Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Modelos Transformer para Inferência em Língua Portuguesa

Giovani Tavares de Andrade

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisora: Profª. Drª Renata Wassermann
Cossupervisor: Me. Felipe Ribas Serras

São Paulo
2023

Agradecimentos

Aos meus queridos pai, mãe, irmão, irmã e amigos,

Pelos abraços nos dias difíceis, pelos sorrisos nos dias de vitória. À minha mãe Cilene e pai Claudiones, a base sólida do meu caminho. Aos meus irmãos de sangue Letícia e Gabriel e aos de alma Jéssica e Ian, a luz e alegria nos momentos sombrios.

Obrigado por serem as estrelas que iluminaram minha jornada até aqui.

Resumo

Giovani Tavares de Andrade. **Modelos Transformer para Inferência em Língua Portuguesa**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

O crescente interesse em Grandes Modelos de Linguagem (LLMs) destaca desafios em sua aplicabilidade devido à limitação de dados para treinamento em especial para línguas de baixo recursos. A arquitetura desses modelos, crucial para o seu sucesso em Processamento de Linguagem Natural (PLN), é um ponto crítico, conforme evidenciado por Brown et al., 2020. Este estudo investigou a capacidade de dois modelos de arquitetura *Transformer* de código aberto (XLM-RoBERTa e BERTimbau) em realizar Inferência em Linguagem Natural (NLI) em português, comparando resultados em diferentes conjuntos de dados. Os modelos e *framework* de sua construção foram disponibilizados no GitHub e Hugging Face, contribuindo para a comunidade lusófona de PLN. O estudo busca comparar os conjuntos de dados para o NLI em português e como as características de pré treinamento de modelos pode influenciar os resultados obtidos neles.

Palavras-chave: transformer. llm. bert. xlm. xlm-roberta. grande modelo de linguagem. multilinguismo. monolinguismo. inferência. processamento de linguagem natural. implicação. contradição. pln.

Abstract

Giovani Tavares de Andrade. **Transformer Models for Portuguese Language**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

The growing interest in Large Language Models (LLMs) highlights challenges in their applicability due to limited data for training specially for low-resource languages. The architecture of these models, crucial to their success in Natural Language Processing (NLP), is a critical point, as evidenced by Brown et al., 2020. This study investigated the ability of two *Transformer* architecture open source models (XLM-RoBERTa and BERTimbau) in performing Natural Language Inference (NLI) in Portuguese, comparing results on different datasets. The models and framework of its construction were made available on GitHub and Hugging Face, contributing to the Portuguese-speaking PLN community. The study seeks to compare datasets for NLI in Portuguese and how model pre-training characteristics can influence the results obtained in them.

Keywords: transformers. llm. bert. large language model. multilingualism. monolingualism.

Lista de Abreviaturas

NLP	*Natural Language Processing* (Processamento de Linguagem Natural)
RNN	*Recurrent Neural Networks* (Redes Neurais Recorrentes)
NLI	*Natural Language Inference* (Inferência de Linguagem Natural)
MLM	*Masked Language Model* (Modelo de Linguagem Marcarada)
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

Lista de Figuras

Lista de Tabelas

Lista de Definições

Sumário

Capítulo 1

Introdução e Metodologia

1.1 Introdução

O crescente interesse nos Grandes Modelos de Linguagem (*Large Language Models -LLMs*, em inglês) é acompanhado pelos desafios que limitam a sua aplicabilidade em diversos casos de uso, cada qual abordado de diferentes formas na literatura ou ainda carentes de maior aprofundamento devido à rápida velocidade que os avanços na área estão ocorrendo. Os dados disponíveis para treinamento são um dos fatores impositores da citada limitação. Como destacado em Brown *et al.*, 2020 — o artigo de apresentação do GPT3, um grande modelo de linguagem precursor daquele sobre o qual o *ChatGPT*[1] é baseado —, tal fator juntamente à arquitetura dos LLMs representa um fator indispensável ao seu sucesso em desempenhar diferentes tarefas de Processamento de Linguagem Natural (PLN, ou NLP na sigla em inglês). Desse modo, a *performance* nessas tarefas depende diretamente de dados disponíveis para que esses modelos sejam de fato aplicados em produção para desempenhá-las diretamente ou apoiar o aprendizado de tarefas mais complexas.

Dentre essas tarefas, a Inferência em Linguagem Natural — aqui referenciada na sigla NLI para a sua nomenclatura em inglês, *Natural Language Inference* — é frequentemente utilizada na avaliação de LLMs no que se denominam *downstream tasks*, que são aquelas realizadas após o pré-treinamento. O pré-treinamento é realizado de modo a permitir que os modelos aprendam características gerais dos idiomas codificados de modo que esse aprendizado possa ser aproveitado em tarefas mais complexas e/ou específicas — as *downstream tasks*. Salvatore, 2020 descreve NLI como uma tarefa de dedução e, sendo assim, modelos que a desempenham satisfatoriamente possuem evidência de terem bons resultados em deduzir informações a partir de sentenças de texto.

[1] https://chat.openai.com/

Sendo assim, o uso responsável de grandes modelos de linguagem em nações lusófonas como o Brasil necessita, dentre outros inúmeros e ainda crescentes requisitos, da compreensão das limitações que tais modelos possuem ao desempenhar o NLI, além de compreensão adequada dos dados de treinamento publicamente disponíveis para a realização dessa tarefa. Neste trabalho estudou-se a capacidade em realizar inferência em linguagem natural em português de dois LLMs de código aberto com a mesma arquitetura, mas com características de pré-treinamento distintas. Juntamente a isso, comparou-se os resultados obtidos por esses modelos em diferentes conjuntos de dados públicos. Além disso, a fim de colaborar com a comunidade de NLP lusófona e permitir expansão do trabalho, o *framework* de geração desses modelos adaptados para o NLI em português e os modelos em si foram disponibilizados no *GitHub*[2] *HuggingFace*[3], ambas sendo plataformas de código aberto que hospedam diferentes modelos de linguagem para a comunidade interessada em inteligência artificial.

Ao longo deste trabalho, Inferência em Linguagem Natural, *Natural Language Inference* e NLI serão utilizados de maneira intercambiável, assim como Processamento de Linguagem Natural, *Natural Language Processing* e NLP.

O presente capítulo está estruturado da seguinte maneira: na seção 1.1.1, a tarefa de inferência em linguagem natural é definida formalmente assim como são descritos os conjuntos de dados utilizados; na seção 1.2, a metodologia que possibilitou as análises dos LLMs e *datasets* para o desempenho do NLI em Língua Portuguesa é descrita.

1.1.1 Inferência em Linguagem Natural e Conjuntos de Dados

Salvatore, 2020 define NLI como a tarefa de sistemas em determinar a relação lógica entre um par de sentenças P e H denominadas *premissa* e *hipótese*, respectivamente. Tal determinação categoriza H como exclusivamente: implicação lógica de P, contradição de P ou neutra em relação a P.

Diversos conjuntos de dados públicos com classificações de pares *premissa* e *hipóteses* como descreveu Salvatore, 2020 estão disponíveis no *Hugging Face*, sendo o *Stanford Natural Language Inference* (SNLI) o maior deles (Bowman *et al.*, 2015). O SNLI contem 550k pares na língua inglesa classificados em implicação lógica (*entailment*), neutra (*neutral*) e contradição (*contradiction*). As classificações do SNLI foram feitas por cerca de 2500 anotadores humanos que receberam legendas (premissas) de imagens reais extraídas do *Flickr30k* corpus (Young *et al.*, 2014) e deveriam criar, a partir de seus conhecimentos

[2] https://github.com/

[3] https://huggingface.co

acerca da realidade, três legendas (hipóteses) alternativas a ela tais que: i. uma deveria ser uma descrição verdadeira da imagem, gerando uma implicação lógica da legenda original (*entailement*); ii. outra deveria ser uma descrição possivelmente verdadeira da imagem, gerando uma hipótese neutra em relação à original (*neutral*); iii. a última deveria contradizer a primeira (*contradiction*).

Neste trabalho, implicação lógica, acarretamento e *entailment* são utilizadas de maneira intercambiável, assim como: contradição e *contradiction*; neutro, neutra e *neutral*.

Williams *et al.*, 2017 descreve dois problemas no SNLI que desfavorecem o seu uso como *benchmark* para modelos de *Machine Learning*. O primeiro é relacionado ao domínio único do qual as premissas são extraídas, sendo ele a de legendas de imagens que descrevem situações reais. Tal domínio é caracterizado, portanto, por sentenças curtas e simples descritoras de cenas visuais que não capturam com frequência o sentido de fenômenos linguísticos importantes como o raciocínio temporal (termos como *ontem*, *hoje* e *amanhã*), de crença (termos como *saber*, *significar*, *conhecer*, etc.) e de afirmação (termos como *deveria*). Essa ausência inibe a capacidade dos modelos de realizar uma ampla gama de inferências fundamentais, bem como de permitir a avaliação de sua capacidade de realizá-las. O segundo problema é derivado do primeiro e caracteriza o SNLI como um *benchmark* "fácil"para os melhores modelos disponíveis atualmente cujos resultados nele se aproximam muito daqueles que humanos conseguiriam obter.

Multi-Genre Natural Language Inference (MNLI)

Os problemas do SNLI descritos motivaram a criação do corpus *Multi-Genre Natural Language Inference* (MultiNLI), denominado MNLI no presente trabalho. Este corpus em língua inglesa foi criado de maneira similar àquela do SNLI — anotadores humanos criaram hipóteses (uma neutra, uma implicação lógica e uma contradição) a partir de premissas. A fim de superar o problema de limitação de domínio característica do SNLI descrita anteriormente, o MNLI utilizou como fonte das premissas dez fontes de textos publicamente disponíveis: transcrições de conversas "face a face"entre pessoas, o relatório público gerado na Comissão Nacional contra Ataques Terroristas aos Estados Unidos realizada em 2004; cinco obras de não-ficção sobre a indústria têxtil e desenvolvimento infantil publicadas pela *Oxford University Press* (OUP); *websites* governamentais; cartas; artigos culturais populares; transcrições de conversas telefônicas; guias turísticos; verbetes sobre linguística para leigos; e obras de ficção cobrindo diferentes gêneros.

Para coletar pares de premissa e hipótese, foi apresentada a um anotador humano uma premissa retirada de um texto fonte e essa pessoa gerou três novas sentenças a partir dela (as hipóteses): uma que é necessariamente verdadeira ou apropriada sempre que a

premissa é verdadeira (emparelhada com a premissa e rotulada como *entailment*), uma que é necessariamente falsa ou inapropriado sempre que a premissa é verdadeira (rotulada como *contradiction*/ contradição) e aquele em que nenhuma das condições se aplica (*neutral*/neutra). Este método de coleta de dados garantiu que as três classes fossem representadas igualmente no MNLI, possuindo 433k pares no total em sua versão em inglês (Williams *et al.*, 2017), caracterizando-o como um *dataset* balanceado.

O conjunto de treinamento do MNLI só possui cinco dos dez gêneros descritos anteriormente, e seu conjunto de testes é dividido em dois: o conjunto de testes *matched*, cuja distribuição é a mesma das cinco do conjunto de treinamento; e o conjunto de testes *mismatched*, que possui exemplos das distrubuições não presentes no conjunto de treino. Essa divisão permite que o MNLI seja utilizado: i. para avaliar modelos classificadores em sua capacidade de compreensão linguística (*Natural Language Understanding* (NLU)) somente em relação à distribuição do conjunto de treinamento (conjunto de testes *matched*); ii. para avaliá-los em sua capacidade de generalização em NLU (conjunto de testes *mismatched*). Isso confere ao MNLI mais uma vantagem sobre o SNLI em que essa divisão do conjunto de testes não está disponível.

No presente trabalho utilizou-se a versão em Língua Portuguesa do MNLI disponível no *Portuguese Language Understanding Evaluation* (PLUE) —denominada neste trabalho de PLUE/MNLI — para avaliar e treinar modelos de redes neurais em NLI. O PLUE/MNLI é uma tradução do *General Language Understanding Evaluation* (GLUE). O GLUE é um *benchmark* com nove conjuntos de dados voltados cada um a uma tarefa de processamento de linguagem natural distinta (NLP, na sigla em inglês), dentre elas o MNLI (Wang, Singh *et al.*, 2019). Para a criação do PLUE, foram utilizados modelos de redes neurais do OPUS-MT (Tiedemann e Thottingal, 2020). O OPUS é um conjundo de diversos *datasets* de textos extraídos da *web* traduzidos entre diversos idiomas, e o OPUS-MT é um repositório público do *GitHub* em que modelos tradutores treinados em diferentes *corpus* do OPUS são disponibilizados. Para os testes, utilizou-se apenas a versão *matched* do MNLI traduzida, denominada neste trabalho de $PLUE/MNLI_{validationMatched}$.

Embora o PLUE/MNLI seja um *dataset* em Língua Portuguesa, sua construção deu-se a partir de traduções automáticas do SNLI. A seguir, apresentam-se conjuntos de dados voltados a NLI criados diretamente na Língua Portuguesa.

ASSIN

Outro conjunto de dados com anotações de NLI utilizado foi aquele disponível na Avaliação de Similaridade Semântica e Inferência Textual (ASSIN) introduzido em Fonseca *et al.*, 2016. Apesar de o ASSIN conter anotações referentes a duas tarefas, NLI e similaridade

semântica, quando nos referimos ao ASSIN nesse trabalho estamos tratando apenas das anotações referentes à NLI. O ASSIN disponível no *HuggingFace*[4] possui uma versão contendo sentenças em português europeu e uma em português brasileiro, possuindo 5k pares anotados cada. Há ainda o ASSIN *full* que equivale à junção de ambas variedades linguísticas e é aquela utilizada no presente trabalho. Todas as versões possuem textos extraídos de notícias disponíveis na *web*.

O ASSIN contém pares de premissa e hipótese classificados como *entailment*, *paraphrase* e *neutral*, diferindo da definição de Salvatore, 2020. A justificativa de Fonseca *et al.*, 2016 para a ausência da anotação de *contradiction* é a raridade de sua observação no *corpus* utilizado para a construção desse conjunto. Nele, a *paraphrase* é a classificação de um par de premissa (P) e hipótese (H) em que H pode ser considerado verdadeiro sempre que P o for e vice-versa — sendo, portanto, equivalente a um *entailment* em ambos os sentidos.

As sentenças que compõem o ASSIN foram extraídas de notícias utilizando o agrupamento por assunto fornecido pelo *Google News*[5]. As notícias em português europeu foram coletadas do Jornal Público[6], enquanto aquelas na versão brasileira do idioma tiveram como fonte o portal G1[7]. Para a exploração de agrupamentos de notícias de modo a adquirir pares de sentenças similares posteriormente classificadas, utilizou-se o *Latent Dirichlet Allocation* (LDA). O LDA é um método de modelagem de espaços vetoriais que pontua pares de documentos segundo a sua similaridade (Fonseca *et al.*, 2016). Os pares considerados similares, então, foram classificados por quatro pessoas como *entailment*, *paraphrase* e *neutral*, sendo considerada a anotação escolhida pela maioria e excluindo-se pares cuja classificação resultou em empate.

ASSIN2

O ASSIN2 é a segunda edição da Avaliação de Similaridade Semântica e Inferência Textual (ASSIN), contendo, tal como o ASSIN, um conjunto de dados anotados para a tarefa de NLI. Neste trabalho, utiliza-se a sigla ASSIN2 como referência para esse conjunto de dados, embora nesse conjunto, assim como no ASSIN, anotações para outra tarefa de NLP também esteja disponível. O ASSIN2 disponível no *HuggingFace*[8] possui uma única versão contendo sentenças em português brasileiro, com 6.5k pares anotados. Os dados desse

[4] https://huggingface.co/datasets/assin

[5] https://news.google.com/home?hl=pt-BR&gl=BR&ceid=BR:pt-419

[6] https://www.publico.pt/

[7] https://g1.globo.com/

[8] https://huggingface.co/datasets/assin2

conjunto são baseados no SICK-BR (Real, Rodrigues *et al.*, 2018), um outro *dataset* que é tradução do SICK. O SICK, tal qual o MNLI, é baseado em legendas de imagens, possuindo dessa forma, menor complexidade linguística que o ASSIN (Real, Erick Fonseca *et al.*, 2019).

O ASSIN2 possui pares de premissa e hipótese classificados como apenas *entailment* ou *neutral* de maneira balanceada, o que significa que ele possui a mesma quantidade de pares pertencentes a tais classes. Essa característica foi obtida de maneira intencional e representa uma melhoria em relação ao ASSIN, no qual há mais pares classificados como *neutral* que *entailment*. A presença de contradições inconsistentes e a menor quantidade de suas amostras no SICK justificaram a ausência dessa anotação no ASSIN2, já que o primeiro foi utilizado como base para o segundo (Real, Erick Fonseca *et al.*, 2019). Essa característica pode beneficiar o uso do ASSIN2 em modelos de *machine learning* como aqueles utilizados neste trabalho.

O processo de anotação do ASSIN2 se deu de forma semelhante àquela da primeira versão do ASSIN, sendo realizada por grupos de quatro pessoas com formação linguística. Assim como todos os outros *datasets* utilizados neste trabalho, o ASSIN2 foi recuperado na plataforma *HuggingFace*.

1.2 Metodologia e Objetivos

Os modelos capazes de realizar o NLI utilizados neste trabalho são modelos de *machine learning* de arquitetura *Transformer*. Mais detalhes sobre esta arquitetura estão disponíveis no próximo capítulo. Dois modelos com características de pré-treinamento distintas foram escolhidos: o XLM-RoBERTa e o BERTimbau. Em especial, investigou-se dois aspectos principais em relação à capacidade dos *Transformers* em realizar NLI em português: i. o impacto do pré treinamento multilingual nos resultados; ii. a variação dos resultados em diferentes *datasets* com características de anotação e classes distintas com o uso da avaliação cruzada descrita nesta seção.

Dessa forma, além da disponibilização pública dos modelos adaptados ao NLI e seu *framework* de geração, responder às perguntas **Sob quais circunstâncias um modelo multilingual tende a ser melhor no NLI em português?** e **Como a *performance* de *Transformers* varia na tarefa de NLI em diferentes distribuições linguísticas/*datasets*?** foram objetivos deste trabalho. Em especial, dadas as diferentes características de domínio, classes presentes e anotação dos *datasets* utilizados, buscou-se verificar como elas influenciaram os resultados.

Cada modelo citado foi ajustado e testado em cada um dos *datasets* descritos na seção Inferência em Linguagem Natural e Conjuntos de Dados. Desse modo, 2 modelos × 3 datasets = 6 modelos ajustados para o NLI foram disponibilizados e documentados no *HuggingFace*. Cada modelo foi avaliado em cada *dataset*, o que resultou em 6 modelos × 3 datasets = 18 conjuntos de resultados. Por exemplo, o XLM-RoBERTa ajustado no ASSIN foi testado no ASSIN, ASSIN2 e MNLI, tal qual foi o BERTimbau ajustado no ASSIN e o mesmo se fez para ajustes em outros conjuntos. Uma vez que os sistemas de classes no qual o modelo foi ajustado e aquele dos *datasets* em que foi testado não eram sempre equivalentes (por exemplo, o ASSIN é o único entre os três que contem paráfrases), um sistema de tradução de classes foi proposto e é descrito na subseção 1.2.2.

1.2.1 *Fine-Tuning* (Ajuste Fino)

A abordagem de *fine tuning* equivale à alteração dos parâmetros de um modelo pré treinado de modo a torná-lo especializado em alguma tarefa em particular. No caso dos modelos de rede neural como os utilizados no presente trabalho, o *fine tuning* é feito com a adição de uma camada de classificação à rede pré-treinada de modo que sua camada de *output* equivalha à saída da tarefa para a qual o *fine tuning* é realizado. Para isso, os parâmetros do modelo e de tal camada são alterados de maneira semelhante àquela do pré treinamento, mas exigindo menos tempo uma vez que o pré treinamento já foi realizado. De modo geral, essa técnica pode ser compreendida como a especialização de um modelo mais geral em alguma tarefa específica — a Inferência em Linguagem Natural, no presente caso.

Os *Transformers* utilizados foram especializados em realizar o NLI com o uso da biblioteca `transformers`[9] da plataforma *HuggingFace*. Essa biblioteca fornece diversas APIs e classes para o *download*, treinamento e *fine tuning* de diversos modelos pré treinados. Outra biblioteca do *HuggingFace*, a `datasets`[10], foi aquela utilizada para o *download* dos conjuntos de dados de NLI utilizados.

Ajuste de Hiperparâmetros

Há diversos hiperparâmetros que caracterizam o treinamento e o *fine tuning* de um modelo de arquitetura *Transformer*. Esses componentes são assim denominados porque seus valores que maximizam a *performance* do modelo em tarefas específicas não podem ser aprendidos em tempo de treinamento como os outros parâmetros da rede, precisando

[9] https://huggingface.co/docs/transformers/index

[10] https://huggingface.co/docs/datasets/index

ser definidos antes dele. Alguns exemplos de hiperparâmetros de redes neurais são o seu número de camadas, a taxa de aprendizagem, o tamanho dos *batches* de treinamento, etc.

A biblioteca `transformers` permite que diversos hiperparâmetros sejam definidos na instanciação da classe responsável pelo *fine tuning*. A otimização de tais parâmetros foi feita de modo a maximizar a acurácia de cada modelo nos *datasets* de avaliação (*evaluation datasets*) de cada um dos conjuntos descritos. A escolha dos hiperparâmetros ajustados e seu espaço de busca para cada modelo foi feita de acordo com os artigos que os introduziram quando disponíveis e/ou de acordo com outras literaturas, o que é descrito na Tabela 1.1 abaixo em que os valores com asterisco correspondem aos valores ótimos utilizados no *fine tuning* dos modelos disponibilizados.

Modelo	Épocas	Tamanho do *batch* por GPU	Taxa de Aprendizagem
Conjunto de fine tuning: $ASSIN_{train}$			
$BERTimbau_{large}$	$(2^*, 3, 4)$	$(8, 16^*, 32)$	$(3e-5, 2e-5, 5e-5^*)$
$XLM-RoBERTa_{base}$	$(1, 2, 3^*)$	$(8, 16^*, 32)$	$(1e-5, 2e-5^*, 3e-5)$
Conjunto de fine tuning: $ASSIN2_{train}$			
$BERTimbau_{large}$	$(2, 3^*, 4)$	$(8, 16, 32^*)$	$(3e-5, 2e-5, 5e-5^*)$
$XLM-RoBERTa_{base}$	$(1, 2, 3^*)$	$(8^*, 16, 32)$	$(1e-5, 2e-5, 5e-5^*)$
Conjunto de fine tuning: $PLUE/MNLI_{train}$			
$BERTimbau_{large}$	$(4, 5, 6^*)$	$(8, 16^*, 32)$	$(3e-5, 2e-5, 5e-5^*)$
$XLM-RoBERTa_{base}$	$(1, 2, 3^*)$	$(8, 16^*, 32)$	$(1e-5, 2e-5^*, 3e-5)$

Fonte: O autor.

Tabela 1.1: *Hiperparâmetros de Treinamento Ajustados*

Como observado na Tabela 1.1, os espaços de busca para os hiperparâmetros ótimos de cada modelo foram praticamente os mesmos para todos os *datasets* utilizados, com exceção do número de épocas de treinamento do PLUE/MNLI no *BERTimbau* ($BERTimbau_{large}$). No caso deste modelo, optou-se por avaliar a sua *performance* no PLUE/MNLI em épocas de tamanhos $4, 5$ e 6 ao invés de $2, 3$ e 4 como se fez com os ASSIN. Isso se deve a testes preliminares em que não se obteve resultados satisfatórios no PLUE/MNLI para o *BERTimbau* ajustado em menos épocas.

Os ajustes de hiperparâmetros foram monitorados pela plataforma *Weights & Biases*[11],

[11] https://wandb.ai/site

uma ferramenta de *Machine Learning Development & Operations* (MLOps) que fornece, dentre outros recursos, uma biblioteca em `Python` em que experimentos como tal podem ser monitorados em conjunto à `transformers`.

1.2.2 *Framework* dos experimentos e Testes Cruzados

Para a construção do *framework* de *fine tuning* do XLM-RoBERTa e do BERTimbau no ASSIN, ASSIN2 e PLUE/MNLI, utilizou-se um *fork* do repositório da biblioteca `transformers`. Especificamente, modificou-se o *script* em que se realiza o *fine tuning* de modelos multilinguais do *HuggingFace* no XNLI (Conneau, Lample *et al.*, 2018). As modificações no *script* foram feitas de modo a possibilitar: i. realização do ajuste de hiperparâmetros; ii. monitoramento dos experimentos com o *Weights & Biases*; iii. uso de arquivos de configuração e integração com a biblioteca `hydra`[12], facilitando a seleção de modelos, *datasets* e outros parâmetros. Como resultado, o `script` final desenvolvido serviu como um *framework* de *fine tuning* de modelos do *Hugging Face* utilizado ao longo de todo o trabalho disponível no repositório público[13] do presente trabalho. Os modelos ajustados, testados e documentados foram disponibilizados na *página do autor*[14] no *HuggingFace* e documentados.

Uma vez realizados o ajuste de hiperparâmetros e o *fine tuning* dos modelos em cada um dos *datasets*, cada um deles foi submetido às avaliações direta e cruzada. Para essa etapa, as previsões dos modelos foram submetidas a um sistema de tradução exposto a seguir. Para um par premissa (p) e hipótese (h) e seja X um conjunto de excertos textuais quaisquer, define-se:

$$E, P, \emptyset, C, N \ : \ X \times X \rightarrow \{True, False\} \tag{1.1}$$

$$F = E, P, \emptyset, C, N \tag{1.2}$$

$$f_{i,j} \in F \text{ as funções booleanas da tabela a seguir} \tag{1.3}$$

- $E(a, b)$: a é uma implicação lógica (*entailment*) de b.
- $P(a, b)$: a é uma paráfrase (*paraphrase*) de b.
- $\emptyset(a, b)$: b não é implicação lógica de a (*None*).
- $C(a, b)$: b é uma contradição de a (*contradiction*)

[12] https://hydra.cc/docs/intro/

[13] https://github.com/giogvn/Natural-Portuguese-Language-Inference

[14] https://huggingface.co/giotvr

- $N(a, b)$: b não é nem implicação lógica nem contradição de a (*Neutral*).

Classe Real		Previsão do Modelo		
Dataset	Classe	ASSIN	ASSIN2	PLUE/MNLI
ASSIN	$E(p,h)$ $P(p,h)$ $\emptyset(p,h)$		$E(p,h)$ $E(p,h) \wedge E(h,p)$ $\emptyset(p,h)$	$E(p,h)$ $E(p,h) \wedge E(h,p)$ $C(p,h) \oplus N(p,h)$
ASSIN2	$E(p,h)$ $\emptyset(p,h)$	$E(p,h) \oplus P(p,h)$ $\emptyset(p,h)$		$E(p,h)$ $C(p,h) \oplus N(p,h)$
PLUE/MNLI	$E(p,h)$ $C(p,h)$ $N(p,h)$	$E(p,h) \oplus P(p,h)$ $\emptyset(p,h)$ $\emptyset(p,h)$	$E(p,h)$ $\emptyset(p,h)$ $\emptyset(p,h)$	

Fonte: O autor.

Tabela 1.2: *Sistema de Tradução entre os* datasets

Na Tabela 1.2, são estabelecidos um conjunto de funções booleanas que permitem mapear as previsões de um modelo às classes reais de um sistema. Para um par premissa e hipótese (p, h), sejam:

$$i_{real} = i \text{ tal que } f_{i,2} = True \ : \ \text{a linha correspondente à classe real de } (p,h) \tag{1.4}$$

$$j_{predict} \ : \ \text{a coluna correspondente ao } \textit{dataset} \text{ em que um modelo foi ajustado} \tag{1.5}$$

$$k_{true} = k \text{ tal que } f_{k,j_{predict}}(p,h) = True \tag{1.6}$$

Em outras palavras, i_{real} é a linha sob a coluna Classe (coluna 2) em que a função booleana é verdadeira para aquele par, ou seja, é o que se espera que o modelo preveja; $j_{predict}$ é a coluna correspondente ao *dataset* que define as previsões que um modelo é capaz de fornecer para (p, h); k_{true} é a linha sob a coluna $j_{predict}$ em que a função booleana assume valor verdadeiro. Dessa forma, dada uma previsão de um modelo para a classe de um par (p, h), tal previsão foi considerada correta se e somente se: a célula i_{real}, $j_{predict}$ é vazia, o que corresponde à avaliação de um modelo no conjunto de testes do próprio *dataset* em que foi ajustado, o que equivale aos testes de um modelo submetido ao *fine tuning* no ASSIN no conjunto de testes do próprio ASSIN, por exemplo; ou se $i_{real} = k_{true}$, ou seja, a tradução da classe prevista pelo modelo para (p, h) corresponde à classe real de tal par.

A Tabela 1.2 permitiu avaliar todos os modelos ajustados em todos os conjuntos de dados disponíveis. Quando um modelo foi testado em um conjunto correspondente ao

domínio em que foi ajustado (modelo ajustado no ASSIN testado no conjunto de testes do ASSIN, por exemplo), isso se denominou **avaliação direta** e corresponde à uma célula vazia na Tabela 1.2. Quando foi testado em outro domínio, a isso se chamou **avaliação cruzada** (modelo ajustado no PLUE/MNLI testado no conjunto de testes do ASSIN, por exemplo). As avaliações cruzadas possibilitaram uma compreensão maior acerca tanto da capacidade de generalização de tais modelos, da qualidade dos *datasets* em viabilizá-la e da relação entre os domínios linguísticos cobertos por cada *dataset*.

Capítulo 2

Revisão Técnica

2.1 *Fine Tuned Models*

Para a realização da tarefa de NLI foram utilizados modelos *Transformer* (Vaswani *et al.*, 2017). *Transformers* são modelos de *deep learning* que usam a arquitetura *encoder-decoder* descrita a seguir na subseção 2.1.1. Os modelos utilizados no presente trabalho são modelos de classificação baseados nos *Transformers*,os quais usualmente só utilizam parte dessa arquitetura e um método de treinamento específico.

Esses modelos possuem a estrutura das redes neurais *sequence-to-sequence* com os melhores desempenhos atualmente (Vaswani *et al.*, 2017) denominada estrutura *encoder-decoder*. Modelos *sequence-to-sequence* (Sequência-para-sequência, em tradução livre ao português) produzem sequências de dados a partir de outros dados sequenciais. Excertos textuais podem ser representados como tais sequências, portanto os *Transformers* são modelos gerativos cujas saída e entrada são sequências textuais reais ou sintéticas.

2.1.1 A arquitetura *Transformer*

Introduzidos em 2017 no artigo intitulado "*Attention is All You Need*"(Vaswani *et al.*, 2017), os *Transformers* submetem a sequência de entrada a uma camada de codificação denominada *encoder* seguida de uma de decodificação denominada *decoder*. A arquitetura *encoder-decoder* resultante desse processo não é exclusividade dos *Transformers*, tendo sido demonstrada em redes neurais de tradução anteriormente (Sutskever *et al.*, 2014) . O denominado "Mecanismo de Atenção"presente em ambas camadas citadas é, por sua vez, parte importante dos aspectos que diferenciam os modelos apresentados em Vaswani *et al.*, 2017 de outros modelos *sequence-to-sequence*.

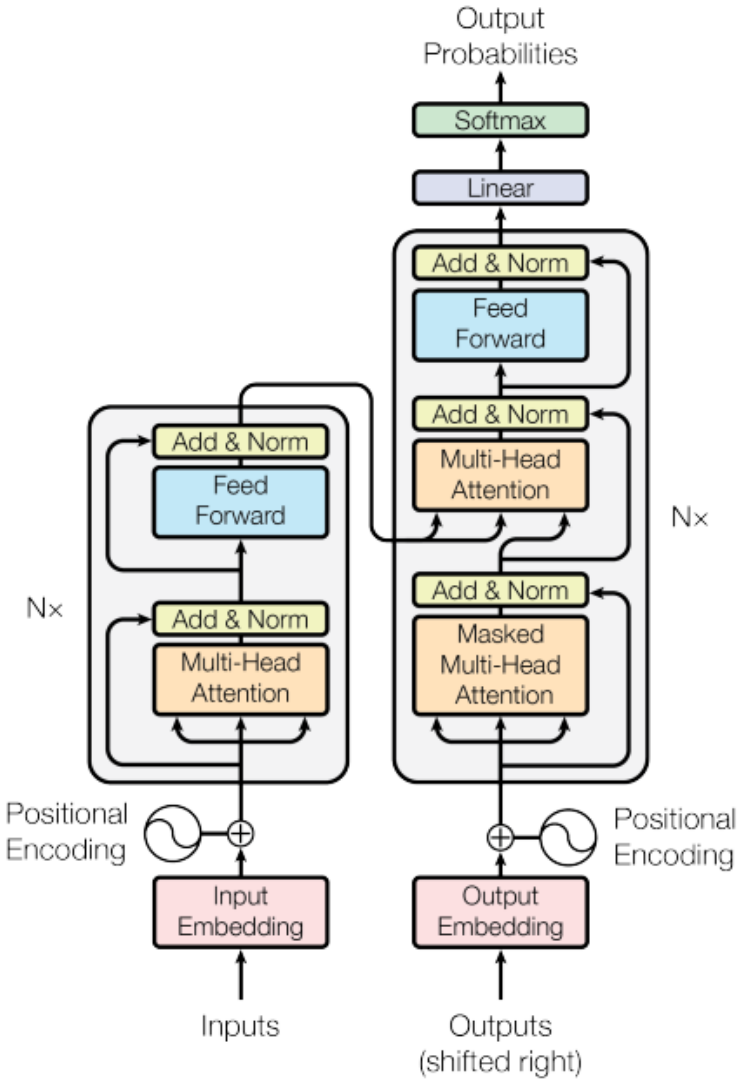


Figura 2.1: *A arquitetura Transformer*

O Mecanismo de Atenção

Definição 1. Atenção

Sejam $q \in \mathbb{R}^{d_{model}}$, $k \in \mathbb{R}^{d_{model}}$ e $v \in \mathbb{R}^{d_{model}}$, a *Função de Atenção (Attention)* é definida abaixo:

$$Attention(q,k,v) = SoftMax(\frac{\langle q,k \rangle}{\sqrt{d_k}})v = a_{q,k,v}$$

onde d_{model} é o número de dimensões/parâmetros do *embedding* (vetor) representante do *token*, também referido como "tamanho do modelo"no presente trabalho.

Da definição acima é possível concluir que $a_{q,k,v} \in \mathbb{R}^{d_{model}}$. Abaixo encontram-se inter-

pretações para q, k e v.

- **Query ($q \in \mathbb{R}^{d_{model}}$)**: é a representação da palavra ou sequência em seu contexto para a qual a atenção $a_{q,k,v}$ está sendo calculada . Usando vetores q, o *Transformer* é capaz de representar diferentes sequências de texto em uma mesma sentença/entrada.
- **Key ($k \in \mathbb{R}^{d_{model}}$)**: é a representação de outra palavra ou sequência no texto sendo processado pelo modelo. A realização do produto escalar entre ele e q permite ao *Transformer* mensurar a similaridade entre a sequência representada pela *query* e outras [*keys*].
- **Value ($v \in \mathbb{R}^{d_{model}}$)**: é a representação da informação contextual carregada por cada palavra no texto. Dessa forma, ao multiplicar o resultado do produto escalar entre q e k por v, o que se obtem é a similaridade entre q e k ponderada pela "importância"do contexto que k carrega, a qual é representado por v. Tal importância depende da tarefa que caracteriza o treinamento do modelo.

Evidentemente, os textos submetidos como *input* aos *Transformers* são formados por diversas palavras e, portanto, o mecanismo de atenção deve ser aplicado a múltiplas *queries*, *keys* e *values* — muitas sequências devem ser comparadas e mensuradas em diferentes contextos. Em Vaswani *et al.*, 2017 é introduzido o *Scaled Dot-Product Attention.* Tal definição sugere como aplicar o mecanismo de atenção de 1 em múltiplos q, k e v simultaneamente, ou seja, como calcular as relações internas existentes entre todas as sentenças de um texto de maneira paralela e, portanto, eficiente. Para isso, todas as *queries* q, *keys* k e *values* v são empilhadas nas matrizes Q, K e V respectivamente. Abaixo se define o *Scaled Dot-Product Attention* como em Vaswani *et al.*, 2017.

Definição 2. ***Scaled Dot-Product Attention***

Sejam $Q \in \mathbb{R}^{m_q \times d_{model}}$, $K \in \mathbb{R}^{m_k \times d_{model}}$ e $V \in \mathbb{R}^{m_k \times d_{model}}$, a função *Scaled Dot-Product Attention* é definida abaixo:

$$Attention(Q, K, V) = SoftMax(\frac{QK^T}{\sqrt{d_k}})V = A_{Q,K,V}$$

m_q é o número de *queries* cujas atenções estão sendo cauculadas e m_k é o número de *keys* com as quais as *queries* serão comparadas.

Analogamente ao que foi feito na Definição 1, é possível inferir $A_{Q,K,V} \in \mathbb{R}^{m_q \times d_{model}}$. Assim, pode-se interpretar a atenção da Definição 2 como uma matriz em que cada linha representa a atenção de uma única *query* relativa a diversas *keys*.

Nas Definições 1 e 2, todas as *queries*, *keys* e *values* têm d_{model} dimensões. Ou seja,

toda a informação é representada em subespaços vetoriais de dimensionalidades iguais às do modelo.

Atenção *Multi-Head*

Vaswani *et al.*, 2017 alega ser benéfico projetar linearmente as *queries*, *keys* e *values* h vezes em diferentes subespaços de dimensões d_k, d_k e d_v, respectivamente. Cada um desses subespaços é denominado *head*. A atenção (Definição 2) é então computada em cada uma dessas projeções, gerando resultados com d_v dimensões cada. Tais resultados são concatenados e então projetados de volta para o espaço com d_{model} dimensões, gerando o resultado final do mecanismo de atenção *Multi-head* (Vaswani *et al.*, 2017).

De maneira abstrata, a projeção das *queries* que compõem Q em diferentes subespaços equivale a representar as relações internas existentes em um texto segundo diferentes perspectivas. Por exemplo, uma das cabeças de atenção ($head_i$ na Definição 3) pode capturar as relações semânticas locais entre sentenças próximas dentro de uma janela de contexto fixa, enquanto outra as captura globalmente. A depender da tarefa de treinamento do modelo, uma cabeça pode ser mais relevante do que outra por capturar as relações mais importantes a ela existentes entre as sentenças do texto. Por exemplo, a tarefa de sumarização de texto requer a representação de relações globais no texto de modo a capturar de maneira sintética parte relevante de toda a sua informação, enquanto que a representação de relações locais são mais significativas para tarefas como o reconhecimento de entidades nomeadas. A alteração dos parâmetros de um modelo característica do *fine-tuning* (Seção 1.2.1) pode em parte ser compreendida como a ação de se alterar os pesos das cabeças de modo que aquelas mais relevantes para a tarefa em questão sejam favorecidas na obtenção da saída.

Definição 3. Atenção *Multi-Head*

Sejam $Q \in \mathbb{R}^{m_q \times d_{model}}$, $K \in \mathbb{R}^{m_k \times d_{model}}$ e $V \in \mathbb{R}^{m_k \times d_{model}}$. A atenção *Multi-Head* é definida abaixo:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^0$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where

$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$$

$$W_i^k \in \mathbb{R}^{d_{model} \times d_k}$$

$$W_i^V \in \mathbb{R}^{d_{model} \times d_v}$$

$$W^0 \in \mathbb{R}^{hd_v \times d_{model}}$$

As projeções referidas são realizadas com as matrizes W acima. O uso da atenção com h *heads* distintas gerando diferentes *queries*, *keys* e *values*, permite aos *Transformers* a representação de sentenças e textos segundo diversos aspectos e relações entre seus componentes. Sendo assim, o mecanismo de atenção *Multi Head* é fundamental para caracterizar essa arquitetura em sua capacidade de aprendizado de diferentes estruturas sintáticas e semânticas de um texto, justificando em parte o desempenho notável de modelos *Transformers* em tarefas de NLP. Os múltiplos atributos caracterizadores de linguagens escritas podem ser abstratamente representados pelas *heads*.

O *encoder*

O *encoder* dos *Transformers* consiste em uma pilha de L camadas idênticas h_i (a saída de h_{i-1} é entrada para h_i) com H dimensões escondidas aplicadas consecutivamente. Cada camada h_i pode ser compreendida como uma rede neural cujas subcamadas entre a de entrada e a de saída possuem H dimensões ou "neurônios". Dessa forma, como definido em Souza *et al.*, 2020, o *encoder* é matematicamente definido por

Definição 4. ***Encoder***

Seja x um vetor de *tokens* e h_i funções que definem redes neurais, o *encoder* é uma função h tal que:

$$h(x) = h_L(h_{L-1}(...h_2(h_1(x))...))$$
$$h_i : \mathbb{R}^{N \times H} \rightarrow \mathbb{R}^{N \times H}$$

Acima, N é o comprimento da sequência/texto de entrada em *tokens*. Cada camada h_i é formada por uma camada de atenção (3) e outra camada denominada *feedforward*, que é um *Multi Layer Perceptron* (MLP) com algumas camadas internas.

A saída $h(x)$ do *encoder* pode ser interpretada como um vetor que codifica uma sequência de texto x de modo que represente o contexto em que x ocorre. Em linhas gerais, o *encoder* é camada responsável por codificar o texto de entrada, capturando informações de seu contexto, estrutura sintática e do idioma. Quando apresentados em Vaswani *et al.*, 2017, os *Transformers* possuíam um total de $L = 6$ camadas idênticas.

Na camada de atenção *Multi-Head* dos *encoders*, as *queries*, *keys* e *values* são recuperadas da última camada de codificação. Ou seja, no *encoder* a atenção é computada entre elementos de um mesmo objeto, i.e., da mesma sentença de texto sendo codificada. Tal característica se denomina autoatenção.

O *decoder*

O *decoder*, representado à direita na Figura 2.1 tem como função a geração de textos a partir da saída das camadas de codificação, ao que se denonima decodificação. Dessa forma, a qualidade do texto gerado pelo *decoder* depende diretamente da qualidade da saída produzida pelo *encoder*.

Os *tokens* podem ser compreendidos como parcelas significativas de sentenças, como palavras, sub-palavras, morfemas e pontuações, e são o que formam a saída do *decoder*. Eles são gerados sequencialmente de modo que a cada instante o *token* produzido depende exclusivamente daqueles gerados nos instantes anteriores. Essa característica é chamada **autorregressão**.

Tal qual os *encoders*, os *decoders* são formados por camadas conectadas. Cada camada do *decoder* aplica a atenção duas vezes, sendo a primeira a autoatenção cujas entradas são *queries, keys* e *values* produzidas na camada anterior do próprio *decoder*.

A propriedade de **autorregressão** característica dos *Transformers* limita o cálculo da primeira autoatenção na camada de decodificação à exploração das relações entre um elemento da sequência de entrada com somente elementos de posições anteriores. Dessa forma, a autoatenção no *decoder* é realizada com *queries, keys* e *values* em que conexões de uma posição com posições posteriores são ocultadas e, por conseguinte, não aprendidas em tempo de treinamento. Dessa forma, os *decoders* são capazes, por exemplo, de gerar a continuação de sequências de texto utilizando apenas uma sequência de entrada.

A autoatenção no *decoder* é portanto denominada *Masked Multi-Head Attention* (Atenção *Multi-Head* Mascadada). Para garantir a ocultação de relações que violem a propriedade autoregressiva, posições em Q, K e V que possibilitariam a computação de relações proibidas — de uma sequência com outra posterior — são transformadas em $-\infty$ durante o treinamento (Vaswani *et al.*, 2017).

A segunda aplicação da atenção no *decoder* utiliza as *keys* e *values* geradas pelo *encoder* e as *queries* produzidas pela última camada de decodificação. Nessa etapa, não há a ocultação de nenhuma parte de K ou V . K resulta do *encoder* e, portanto, QK^T representa apenas relações entre a saída da última camada do *decoder* com a entrada do *encoder*. Sendo assim, nessa etapa a atenção explora apenas relações entre a saída do *encoder* e do *decoder* o que, em última instância, significa relacionar a entrada recebida pelo modelo e codificada pelo *encoder* com o texto que ele produziu como "resposta" até então.

Os benefícios da atenção

As camadas de autoatenção no *encoder* conectam todas as posições de uma sentença entre si. Para isso, um número constante de operações é realizado para cada par de posições (Vaswani *et al.*, 2017). É válido verificar que a multiplicação QK^T em 3 requer um número fixo de operações quaisquer que sejam as linhas de Q e K multiplicadas. Consequentemente, relações entre a primeira e a segunda palavras de um texto são codificadas e decodificadas usando o mesmo número de operações que o mesmo processo realizado entre a primeira e palavras mais distantes.

Na arquitetura de Redes Neurais Recorrentes (RNN - *Recurrent Neural Networks*) (Jurafsky e Martin, 2021), utilizada em modelos de linguagem como em Merity *et al.*, 2017, o processo de cálculo dessas relações é de complexidade $\mathcal{O}(n)$, em que n é distância no texto entre as palavras para as quais é calculada a relação ou dependência (Vaswani *et al.*, 2017). Essa complexidade revela dois aspectos característicos de RNNs, uma outra arquitetura para modelos de linguagem. O primeiro refere-se ao tempo linearmente crescente para o cálculo das dependências internas de um texto. O segundo diz respeito ao número de operações a que a informação contida em uma palavra é submetida até que a sua dependência com outra seja calculada.

Em linhas gerais, no caso das RNNs, quanto maior a distância entre as sentenças, menor é a qualidade do aprendizado da dependência entre as palavras envolvidas. Isso está relacionado a problemas como o *vanishing* e *exploding gradients* (Hanin, 2018). As RNNs são projetadas para capturar dependências temporais entre as posições de um texto, as quais podem ser muito distantes. À medida que uma RNN é treinada com uma sequência longa de palavras, o gradiente pode desaparecer (tornar-se extremamente pequeno) ou "explodir"(tornar-se extremamente grande) devido às multiplicações consecutivas características do treinamento de redes neurais com *backpropagation*. Na prática, isso significa a perda de informações contidas nas dependências entre palavras distantes e, portanto, resulta na piora no aprendizado que esses modelos adquirem sobre tais relações e que por sua vez são fundamentais para que atinjam um bom desempenho em tarefas de NLP.

Em suma, a complexidade $\mathcal{O}(1)$ do cálculo das dependências internas de um texto nos *Transformers* realizado pelas camadas de atenção sintetiza o benefício em utilizar tal arquitetura dos pontos de vista computacional e do desempenho em diferentes tarefas, sendo este relacionado à sua alta capacidade em aprender relações entre partes distantes de uma sentença. Dessa forma, optou-se pela utilização de modelos baseados na arquitetura *Transformer* para a realização da tarefa de NLI nos conjuntos de dados descritos na Seção 1.1.1. Os *Transformers* escolhidos são descritos nas seções seguintes deste capítulo.

É válido ressaltar que os modelos baseados em *Transformers* não são necessariamente formados pelo *encoder* e pelo *decoder*. Os *Masked Language Models* baseados nessa arquitetura descritos a seguir são modelos formados apenas pela camada de codificação característica dos *Transformers*. Portanto, eles aplicam do mecanismo de atenção bidirecional sem a autoregressividade explicada dos *decoders*.

2.1.2 *Masked Language Models* (Modelos de Linguagem Mascarada

MLMs (sigla para *Masked Language Models*, em inglês) são *encoders* em cujo treinamento uma porcentagem dos *tokens* de sua entrada é ocultada. Como explicado, *tokens* são representações de parcelas que compõem uma sentença de texto ("ser humano"pode ser considerado um *token* único, por exemplo). A ocultação ocorrida em MLMs é obtida pela substituição de alguns *tokens* por máscaras genéricas denotadas por $[MASK]$. Os *tokens* gerados como saídas do modelo correspondem àqueles substituídos por $[MASK]$ na entrada. Abaixo, define-se a função que caracteriza o aprendizado de um *Masked Laguage Model* de parâmetros θ (Wang e Cho, 2019).

Definição 5. ***Pseudo Log-Likelihood Learning***

Seja $X_{\setminus t} = (x_1, x_2, ..., x_{t-1}, x_{t+1}, ..., x_{|X|})$ e a função *pseudo log-likelihood, PLL* tais que:

$$PLL(\theta, D) = \frac{1}{D} \sum_{X \in D} \sum_{t=1}^{|X|} log(p(x_t | X_{\setminus t})) \tag{2.1}$$

$$\theta \text{ são os parâmetros do modelo} \tag{2.2}$$

$$D \text{ é um conjunto de dados de treinamento} \tag{2.3}$$

O aprendizado dos modelos de linguagem mascarada é baseado na maximização da função *PLL* acima. De maneira simplificada, a *PLL* representa a predictibilidade de cada *token* x_t presente em determinado conjunto de dados de treinamento. Dentre esses x_t há os *tokens* gerados pelo modelo para substituir as $[MASK]$ ocultantes de *tokens* aleatoriamente escolhidos no conjunto de treinamento e, portanto, na entrada. Em linhas gerais, ao se maximizar a função *PLL*, define-se quais parâmetros θ caracterizam um modelo que atribui às $[MASK]$ da entrada os *tokens* x_t com maior predictibilidade. Modelos capazes de gerar conjuntos de *tokens* de maior predictibilidade são capazes de substituir partes ocultadas de um texto de entrada por *tokens* que, em conjunto àqueles que estavam explícitos na entrada,

formam um texto de distribuição de probabilidade próxima à real, ou seja, verossímil.

A não inclusão de x_t entre as dimensões de $X_{\setminus t}$ é fundamental para caracterizar o aprendizado dessa tarefa e representa o efeito de mascarar a entrada antes de submetê-la ao modelo. Caso tal ocultação não fosse realizada, o contexto de cada *token* calculado pelo mecanismo de atenção (2) incluiria o próprio *token* previsto. Esse efeito descaracterizaria a capacidade do modelo em ser capaz de predizer que palavra ou sentença melhor compõe partes ocultas de um texto ou sentença dado apenas as posições anteriores e posteriores àquela que se deseja predizer, o que é necessário em tarefas de completar textos automaticamente, por exemplo.

É válido ressaltar que, em geral, *tokens* aleatoriamente selecionados para serem ocultados na entrada — determinado $x \in D$ — têm sua previsão possibilitada apenas se disponíveis explicitamente pelo modelo em outras partes da mesma entrada. Em outras palavras, isso significa que um *Masked Language Model* só é capaz de gerar os *tokens* (i.e. palavras) que tenha "visto"durante o treinamento. Por outro lado, há exceções: palavras compostas de um radical e um morfema podem ser geradas pelo modelo mesmo que jamais vistas durante o treinamento, por exemplo a palavra "patos"pode ser gerada mesmo que no conjunto de treinamento apenas o radical "pato"e o morfema "#s"para plural estivessem disponíveis. No entanto, essa exceção não se aplica a qualquer palavra e, portanto, para evitar que um mesmo *token* seja ocultado em todas as suas aparições na entrada e, por conseguinte, nunca seja gerado pelo modelo mesmo fornecido o contexto apropriado, aleatoriza-se a sua ocultação: se um *token* é selecionado para ser mascarado em determinada parte da entrada, sua ocultação de fato só ocorre com determinada probabilidade menor do que 1. Assim, minimiza-se a possibilidade de que um mesmo *token* seja ocultado em todas as suas ocorrências em D.

Os *Masked Language Models*, em contraste com modelos de linguagem unidirecionais, representam os contextos de textos utilizando a atenção bidirecional. Isso significa que, ao aprender a predizer $[MASK]$ em "Eu gosto de $[MASK]$ maçãs", MLMs representam o contexto utilizando "Eu gosto de"e "maçãs"simultaneamente. O atenção unidirecional por sua vez, representaria o contexto apenas com "Eu gosto de"(*left-to-right attention*) ou somente "maçãs"(*right-to-left attention*). É a atenção bidirecional que permite que MLMs tenham bons resultados em tarefas em que a compreensão do contexto bidirecional é crucial.

A seguir são apresentados os dois MLMs utilizados no presente trabalho: o XLM-RoBERTa e o BERTimbau.

2.1.3 XLM-RoBERTa

XLM-RoBERTa é o nome do *Transformer* multilingual criado pelo *Facebook AI* (atual *AI by Meta*) introduzido em Conneau, Khandelwal *et al.*, 2020. Trata-se de um modelo de linguagem mascarada pré-treinado em 100 idiomas, caracterizando-o como um modelo multilingual. O XLM-RoBERTa obtem resultados expressivos em tarefas de NLP como Reconhecimento de Entidades Nomeadas (*Named Entity Recognition*— NER), Perguntas e Respostas (*Question Answering*— QA) e tarefas de classificação de textos, sendo que dentre estas, em particular, inclui-se o NLI. Os dados de treinamento (*corpus*) do XLM-RoBERTa foram extraídos com filtragens da *Common Crawl*, uma organização sem fins lucrativos que fornece arquivos e dados extraídos da internet de maneira pública.

Outros *Transformers* multilinguais como o mBert (Devlin *et al.*, 2019) e o XLM (Lample e Conneau, 2019) demonstravam bons resultados em diferentes tarefas de NLP à época da divulgação do XLM-RoBERTa. Aqueles modelos foram treinados com textos de diferentes línguas extraídos do *Wikipedia* — os quais são pouco diversos para línguas de baixos recursos, i.e., com poucos dados de treinamento disponíveis. A maior diversidade linguística é uma das razões apontadas no artigo de introdução do XLM-RoBERTa que explicam a sua superioridade em resultados de tarefas nesses idiomas (Lample e Conneau, 2019). A denominada "maldição do multilinguismo"(*curse of multilinguality*) é uma das explicações fornecidadas pelo artigo de introdução do modelo para o resultado inferior de outros modelos em idiomas com poucos conjuntos de dados disponíveis, o que é parcialmente superado pelo XLM-RoBERTa. Já que a língua portuguesa é considerada uma língua de baixos recursos principalmente se comparada ao inglês, isso justifica parte da escolha de seu uso no presente trabalho.

Diluição de Capacidade e a Maldição do Multilinguismo

O tamanho e a capacidade de um modelo *Transformer* são definidas pelo seu número de parâmetros. Essa quantidade divide-se entre as dimensões que formam as camadas da rede neural voltadas à representação da linguagem em um espaço vetorial comum — denominadas camadas de *embeddings* — e aquelas do *Transformer* propriamente dito. O número de parâmetros possui relação direta com a *performance* que o modelo tem em tarefas de NLP, embora não seja o único fator que a influencie (Conneau, Khandelwal *et al.*, 2020). O tamanho do vocabulário (quantidade de termos únicos no conjunto de dados de treinamento) é outro fator importante nessa relação.

As línguas de poucos recursos são aquelas que possuem pouca quantidade relativa de amostras durante o treinamento em diferentes tarefas quando comparadas ao inglês. Os resultados dos *Transformers* em tais idiomas tendem a ser piores do que aqueles em

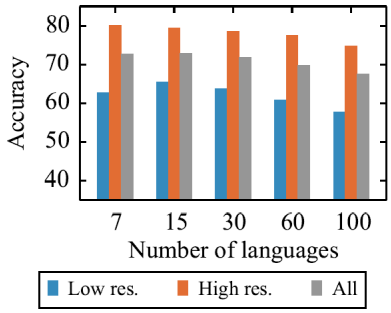
línguas de muitos recursos. Conneau, Khandelwal *et al.*, 2020 expõe a importância de se aumentar o tamanho do modelo quando se aumenta o número de línguas utilizadas durante seu treinamento.

Por outro lado, fixado o total de parâmetros, Conneau, Khandelwal *et al.*, 2020 discute a existência de um regime em que o aumento de idiomas de muitos recursos similares aos de baixo implica na melhoria da *performance* dos *Transformers* em idiomas de baixos recursos, possivelmente porque idiomas de baixos recursos se beneficiam do aprendizado de outras línguas similares presentes durante o treinamento. Fora desse regime, o aumento de idiomas causa prejuízo à *performance* geral em todas as línguas, efeito que se denomina diluição de capacidade: o aumento da quantidade de dados a serem representados no treinamento inerente a inclusão de mais línguas leva à necessidade do aumento do tamanho das camadas de *embeddings* ao custo dos parâmetros voltados ao *Transformer* propriamente dito. Por conseguinte, aumentar a quantidade de idiomas em tempo de treinamento degrada a *performance* geral do modelo se seu total de parâmetros não for apropriado.

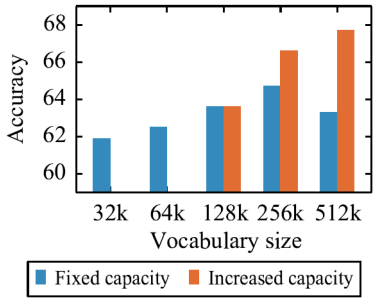
Há, portanto, uma relação de *trade-off* entre os benefícios trazidos pelo multilinguismo e a diluição de capacidade caracterizada pela redução das dimensões das camadas do *Transformer*. Essa relação é observada quando, fixado o número de parâmetros de um modelo, o aumento da variabilidade de idiomas (aumento da camada de *embeddings*) implica na melhoria de seus resultados em línguas de baixos recursos até certo ponto a partir do qual os resultados em línguas de baixos recursos se degradam, caracterizando um regime em que as camadas voltadas ao *Transformer* não são grandes o suficiente para a melhoria ou até manutenção de seus resultados.

Abaixo, a figura 2.2a relaciona o número de idiomas (*Number of languages*) no treinamento do XLM treinado em 100 idiomas (XLM-100) (Lample e Conneau, 2019) com a acurácia que ele obtem no XNLI (Conneau, Lample *et al.*, 2018) — uma tarefa de classificação de texto similar ao NLI.

Conneau, Khandelwal *et al.*, 2020 refere-se ao que é observado na figura 2.2a como "maldição do multilinguismo"e pode ser minimizado pelo aumento do tamanho do modelo e/ou pelo aumento do vocabulário — o que é observado na figura abaixo.



(a) *Línguas de baixo recurso (*low res.*) se beneficiam do treinamento em múltiplos idiomas até que a diluição de capacidade passa a degradar os resultados gerais. Imagem extraída de* Conneau, Khandelwal *et al., 2020*



(a) *Relação entre o tamanho do vocabulário e a acurácia do XLM-100 no XNLI. Imagem extraída de* Conneau, Khandelwal *et al., 2020*

Embora as imagens acima sejam referentes ao XLM (um *Transformer* treinado com um *corpus* extraído do Wikipedia) e não ao XLM-RoBERTa, Conneau, Khandelwal *et al.*, 2020 afirma que o mesmo efeito é observado no modelo utilizado neste trabalho, sendo o comportamento observado característico de modelos multilinguais e não restrito àquele a que os gráficos se referem. O que se visualiza na figura 2.3a é similar àquilo da figura 2.2a (diluição de capacidade): o aumento do vocabulário melhora a *performance* do *Transformer* até certo ponto a partir do qual o efeito contrário é observado, o que é superado pelo aumento da capacidade (*increased capacity*) do modelo.

A figura 2.3a sugere que modelos multilinguais beneficiam-se da alocação de maiores proporções do total de parâmetros às camadas de representação — mesmo que isso reduza o tamanho do *Transformer*. A *performance* do XLM-100 aumenta até o vocabulário de tamanho $256K$, a partir do qual a diluição de capacidade se impõe e os resultados passam a ser piores. Isso é superado, então, pelo aumento do número total de parâmetros.

Modelos Multilinguais *vs* Monolinguais

Conneau, Khandelwal *et al.*, 2020 evidencia que uma das críticas a modelos multilinguais é sua *performance* inferior em relação aos seus pares monolinguais. Essas críticas afirmam que um modelo treinado exclusivamente em uma língua X tende a obter resultados superiores que aqueles multilinguais treinados em um conjunto de idiomas dentre os quais está X.

Para isolar a capacidade de modelos multilinguais em representar idiomas de baixos recuros, Conneau, Khandelwal *et al.*, 2020 compara a *performance* de BERTs monolinguais em realizar a tarefa de XNLI com aquela do XLM-7 (treinado em 7 línguas). O que se observa é que o XLM-7 obtem, em média, acurácia 2.5% superior àquela do BERT. Esse resultado demonstra que é possível construir modelos multilinguais que representem línguas de baixos recuros com melhor qualidade que suas contrapartidas monolinguais.

Embora a Língua Portuguesa não seja uma língua de recursos extremamente baixos, a verificação de tal afirmação para tarefas no idioma compôs a motivação do presente trabalho.

Por que se decidiu fazer o *fine-tuning* do XLM-RoBERTa para o NLI?

O XLM-RoBERTa explorou o efeito de diluição de capacidade, fixando o tamanho de vocabulário em $250K$ para as duas versões do modelo disponibilizadas no *Hugging Face*: 270 milhões de parâmetros no XLM-RoBERTa *base* (versão utilizada neste trabalho) e 550 no *large*. No XNLI, o XLM-RoBERTa *base* superou o *Multilingual BERT* (mBERT) e o XLM-100 em todas as línguas de baixos recursos com exceção do Swahili em que foi superado por ambos. Por outro lado, o XLM-RoBERTa *large* os superou em todas as línguas na mesma tarefa. Ambos mBERT e XLM-100 foram treinados em textos extraídos do Wikipedia em 100 línguas distintas, dentre elas a portuguesa, possuindo $172M$ e $570M$ de parâmetros, respectivamente.

Os resultados superiores do XLM-RoBERTa na tarefa de XNLI quando comparados àqueles do XLM-100 e do mBERT, além da melhor *performance* de modelos multilinguais na mesma tarefa obtida por Conneau, Khandelwal *et al.*, 2020 e explicada anteriormente justificaram o uso do XLM-RoBERTa no presente trabalho — o modelo mostrou-se contrapartida multilingual promissora ao BERTimbau, um MLM monolingual treinado na Língua Portuguesa.

2.1.4 BERTimbau

BERTimbau é o nome do conjunto de *Transformers* baseados no BERT (Devlin *et al.*, 2019) introduzido em 2020 em Souza *et al.*, 2020. Seu treinamento foi baseado em duas tarefas: Modelo de Linguagem Mascarada e predição de próxima palavra (*Next Sentence Prediction - NSP*). Todos os seus dados de pré treinamento foram extraídos do brWac, um *corpus* com $2.68B$ de *tokens* de $3.53M$ de páginas da *web*. Além de ser o maior *corpus* público em Língua Portuguesa, a metodologia utilizada em sua organização garantiu diversidade e qualidade de sua linguagem do ponto de vista dos domínios que cobre, o que é importante para o pré-treinamento de modelos baseados no BERT (Souza *et al.*, 2020).

Uma das abordagens utilizadas na avaliação deste modelo em seu artigo de apresentação é a comparação de seus resultados em diferentes tarefas de NLP com aqueles do mBERT. Essas comparações confirmariam as vantagens que modelos pré-treinados em português teriam sobre os multilinguais — como é o mBERT. Essa visão compete com aquela exposta em Conneau, Khandelwal *et al.*, 2020 e explicada anteriormente, em que resultados superiores com modelos multilinguais (XLM-100 e XLM-RoBERTa) são observados especialmente para línguas de baixo recurso quando comparados aos resultados obtidos por modelos monolinguais equivalentes.

BERT

O BERT (*Bidirectional Encoder Representation from Transformers*) é um *Transformer* formado apenas pela camada de codificação. Recuperando o *encoder* na Definição 4, a entrada do BERT pode ser definida: i. por um conjunto de *tokens* contidos em uma sentença acrescidos de dois *tokens* especiais, o [CLS] e o [SEP] (Souza *et al.*, 2020); ii. por dois conjuntos de *tokens* contidos em sentenças distintas separados por [SEP], sendo apenas o primeiro acrescido de [CLS]. Assim, sejam as sequência de *tokens* $(x_1, ..., x_N)$ e $(y_1, ..., y_M)$, o BERT(função h em 4) receberá como entradas:

$$[CLS]x_1...x_N[SEP] \text{ quando i,} \tag{2.4}$$

$$[CLS]x_1...x_N[SEP]y_1...y_M[SEP] \text{ quando ii} \tag{2.5}$$

Onde x_i e y_i são *tokens* de um mesmo vocabulário e N e M são os números de *tokens* nas sentenças x e y, respectivamente (Souza *et al.*, 2020).

Finalmente, é possível definir a camada de *encoder* do BERT recuperando a Definição 4:

Definição 6. BERT *Encoder*

Seja $x = (x_1, ..., x_N)$ um vetor de *tokens*, a função $h_{BERT} : \mathbb{R}^{(N+2)\times H} \rightarrow \mathbb{R}^{(N+1)\times H}$ de codificação do BERT é tal que:

$$h_{BERT}([CLS], x_1, ..., x_N, [SEP]) = (c, T_1, ..., T_N)$$

Na definição acima, $T_i \in \mathbb{R}^H$ é a representação codificada do *token* x_i na sequência x e $c \in \mathbb{R}^H$ é a representação para o [CLS]. No pré-treinamento do BERT, as saídas c e T_i são aprendidas com as tarefas de MLM e NSP simultaneamente: as saídas c das camadas de autoatenção são entradas para a rede *feedforward* responsável pela tarefa de NSP, enquanto os vetores T_i alimentam aquela que realiza o MLM. Para que o aprendizado ocorra simultaneamente, a soma das funções de perda *cross-entropy* de cada tarefa é minimizada (Souza *et al.*, 2020). Dessa forma, a arquitetura do BERT pode ser compreendida como um conjunto de camadas de autoatenção (h_{BERT}) cujas saídas servem de entrada para redes que realizam as tarefas de NSP e NLI, denominadas *heads* de classificação, o que permite o aprendizado.

Para aplicar o BERT em alguma tarefa específica, c e/ou T_i podem: i. ser utilizados diretamente como entradas de modelos especializados em tais tarefas; ii. ser modificados com a técnica de *fine-tuning* que altera os parâmetros da rede neural de modo que sua saída corresponda à tarefa em questão.

Em linhas gerais, o *token* especial [CLS] pode ser compreendido como aquele que codifica em uma única representação/saída toda a sequência de entrada, sendo útil para tarefas que lidam com textos a nível de sequência e não de *tokens* específicos (Souza *et al.*, 2020)— como é o caso do NLI.

***Performance* do BERTimbau em Reconhecer Implicação Textual/NLI**

Para avaliar os resultados do BERTimbau em diferentes tarefas denonimadas *downstream tasks*, as *heads* de classificação utilizadas no pré-treinamento descritas anteriormente são desacopladas das camadas de autoatenção e substituídas por redes (*head*) que serão treinadas nas tarefas específicas.

O *fine-tuning* do BERTimbau para a tarefa de Reconhecimento de Implicação Textual (*Recognizing Textual Entailment* - RTE) é um tipo de NLI em que um par (premissa, hipótese) é classificado em $ENTAILMENT$ ou $NONE$. Como descrito em Souza *et al.*, 2020, o RTE foi feito conjuntamente à tarefa de STS. Para isso, duas camadas lineares independentes

foram acopladas às camadas de autoatenção do BERTimbau — uma voltada para o RTE e outra para o STS. Camadas lineares são redes neurais que realizam uma única transformação linear nos vetores de entrada para produzir o vetor de saída. Para que um único treinamento fosse realizado nesse esquema multitarefa, as funções de perda de ambas camadas lineares foram somados com iguais pesos, o que significa que o *fine-tuning* priorizou o RTE e o STS igualmente (Souza *et al.*, 2020).

O ASSIN2 foi utilizado como a base de dados de RTE para o *fine-tuning* do BERTimbau no esquema descrito acima — um dos conjuntos de dados utilizado neste trabalho. Abaixo encontram-se os resultados expostos no artigo do modelo.

Row	Model	STS		RTE	
		Pearson (⋆)	MSE	F1 (⋆)	Accuracy
1	mBERT + RoBERTa-Large-en (Averaging) [40] †	0.83	0.91	84	84.8
2	mBERT + RoBERTa-Large-en (Stacking) [40] †	0.785	0.59	88.3	88.3
3	mBERT (STS) and mBERT-PT (RTE) [39] ‡	0.826	0.52	87.6	87.6
4	USE+Features (STS) and mBERT+Features (RTE) [13]	0.800	**0.39**	86.6	86.6
5	mBERT+Features [13]	0.817	0.47	86.6	86.6
6	mBERT (ours)	0.809 (0.004)	0.58 (0.04)	86.8 (0.4)	86.8 (0.4)
7	BERTimbau Base	0.836 (0.007)	0.58 (0.03)	89.2 (0.8)	89.2 (0.8)
8	BERTtimbau Large	**0.852 (0.003)**	0.50 (0.03)	**90.0 (0.4)**	**90.0 (0.4)**

(a) *Pontuações do BERTimbau nas tarefas STS e RTE no conjunto de dados ASSIN2 e comparações com o mBERT. Melhores pontuações em negrito. Os valores relatados são a média de múltiplas execuções (desvio padrão entre parênteses) com diferentes sementes aleatórias. A estrela denota as métricas principais, '†' denota a técnica de* ensemble *e '‡' versões do modelo que tiveram mais dados de treinamento do que as versões 'puras'. Tabela extraída de Souza et al., 2020*

A técnica de *ensemble* vista na tabela acima combina a saída de dois ou mais modelos para uma mesma entrada para formar uma única saída final a fim de minimizar erros. Há mais detalhes sobre como essa técnica pode ser aplicada em Ganaie *et al.*, 2022.

Como se observa acima, o BERTimbau —tanto em sua versão *base* (110M de parâmetros) quanto em sua versão *large* (330M de parâmetros) — obteve resultados superiores em RTE e STS àqueles do mBERT.

Por que se decidiu usar o BERTimbau na tarefa de NLI?

O fato de que esse modelo foi bem avaliado para o NLI na Língua Portuguesa é um indicativo de sua boa *performance* em realizar tal tarefa, o que foi parte da motivação da escolha do BERTimbau como um dos modelos submetidos ao *fine-tuning* no presente trabalho.

Além disso, a aparente contradição existente ao que se afirma em Conneau, Khan-delwal *et al.*, 2020 e no artigo de apresentação do BERTimbau em relação à superioridade (ou não) de modelos multilinguais em relação a seus pares monolinguais evidencia a neces-

sidade de maiores análises sobre tal afirmação com atenção especial à Língua Portuguesa. Propôs-se, no lugar de confirmá-la, verificar sob quais condições ela pode ser válida ou não. Em especial, em Souza *et al.*, 2020 o BERTimbau — um modelo monolingual — de fato supera o mBERT (um modelo multilingual) no NLI realizado no ASSIN2, embora isso tenha sido alcançado em um experimento em que tal tarefa foi realizada simultaneamente ao STS.

Realizar os testes com o BERTimbau na tarefa de NLI isoladamente (sem o STS ou qualquer outra tarefa simultânea) trouxe resultados mais explicativos sobre a possível superioridade (ou não) de modelos multilinguais nesse tipo específico de tarefa na Língua Portuguesa.

Capítulo 3

Resultados e Discussões

Este capítulo é dividido da seguinte forma: na Seção 3.1, apresentam-se os resultados do *fine tuning* dos modelos e de sua avaliação; na Seção 3.2 os resultados das avaliações direta e cruzada são discutidos de modo a comparar os *datasets* utilizados.

3.1 Resultados

Abaixo, apresentam-se os resultados obtidos no conjunto de teste (*test set*) do *fine tuning* dos modelos estudados. Tais resultados correspondem à *performance* obtida pelo modelo submetido ao *fine tuning* completo no conjunto de treinamento correspondente a cada um dos conjuntos de teste apresentados

	BERTimbau		XLM-RoBERTa	
Test set	Accuracy	Loss	Accuracy	Loss
$ASSIN_{test}$	0.91	0.25	0.87	0.38
$ASSIN2_{test}$	0.96	0.13	0.95	0.17
$PLUE/MNLI_{validationMatched}$	0.84	0.43	0.82	0.51

Fonte: O autor.

Tabela 3.1: *Resultados finais do Fine Tuning*

A avaliação dos modelos submetidos ao *fine-tuning* foi dividida em duas subcategorias: i. avaliação na mesma distribuição dos dados de *fine tuning*, denominada avaliação direta e correspondente aos conjuntos de células vazias na Tabela 1.2; ii. avaliação em distribuições distintas daquela dos dados de *fine tuning*, denonimada avaliação cruzada. Na Tabela 3.2 quando o *Conjunto de fine tuning* é igual ao *Conjunto de testes* os resultados correspondem a uma avaliação direta e, do contrário, correspondem a uma avaliação cruzada.

Model	Accuracy	F1 Score	Precision	Recall
Conjunto de fine tuning: $ASSIN_{train}$ | Conjunto de Testes: $ASSIN_{test}$				
$XLM-RoBERTa_{base}$	0.898	0.895	0.894	0.898
$BERTimbau_{large}$	0.918	**0.918**	0.919	0.918
Conjunto de fine tuning: $ASSIN_{train}$ | Conjunto de Testes: $ASSIN2_{test}$				
$XLM-RoBERTa_{base}$	0.706	0.698	0.731	0.706
$BERTimbau_{large}$	0.730	**0.720**	0.769	0.730
Conjunto de fine tuning: $ASSIN_{train}$ | Conjunto de Testes: $PLUE-MNLI_{validationMatched}$				
$XLM-RoBERTa_{base}$	0.463	0.372	0.341	0.463
$BERTimbau_{large}$	0.488	**0.395**	0.354	0.488
Conjunto de fine tuning: $ASSIN2_{train}$ | Conjunto de Testes: $ASSIN2_{test}$				
$XLM-RoBERTa_{base}$	0.864	0.864	0.868	0.864
$BERTimbau_{large}$	0.893	**0.893**	0.898	0.893
Conjunto de fine tuning: $ASSIN2_{train}$ | Conjunto de Testes: $ASSIN_{test}$				
$XLM-RoBERTa_{base}$	0.779	**0.779**	0.799	0.779
$BERTimbau_{large}$	0.776	0.734	0.782	0.776
Conjunto de fine tuning: $ASSIN2_{train}$ | Conjunto de Testes: $PLUE-MNLI_{validationMatched}$				
$XLM-RoBERTa_{base}$	0.709	**0.668**	0.710	0.709
$BERTimbau_{large}$	0.677	0.582	0.735	0.677
Conjunto de fine tuning: $PLUE/MNLI_{train}$ | Conjunto de Testes: $PLUE-MNLI_{validationMatched}$				
$XLM-RoBERTa_{base}$	0.823	0.823	0.825	0.823
$BERTimbau_{large}$	0.836	**0.834**	0.836	0.836
Conjunto de fine tuning: $PLUE/MNLI_{train}$ | Conjunto de Testes: $ASSIN_{test}$				
$XLM-RoBERTa_{base}$	0.772	**0.683**	0.613	0.772
$BERTimbau_{large}$	0.723	0.668	0.630	0.728
Conjunto de fine tuning: $PLUE/MNLI_{train}$ | Conjunto de Testes: $ASSIN2_{test}$				
$XLM-RoBERTa_{base}$	0.862	0.862	0.862	0.862
$BERTimbau_{large}$	0.871	**0.870**	0.883	0.871

Fonte: O autor.

Tabela 3.2: *Resultados das Avaliações Direta e Cruzada*

Model	Entailment (E)	Paraphrase (P)	None ($\emptyset$)	Contradiction (C)	Neutral (N)
Conjunto de fine tuning: $ASSIN_{train}$ | Conjunto de Testes: $ASSIN_{test}$					
$XLM-RoBERTa_{base}$	0.774	0.631	0.950	-	-
$BERTimbau_{large}$	0.826	0.721	0.959	-	-
Conjunto de fine tuning: $ASSIN_{train}$ | Conjunto de Testes: $ASSIN2_{test}$					
$XLM-RoBERTa_{base}$	0.747	-	0.649	-	-
$BERTimbau_{large}$	0.774	-	0.667	-	-
Conjunto de fine tuning: $ASSIN_{train}$ | Conjunto de Testes: $PLUE-MNLI_{validationMatched}$					
$XLM-RoBERTa_{base}$	0.781	-	-	-	0.472
$BERTimbau_{large}$	0.789	-	-	-	0.500
Conjunto de fine tuning: $ASSIN2_{train}$ | Conjunto de Testes: $ASSIN2_{test}$					
$XLM-RoBERTa_{base}$	0.871	-	0.857	-	-
$BERTimbau_{large}$	0.899	-	0.887	-	-
Conjunto de fine tuning: $ASSIN2_{train}$ | Conjunto de Testes: $ASSIN_{test}$					
$XLM-RoBERTa_{base}$	0.488	0.384	0.892	-	-
$BERTimbau_{large}$	0.300	0.312	0.889	-	-
Conjunto de fine tuning: $ASSIN2_{train}$ | Conjunto de Testes: $PLUE-MNLI_{validationMatched}$					
$XLM-RoBERTa_{base}$	0.806	-	-	-	0.417
$BERTimbau_{large}$	0.798	-	-	-	0.188
Conjunto de fine tuning: $PLUE/MNLI_{train}$ | Conjunto de Testes: $PLUE-MNLI_{validationMatched}$					
$XLM-RoBERTa_{base}$	0.839	-	-	0.838	0.790
$BERTimbau_{large}$	0.856	-	-	0.854	0.796
Conjunto de fine tuning: $PLUE/MNLI_{train}$ | Conjunto de Testes: $ASSIN_{test}$					
$XLM-RoBERTa_{base}$	0.770	0.709	0.942	-	-
$BERTimbau_{large}$	0.579	0.436	0.879	-	-
Conjunto de fine tuning: $PLUE/MNLI_{train}$ | Conjunto de Testes: $ASSIN2_{test}$					
$XLM-RoBERTa_{base}$	0.863	-	0.860	-	-
$BERTimbau_{large}$	0.881	-	0.859	-	-

Fonte: O autor.

Tabela 3.3: *F1 Score por Classe das Avaliações Direta e Cruzada*

3.2 Discussões

Como é comum na literatura, utilizou-se o *F1 Score* (Definida como *F-Measure* em Hossin e Sulaiman, 2015) como critério de seleção de modelos com melhores resultados tanto na Tabela 3.2 quanto na Tabela 3.3. Assim, para cada par *Conjunto de fine tuning* e *Conjunto de Testes* foi considerado como melhor modelo aquele com maior *F1 Score*. É válido ressaltar que todas as discussões em relação aos testes cruzados dependem diretamente do sistema de tradução desenvolvido no presente trabalho.

Analisando-se a Tabela 3.2, nota-se que o BERTimbau obtem os melhores resultados em todas as avaliações diretas realizadas. Sendo assim, o XLM-RoBERTa demonstrou-se pior em realizar a tarefa de NLI em conjuntos de testes cujos domínios linguísticos são iguais àqueles do conjunto utilizado para o *fine tuning*. Por outro lado, das 6 avaliações cruzadas realizadas de acordo com a Tabela 1.2, o XLM-RoBERTa obteve maior $F1Score$ que o BERTimbau na metade delas.

O fato de que o XLM-RoBERTa (um modelo multilingual) foi capaz de superar os resultados do BERTimbau em alguns dos testes cruzados, sendo o último um modelo maior e monolingual, é uma evidência de que em relação à capacidade de generalização, o monolinguismo e maior quantidade de parâmetros não necessariamente caracterizam modelos com os melhores resultados em relação ao NLI na Língua Portuguesa. Em particular, na Tabela 3.2 nota-se que o XLM-RoBERTa foi capaz de reconhecer as paráfrases do ASSIN (*paraphrase*) com melhor *F1 score* que o BERTimbau nos testes cruzados. Ou seja, o XLM-RoBERTa superou a capacidade de reconhecer paráfrases indiretamente que aquela do BERTimbau.

Nos testes cruzados para *Conjunto de fine tuning:* $PLUE/MNLI_{train}$ e *Conjunto de Testes:* $ASSIN_{test}$, o XLM-RoBERTa obteve $F1Score$ de 0.709 para o reconhecimento de paráfrase, enquanto o BERTimbau obteve 0.436 — 38.5% a menos (Tabela 3.3). Nessa avaliação, o reconhecimento dessa classe passou pelo seguinte teste: um par de premissa (p) e hipótese (h) foi classificado como paráfrase se $E(p,h) \wedge E(h,p)$, já que o $PLUE/MNLI_{train}$ não contem dados classificados diretamente nessa classe. Sendo assim, o reconhecimento de tal classe passa pela classificação de paráfrase como sendo um acarretamento lógico em ambos os sentidos — o que é feito pela condição citada. O fato de que, para esses conjuntos de *fine tuning* e de testes, o XLM-RoBERTa superou os resultados do BERTimbau com folga indica que sua compreensão dos acarretamentos lógicos presentes no $ASSIN_{test}$ foi superior. Mais especificamente, esse resultado indica que o XLM-RoBERTa obteve maior sucesso na transferência do reconhecimento dos *entailment* presentes no domínio linguístico do $PLUE/MNLI_{train}$ para aqueles do $ASSIN_{test}$ que o BERTimbau. A identificação da paráfrase

em um par premissa (p) e hipótese (h) através das atribuições de $E(p, h)$ **e** $E(h, p)$ denota melhor qualidade na compreensão semântica do *entailment* — se um modelo aprendeu o sentido do acarretamento lógico com qualidade, deve ser capaz de reconhecê-lo em *ambos os sentidos* em um par classificado como paráfrase. De fato, na Tabela 3.3 nota-se que o XLM-RoBERTa ajustado no PLUE/MNLI obteve $F1Score$ de 0.770 no reconhecimento de *entailment*, enquanto o BERTimbau obteve somente 0.579.

Quando todas as classes são levadas em consideração na seleção dos modelos com os melhores resultados, os dados da Tabela 3.2 revelam os seguintes *F1 Score* médios tomados sobre os resultados das avaliações direta e cruzada realizadas com os XLM-RoBERTA e BERTimbau, respectivamente: para o *fine tuning* no $ASSIN - train$, 0.655 e 0.677; no $ASSIN2_{train}$, 0.770 e 0.736; no $PLUE/MNLI_{train}$, 0.789 e 0.790. Tais resultados demonstram que quando levadas em consideração todas as classes, os melhores resultados médios são obtidos, em ambos os modelos, naqueles submetidos ao *fine tuning* no $PLUE/MNLI_{train}$, embora tal *dataset* tenha sido construído por meio da tradução automática do MNLI para o idioma português. Ou seja, embora os processos de anotação do ASSIN e ASSIN2 tenham sido feitos com a supervisão e validação de especialistas humanos, tais *datasets* se mostraram piores, em geral, que o PLUE/MNLI para o *fine tuning* do XLM-RoBERTa e do BERTimbau. Dão-se duas hipóteses não conflitantes para tal resultado: a primeira é relacionada aos tamanhos do ASSIN, ASSIN2 e PLUE/MNLI, possuindo $5k$, $6.5k$ e $393k$ pares de premissa e hipótese para treinamento, respectivamente. Ou seja, o PLUE/MNLI possui muito mais dados para treinamento e isso favorece a sua *performance* em conjuntos de testes; a segunda relaciona-se à diversidade linguística do PLUE/MNLI quando comparada aos outros *datasets*, o que pode ter beneficiado modelos submetidos ao *fine tuning* nele em detrimento do ASSIN e ASSIN2 que possuem domínios mais restritos.

Analisando-se os conjuntos de testes isoladamente, em todos os testes realizados, os modelos com as melhores *performances* em cada um deles foram aqueles cujo *fine tuning* foi feito no conjunto de treinamento equivalente. Ou seja, o modelo com os melhores resultados em um *dataset* $D_{0_{test}}$ foi aquele submetido ao *fine tuning* no $D_{0_{train}}$, o que é coerente — modelos adaptados do domínio linguístico de $D_{0_{test}}$ tendem a obter melhores resultados nele se comparados àqueles ajustados em um domínio distinto D_1.

No entanto, houve um resultado em que os modelos testados em um *dataset* $D_{0_{test}}$ tiveram resultados semelhantes entre aqueles ajustados em $D_{0_{train}}$ (avaliação direta) e aqueles ajustados em D_1 (avaliação cruzada). Para o ajuste no $PLUE/MNLI_{train}$, tanto o XLM-RoBERTa quanto o BERTimbau, quando testados no $ASSIN2_{test}$ (avaliação cruzada), obtiveram resultados semelhantes aos resultados de suas versões ajustadas no $ASSIN2_{train}$ (avaliação direta). Na avaliação direta, o XLM-RoBERTA ajustado obteve *F1 Score* de 0.864,

e na cruzada esse resultado foi de 0.862 — uma diferença de apenas 0.1%. Já o BERTimbau obteve *F1Score* de 0.893 e 0.870 nas avaliações direta e cruzada, respectivamente, o que corresponde a uma diferença de apenas 2.3%. Tais resultados indicam que o PLUE/MNLI cobre parte significativa do domínio linguístico do ASSIN2 com boa aproximação.

Por outro lado, o XLM-RoBERTa ajustado no $ASSIN2_{train}$ alcançou *F1 Score* de 0.668 na avaliação cruzada com o conjunto de testes $PLUE/MNLI_{validationMatched}$, enquanto o mesmo modelo ajustado no $PLUE/MNLI_{train}$ obteve 0.823 no mesmo conjunto, expondo uma diferença de 15.5%. O BERTimbau ajustado no $ASSIN2_{train}$, por sua vez, obteve *F1 Score* de 0.582 na mesma avaliação cruzada, e aquele ajustado no $PLUE/MNLI_{validationMatched}$ obteve 0.834 na avaliação direta, o que equivale a uma diferença de 25.2%. Dessa forma, é possível concluir que embora o PLUE/MNLI cubra satisfatoriamente o domínio linguístico característico do ASSIN2, o contrário não é verdadeiro. Ou seja, os testes realizados demonstraram que os resultados do XLM-RoBERTa e BERTimbau avaliados no ASSIN2 são próximos entre suas versões submetidas ao *fine tuning* no próprio ASSIN2 e no PLUE/MNI. Em contraste, quando eles são avaliados no $PLUE/MNLI_{validationMatched}$, os modelos ajustados no ASSIN2 não conseguem resultados tão satisfatórios quanto daqueles ajustados no PLUE/MNLI.

Quando são feitas comparações entre o ASSIN e PLUE/MNLI, os modelos se compartam de maneira distinta à anterior. O XLM-RoBERTa ajustado no $PLUE/MNLI_{train}$ obteve *F1 Score* de 0.683 no $ASSIN_{test}$, enquanto aquele ajustado no $ASSIN_{train}$ obteve 0.895 no mesmo conjunto de testes — uma diferença de 34%. O BERTimbau ajustado no $PLUE/MNLI_{train}$ obteve *F1 Score* de 0.668 quando avaliado no $ASSIN_{test}$, enquanto o ajustado no $ASSIN_{train}$ chegou a 0.918 — uma diferença de 27.2%. Ou seja, há evidência de que o PLUE/MNLI não inclui o domínio linguístico do ASSIN. Dado o tamanho superior do PLUE/MNLI em relação ao ASSIN, espera-se que o contrário também não seja verdadeiro para nenhum dos modelos estudados e, de fato, as evidências confirmaram tais expectativas. O XLM-RoBERTa ajustado no $ASSIN_{train}$ obteve *F1 Score* de 0.372 no $PLUE/MNLI_{validationMatched}$, enquanto o ajustado no $PLUE/MNLI_{train}$ obteve 0.823 na mesma base de testes — uma diferença de 54.8%. Já o BERTimbau ajustado no $ASSIN_{train}$ obteve 0.395 e 0.834 para os ajustes no $ASSIN_{train}$ e $PLUE/MNLI_{train}$, respectivamente — uma diferença de 52.6%.

Finalmente, quando essas comparações são feitas entre os modelos ajustados no $ASSIN_{train}$ e $ASSIN2_{train}$, o que se observou é que nem o ASSIN cobre o domínio linguístico do ASSIN2 e nem o inverso. O XLM-RoBERTa ajustado no $ASSIN_{train}$ obteve *F1 Score* de 0.698 nos testes no $ASSIN2_{test}$, enquanto o ajustado no $ASSIN2_{train}$ obteve 0.0.864 no mesmo teste — uma diferença de 19.2%. O BERTimbau ajustado no $ASSIN_{train}$, por sua vez, obteve *F1 Score* de 0.720 no $ASSIN2_{test}$ e aquele submetido ao *fine tuning* no $ASSIN2_{train}$

obteve 0.893 — uma diferença de 19.4%. Quando são analisados os testes no $ASSIN_{test}$ para verificar a cobertura do ASSIN2 em relação ao ASSIN, grandes diferenças também foram observadas. O XLM-RoBERTA ajustado no $ASSIN_{train}$ obteve 0.895 de *F1 Score* quando testado no $ASSIN_{test}$, enquanto aquele ajustado no $ASSIN2_{train}$ obteve apenas 0.779 — uma diferença de 13%. O BERTimbau ajustado no $ASSIN_{train}$ obteve 0.918, enquanto aquele ajustado no $ASSIN2_{train}$, apenas 0.734 no mesmo teste — uma diferença de 20%.

3.3 Conclusão

A superioridadade dos resultados do $XLM - RoBERTa_{base}$, um modelo com $60M$ de parâmetros a menos que o $BERTimbau_{large}$ em alguns dos testes cruzados demonstram que a maldição do multilinguismo descrita por Conneau, Khandelwal *et al.*, 2020 não se aplica ao NLI em Língua Portuguesa em algumas circunstâncias. Pelo contrário, o fato de que o $XLM - RoBERTa_{base}$ foi capaz de utilizar o conhecimento dos *entailments* do ASSIN2 para reconhecer aqueles do ASSIN melhor que o BERTimbau, pode representar evidências de que em alguns domínios multilinguismo é benéfico para a capacidade de generalização de modelos para o NLI em português.

Por outro lado, o fato de que o BERTimbau demonstrou melhores resultados em todas as avaliações diretas realizadas levanta a hipótese de que, se avaliados no mesmo domínio em que foram treinados, modelos monolinguais tendem a superar os multilinguais no NLI em português.

Por fim, a verificação de que ambos XLM-RoBERTa e BERTimbau obtiveram resultados que indicam cobertura do domínio do ASSIN2 — com $6.5k$ pares anotados — por parte do PLUE/MNLI — com $433k$ pares anotados — indicam que o processo automatizado de geração do PLUE/MNLI pela tradução de máquina do MNLI não degradou as anotações de *entailment* para o domínio de pouca complexidade do ASSIN2 (legendas de fotos), embora esse domínio não esteja presente no PLUE/MNLI. Ou seja, há evidência de que incluir esse tipo de domínio no PLUE/MNLI não resultaria em melhores resultados na tarefa. Por outro lado, o mesmo não se verificou em relação ao ASSIN, o que pode ter sido causado tanto pelo processo de tradução automática ou à ausência do domínio do ASSIN entre aqueles representados no PLUE/MNLI. Ou seja, há evidência de que a inclusão de pares de premissa e hipótese provenientes de notícias melhoraria o reconhecimento de *entailment* pelos modelos testados.

Referências

[Bowman *et al.* 2015] Samuel R Bowman, Gabor Angeli, Christopher Potts e Christopher D Manning. "The snli corpus". Em: (2015) (citado na pg. 2).

[Brown *et al.* 2020] Tom Brown *et al.* "Language models are few-shot learners". Em: *Advances in Neural Information Processing Systems*. Ed. por H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan e H. Lin. Vol. 33. Curran Associates, Inc., 2020, pgs. 1877–1901. url: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf (citado na pg. 1).

[Conneau, Khandelwal *et al.* 2020] Alexis Conneau, Kartikay Khandelwal *et al.* *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL] (citado nas pgs. 22–26, 28, 37).

[Conneau, Lample *et al.* 2018] Alexis Conneau, Guillaume Lample *et al.* *XNLI: Evaluating Cross-lingual Sentence Representations*. 2018. arXiv: 1809.05053 [cs.CL] (citado nas pgs. 9, 23).

[Devlin *et al.* 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee e Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL] (citado nas pgs. 22, 26).

[Fonseca *et al.* 2016] E Fonseca, L Santos, Marcelo Criscuolo e S Aluisio. "Assin: avaliacao de similaridade semantica e inferencia textual". Em: *Computational Processing of the Portuguese Language-12th International Conference, Tomar, Portugal*. 2016, pgs. 13–15 (citado nas pgs. 4, 5).

[Ganaie *et al.* 2022] M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer e P.N. Suganthan. "Ensemble deep learning: a review". Em: *Engineering Applications of Artificial Intelligence* 115 (out. de 2022), pg. 105151. doi: 10.1016/j.engappai.2022.105151. url: https://doi.org/10.1016%2Fj.engappai.2022.105151 (citado na pg. 28).

[Hanin 2018] Boris Hanin. "Which neural net architectures give rise to exploding and vanishing gradients?" Em: 31 (2018). Ed. por S. Bengio *et al.* url: https://proceedings.neurips.cc/paper_files/paper/2018/file/13f9896df61279c928f19721878fac41-Paper.pdf (citado na pg. 19).

[Hossin e Sulaiman 2015] Mohammad Hossin e Md Nasir Sulaiman. "A review on evaluation metrics for data classification evaluations". Em: *International journal of data mining & knowledge management process* 5.2 (2015), pg. 1 (citado na pg. 34).

[Jurafsky e Martin 2021] Daniel Jurafsky e James H. Martin. *Speech and Language Processing*. https://web.stanford.edu/ jurafsky/slp3/, 2021 (citado na pg. 19).

[Lample e Conneau 2019] Guillaume Lample e Alexis Conneau. *Cross-lingual Language Model Pretraining*. 2019. arXiv: 1901.07291 [cs.CL] (citado nas pgs. 22, 23).

[Merity *et al.* 2017] Stephen Merity, Nitish Shirish Keskar e Richard Socher. "Regularizing and optimizing lstm language models". Em: *arXiv preprint arXiv:1708.02182* (2017) (citado na pg. 19).

[Real, Erick Fonseca *et al.* 2019] Livy Real, Erick Fonseca e Hugo Gonçalo Oliveira. "Organizing the assin 2 shared task." Em: *ASSIN@ STIL*. 2019, pgs. 1–13 (citado na pg. 6).

[Real, Rodrigues *et al.* 2018] Livy Real, Ana Rodrigues *et al.* "Sick-br: a portuguese corpus for inference". Em: *Computational Processing of the Portuguese Language: 13th International Conference, PROPOR 2018, Canela, Brazil, September 24–26, 2018, Proceedings 13*. Springer. 2018, pgs. 303–312 (citado na pg. 6).

[Salvatore 2020] Felipe de Souza Salvatore. "Analyzing natural language inference from a rigorous point of view". Tese de dout. Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2020. doi: 10.11606/t.45.2020.tde-05012021-151600 (citado nas pgs. 1, 2, 5).

[Souza *et al.* 2020] Fábio Souza, Rodrigo Nogueira e Roberto Lotufo. "Bertimbau: pretrained bert models for brazilian portuguese". Em: out. de 2020, pgs. 403–417. isbn: 978-3-030-61376-1. doi: 10.1007/978-3-030-61377-8_28 (citado nas pgs. 17, 26–29).

[Sutskever *et al.* 2014] Ilya Sutskever, Oriol Vinyals e Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 [cs.CL] (citado na pg. 13).

[Tiedemann e Thottingal 2020] Jörg Tiedemann e Santhosh Thottingal. "OPUS-MT – building open translation services for the world". Em: *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*. Ed. por André Martins *et al.* Lisboa, Portugal: European Association for Machine Translation, nov. de 2020, pgs. 479–480. url: https://aclanthology.org/2020.eamt-1.61 (citado na pg. 4).

[Vaswani *et al.* 2017] Ashish Vaswani *et al.* "Attention is all you need". Em: *Advances in neural information processing systems* 30 (2017) (citado nas pgs. 13, 15–19).

[Wang e Cho 2019] Alex Wang e Kyunghyun Cho. *BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model*. 2019. arXiv: 1902.04094 [cs.CL] (citado na pg. 20).

[Wang, Singh *et al.* 2019] Alex Wang, Amanpreet Singh *et al. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL] (citado na pg. 4).

[Williams *et al.* 2017] Adina Williams, Nikita Nangia e Samuel R Bowman. "A broad-coverage challenge corpus for sentence understanding through inference". Em: *arXiv preprint arXiv:1704.05426* (2017) (citado nas pgs. 3, 4).

[Young *et al.* 2014] Peter Young, Alice Lai, Micah Hodosh e Julia Hockenmaier. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". Em: *Transactions of the Association for Computational Linguistics* 2 (fev. de 2014), pgs. 67–78. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00166. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00166/1566848/tacl_a_00166.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00166 (citado na pg. 2).