disciplina de Processamento de Linguagem Natural,
do curso de Engenharia da Computação do Centro de Informática,
da Universidade Federal da Paraíba.

Professor: Yuri de Almeida Malheiros Barbosa

Setembro de 2025

Sumário

1 INTRODUÇÃO

1.1 Contextualização e Apresentação do Problema

O surgimento formal da área de Teoria da Informação, em 1948, com o artigo de Claude Shannon, "*A Mathematical Theory of Communications*"levou ao desenvolvimento de metodologias voltadas para o problema da compressão de dados, com a árvore de *Huffman* e suas variantes. Assim, em 1984, o trabalho "*Data compression using adaptive coding and partial string matching*"introduziu o algoritmo do PPM, cuja ideia central consiste na construção de um modelo contextual adaptativo sobre a mensagem recebida.

Já linguagem natural é o termo que se refere à linguagem usada para comunicações realizadas por seres humanos. Nesse sentido, o campo de Processamento de Linguagem Natural (PLN) surgiu junto aos computadores, em meados de 1940, quando a tradução entre línguas foi um dos primeiros problemas submetidos às máquinas. Desde então, seu objetivo tem sido desenvolver métodos capazes de extrair representações e significados de textos escritos em linguagem natural.

Essas duas áreas se cruzam em diversos pontos. Considerando que modelos de compressão podem ser aplicados a mensagens em linguagem natural, tais modelos devem se adaptar às probabilidades dos símbolos usados, o que implica em uma "compreensão" da linguagem humana. Além disso, é possível observar intersecções claras entre os conceitos de Teoria da Informação e PLN quando se consideram métricas como Entropia Cruzada e a utilização de Cadeias de Markov.

Nesse contexto, o presente trabalho busca analisar e comparar métodos da Teoria da Informação e do Processamento de Linguagem Natural, em particular os modelos Prediction by Partial Matching (PPM) e Sequence-to-sequence (Seq2seq), no cenário de geração de texto. A intenção é verificar, a partir de um mesmo dataset utilizado no treinamento desses modelos, os respectivos resultados, métricas e vantagens e desvantagens de cada abordagem.

1.2 Objetivos

Assim sendo, formalizamos a seguir os objetivos visados no presente relatório:

- Aplicar o mesmo dataset, com mesmo pré-processamento, aos modelos PPM e Seq2seq, de modo a garantir condições justas de comparação.
- Avaliar o desempenho dos modelos na tarefa de geração de texto por meio de métricas adequadas, como perplexidade, entropia cruzada e qualidade perceptiva do texto gerado.

- Estabelecer uma comparação entre os dois métodos, destacando semelhanças e diferenças no comportamento, eficiência e adequação de cada um para o problema proposto.
- Identificar e discutir as vantagens e limitações inerentes a cada modelo no contexto da geração de texto, considerando aspectos como qualidade, coerência, custo computacional e escalabilidade.
- Contribuir para uma melhor compreensão das intersecções entre métodos clássicos de compressão e modelos modernos de PLN, evidenciando possíveis complementaridades ou inspirações para trabalhos futuros.

2 DADOS E PRÉ-PROCESSAMENTO

2.1 Dataset utilizado

Inicialmente, foram realizados testes com o modelo PPM usando apenas 100mb de arquivos de texto retirados do projeto Gutenberg. Verificou-se que os resultados não foram excelentes, então, nesse sentido, buscou-se, para os testes atuais, utilizar uma maior quantidade de dados e de variadas fontes que apresentassem textos em inglês.

Assim, foram utilizados arquivos do projeto Gutenberg, já mencionado acima, a partir de um script para download dos .txt. Usando verificações quanto às quantidades dos caracteres válidos (no caso, que estivessem presentes no alfabeto da língua inglesa), quanto à quantidade de símbolos numéricos.

Para além desses dados, foram utilizados dois datasets disponibilizados na plataforma do Kaggle. O primeiro contém artigos extraídos do Medium, e deste foram usados quase 300mb de arquivos. Já o segundo apresenta estórias de ficção científica retiradas do projeto Writing with the machine, que foi utilizado em sua totalidade.

Para uso nos treinamentos dos modelos, foi criado um script para realizar a concatenação dos conteúdos presentes em um diretório e criaram-se três arquivos concatenados com os textos de cada um dos conjuntos de dados. Ao fim, com o objetivo de extrair a métrica de perplexidade apresentado pelo modelo PPM, foi usado um arquivo único com todos os dados concatenados.

2.2 Pré-processamento dos Dados

Considerando a separação e adaptação dos arquivos já explicitadas acima, agora são abordados os métodos para pré-processamento aplicados aos dados.

O processo utilizado consistiu em uma limpeza com a normalização dos caracteres, onde foram mantidos apenas os símbolos do alfabeto latino presentes na língua inglesa, além dos símbolos da vírgula, ponto e espaço (',' '.' e ' ', respectivamente). Por fim, houve a substituição de caracteres de espaço, estes sendo tabulação, quebra de linha, entre outras representações incluídas no padrão $\backslash s$ das expressões regulares, por um único espaço ' '.

Por fim, foram estruturados os dados de forma adequada para o treinamento do seq2seq. Inicialmente, cada caractere foi convertido para sua respectiva representação numérica utilizando os dicionários `char2idx` e `idx2char`, gerando assim um vetor de inteiros correspondente ao texto original. Esse vetor foi transformado em um objeto do tipo `tf.data.Dataset`, que possibilita o manuseio eficiente durante o treinamento.

As sequências de entrada foram organizadas em blocos de tamanho fixo, garantindo que cada amostra possua o contexto e o próximo símbolo a ser previsto. Em seguida, cada bloco foi dividido em duas partes: o texto de entrada, composto por todos os caracteres exceto o último, e o texto de saída, representado pelo último caractere, o qual deve ser previsto pelo modelo.

Finalmente, o dataset foi dividido em subconjuntos de treinamento e validação, utilizando 90% dos dados para treino e 10% para validação. Cada subconjunto foi estruturado em lotes, com descarte do último lote caso não atingisse o tamanho definido, além da aplicação da técnica de *prefetching*, usada aqui para que seja possível carregar lotes menores em memória no lugar de carregar o dataset inteiro, o que leva ao uso excessivo de memória, considerando o grande tamanho dos datasets usados no presente trabalho.

3 METODOLOGIA

3.1 Sobre modelo PPM (Prediction by Partial Matching)

A implementação segue o modelo PPM-C (*Prediction by Partial Matching*, versão C), em conjunto com a técnica de codificação aritmética para compressão de dados. O algoritmo foi estruturado em torno de uma árvore de contextos (trie), onde cada nó armazena informações estatísticas sobre a ocorrência de símbolos em determinados contextos de comprimento limitado.

Na prática, cada nó da trie guarda:

- O símbolo representado;
- A contagem de ocorrências;
- Referências para filhos (possíveis símbolos seguintes no contexto);

- Ponteiro para o contexto reduzido (*vine*), permitindo retroceder para contextos menores em caso de *escape*;
- O pai e o nível do contexto.

A lógica principal está na classe que gerencia a árvore (*ArvorePPM_C*), a qual mantém o estado atual do modelo durante compressão e descompressão. Nela, destacam-se:

- Codificação de símbolos: ao receber um novo símbolo, o algoritmo percorre os contextos do maior para o menor, procurando probabilidades já registradas. Caso o símbolo não exista, emite-se um escape e reduz-se o contexto, até atingir o nível zero.
- Decodificação: segue a mesma lógica em sentido inverso, utilizando os intervalos fornecidos pelo decodificador aritmético.
- Atualização do modelo: após cada símbolo processado, a árvore é atualizada com a nova ocorrência, permitindo adaptação dinâmica ao fluxo de entrada.
- Controle de frequências: para evitar estouro numérico, as contagens dos nós são reescaladas sempre que atingem um limite pré-definido.
- Símbolos especiais: além dos bytes de entrada, são utilizados símbolos reservados para indicar escape e fim da entrada.

Como recurso complementar, a implementação também realiza o cálculo da entropia empírica da fonte, da perplexidade do modelo e do comprimento médio em bits por símbolo, possibilitando avaliar o desempenho teórico da compressão em relação ao limite de Shannon.

Baseado nos recursos explicados acima, é implementada a construção da árvore sem que haja compressão do conteúdo de entrada a fim de utilizar no contexto de geração de texto.

3.2 Sobre modelo Sequence-to-sequence

O modelo desenvolvido segue a abordagem de predição sequencial, na qual a geração de cada novo símbolo é feita com base na sequência de símbolos anteriores. A cada passo, o sistema utiliza um contexto (uma parte do texto já visto/gerado) para inferir qual elemento deve sucedê-lo, repetindo este processo de forma iterativa. Trata-se, portanto de um processo no qual a saída de um instante é realimentada como entrada para o instante seguinte.

A preparação dos dados inicia-se pelo tratamento do corpus, processo já descrito na seção anterior. Em seguida, o texto é segmentado em múltiplos pares de exemplo, cada um composto por uma sequência de entrada e o símbolo seguinte correspondente.

A arquitetura neural utilizada possui três componentes principais:

- **Camada de Embedding:** cada símbolo do texto é transformado em um vetor numérico de dimensão reduzida, permitindo uma representação compacta e significativa de cada elemento.
- **Camada LSTM:** processa os vetores gerados pela camada anterior, capturando padrões sequenciais e dependências temporais dentro da sequência.
- **Camada de Saída:** gera uma distribuição de probabilidades sobre os símbolos possíveis, determinando qual será escolhido como próximo na sequência.

Os principais hiperparâmetros que controlam a capacidade e o comportamento do modelo são:

- **Dimensão do Embedding:** define o tamanho dos vetores numéricos que representam cada símbolo, influenciando a riqueza da codificação e a capacidade do modelo em capturar relações entre símbolos.
- **Número de unidades na LSTM:** determina a memória interna da camada recorrente, ou seja, até que ponto a rede consegue considerar o contexto anterior para prever o próximo símbolo.
- ***Batch Size*** estabelece quantos exemplos são processados juntos antes de cada atualização dos parâmetros;
- **Temperatura:** controla o grau de aleatoriedade na escolha do próximo símbolo durante a geração de texto, influenciando a criatividade e diversidade do resultado final.

Durante a fase de geração, o sistema recebe uma sequência inicial fornecida pelo usuário e, a partir dela, passa a prever sucessivos símbolos, atualizando continuamente o contexto considerado. O processo de amostragem inclui um parâmetro de temperatura, que ajusta a distribuição de probabilidades: valores baixos favorecem escolhas mais determinísticas, enquanto valores altos aumentam a diversidade e imprevisibilidade do texto resultante.

A avaliação do modelo combina métricas quantitativas e qualitativas. Além da *loss* de treinamento e da perplexidade, são aplicadas medidas de diversidade lexical, como a

proporção de palavras únicas e de pares únicos em relação ao total gerado. Dessa forma, não apenas a adequação estatística do modelo é verificada, mas também sua capacidade de produzir textos variados, evitando repetições excessivas.

3.3 Experimentos para Avaliação

3.3.1 Objetivos dos Experimentos

A partir dos experimentos realizados, tem-se como objetivo conseguir extrair métricas que permitam a comparação direta entre os métodos utilizados, isto é, os modelos PPM e Seq2seq. Dessa forma, pretende-se obter métricas como perplexidade e distinct-n para unigramas e bigramas. Para além dessas métricas, são consideradas também avaliações qualitativas sobre os textos gerados.

3.3.2 Experimentos com Datasets

Inicialmente, foram realizados testes utilizando quase 700mb de arquivo extraídos do Projeto Gutenberg. A partir dessa abordagem, houve uma limitação quanto à qualidade dos dados, que continham textos, além de em inglês, nas línguas francesa, alemã e chinesa. Dessa forma, verificou-se uma baixa qualidade sobre a geração de texto de ambos os modelos.

Com isso, a abordagem foi alterada e obteve-se o conjunto de dados atual, utilizando documentos do Projeto Gutenberg considerando uma extração mais refinada, com mais verificações quanto à qualidade do texto extraído, além dos outros datasets já citados acima.

3.3.3 Configuração

O hardware utilizado para treinamento dos modelos apresenta a seguinte configuração: Intel Core i5-1135G7, 4 núcleos e 8 threads, 8 MB de cache de 2.40GHz até 4.20GHz, 16GB memória RAM. Em relação às tecnologias, utilizou-se Python, TensorFlow e Keras, além do pacote re de expressões regulares do Python e do nltk.

Sobre o PPM, inicialmente foram experimentados diversos valores para o contexto K, buscando o parâmetro que fornecesse melhor compressão e, por consequência, melhor geração de texto. Portanto, verificaram-se os valores de K de 1 a 8 (ponto onde foi atingido o limite de memória da máquina usada). É importante considerar as limitações impostas pela implementação própria, que não utiliza de otimizações e métodos que agilizem os cálculos e operações realizados.

Em relação ao seq2seq, o treinamento foi realizado com os seguintes parâmetros: sequência de entrada de tamanho 100, passo de 1, batch size de 32, ao longo de 6 épocas. Essa configuração busca equilibrar desempenho computacional e qualidade do modelo gerado.

3.3.4 Métricas Utilizadas

Como já dito, uma das métricas utilizadas foi perplexidade, que é basicamente uma medida do quão bem um modelo probabilístico prediz uma amostra ao quantificar a incerteza dele sobre o próximo token em uma sequência construída. Assim, a perplexidade é uma medida da quantidade de opções que o modelo considera para o próximo token, onde menos valores demonstram menos oppções e, portanto, previsões mais confiáveis.

A métrica Distinct-n, por sua vez, é uma medida da diversidade sobre o texto gerado e é a razão entre o número de n-gramas únicos sobre o total de n-gramas no texto gerado.

Em suma, essas métricas, aqui apresentadas brevemente, foram utilizadas com o intuito de fornecerem uma maneira de comparação direta entre os textos gerados por ambos os modelos trazidos no presente projeto.

3.3.5 Procedimento para Treinamentos

Os procedimentos tomados para o PPM foram simples e se tratam apenas da compressão do dataset para obtenção da perplexidade e do treinamento em si do modelo da árvore implementada, de forma a construir as probabilidades para um contexto K = 4 com base no conjunto de dados passado.

Para o Seq2seq, o texto foi convertido em uma sequência de inteiros, considerando um alfabeto fixo de caracteres. Em seguida, os dados foram organizados em sequências de tamanho 100 e o alvo (o próximo caractere). O conjunto resultante foi dividido em treino (90%) e validação (10%), ambos preparados com batching e prefetching para otimizar o carregamento durante o treinamento do modelo. Ao término do treinamento, a perplexidade foi calculada a partir da loss de validação final, servindo como métrica de avaliação da qualidade do modelo.

3.3.6 Limitações e Desafios

Os desafios encontrados baseiam-se nas dificuldades observadas quanto à qualidade dos datasets encontrados, devido à grande presença de ruído e de textos em línguas variadas. Essa questão dificultou a qualidade do texto gerado por ambos os modelos, o

que demonstra a importância de um conjunto de dados bem selecionado e preparado para bons resultados nessa tarefa.

Além dessas limitações, os modelos treinados, principalmente o PPM, têm grande dependência de máquinas mais poderosas, que costumam ser pouco acessíveis. Então, foram necessárias muitas horas de treinamento para estes, considerando que usaram-se muitos dados (com o intuito de melhorar a diversidade e qualidade na geração de texto).

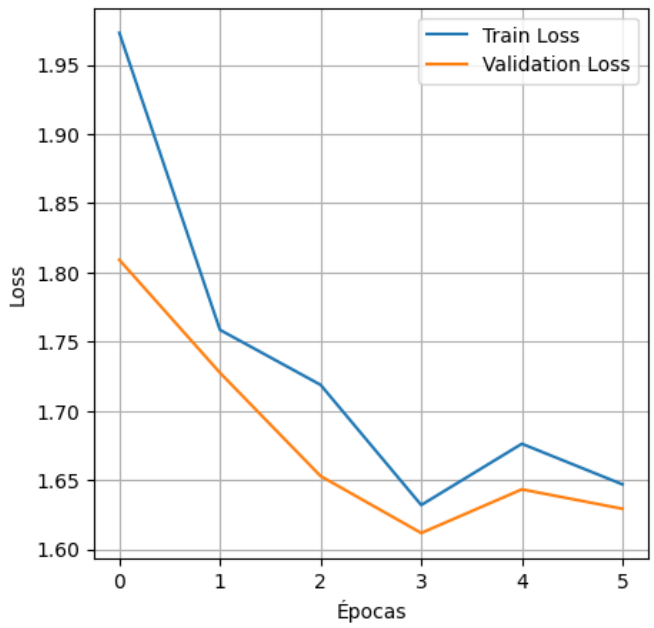
Por fim, destaca-se a dificuldade em aplicar métricas objetivas para avaliar a coerência dos textos gerados, uma vez que os modelos trabalham diretamente com tokens de caracteres individuais. Dessa forma, tornou-se necessário recorrer à análise qualitativa realizada por avaliadores humanos, que permitiu identificar aspectos de fluidez, consistência e inteligibilidade nas amostras produzidas.

4 RESULTADOS E DISCUSSÃO

4.1 Resultados Quantitativos

Após a realização dos testes e experimentos com os modelos PPM e seq2seq, foi possível extrair as métricas e realizar as comparações entre ambos. Então, apresentando os resultados obtidos, o PPM demonstrou perplexidade de 4.5214. Já sobre as métricas de Distinct-n resultou em 0.7714, para unigramas, e 0.9904, para bigramas.

O modelo seq2seq, por sua vez, alcançou perplexidade de 5.1004 e as métricas de Distinct-n: 0.5938 para unigramas e 0.9579 para bigramas. O gráfico representando as losses obtidas no treinamento e validação pode ser verificado logo abaixo.



Sobre os tempos obtidos para os modelos, foram feito testes com uma amostra do dataset (utilizando apenas os arquivos extraídos do Projeto Gutenberg de 104,2 MB) voltados apenas para a obtenção dos tempos. O PPM foi treinado em um total de 3888.058 segundos. Já o seq2seq levou 1017.9 segundos para treinamento.

A tabela abaixo exibe a comparação entre os modelos e as métricas utilizadas:

Modelo	Perplexidade	Distinct-1	Distinct-2	Tempo de Treinamento [s]
PPM	**4.5214**	**0.7714**	**0.9904**	3888.058
Seq2seq	5.1004	0.5938	0.9579	**1017.9**

Tabela 1: Comparação dos resultados obtidos para perplexidade, diversidade (Distinct-n) e tempo de treinamento entre os modelos avaliados.

4.2 Resultados Qualitativos

Abaixo, alguns textos foram gerados com ambos os modelos a partir da seed "the king" e com tamanho 400 da sequência gerada.

- Com PPM, que obteve os tempos de 0.0074 segundos e 0.0099 segundos, respectivamente, para a geração dos textos abaixo. O segundo texto gerado foi obtido a partir do sorteio dos símbolos a partir de suas probabilidades, mas com uma alteração para aumentar as maiores probabilidades e diminuir probabilidades pequenas (esse experimento foi realizado para verificar se ocorreriam melhorias nos resultados).

```
'the king show there offerent unknow that that has nementime.
   askets when crating it entinued it up of history of other legsy
   passwords enable comfortion it in complate someone runs of
   value man object the focus of miliationship on exchangement
   ideas women only frience, we calm, but where call rife any
   expect. the shaggy combia, at he time we nowhat superience of
   soltags you stilltime passigns. you no, his'
```

```
'the king to find of the conside this is can to see and she
   transactively different the new the box from the policense of
   the same to methick up the and not be the competition with me
   to the presented in the conting to set and really remain
   probably to the science is that is show face the face in a
   community way the technology in the some in the distring the
   are your set user how the presideral police were'
```

- Abaixo, textos gerados com o seq2seq, que, por sua vez, necessitou de 19 segundos, em média, para a geração de texto.

```
'the king to the more the moving the done and her didn t
   understand and desperes when the man be an experion and its all
   the good and good for a moved a and one because when the slump
   and a shaped, and she had a the most and a completic and
   decable of a pattern be a man of the look when the single what
   are the guin an even he standed and child the programs of the
   go of the light and in the paping because t'
```

```
'the king of him. so when the bases of the per below the land a
   and s do and result and found of the out the day. i do
   everything were libry, and he start and seemed a be a starting
   as we sat on the reminded and even as i saled the the started
   the stranged up and something and be a beginning and for a
   remard, for your other our lead about it are her the course and
   it in a could and a could be a far she was'
```

Analisando os textos obtidos com o PPM, percebe-se que a geração consegue entregar muitas palavras existente na língua inglesa (apesar de surgirem também palavras que não existem), mas é um texto que não obedece às regra gramaticais da língua e não apresenta mínima coerência quanto ao conteúdo presente. Sua vantagem é a geração em um tempo mínimo, sendo um modelo muito veloz no quesito da própria geração (porém, com um treinamento com tempo bastante elevado). Portanto, temos resultados obtidos com rapidez, com grande diversidade de palavras, mas com quase nenhuma coerência ou semântica.

Já em relação aos textos obtidos com o modelo seq2seq, são percebidas as muitas palavras geradas que não existem, como espécies de alucinações do modelo. De certa forma, percebe-se uma construção mínima de frases e orações, mas com um vocabulário impreciso e irreal. Para tal geração, o tempo de 19 segundos demonstra o elevado custo computacional das redes neurais.

4.3 Discussão Final

Comparando os resultados obtidos, é possível concluir que o PPM se destaca pela rapidez e pela maior diversidade lexical, mas sofre com a falta de coerência e pela criação de termos inexistentes, reflexo direto de sua limitação em lidar com dependências de longo alcance. Já o seq2seq, apesar de demandar mais tempo de processamento, apresentou melhor desempenho qualitativo ao produzir textos mais estruturados e próximos da linguagem natural, ainda que com repetições, falta de diversidade e também geração de palavras inexistentes.

No contexto dos testes realizados, o seq2seq se mostrou mais eficiente, pois a qualidade do texto gerado é mais relevante do que a velocidade de processamento isolada, especialmente quando o objetivo é aproximar-se da fluidez da escrita humana.

Além disso, há espaço para melhorias significativas no seq2seq com ajustes de parâmetros e técnicas adicionais. O aumento do número de épocas de treinamento, a utilização de camadas adicionais de LSTM, maior dimensionalidade nos embeddings ou mesmo a adoção de mecanismos de regularização (como dropout) poderiam melhorar a diversidade e reduzir repetições. Outro ponto importante seria a curadoria do dataset: a eliminação de ruídos e a homogeneização do corpus em termos de idioma certamente ajudariam a aumentar a coerência e a riqueza do texto gerado.

Assim, enquanto o PPM evidencia claramente as restrições impostas pelo modelo estatístico de memória limitada, o seq2seq demonstra o potencial das arquiteturas neurais, desde que apoiadas em dados de qualidade e configurações adequadas, podendo superar largamente o desempenho de abordagens tradicionais.

REFERÊNCIAS

[1] MATERIAL DA DISCIPLINA. Introdução à Teoria da Informação. Leonardo Vidal. João Pessoa: Universidade Federal da Paraíba, 2025. Apostila.

[2] BRASILEIRASPLN. Livro de Processamento de Linguagem Natural, 1ª Edição, Parte 1, Prefácio. Disponível em: `https://brasileiraspln.com/livro-pln/1a-edicao/parte1/prefacio.html`. Acesso em: 29 set. 2025.

[3] MELO, Wendel. Tutorial NLTK. Universidade Federal de Uberlândia. Disponível em: `https://www.facom.ufu.br/~wendelmelo/terceiros/tutorial_nltk.pdf`. Acesso em: 29 set. 2025.

[4] FRONTIERS. Perplexity and Communication Studies. Frontiers in Communication, 2022. Disponível em: `https://www.frontiersin.org/journals/communication/articles/10.3389/fcomm.2022.657725/full`. Acesso em: 29 set. 2025.

[5] KLAAS, Janne. SciFi Stories Text Corpus. Kaggle Dataset. Disponível em: `https://www.kaggle.com/datasets/jannesklaas/scifi-stories-text-corpus`. Acesso em: 29 set. 2025.

[6] CHIUSANO, Fabio. Medium Articles Dataset. Kaggle Dataset. Disponível em: `https://www.kaggle.com/datasets/fabiochiusano/medium-articles`. Acesso em: 29 set. 2025.

[7] COMET. Perplexity for LLM Evaluation. Disponível em: https://www.comet.com/site/blog/perplexity-for-llm-evaluation/. Acesso em: 29 set. 2025.

[8] ARYA.AI. LLM Evaluation Metrics. Disponível em: https://arya.ai/blog/llm-evaluation-metrics. Acesso em: 29 set. 2025.