FELIPE COELHO DE ABREU PINNA
GABRIEL TAKESHI MEDEIROS YAMASAKI
IAN ALKMIN SANTOS LA ROSA

Reconhecimento de Entidades Nomeadas na Língua Portuguesa

São Paulo

2020

FELIPE COELHO DE ABREU PINNA
GABRIEL TAKESHI MEDEIROS YAMASAKI
IAN ALKMIN SANTOS LA ROSA

Reconhecimento de Entidades Nomeadas na Língua Portuguesa

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo

São Paulo

2020

FELIPE COELHO DE ABREU PINNA
GABRIEL TAKESHI MEDEIROS YAMASAKI
IAN ALKMIN SANTOS LA ROSA

Reconhecimento de Entidades Nomeadas na Língua Portuguesa

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo

Área de Concentração: Engenharia da Computação

Orientador: Prof. Dr. Ricardo Luis de Azevedo da Rocha

São Paulo

2020

Felipe Coelho de Abreu Pinna, Gabriel Takeshi Medeiros Yamasaki, Ian Alkmin Santos La Rosa

Reconhecimento de Entidades Nomeadas na Língua Portuguesa/ Felipe Coelho de Abreu Pinna, Gabriel Takeshi Medeiros Yamasaki, Ian Alkmin Santos La Rosa. – São Paulo, 2020-

78p. : il. (algumas color.) ; 30 cm.

Orientador: Ricardo Luis de Azevedo da Rocha

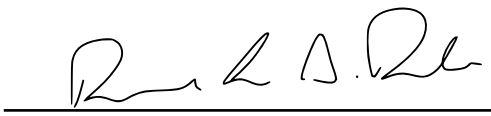
– Universidade de São Paulo – USP
Escola Politécnica
Departamento de Computação e Sistemas Digitais – PCS, 2020.

1. Engenharia. 2. Engenharia Elétrica. 3. Engenharia de Computação. 4. Curso de Gradução 5. Ensino e Aprendizagem I. Orientador. II. Universidade de São Paulo. Escola Politécnica. Departamento de Computação e Sistemas Digitais

FELIPE COELHO DE ABREU PINNA
GABRIEL TAKESHI MEDEIROS YAMASAKI
IAN ALKMIN SANTOS LA ROSA

Reconhecimento de Entidades Nomeadas na Língua Portuguesa

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo



Ricardo Luis de Azevedo da Rocha
Orientador

São Paulo
2020

RESUMO

Um dos principais tópicos em Processamento de Linguagem Natural é o processo de Reconhecimento de Entidades Nomeadas. Esse processo trata da identificação de entidades nomeadas e de sua classificação em categorias pré-selecionadas. As entidades nomeadas são termos da linguagem que se referem a objetos, pessoas, lugares, conceitos, entre outros e que possuem um nome específico. Este projeto tem como objetivo a realização desse reconhecimento na Língua Portuguesa, utilizando tecnologias de aprendizado de máquina e redes neurais, alimentadas pore um corpus manualmente construído e classificado. As melhores redes neurais produzidas alcançaram uma taxa de acerto por volta de 97% , mostrando a viabilidade desse reconhecimento na Língua Portuguesa.

Palavras-chave: Reconhecimento de Entidades Nomeadas. Rede Neural. Língua Portuguesa. Aprendizado de Máquina. Processamento de Linguagem Natural. Corpus

ABSTRACT

One important topic in the Natural Language Processing area is Named Entity Recognition. This process detects named-entities and categorizes them in one of any previously decided upon categories. Named entities are individual terms of language that refer to objects, places, people, concepts, and othres that have a specific name. This project aims to perform Named Entity Recognition in the Portuguese Language, utilizing machine learning and neural network techniques, supplied by a handmade, curated and classified corpus. The best neural networks produced managed to achieve around 97% precision, showing the viability of Named Entitiy Recognition in Portuguese.

Keywords: Named Entity Recognition. Neural Netwroks. Portuguese Language. Machine Learning. Natural Language Processing. Corpus

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

NER	*Named Entity Recognition*
IA	Inteligência Artificial
PLN	Processamento de Linguagem Natural
GRU	*Gated Recurrent Unit*
LSTM	*Long Short-Term Memory*
BILSTM	*Bidirectional Long Short-Term Memory*
RNN	*Recurrent Neural Network*
CPF	Cadastro de Pessoas Físicas
CNPJ	Cadastro Nacional de Pessoas Jurídicas
NLTK	*Natural Language Toolkit*
BoW	*Bag of Words*
TF-IDF	*Term Frequency - Invesrse Document Frequency*
PMI	*Pointwise Mutual Information*
ReLU	*Rectified Linear Unit*
CPU	*Central Processing Unit*
GPU	*Graphics Processing Unit*
TPU	*Tensor Processing Unit*
HTML	*HyperText Markup Language*
XML	*eXtensible Markup Language*
NILC	Núcleo Interinstitucional de Linguística Computacional
API	*Application Programming Interface*

LISTA DE SÍMBOLOS

σ	Função sigmoide
Y	Conjunto de classes
$\mathbf{y}$	Vetor de saída da rede
y_i	i-ésimo item do vetor $\mathbf{y}$
K	tamanho do vetor $\mathbf{y}$

SUMÁRIO

1 INTRODUÇÃO

Processamento de linguagem natural é um campo de estudos que visa tornar possível que computadores sejam capazes de analisar, processar e até mesmo replicar comunicação entre humanos, abrangendo diversas áreas como linguística, inteligência artificial, entre outros. Dentro deste campo de estudos existem diversos problemas envolvendo a representação, extração e compreensão de textos produzidos por humanos em linguagem natural.

Um deles é o reconhecimento de entidades nomeadas (*named entity recognition*, ou NER), que tem como objetivo a localização e classificação de entidades durante a extração de informação de dados não estruturados, tais como documentos científicos. Um exemplo de entidade que se pode destacar em tais documentos são nomes de pessoas, que se identificados podem ser usados para relacionar trabalhos de diferentes origens que tratem de elementos semelhantes.

1.1 Motivação

Embora o estudo no campo de linguagem natural seja antigo (FIRTH, 1957), houve importantes avanços durante a última década nessa área. Principalmente com a onda de interesse em inteligência artificial, especialmente no aprendizado de máquina, impulsionada pelo maior poder de computação de *hardware* moderno e acesso a maior número de dados.

Atualmente técnicas de processamento de linguagem natural são utilizadas em produtos de mercado por meio de *chatbots* e assistentes virtuais, que comumente são encontrados em dispositivos móveis, sites de comércio eletrônico, serviços de atendimento a cliente, entre outros. Essa aplicação permite que a interação com o usuário seja mais fácil e fluida, por usar o meio de comunicação natural do ser humano, a linguagem humana.

NER é extremamente útil na categorização de dados, especialmente em arquivos de texto ou gravações de voz, ainda que no segundo caso softwares de reconhecimento de voz sejam necessários. Com o progresso das tecnologias de informação, cada vez mais dados estão disponíveis para análise e com o crescimento do interesse por *analytics*, NER se torna cada vez mais essencial. Apesar disso, os principais esforços na área se concentram na língua inglesa, tendo de longe o maior número de recursos relacionados ao campo, enquanto outras línguas como o português acabam tendo recursos muito mais limitados.

1.2 Objetivo

Este trabalho busca estudar as técnicas de processamento de linguagem natural necessárias para a tarefa de reconhecimento de entidades nomeadas (NER), considerando pré-requisitos como *word embedding* e possíveis aplicações e complementações a opções de análise léxica, sintática e semântica automática de linguagem natural. Serão apresentados os algoritmos clássicos e recentes da área e como eles podem ser aplicados para o contexto da língua portuguesa, uma vez que os desenvolvimentos científicos na área de PLN é focado primeiramente no idioma inglês.

Em um primeiro momento serão estudados algoritmos e técnicas. Para demonstrar seu uso será desenvolvido um conjunto de testes para a língua portuguesa, onde eles serão aplicados, a fim de compreender quais as vantagem e limitações de cada um. Esse conjunto de testes necessitará de uma base de dados linguísticos em português para que sejam bem definidos e as técnicas possam ser comparadas. Este também é outro problema, pois não existem tantos conjuntos de dados disponíveis como na língua inglesa, no entanto existem algumas fontes disponíveis como (HARTMANN et al., 2017), (SERRANI; UEBEL, 2011) e (MILIDIÚ; DUARTE; CAVALCANTE, 2007) que podem ser usadas nesse trabalho ou podem ser ampliadas durante o curso do mesmo.

Em um segundo momento, será proposto um novo algoritmo ou alguma melhoria a algum algoritmo existente, com o objetivo de aprimorar a aplicação de tecnologias de processamento de linguagem natural para a língua portuguesa. A solução proposta deverá ser comparada com as demais estudadas anteriormente para avaliar a eficácia da proposta e determinar em quais cenários ela poderá ser aplicada.

1.3 Definição do Problema

O reconhecimento de entidade nomeada é um caso da tarefa de extração de informação que busca tornar o texto não estruturado em dados estruturados. Primeiramente, é preciso definir quais **entidades nomeadas** que serão identificadas, que de modo geral pode ser qualquer termo que pode ser referido com um nome próprio. A tarefa de NER é **encontrar** as **menções** de entidades nomeadas no texto e **labels** do seu tipo (JURASFKY; MARTIN, 2019). Entidades nomeadas comuns são pessoas, lugares e organizações, mas podem existir outras menos comuns como nomes de medicamentos ou produtos financeiros.

Após as entidades serem extraídas elas podem ser ligadas a objetos reais pelas tarefas de resolução de correferência (*coreference resolution*) e ligação de entidades (*entity linking*). Vale ressaltar que essa tarefa lida diretamente com o texto escrito, que pode ser provido pela transcrição automática da voz por meio de um processo de reconhecimento de fala, que não será abordado no momento.

Essa tarefa está localizada, no campo de aprendizado de máquina, no grupo de problemas de rotulação de sequências. Em geral, a entrada é uma sequência de dados e a saída é um rótulo, ou classificação, para cada elemento da sequência de entrada. No problema de reconhecimento de entidades nomeadas, a sequência de entradas pode ser a representação do texto em linguagem natural, operando o texto como uma sequência de palavras ou caracteres, e a classificação de saída será o rótulo da entidade nomeada.

Como entidades podem ser compostas de mais de uma palavra, é útil realizar o IOB tagging, em que os rótulos podem ser o começo de uma entidade (B), interno a uma entidade (I) ou externo (O) que não pertence a nenhuma entidade. As soluções para esse problema se apresentam em três formas: baseadas em regras, baseadas em atributos e algoritmos neurais, sendo que os melhores resultados atualmente são produzidos por redes neurais treinadas em grandes massas de dados.

Ainda resta o problema de que as soluções existentes são principalmente focadas na língua inglesa e os demais idiomas, como o português, são adaptados e possuem menor qualidade de resultado. Também, existem entidades que são específicas para o contexto brasileiro, como o número de CPF ou CNPJ, para os quais não existe soluções de NER atualmente.

1.4 Justificativa

Considerando a falta de soluções de processamento de linguagem natural que tratem nativamente do português, dado que as melhores soluções são desenvolvidas na língua inglesa e adaptadas para o português. Além da quantidade de dados disponíveis para o português para o treinamento de modelos ser menor, não existem boas alternativas para o reconhecimento de entidades nomeadas que sejam específicas ao contexto da língua portuguesa, tais como documentos específicos do Brasil como CPF e CNPJ. Assim, surge a necessidade de pesquisas na área de processamento de linguagem natural em português.

1.5 Organização do Trabalho

A organização do trabalho é a seguinte: no Capítulo 2 são apresentados os conceitos de PLN e aprendizado de máquina necessários para a tarefa de NER, no Capítulo 3 as tecnologias que serão usadas para o desenvolvimento de uma solução, Capítulo 4 comenta sobre a metodologia de trabalho, Capítulo 5 obtém a especificação de requisitos da solução e onde ela se encontra no mundo, o Capítulo 6 descreve os experimentos realizados e resultados obtidos, e as descobertas e conclusões decorrentes são sintetizadas no Capítulo 7.

2 CONCEITOS

A tarefa de reconhecimento de entidades nomeadas exige uma preparação dos dados antes de aplicar os algoritmos de inteligência artificial, envolvendo pré-processamento dos dados textuais e representação numérica e vetorial dos textos e sentenças. A seguir são apresentados os temas de cada etapa que serão estudados neste trabalho.

2.1 Tratamento de dados em linguagem natural

O tipo de dado que será utilizado durante o trabalho são textos em linguagem natural, que são produzidos pela expressão escrita ou falada da linguagem de humanos. Não é adequado aplicar algoritmos diretamente sobre os textos, visto que existem diversos artefatos que podem dificultar a extração de informações, que são formalismos da linguagem usados para outros seres humanos lerem e interpretarem o texto. Quando uma máquina deseja extrair informações é mais adequado simplificar o texto por meio de técnicas simples e conhecidas, mas que permitem obtenção de melhores resultados nos futuros algoritmos de PLN. A seguir se apresenta uma sequência de passos de pré-processamento de textos em linguagem natural que proporcionam maior facilidade na extração de informações:

- *tokenização*
- *stop words*
- *stemmização*
- *lemmatização*
- campos especiais

A primeira etapa na maioria dos *pipelines* de processamento de linguagem natural é a tokenização. O objetivo é separar o texto e as sentenças em “tokens”, que podem ser compreendidos como palavras. Usualmente espaços em branco são utilizados nos textos em português para a separação de palavras, mas sinais de pontuação também delimitam tokens (JURASFKY; MARTIN, 2019).

Na língua portuguesa, como em muitas outras, existem palavras que são usadas com muita frequência em quase todos os textos escritos. Artigos como “a” e “o” e pronomes como “que”, “ele” e “ela” são usados em quase todos os textos da língua portuguesa e trazem pouca informação, enquanto palavras menos comuns e mais específicas ao domínio do texto permitem diferenciar mais facilmente dois textos diferentes. Logo, é comum remover essas *stop words* ao analisar textos em linguagem natural para

focar nas palavras com mais significado e que proporcionam mais informação para a análise. Nessa tarefa são utilizadas listas de *stop words* conhecidas e disponíveis em bibliotecas como NTLK (BIRD; KLEIN; LOPER, 2009).

Outro passo é reduzir a palavra a sua raiz, que pode ser feito por meio de stemmização ou lemmatização. Embora esses conceitos sejam semelhantes a lemmatização busca obter o *lemma* das palavras mais precisamente por meio de mapas de lemmatização construídos por especialistas na linguagem, enquanto stemmização apenas retira os sufixos no final das palavras (JURASFKY; MARTIN, 2019). Quando tenta-se compreender a semântica de uma sentença é mais fácil estudar as raízes das palavras do que as diversas variações que podem ocorrer no idioma. Por exemplo, as diversas conjugações verbais são palavras diferentes que podem ser reduzidas a mesma raiz com pequena perda no significado completo.

Outra possibilidade de pré-processamento de textos em linguagem natural é a substituição de campos especiais. Isso pode ser interessante quando existe grande variação de tokens que possuam o mesmo significado. Campos como emails, CPF e números podem ser facilmente capturados por expressões regulares e o valor específico não é importante para a interpretação da sentença. Assim, é comum substituir todas as ocorrências desses tokens por um único token padrão que represente o campo.

As etapas de processamento apresentadas são básicas e amplamente utilizadas em sistemas que interajam por meio da linguagem natural. Cada aplicação específica pode escolher quais etapas de pré-processamento realiza, mas as mencionadas acima são usadas na maioria dos casos.

2.2 Vetorização de palavras e sentenças

Para possibilitar a comparação e análise de sentenças e palavras pelo computador utiliza-se o método da vetorização, que busca obter uma representação numérica do conteúdo ou significado daquilo que se está vetorizando. Resultando em um conjunto de números representativos do objeto inicial, de muito mais fácil manipulação, que pode ser utilizado nos estudos de linguagem natural.

Transformando o texto ou palavra em um conjunto de números, pode-se utilizá-lo como um vetor, e o conjunto dos vetores formados pela aplicação do processo em várias sentenças ou palavras caracterizam um espaço vetorial, sendo assim possível realizar quaisquer operações que seriam possíveis em espaços vetoriais de mesmas dimensões, tais como produto escalar, valor absoluto, entre outras operações que se mostram úteis em PLN.

Com isso é possível obter uma representação vetorial das palavras ou das sentenças inteiras, que permite realizar operações aritméticas vetoriais (PENNINGTON;

SOCHER; MANNING, 2014) ou aplicar algoritmos de inteligência artificial e aprendizado de máquina. Existem diversas maneiras de vetorizar palavras, variando desde simples contagem de frequências a grandes redes neurais treinadas com uma enorme quantidade de dados como páginas da Wikipédia (HARTMANN et al., 2017). A seguir são apresentadas algumas das formas de vetorização mais comuns:

- *Bag of Words* (BoW)
- N-grams
- TF-IDF
- PMI
- GloVe
- Word2Vec

Uma das implementações mais simples de vetorização é o modelo *Bag of Words* (BoW). O BoW é uma vetorização simples de sentenças em que o vetor resultante é o resultado apenas do conjunto de palavras (ou tokens) presentes no texto e de suas respectivas multiplicidades. Devido a sua simplicidade, suas aplicações são limitadas, porém BoW pode ser bem utilizada em classificações e caracterização de textos. As limitações do modelo são, no entanto, óbvias. Por exemplo, BoW não leva em consideração ordem e posicionamento de palavras em uma sentença, o que torna a vetorização resultante menos precisa, uma vez que a relação entre os termos não é levada em conta. O modelo também acaba valorizando demasiadamente palavras comuns como pronomes e artigos, porém isso pode ser mitigado utilizando as *stop words* já mencionadas.

Uma maneira de diminuir o impacto das limitações do modelo *Bag of Words* é a implementação do que é chamado de modelo *N-grams*. Este modelo é uma alternativa ao BoW em que ao invés de utilizar apenas termos isolados do resto do texto, o *N-grams* utiliza grupos de termos consecutivos na amostra para vetorizá-las. Dessa forma, o *N-grams* consegue levar em conta contexto da utilização dos termos, o que é uma vantagem sobre o modelo BoW, sendo portanto uma vetorização com maior precisão. Ao realizar *N-grams* um tamanho para os grupos de palavras que serão comparados deve ser selecionado, com esse determinando o valor de N no nome do modelo aplicado. Poderia se dizer que o BoW se trata na realidade do caso específico de *N-grams* em que N é 1, ou uma vetorização por *unigram*.

Outra maneira de aprimorar a vetorização é pela aplicação de TF-IDF (*term frequency - inverse document frequency*). TF-IDF é um método pelo qual podemos

discriminar a importância de termos para a vetorização de tal forma que o valor dado a uma palavra qualquer na classificação é inversamente proporcional à porcentagem das amostras em que o termo aparece, assim diminuindo o valor de palavras frequentes mas de uso tão comum ao ponto de não serem interessante pelo ponto de vista de classificação, enquanto acrescenta nuances às demais palavras do léxico da língua, dando forte valor à vocabulário especializado, criando uma vetorização ponderada(JURASFKY; MARTIN, 2019).

Uma alternativa de ponderação dos dados é a aplicação de PMI (*Pointwise Mutual Information*). O PMI é uma análise que atribui valor à associação entre dois termos de acordo com quão frequente é a coocorrência entre os dois, através de uma análise da frequência de ocorrência das palavras individualmente nos documentos analisados e a sua ocorrência em conjunto nos mesmos, se comparadas a um valor esperado padrão. Como uma associação negativa é em geral considerada como um parâmetro não confiável na maioria dos casos, é mais comum o uso de PMI positivo, ou PPMI, no qual todos os valores negativos são substituídos por 0, em aplicações de PLN (JURASFKY; MARTIN, 2019).

GloVe (PENNINGTON; SOCHER; MANNING, 2014) e Word2Vec (MIKOLOV et al., 2013) são métodos de vetorização bem diferentes dos anteriormente apresentados. Enquanto o modelo BoW, por exemplo, resulta em vetores de dimensões tão grandes quanto for necessário para que qualquer palavra tenha sua própria dimensão, os vetores gerados pro GloVe ou Word2Vec são muito menores, frequentemente entre 100 e 1000 dimensões. Como o tamanho de cada vetor no BoW é diretamente proporcional ao tamanho do vocabulário, quando o vocabulário é extenso isso causa um grande consumo de recursos computacionais, enquanto esses outros métodos permitem limitar a dimensionalidade do espaço vetorial. Embora isso pareça uma desvantagem, na realidade métodos como BoW produzem matrizes de palavras esparsas, enquanto GloVe e Word2Vec produzem matrizes mais densas que concentram mais informações e possuem propriedades aritméticas melhores.

Esses dois modelos formam seus vetores aproximando de termos dos quais ele aparece próximo nos documentos amostrados, e os afastando de termos que não aparentam ter relação semântica entre si. Ao fazer isso, a vetorização aproxima vetores de termos que frequentemente ocorrem em mesmo contexto, aproximando-os semanticamente e gerando uma vetorização precisa, com palavras similares apresentando proximidade no espaço vetorial, e criando interações semânticas entre diferentes vetores. Estes modelos são preferíveis em aplicações que utilizem aprendizado de máquina ou processos parecidos, uma vez que os vetores de menor dimensionalidade agilizam o processo do aprendizado de máquina significativamente.

2.3 Redes Neurais

Atualmente a área de processamento de linguagem natural é dominada por algoritmos de aprendizado de máquina. No problema de NER isso não é diferente onde o estado da arte utiliza técnicas de aprendizado profundo ou *Deep Learning* (GOODFELLOW; BENGIO; COURVILLE, 2016). A classe de problemas mais usada é aprendizado supervisionado, na qual existem uma conjunto de dados rotulados de relações de entrada e saída de uma rede neural. Nessa classe o humano opera como professor da máquina, ele mostra exemplos conhecidos nos quais a máquina tentará encontrar padrões para prever rótulos de exemplos desconhecidos. Outras classes de problemas de aprendizado de máquina são aprendizado não supervisionado e aprendizado por reforço.

A estrutura de rede neural mais comum é a rede *feed-forward*, nela os neurônios são organizados em camadas e a saída dos neurônios de uma camada é ligada a entrada dos neurônios da próxima camada. A Figura 1 ilustra uma rede desse modelo com uma camada de entrada (verde), uma de saída (vermelho) e duas camadas ocultas (azul). Cada um dos arcos que ligam dois neurônios possui um peso associado, caso esses pesos sejam selecionados adequadamente, uma rede com um número suficiente de neurônios e uma função de ativação não linear pode aproximar uma grande família de funções (GOLDBERG, 2016).

O processamento dos neurônios é uma transformação linear da entrada para a saída, os únicos pontos em que existem não linearidades nas redes neurais são nas funções de ativação. Uma função de ativação comum é a sigmoide que mapeia todos os valores reais no intervalo $[0, 1]$ por meio da seguinte função:

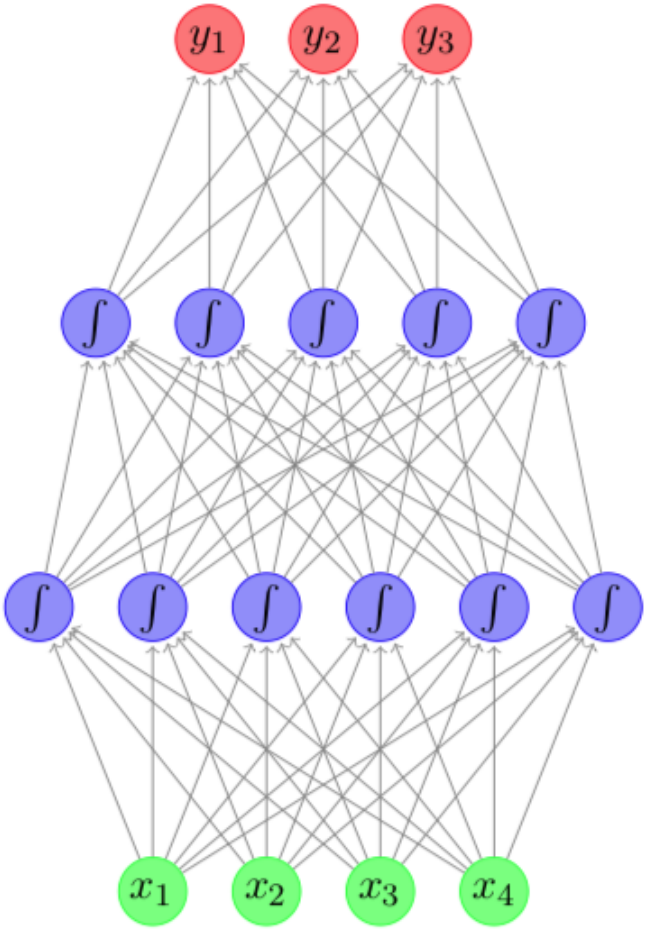
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Outras funções de ativação usadas são: tangente hiperbólica, *softmax* e ReLU (GLOROT; BORDES; BENGIO, 2011).

A predição mais imediata que pode ser realizada para NER é utilizando uma rede *feed-forward* simples. Esta recebe de entrada um vetor correspondente de uma palavra, que foi gerado na etapa de tratamento do texto, ou mais normalmente um grupo de concatenação fixo, com alguns vetores anteriores e posteriores. Dessa entrada se recebe uma predição na camada final da rede sobre o tipo de entidade que a entrada é, os tipos possíveis de resposta são delimitados por quem faz a análise.

No entanto, para a NER efetiva de linguagens naturais é normalmente necessário um método diferente para uso de redes neurais, dado que muitas vezes informações relevantes para o reconhecimento de uma entidade podem se espalhar por todo um parágrafo ou texto, já que na abordagem anterior a janela de contexto é limitada a

Figura 1 – Rede neural *feed-forward* com duas camadas ocultas



Fonte: (GOLDBERG, 2016)

algumas palavras vizinhas, assim requisitando um tipo diferente de rede neural. Uma das soluções possíveis é o uso de *Recurrent Neural Networks*, ou RNNs.

A RNN opera no princípio de memória, mantendo cumulativamente parte da informação processada no passado. Nesse tipo de rede neural as saídas da primeira camada, além de serem passadas para a camada seguinte, também são retroalimentadas na primeira camada. Com esta configuração, se inicia a predição das entidades do texto com um vetor inicial, que é processado pela rede neural normalmente e produz uma predição final, em seguida se processa o próximo vetor do texto porém com a retroalimentação do processamento da primeira camada do vetor anterior. Com isso a rede processa essa entrada e produz em sua camada final uma predição levando em conta a memória passada, depois se coloca o próximo vetor na entrada da camada inicial e assim em diante. Assim, existe uma dependência da ordenação das palavras, já que palavras passadas constroem uma memória que afeta a predição da palavra presente.

Porém a RNN apresenta um problema, quanto maior é o texto a ser processado, maior é o acumulo de informação ao longo de seu processamento, assim diluindo cada vez mais qualquer pedaço útil de informação que exista na memória do processamento. Consequentemente, são extensivamente utilizadas para a NER as redes neurais LSTM, ou *Long Short-Term Memory*, tais redes neurais são um tipo de RNN, que possuem mecanismos de esquecimento de parte da memória acumulada no processamento de um texto, possibilitando assim que apenas as melhores, ou mais recentes informações sejam acumuladas (CHIU; NICHOLS, 2016).

É a rede neural LSTM e suas variações, como por exemplo LSTM bidirecional, que trabalha com duas redes neurais simultâneas processando o texto em sentidos diferentes, que são as mais comumente utilizadas na NER, e também será o tipo de rede principalmente empregado neste projeto.

3 TECNOLOGIAS

Para criação de soluções que lidem com linguagem natural, existem diversas tecnologias para auxiliar no processo de desenvolvimento. Tarefas comuns como apresentadas na seção 2.1 são implementadas por bibliotecas de software específicas para PLN. A seguir apresentamos algumas das tecnologias que serão usadas para a pesquisa da solução de reconhecimento de entidades nomeadas em português.

3.1 Python

Python é uma linguagem de programação de fácil aprendizado, simples e poderosa. A sintaxe elegante promove a rápida prototipação e teste de programas em diversas plataformas, sendo ideal para scripts e desenvolvimento rápido (Python Software Foundation, 2020). Por conta dessas facilidades, Python é uma linguagem muito utilizada no meio acadêmico e de pesquisa em aprendizado de máquina, pois necessitam de muita experimentação e o rápido desenvolvimento permite testar diversas alternativas sem muitas preocupações na implantação do software em um primeiro momento.

Como essa linguagem é predominante na área de aprendizado de máquina, diversas bibliotecas especializadas em *machine learning* surgiram para acelerar o desenvolvimento. Uma das principais bibliotecas, que foi desenvolvida pela Google, é a TensorFlow (ABADI et al., 2015), que foi inicialmente desenvolvida para Python e posteriormente ganhou outras implementações em diversas linguagens. Ela auxilia na criação e treinamento de redes neurais, assim como estruturas de dados em tensores que são otimizadas para operações matriciais e execução em GPUs. Outras bibliotecas semelhantes que ganharam notoriedade na comunidade de aprendizado de máquina em Python são PyTorch (Torch, 2019) e Caffe (JIA et al., 2014), que são alternativas ao Tensorflow possuindo grande parte das funcionalidades também disponível.

3.2 Tensorflow

Tensorflow é uma das bibliotecas mais dominantes na área de aprendizado profundo desde 2015, quando deixou de ser um projeto interno do Google e foi liberado para uso comunitário na modalidade de *open source* (ABADI et al., 2015). Embora atualmente existam implementações para outras linguagens será utilizada a versão original como biblioteca da linguagem Python para ser possível aproveitar as demais bibliotecas disponíveis para PLN e aprendizado de máquina. Devido a sua facilidade na criação, treinamento e avaliação de redes neurais e bom desempenho ela foi uma escolha segura para o desenvolvimento das soluções.

Outro fator que impactou a escolha dessa biblioteca entre as alternativas dis-

poníveis é a sua boa integração com o ambiente de execução utilizado, o Google Colaboratory. Por serem fornecidos pela mesma empresa a biblioteca consegue utilizar eficientemente os recursos disponíveis nas máquinas virtuais disponíveis, podendo executar em CPU, GPU ou TPU sem que haja necessidade de grandes configurações por parte dos desenvolvedores. Assim, a biblioteca Tensorflow se apresentou ideal para as condições de uso encontradas neste projeto.

3.3 Natural Language Toolkit

Além dessas bibliotecas para aprendizado profundo para a criação de redes neurais, existem aquelas voltadas para a linguagem natural, outro ponto em que a linguagem Python também é amplamente utilizada. Essas bibliotecas tem conjuntos de dados de linguagem e/ou algoritmos comuns de PLN que são bem estabelecidos e comuns a diversas aplicações.

Uma das principais bibliotecas Python para PLN é a Natural Language Toolkit (NLTK) (BIRD STEVEN; KLEIN, 20009), uma plataforma para construir programas que lidam com dados em linguagem natural. Ela possui corpus de textos e recursos léxicos como a WordNet (Princeton University, 2010) para diversos idiomas incluindo o português. Além desses dados, existem as funcionalidades de classificação, tokenização, *stemming*, *tagging*, *parsing* e raciocínio semântico.

O uso dessa tecnologia pode auxiliar no desenvolvimento de uma solução para português fornecendo dados linguísticos e implementações prontas de algoritmos existentes. O conjunto de dados em português disponível são textos de livros de domínio aberto. Também, existe uma lista de *stop words* que poderá ser usada durante a preparação dos dados neste trabalho. Além desses dados, a tokenização será muito útil, já que não será preciso desenvolver esse processamento desde o princípio. As demais funções de tratamento de texto como *stemming* e *tagging* podem ser experimentadas para avaliar sua eventual contribuição na qualidade da solução.

3.4 Beautiful Soup

Para a obtenção de um corpus inicial de textos para serem tratados e classificados foi decidido o uso de textos de notícias e artigos da internet, utilizando de *web scraping*. O Beautiful Soup é um pacote Python para análise de documentos HTML e XML, facilitando a interpretação das páginas de notícias, auxiliando na retirada do texto.

3.5 Wikipedia

O conjunto de dados utilizado é muito importante para a acurácia da solução em qualquer campo de Inteligência Artificial, quanto maior o conjunto de dados e melhor a qualidade dos dados, melhor o resultado será. Para PLN isso não é diferente, grandes avanços nas pesquisas se devem em grande parte a grandes corpus de textos disponíveis para uso acadêmico.

Existem algumas bases de dados rotulados no idioma português como (SERRANI; UEBEL, 2011). Um dos maiores conjuntos de dados textuais em linguagem natural disponíveis na internet é a Wikipedia (WIKIPEDIA, 2020), onde diversas páginas estão escritas em linguagem natural em português, que podem ser usadas na solução projetada. Uma possibilidade de uso desses dados é como fator de decisão, por exemplo quando se tem dúvida sobre alguma entidade, a busca na Wikipedia pode contribuir na classificação correta, utilizando alguma técnica de comparação de documentos como distância cossenoidal (JURASFKY; MARTIN, 2019).

3.6 Scikit-learn

Scikit-learn é um bilbioteca de aprendizado de máquina em Python com um grande conjunto de algoritmos do estado da arte para aprendizado supervisionado e não supervisionado (PEDREGOSA et al., 2011). Por conta de suas facilidade de uso, performance, documentação e consistência de API, se tornou uma escolha popular para projetos e pesquisas de aprendizado de máquina. Como este projeto fará maior uso de redes neurais, os modelos serão criados por meio da biblioteca Tensoflow, contudo a biblioteca Scikit-learn apresenta uma série de funções auxiliares que serão empregadas como manipulação de *datasets*, seleção de parâmetros e medição de resultados.

4 METODOLOGIA DE TRABALHO

A metodologia de trabalho está dividida em duas etapas, a primeira de pesquisa na literatura que levou à proposta de arquitetura e a segunda de desenvolvimento e testes da arquitetura proposta. Durante a fase de pesquisa foram buscadas informações em livros, artigos, publicações acadêmicas e documentação de softwares *open source* relativos a área de pesquisa. Esses documentos foram estudados por meio da técnica de fichamentos. Adquirido o conhecimento suficiente foi feita uma proposta de arquitetura que soluciona o problema de reconhecimento de entidades nomeadas em português, com base nas referências obtidas, adaptando as soluções encontradas para o contexto do idioma e quantidades de dados e recursos disponíveis.

Durante o processo de implementação da arquitetura foi utilizado o *pair programming*, método de desenvolvimento de software colaborativo onde mais de uma pessoa trabalham em conjunto em programas complexos de forma a reduzir erros e melhorar a qualidade do produto. Após a implementação básica da solução, o trabalho foi dividido de forma que cada integrante trabalhou em uma funcionalidade isoladamente e foi possível a realização de vários testes em paralelo de forma a realizar mais tentativas que eventualmente aprimoraram os resultados obtidos.

A solução proposta foi o uso de *web scraping* para coleta de textos de artigos e notícias em sites de notícia na internet, esses textos em seguida foram tratados e suas entidades classificadas manualmente. Usando diferentes tipos de redes neurais e diferentes tamanhos de *dataset* foi realizado o treinamento e validação de seus modelos. Como avaliação dos modelos foram comparadas a diferença de sua taxas de acerto geral com a *baseline* usando a classe '0 - Entidade Não Nomeada', assim como as taxas de acerto individuais de cada classe, representados em matrizes de confusão.

4.1 *Baseline* e Taxa de Acerto

A avaliação de uma solução de *machine learning* deve ser feita por meio de comparações com outras soluções nas condições mais próximas possíveis. Dessa maneira, é útil obter um *baseline* simples para estabelecer um mínimo resultado necessário que deve ser superado pela solução proposta. No problema de classificação, uma medida fácil de obter é a taxa de acerto de sempre "chutar" a classe mais provável. A taxa de acerto, que é uma das métricas usada neste trabalho, também referenciada como acurácia, é calculada da seguinte maneira:

$$\text{Taxa de acerto} = \frac{\text{Número de acertos}}{\text{Número total de exemplos}} \tag{4.1}$$

Para a taxa de acerto base o número de acertos é igual ao número de ocorrência da classe mais frequente. Então a taxa de acerto desse *baseline* é igual a frequência

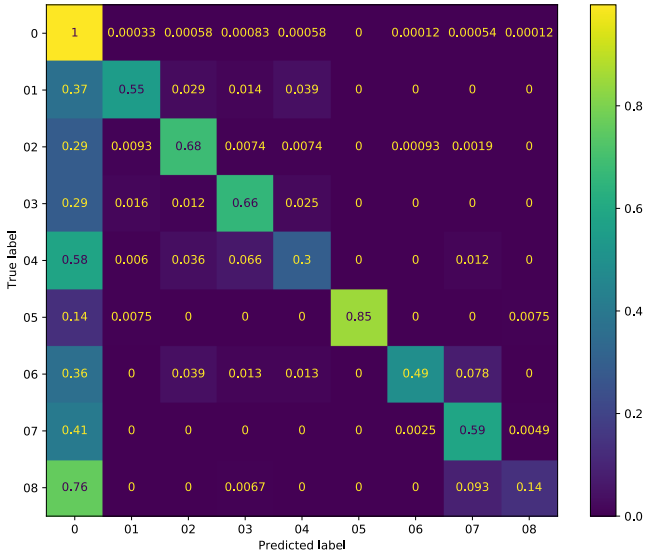
da classe mais frequente. Para o conjunto de dados utilizado, a classe mais comum é a classificação ‘0’, que significa uma entidade não-nomeada. Por exemplo, se a *baseline* de um dado *dataset*, usando a classe ’0’, é 88% , e um modelo treinado de rede neural avalia esse *dataset* e alcança uma taxa total de acerto de 85% , ele estaria 3% abaixo do *baseline*, assim sendo menos útil que o simples “chute”.

4.2 Matriz de Confusão

Para maior análise dos resultados foram utilizadas matrizes de confusão. Estas construções mostram em seu eixo vertical o ’*True Label*’ da entidade, e em seu eixo horizontal o ’*Predicted Label*’. Cada linha da matriz soma um total de 1.00 (100%), cada célula apresenta a porcentagem de ocorrência de cada caso específico. Desse modo, as únicas células da matriz que representam classificações corretas são as células da diagonal principal, onde o ’*True Label*’ e o ’*Predicted Label*’ são o mesmo, todas as outras células configuram classificações erradas, e assim pela matriz de confusão é possível identificar não apenas a taxa de acerto de cada classe, mas como ela está sendo interpretada no caso de erro.

Um exemplo de matriz de confusão é mostrada na Figura 2. Por meio do estudo dessas matrizes é possível verificar se a taxa de acerto de um determinado modelo é realmente traduzida em melhores classificações, pois a taxa de acerto global pode esconder classificações erradas para categorias menos frequentes, coisa que ficará evidente em uma matriz de confusão. Também, será possível analisar o desempenho das soluções para cada classe e compará-la com outros modelos, que pode informar maneiras de aprimorar o resultado global, uma vez que é mais explícito os tipos de erros cometidos.

Figura 2 – Exemplo de uma Matriz de confusão



Fonte: os autores

5 ESPECIFICAÇÃO DE REQUISITOS

O resultado que será entregue como demonstração da solução da tarefa de reconhecimento de entidades nomeadas em português será um programa capaz de identificar e rotular todas as entidades relevantes em um texto. Utilizando as tecnologias apresentadas na Capítulo 3 esse programa terá como entrada um documento de texto em linguagem natural, como um documento em formato .txt ou .pdf e produzirá as anotações das entidades. As anotações são compostas pelo rótulo da entidade (pessoa, organização, lugar, etc.) e as posições da frase em que essa entidade ocorre, essa delimitação das posições poderá ser feita indicando os índices dos caracteres inicial e final da entidade.

Suponha um documento de texto com apenas uma frase: “Apple está buscando comprar startup do Reino Unido por U$ 1 bilhão”. Um resultado de anotações corretas para essa frase é apresentado na Tabela 1. Para outros documentos espera-se obter resultados semelhantes contendo todas as entidades relevantes dentro de um conjunto de entidades nomeadas pré-estabelecido.

Tabela 1 – Exemplo de anotação produzida por NER

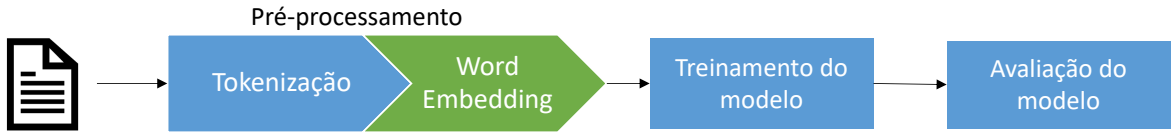
Texto	Início	Fim	Rótulo
Apple	0	5	organização
Reino Unido	39	50	lugar
U$ 1 bilhão	55	66	valor monetário

A qualidade da solução obtida será avaliada por meio de métricas de desempenho de algoritmos de aprendizado de máquina. Esse resultado será comparado com outras soluções existentes no mercado e com os melhores algoritmos do estado da arte na língua inglesa. O objetivo é que a acurácia e precisão sejam comparáveis com outras iniciativas da língua portuguesa, considerando o tamanho do conjunto de dados disponível para o trabalho.

O processo de desenvolvimento do sistema de NER segue o diagrama da Figura 3. O conjunto de dados textuais, coletados a partir de páginas de Wikipédia ou de notícias disponibilizadas nos sites de grandes veículos tais como a Folha de S. Paulo ou o Estado de S. Paulo, passam primeiramente pela etapa de pré-processamento, que prepara os dados para uso do algoritmo de aprendizado de máquina. A proposta de *pipeline* de pré-processamento começa com a tokenização e a separação do texto em sentenças. Os tokens serão manualmente classificados pelos integrantes para que a rede neural utilize esta base de dados classificados no seu treinamento.

Após os dados estarem processados, pode-se iniciar a fase de treinamento da rede neural para classificação. Será usado um modelo de rede LSTM, pois apresenta bons resultados em diversas tarefas de PLN. A técnica de treinamento supervisionado, onde os dados rotulados são fornecidos ao modelo para ele aprender as classificações

Figura 3 – Diagrama da fase de desenvolvimento do modelo



Fonte: os autores

por meio de exemplos, será usada no treinamento. O treinamento ocorre por diversos episódios ou *epochs* em que o *dataset* de treinamento é transcorrido diversas vezes até ou número fixo ou até a acurácia do modelo estabilizar.

Finalizado o treinamento, será realizada a avaliação do modelo treinado usando outras dados, que não foram usados durante o treinamento. Essa etapa permite verificar se o modelo realmente aprendeu os padrões e é capaz de generalizar a classificação ou o aprendizado não foi efetivo e houve *overfitting* nos dados de treinamento. Essa é uma maneira de identificar se os resultados produzidos pelo sistema estão adequados.

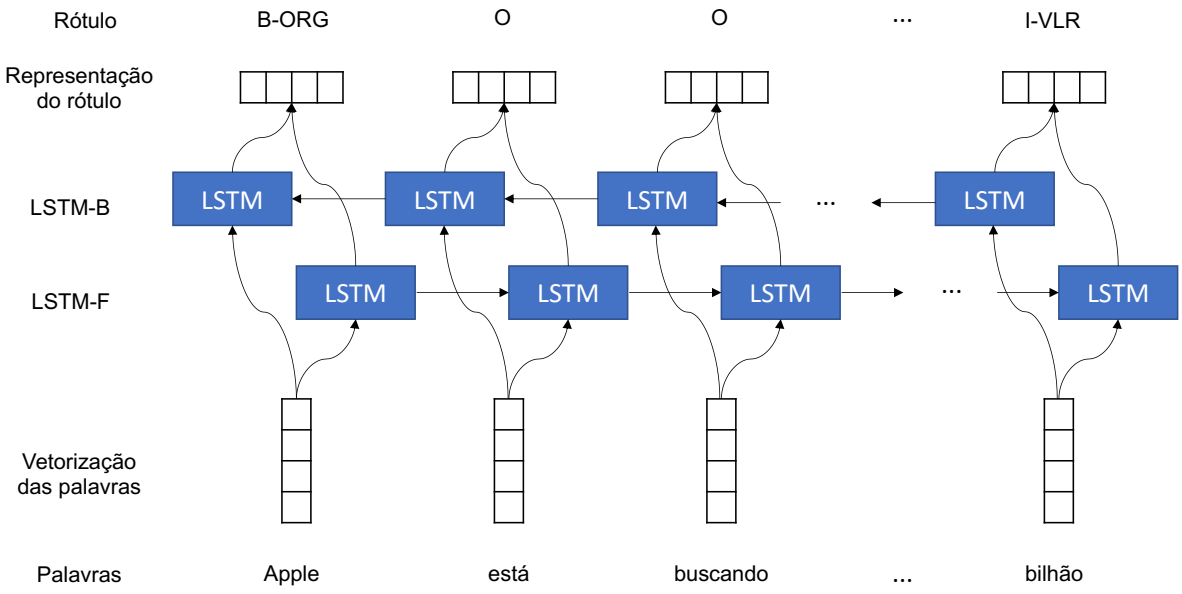
A separação dos dados deve ser feita em três partes: treinamento, teste e validação. Uma divisão comum na prática que será adotada é 80% dos dados para treinamento, 10% dos dados para teste e os 10% restante para validação.

O modelo de rede neural que será utilizado é a rede LSTM bidirecional, como mostrada na Figura 4. A entrada serão as representações vetoriais das palavras da frase a ser analisada e a saída os rótulos de cada palavra de acordo com a classificação IOB, codificados com a codificação *one-hot*. Existe um conjunto de células LSTM ligadas na ordem que as palavras aparecem (“para frente”), denominamos essas de LSTM-F, e existe outro conjunto de células ligadas na ordem inversa (“para trás”), essas são as chamadas LSTM-B. Isso forma uma camada de nós LSTM bidirecional ou BILSTM. O modelo que será implementado terá mais do que uma camada, mas não deve passar de uma ou poucas dezenas, pois modelos grandes causam treinamentos instáveis e lentos.

Também serão rodados experimentos com redes neurais GRU, que usam redes RNN e que só utilizam de um conjunto de células que estão ligadas na ordem de aparecimento das palavras, sem outra que percorre na ordem inversa, e que portanto apresenta treinamento mais rápido que redes LSTM bidirecionais, porém em geral apresentam pior acurácia se comparadas; e com embeddings pré-feitos e sua eficácia será comparada à da rede BILSTM.

Os resultados obtidos se mostraram de acordo com o esperado, com a rede neural BILSTM em geral apresentando o melhor desempenho entre os experimentos do grupo.

Figura 4 – Rede LSTM bidirecional para classificação de NER



Fonte: os autores

6 EXPERIMENTOS

A seguir são apresentados uma série de experimentos de Reconhecimento de Entidades Nomeadas para a língua portuguesa. Foram explorados diferentes conjuntos de dados e diferentes arquiteturas de modelos de redes neurais e parâmetros, com o objetivo de encontrar qual configuração produz o melhor resultado e compreender como treinar modelos de aprendizado de máquina para tarefas de NER na língua portuguesa. Os códigos e dados utilizados estão disponíveis para pesquisas futuras no Github <https://github.com/IanLaRosa/PCS3560_SEM_2020_S15>.

6.1 Ambiente de experimentação

Todos os experimentos desenvolvidos foram criados em um ambiente de desenvolvimento colaborativo e disponibilizado como serviço, que facilita a reprodução dos resultados obtidos. O ambiente utilizado foi o Google Colaboratory[1], onde é disponibilizado um acesso a uma máquina virtual preparada para executar tarefas de aprendizado de máquina com disponibilidade de acesso a GPUs para aceleração dos algoritmos paralelos. Essa solução provê uma facilidade na unificação do ambiente de execução e compartilhamento de resultados. Contudo existem limitações em relação ao tempo máximo de utilização da plataforma, perda de arquivos não baixados ao fechar a conexão e limites de hardware de disco e memória.

Foi utilizada a linguagem Python na versão 3.6 para a codificação dos experimentos e a biblioteca Tensorflow versão 2.3 para criação e treinamento dos modelos de aprendizado profundo.

6.2 Wikipédia

Os experimentos desenvolvidos nesta seção utilizam os dados textuais provenientes das páginas da Wikipédia[2] em português. O conjunto de dados anotados utilizado foi obtido de (NOTHMAN et al., 2013), de seu trabalho de NER para diversos idiomas. Essa base de dados é de padrão prata (*silver standard*), em oposição ao padrão ouro (*gold standard*). A diferença entre os dois é que os dados linguísticos padrão ouro são anotados manualmente, que requer muito esforço ou consumo de recursos e tempo, já dados padrão prata são anotados por algoritmos do estado da arte prévio ou obtidos por meio de heurísticas criadas pelos pesquisadores (FILANNINO; BARI, 2015).

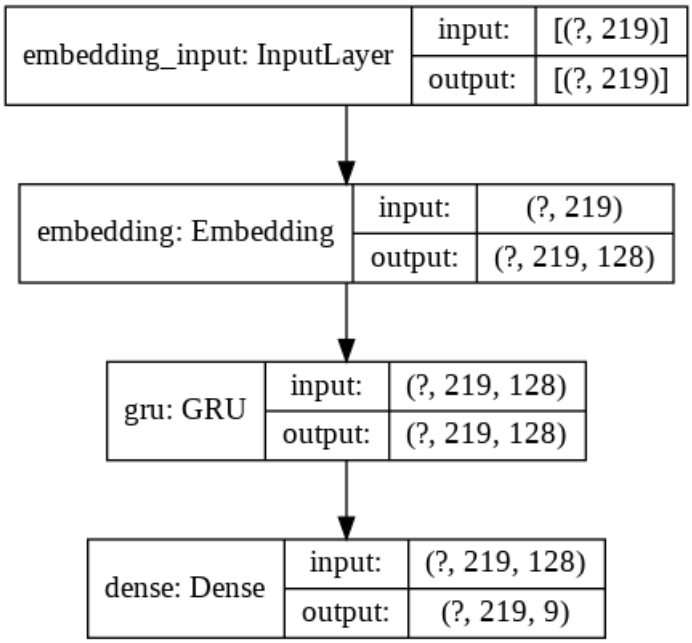
[1] <http://colab.research.google.com>
[2] <https://www.wikipedia.org>

6.2.1 Rede GRU simples

O primeiro teste que foi realizado é uma rede neural recorrente simples. O objetivo foi criar o modelo mais simples para o problema de NER, que pode ser compreendido de rotulagem de sequência, ou mapeamento de sequência para sequência (seq2seq) onde a sequência de entrada de palavras é traduzida em uma sequência de rótulos. Para esse problema, a formulação mais natural é utilizar redes neurais recorrentes, assim a escolha de uma rede *Gated Recurrent Unit* (GRU) (CHUNG et al., 2014) é natural, por ser um dos modelos de rede neural recorrente (RNN) mais simples e convergência mais rápida.

O *embedding* das palavras nesse experimento são aprendidos juntamente com o treinamento. A primeira camada da rede utilizada será uma camada de *embedding*, que tem a função transformar o índice do token de entrada, que representa uma palavra do vocabulário, em um vetor de representação dessa palavra. Em seguida temos a camada recorrente do tipo GRU, que produzirá a representação oculta interna da rede da sentença de entrada. Por fim uma camada densa para produzir a saída da rede, que realiza uma projeção do vetor interno sobre as possíveis categorias de entidades nomeadas. A Figura 5 mostra a arquitetura da rede usada nesse teste. Usamos 128 unidades nas camadas internas da rede, aumentando esse parâmetro permite a rede aprender mais padrões ou classificações mais complexas, contudo aumenta a quantidade de parâmetros da rede, que deixa o treinamento mais lento e requer mais memória.

Figura 5 – Modelo da rede neural usada neste teste



Fonte: os autores

A entrada da rede é um vetor de números inteiros, onde cada número representa uma das palavras do vocabulário. Para produzir esses vetores, foi criado o vocabulário, que é um conjunto com todas as palavras encontradas nos textos do *dataset*. A cada palavra é associado um número inteiro distinto. Nesse conjunto da Wikipédia utilizado, o vocabulário é composto por 116084 palavras únicas. Assim, uma frase nova é separada em palavras e cada palavra é substituída pelo seu número identificador. Realizamos *padding* nos vetores, adicionando zeros no final para que todos os vetores de entrada tenham o mesmo tamanho. O tamanho escolhido foi igual ao da maior sentença encontrada no *dataset*, que possui 219 tokens.

A saída da rede é uma sequência de 219 vetores (mesmo tamanho da entrada). Cada um desses vetores possui nove posições associadas a uma classificação de NER. Foi utilizada a codificação *one-hot*, onde o vetor possui tamanho igual o número de classes, no caso deste experimento possuímos 9 classes diferentes. A codificação *one-hot* transforma uma classe em um vetor com zeros em todas as posições exceto em uma posição associada a essa classe onde é inserido o valor de 1.

Por exemplo, se tivermos 4 possíveis classes $Y = \{c_1, c_2, c_3, c_4\}$ a codificação para a classe c_1 será $\mathbf{y_1} = [1, 0, 0, 0]$, para a classe c_2 será $\mathbf{y_2} = [0, 1, 0, 0]$, para c_3 será $\mathbf{y_3} = [0, 0, 1, 0]$ e para a classe c_4 será $\mathbf{y_4} = [0, 0, 0, 1]$. Esses vetores que serão usados para treinar os modelos realizando a comparação entre a saída produzida pela rede com esses vetores de codificação das classes. No experimento considerado temos 9 possíveis classificações, assim os vetores de saída y possuem 9 posições.

A vantagem da utilização dessa codificação para classificação é a facilidade de transformar as classes em vetores e vetores em classes (basta escolher a posição de maior valor para obter a classe). Além disso, aplicando a função *softmax* na saída da rede podemos interpretar o resultado como probabilidade ou confiança do modelo em alguma classificação.

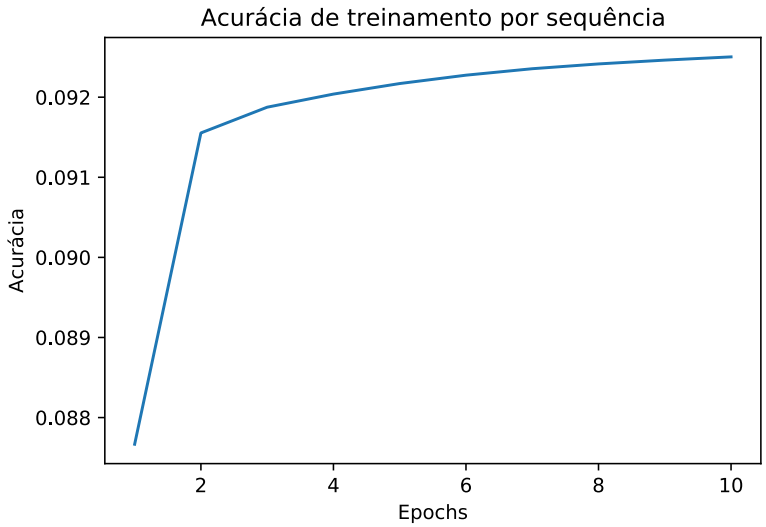
$$softmax(\mathbf{y}) = \frac{e^{y_i}}{\sum_{i=0}^{K} e^{y_i}} \tag{6.1}$$

onde $\mathbf{y}$ é o vetor de saída da rede, y_i é a posição i do vetor $\mathbf{y}$ e K é o tamanho do vetor.

No treinamento foi usada uma partição de 80% de todos os dados do *dataset*. O treinamento foi executado por 10 epochs, produzindo as métricas de acurácia da Figura 6 e erro da Figura 7. O otimizador utilizado foi o Adam e o cálculo do erro o *Categorical CrossEntropy*. Esses parâmetros foram escolhidos com base nos padrões encontrados na literatura e avaliações empíricas para a tarefa trabalhada. Em experimentos apresentados futuramente serão explorados outros parâmetros para o treinamento dos modelos.

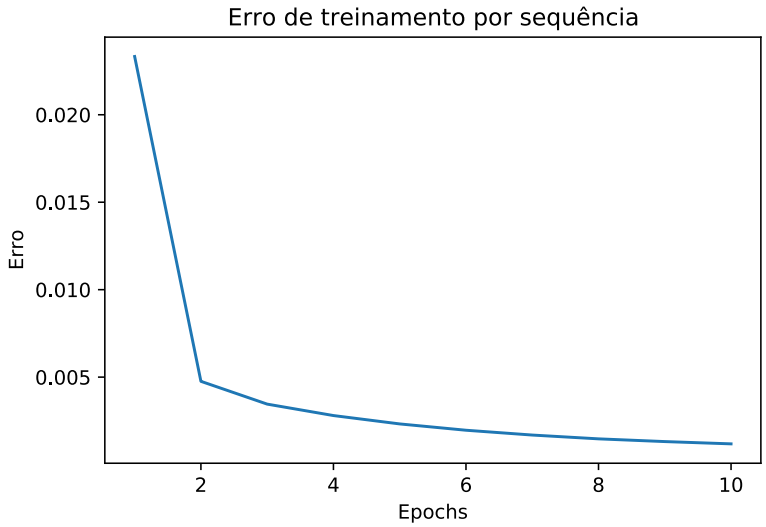
Para medir a eficácia do modelo é preciso calcular as métricas de avaliação em

Figura 6 – Gráfico da acurácia sobre o conjunto de treinamento



Fonte: os autores

Figura 7 – Gráfico do erro sobre o conjunto de treinamento



Fonte: os autores

um conjunto de dados diferente do conjunto no qual o modelo foi treinado, pois pode ocorrer a “memorização” dos dados e não haja uma verdadeira generalização. Esse conjunto é chamado de conjunto de validação, que é composto de 10% dos exemplos no *dataset*, que não foram usados durante o treinamento. Sobre esse conjunto obtemos acurácia de 9,14% por sentenças, isso é a proporção de sentenças onde todos os rótulos foram corretos. Já a taxa de acerto por token foi de 94,94%, indicando que o modelo aprendeu algo pois está acima da *baseline* simples obtida anteriormente.

6.3 Dados de Notícia

Os experimentos descritos a seguir utilizam um conjunto de dados coletados de notícias de jornal. O corpus foi criado pelos autores, desde a coleta dos textos até a anotação das entidades. Essa é uma das contribuições deste trabalho, disponibilizando mais um corpus para a língua portuguesa que poderá ser usado em pesquisas futuras.

As notícias utilizadas no corpus foram coletadas de publicações abertas nos sites do Estadão de São Paulo[3] e Folha de São Paulo[4], no período entre julho e outubro de 2020. Das notícias coletadas foi extraído um conjunto de dados contendo 304.743 palavras que será usado nos experimentos descritos a seguir.

6.3.1 Coleta de dados e preparação do corpus

Dispondo dos textos coletados de artigos de noticiário, é realizado um passo prévio de preparação para que sejam anotados manualmente. Primeiramente são eliminados quaisquer segmentos de texto que se encontram entre parênteses ou colchetes, e os próprios parênteses e colchetes. Em seguida são retirados alguns caracteres especiais: ' "' , ' ' ' , ' ! ' , ' ? ' , ' : ' e ' * '. Por fim o texto é separado em uma lista com cada elemento que o compõe (palavras e pontuações restantes) e são identificadas e numeradas suas frases. É considerada uma frase normalmente qualquer porção de texto delimitada por ponto final '.' .

Uma vez preparado o corpus se inicia a anotação manual, onde cada elemento do texto é avaliado em uma das 9 diferentes categorias de entidade consideradas neste projeto, e onde são realizadas quaisquer correções adicionais à estrutura resultante da preparação dos dados. As 9 categorias consideradas são as seguintes:

- 1 Localidade : entidades que nomeiam países, cidades, estados, regiões geográficas, ou quaisquer outras localidades espaciais definidas
- 2 Pessoa : entidades que nomeiam seres humanos, fictícios ou não
- 3 Organização : entidades que nomeiam grupos de pessoas ou atores estruturados, empresas, conselhos, assembleias, entre outros
- 4 Miscelânea : entidades nomeadas que não se representam em nenhuma das outras categorias propostas
- 5 Unidade monetária : entidades que representam quantias monetárias
- 6 Percentual : entidades que representam frações percentuais

[3] <https://www.estadao.com.br>
[4] <https://www.folha.uol.com.br>

- 7 Data : entidades que delimitam um ponto ou período especifico cronológico que ocorreu ou virá a ocorrer
- 8 Tempo : entidades que delimitam um quantidade genérica de tempo
- 0 Entidade não-nomeada : todas as palavras e elementos do texto que não são próprios de nomeação ou categorização.

6.3.2 *Baseline*

Como mencionado anteriormente, é importante estabelecer um *baseline* para comparação com as soluções de *machine learning*. Esse *baseline* deve ser calculado para cada conjunto de dados, não é correto utilizar *baseline* de diferentes *datasets* para avaliar um determinado modelo, pois ela muda em função da natureza dos dados textuais. A *baseline* calculada para o conjunto de dados de notícias produzidas foi igual a 88,43%, calculada de acordo com a Equação 4.1.

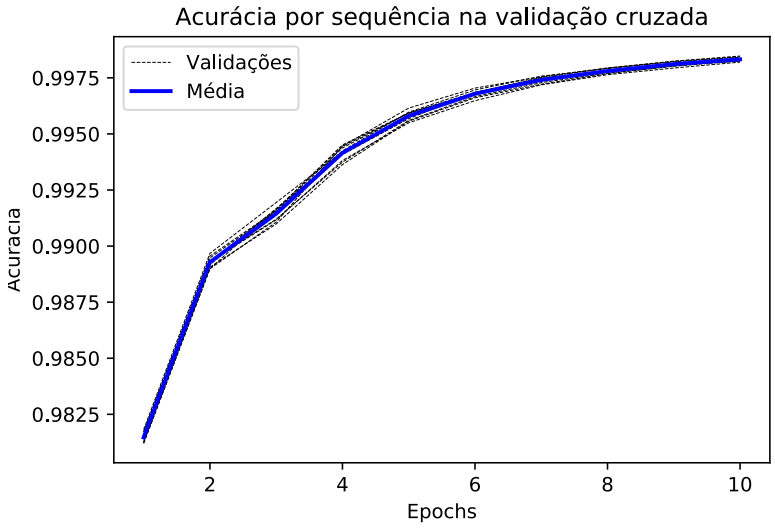
6.3.3 Rede GRU com validação cruzada

O primeiro teste realizado nesse conjunto de dados é com mesmo modelo no experimento descrito na subseção 6.2.1. As diferenças neste caso são o tamanho do vocabulário e o tamanho máximo das sentenças. Neste experimento o vocabulário é composto por 17792 palavras únicas e a maior sentença possível é de 246 tokens.

Foi adicionada a validação cruzada nesse experimento para diminuir a variabilidade entre ensaios e aumentar a confiança nos resultados obtidos. A validação cruzada k-fold empregada divide o conjunto de treinamento em k pastas e em cada passo é deixada de fora uma das pastas para validação e o modelo é treinado nas demais pastas. Embora a escolha do melhor k seja complicada, foi utilizado o valor $k = 10$, que é um valor comum onde existe um balanço entre a confiança do resultado e o tempo de execução do experimento.

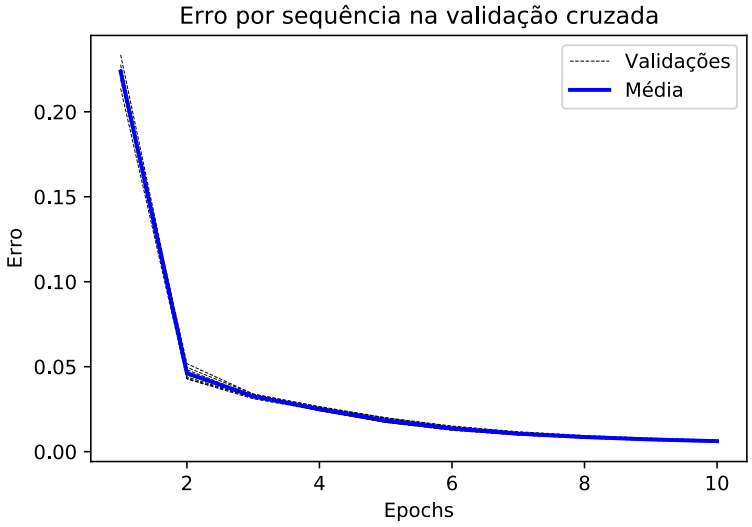
O treinamento feito com 80% dos dados, por 10 epochs e com validação cruzada produziu as métricas de acurácia da Figura 8 e erro da Figura 9. O otimizador utilizado foi o Adam e o cálculo do erro *Categorical CrossEntropy*. Com a validação cruzada o treinamento é repetido $k = 10$ vezes, cada um desses ensaios podem ser observados nos gráficos como as linhas pontilhadas pretas. Embora todas sigam o mesmo padrão existe algumas variabilidades, assim realizar a vadiação cruzada e calcular a média se aproxima do resultado esperado do modelo, mostrado no gráfico pela linha azul. Observando experimentos individuais é possível obter resultados diferentes com os mesmos parâmetros, enquanto a média é menos sensível a essa variabilidade e representa melhor o resultado esperado por esse modelo.

Figura 8 – Gráfico de acurácia sobre o conjunto de treinamento com validação cruzada do *dataset* de 300 mil palavras



Fonte: os autores

Figura 9 – Gráfico de erro sobre o conjunto de treinamento com validação cruzada do *dataset* de 300 mil palavras

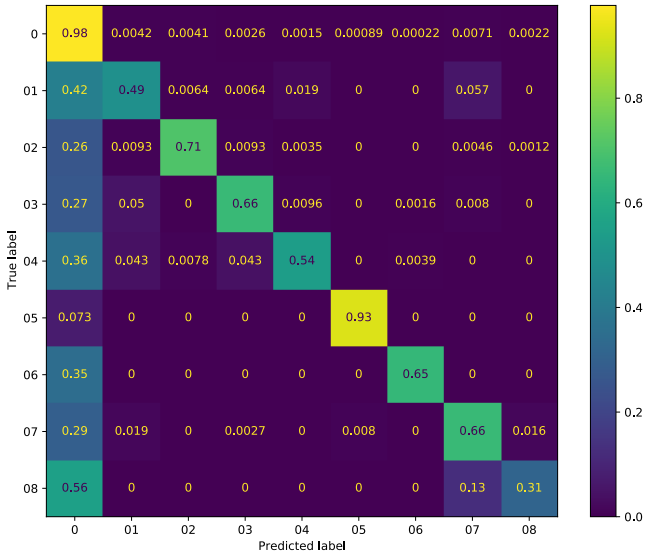


Fonte: os autores

Para cada uma das k divisões dos dados feita pela validação cruzada, foi calculado um valor de acurácia sobre o conjunto de validação. Assim, obtemos diversos valores cuja média produz uma aproximação mais realista da acurácia do modelo desenvolvido em um caso real. O resultado de acurácia na validação obtida foi de 93,20%, esse resultado é uma média simples dos 10 valores calculados em cada ensaio.

Uma maneira de analisar melhor o desempenho do modelo é observar os acertos e erros de cada classe. Uma matriz de confusão permite obter esse tipo de informação, onde se pode observar como estão sendo distribuídas as classificações para cada classe correta. Esse experimento produziu a matriz de confusão da Figura 10 sobre o conjunto de validação. Uma matriz de confusão mostra como um modelo está produzindo classificações para cada uma das classificações corretas. Ela é uma matriz quadrada com o número de linhas e colunas igual ao número de classes, as linhas representam as classificações corretas de referência e as colunas as classificações feitas pelo modelo. Dessa maneira a posição (0,0) da matriz indica o número de exemplos da classe 0 que foram corretamente classificado, já a posição (0, 1) indica os elementos da classe 0 que foram classificados de forma errada como classe 1. Assim, a melhor possível configuração são todos os elementos concentrados na diagonal principal e as demais entradas zeradas, isso indicaria que todos os exemplos foram corretamente classificados.

Figura 10 – Matriz de confusão de classificação sobre o conjunto de validação do *dataset* de 300 mil palavras



Fonte: os autores

A matriz de confusão apresentada foi balanceada para cada classe, de forma que a soma sobre uma linha seja igual a 1 e os valores das células sejam porcentagens de classificações. Isso foi realizado pois como os exemplos não são bem distribuídos, a classe 0 é muito mais frequente que as demais, apresentar os valores absolutos não mostraria o comportamento correto pelas demais partes.

Observando a matriz de confusão para o modelo treinado notamos que a

diagonal principal possui valores altos, indicando que as classificações realizadas estão corretas. Nota-se que o modelo está tendendo a prever muitos labels nulos para todas as classes, indicado pela coluna 0 que também está bem populada, essa tendência de tender para a classe mais frequente é comum em modelos de aprendizado de máquina, intuitivamente se houver alguma dúvida na classificação a maior chance de acerto sem nenhuma informação é classificar como o classe mais frequente. Pode-se afirmar que as entidades de nenhuma entidade (0) e valor monetário (5) são as melhores classificadas pelo modelo, por terem maiores valores na diagonal principal nessa posição. Por outro lado a classificação de tempo (8) é a pior classificação pois o valor de 0,31 na diagonal principal é baixo e a confusão com a classe 0 é muito grande para as entidades de tempo.

Nos próximos experimentos poderá ser comparada a taxa de acerto por token com o *baseline* e com a obtida com esse modelo GRU simples. Também comparando as matrizes de confusão poderá ser observado em quais classes as classificações foram melhoradas ou pioradas e se a tendência de super classificar a classe nula continua ocorrente nos próximos expeirmentos.

6.3.4 Rede LSTM bidirecional

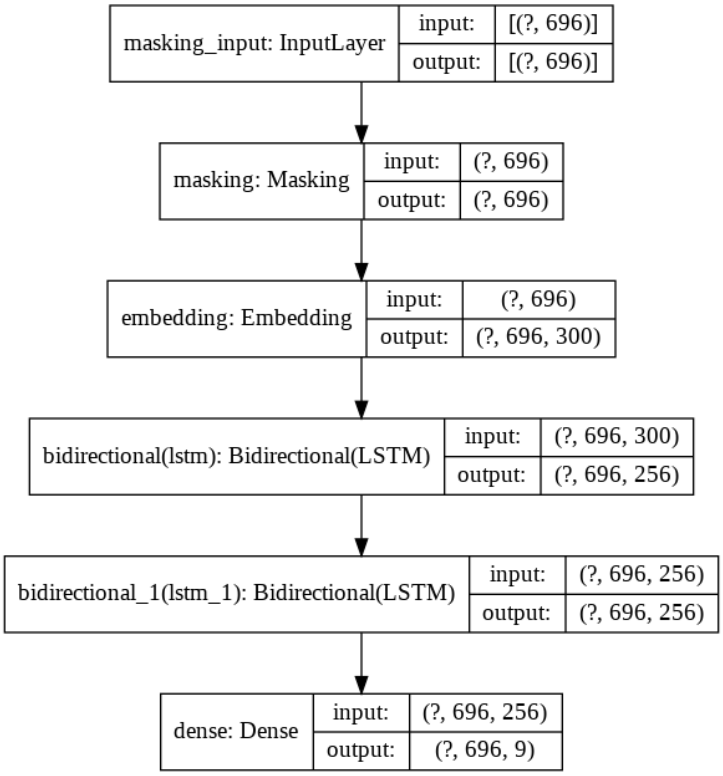
O próximo experimento realizado foi substituir o tipo de RNN usada nas camadas internas para a *Long Short-Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997). LSTM e GRU são os dois tipos de camadas recorrentes mais utilizadas na literatura. Enquanto a GRU utiliza menos parâmetros e é mais rapidamente treinada, a LSTM possui mais parâmetros e é mais lentamente treinada, mas para muitos casos, como em NER, os melhores resultados são obtidas como redes LSTM.

Anteriormente foi utilizada uma rede RNN unidirecional, isso significa que a sequência de entrada é sempre percorrida no sentido do tempo crescente, para sentenças isso significa na ordem natural de leitura da esquerda para direita. Dessa maneira a saída de um token depende dos tokens anteriores. As camadas internas deste experimento são bidirecionais, ou seja, para cada camada existe uma LSTM que percorre os tokens do começo ao fim e outra que percorre os tokens do fim em direção ao começo. Assim, a saída de um token depende tanto dos tokens anteriores quanto dos tokens posteriores. Ao final, a saída da camada é a concatenação das redes *forward* (do início ao fim) e *backward* (do fim ao início). Essa técnica apresenta bons resultados para PLN, pois é possível capturar melhor o contexto de uma palavra, tanto o contexto anterior quanto o posterior. Esse tipo de modelo é chamado de LSTM bidirecional, ou utilizando a sigla BILSTM (*BIdirectional LSMT*). Com isso é possível utilizar informações do contexto inteiro da frase para determinar a classificação de uma palavra.

As camadas de entrada, *embedding* e saída permaneceram iguais aos experi-

mentos anteriores. O modelo da rede é mostrado na Figura 11. Pode-se observar que foi utilizada uma segunda camada oculta, que apresentou um resultado pouco melhor que apenas com uma camada oculta. O estudos sobre a quantidade de camadas que melhor se adequa ao problema será explorado mais adiante.

Figura 11 – Modelo de rede neural BILSTM



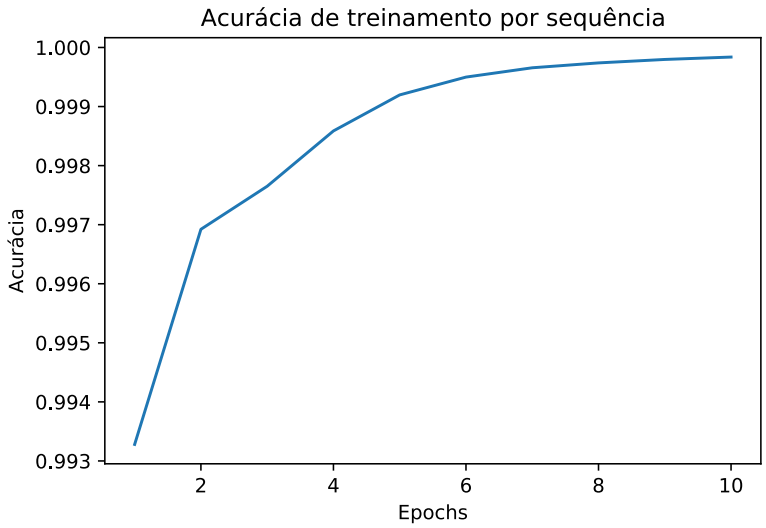
Fonte: os autores

Outra pequena diferença introduzida nesse experimento é o *dropout* nas camadas internas. Isso faz com que durante o treinamento algumas conexões da rede sejam forçadas para zero independentemente da entrada, a escolha de quais conexões é feita de maneira aleatória. Essa técnica é uma maneira de evitar que haja *overfitting* do modelo nas dados de treinamento, pois força com que não seja possível confiar em apenas um caminho interno para transmitir a informação já que ele pode ser aleatoriamente desativado por um passo no treinamento. Nesse experimento foi usado o valor de 0,2, que significa que existe uma chance de 20% de *droptout* de um ramo. Usualmente os valores de *dropout* utilizados na literatura variam de 0 a 50%, utilizar valores maiores dificultam o *overfitting* nos dados de treinamento, mas também torna a convergência menos estável e mais demorada.

O treinamento do modelo foi feito sem validação cruzada, pois o treinamento da BILSTM é mais demorado e por questões de agilidade esse experimento foi apenas feito com validação simples. Os gráficos de treinamento de acurácia e erro são mostrados

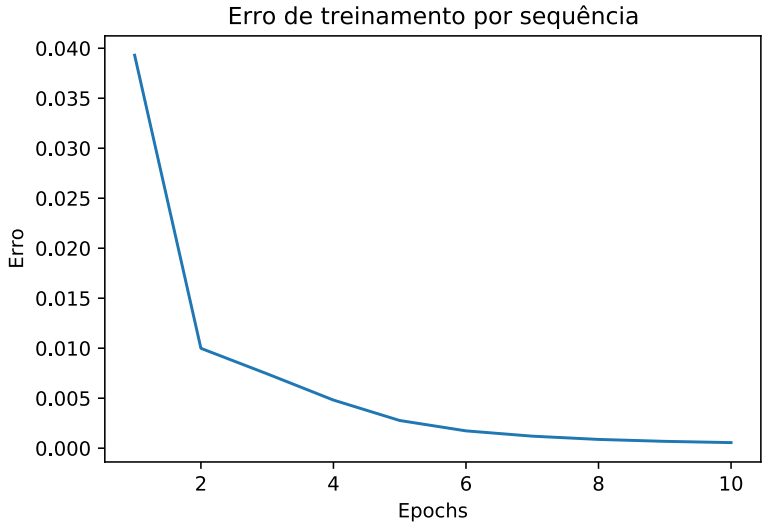
nas Figura 12 e Figura 13, respectivamente. Os outro parâmetros foram mantidos os mesmos, como 10 epochs, otimizador Adam e cálculo de erro *Categorical CrossEntropy*.

Figura 12 – Gráfico de acurácia sobre o conjunto de treinamento da rede BILSTM do *dataset* de 300 mil palavras



Fonte: os autores

Figura 13 – Gráfico de erro sobre o conjunto de treinamento da rede BILSTM do *dataset* de 300 mil palavras



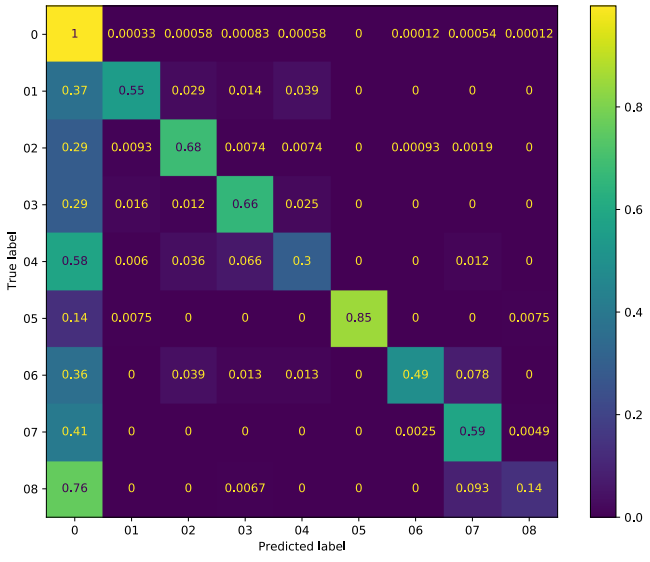
Fonte: os autores

Os gráficos de acurácia e erro obtidos neste experimento possuem formato semelhante aos obtidos em experimentos anteriores, contudo o ajuste aos dados de treinamento é maior do que obtido com a rede GRU, enquanto na Figura 8 na segunda *epoch* temos o valor próximo de 0,990, com a rede LSTM temos o valor de 0,997. O

mesmo pode ser observado nas curvas de erro, a curva de rede LSTM possui valores menores do que a GRU para todos *epochs*, indicando uma aproximação melhor dos dados. Isso é um indicador que este modelo de rede consegue se aproximar melhor das características presentes nos dados de treinamento, que é algo desejável desde que o *overfitting* seja evitado.

O resultado obtido na validação com esse modelo foi de 94,9% de taxa de acerto de tokens. Esse valor é maior do que o de 93,20% obtido no experimento anterior, confirmando que a rede BILSTM desenvolvida realizou uma melhor aproximação dos dados e foi capaz de generalizar para exemplos desconhecidos durante a validação quando comparada com o modelo anterior. Observando a matriz de confusão do modelo BILSTM mostrada na Figura 14, podemos observar que seu comportamento é bem semelhante ao resultado anterior da rede GRU, quanto a distribuição de elementos na diagonal principal e na primeira coluna. Nota-se que para algumas classes como miscelânea (4) e horário (8) existe uma grande quantidade de classificações erradas, por outro lado a classe 0 teve muito poucas classificações erradas, que contribuiu para o melhor resultado agregado de 94,9%.

Figura 14 – Matriz de confusão de classificação sobre o conjunto de validação da rede BILSTM do *dataset* de 300 mil palavras



Fonte: os autores

6.3.5 Embedding de palavras pré-treinados

Nos experimentos anteriores a representação vetorial das palavras foi aprendida juntamente com o treinamento do modelo na camada de *Embedding*. Nesse experi-

mento usamos *embeddings* de palavras em português já treinadas previamente. Foi utilizado um *embedding* Word2Vec (MIKOLOV et al., 2013) de 300 dimensões, ou seja cada palavra é representada como um vetor de 300 valores. Esse *embedding* utilizado é disponibilizado pelo Núcleo Interinstitucional de Linguística Computacional (NILC) (HARTMANN et al., 2017) [5].

Como estão sendo utilizadas vetorizações de palavras disponíveis, o vocabulário do conjunto de dados criado pode não ser compatível com as palavras para as quais sabemos a vetorização. Das 24912 palavras únicas que formam o vocabulário do *dataset* desenvolvido, 23306 palavras possuem vetores correspondentes nos *embeddings* disponíveis e 1608 não possuem vetores próprios. Nesses casos todas elas foram representadas de maneira igual, como o vetor nulo. Foi realizado um passo de normalização dos valores numéricos, de forma a garantir que sua representação vetorial fosse a mesma para todos os números, sem importar seu valor, pois o significado em uma frase de um numeral não depende do valor exato, mas sim do papel que o numeral possui no contexto.

A configuração da rede neural utilizada é igual àquela utilizado no experimento anterior mostrada na Figura 11. A principal diferença é que a camada de *Embedding* não será treinada, inicializamos sua configuração com os *embeddings* do NILC e não atualizaremos seus pesos internos durante o treinamento. Apenas atualizaremos os pesos das camadas BILSTM e densa seguintes. Os parâmetros de quantidade de neurônios nas camadas ocultas, *dropout*, função de ativação, otimizador e cálculo de erro permaneceram as mesmas, pois o objetivo desse experimento é avaliar o efeito do uso de *embeddings* prontos.

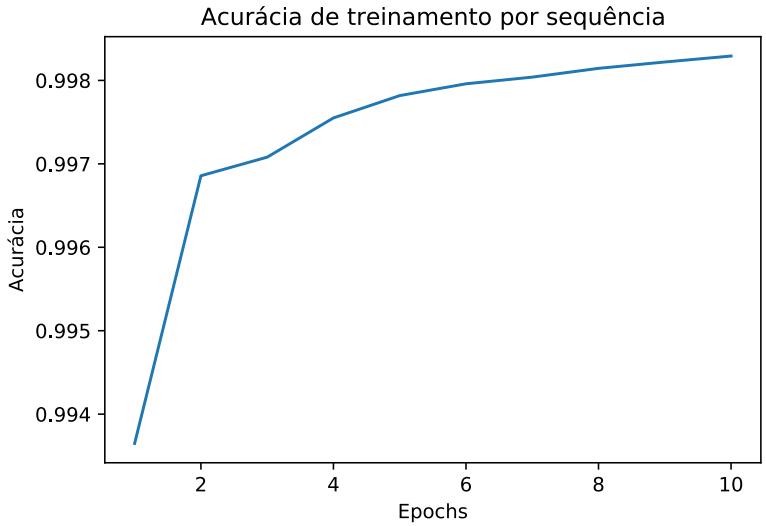
O treinamento foi feito da mesma forma que no experimento anterior por 10 *epochs*. Os gráficos de acurácia e erro são mostrados nas Figura 15 e Figura 16 respectivamente.

Os gráficos obtidos apresentam o comportamento esperado de aumento de acurácia e diminuição de custo, indicando a convergência dos pesos no processo de treinamento. Nota-se que a variação da primeira para a segunda epoch foi mais brusca neste experimento do que nos demais, fazendo com que os gráficos apresentem um vértice no ponto na segunda epoch. Isso pode ser justificado pelo uso dos *embeddings* pré-treinados já que como não são aprendidas as representações das palavras, o treinamento deve ficar mais fácil já que as vetorizações já possuem as informações semânticas das palavras desde o começo do treinamento e os *embeddings* não variam ao longo do tempo.

Na validação do modelo treinado foi obtido uma taxa de acerto do token de 93,9%. O resultado obtido com esses *embeddings* foi pior do que obtido aprendendo a

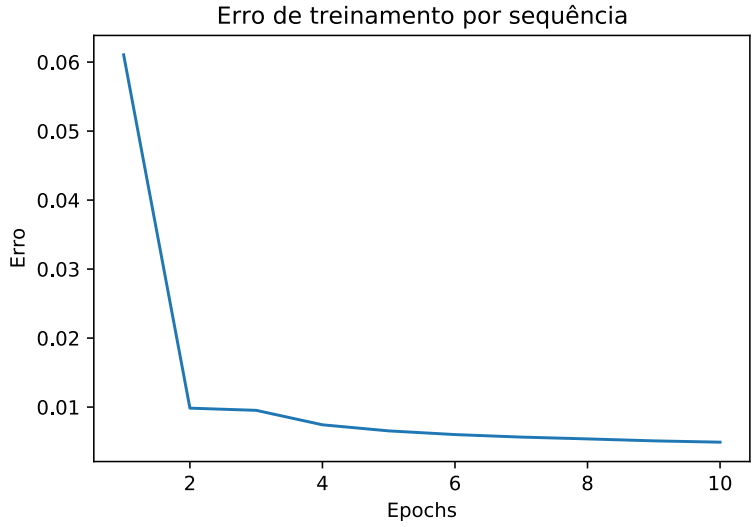
[5] O repositório de *word embeddings* está disponível em <http://nilc.icmc.usp.br/embeddings>

Figura 15 – Gráfico de acurácia sobre o conjunto de treinamento com *embeddings* pré-treinados do *dataset* de 300 mil palavras



Fonte: os autores

Figura 16 – Gráfico de erro sobre o conjunto de treinamento com *embeddings* pré-treinados do *dataset* de 300 mil palavras

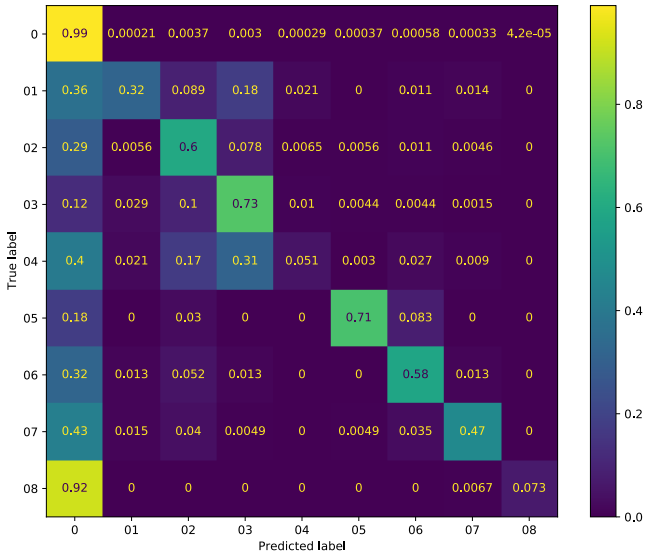


Fonte: os autores

representação das palavras juntamente com o treinamento do modelo. Isso também é confirmado comparando a matriz de confusão obtida nesse experimento mostrada na Figura 17 que possui elementos mais dispersos do que no experimento anterior. São observados grandes erros para as classes local (01), miscelânea (04) e tempo (08).

Uma possível justificativa para esse resultado é a diferença entre o vocabulário do *dataset* produzido para treinamento e o vocabulário no qual o *word emedding* foi

Figura 17 – Matriz de confusão de classificação sobre o conjunto de validação com *embeddings* pré-treinados do *dataset* de 300 mil palavras



Fonte: os autores

preparado. Embora também tenham sido utilizados textos de notícias na preparação do *embeddings* também foram usados textos de outras naturezas como de enciclopédias e obras literárias. Também, houve um número considerável de palavras (6,45%) que não tiveram vetorização disponível e tiveram de ser representadas com o vetor nulo. Como muitas das entidades nomeadas são nomes próprios é possível que justamente essas palavras que estamos mais interessados não possuam cobertura nos *embeddings* do NILC e componham grande parte das palavras sem vetorização.

6.3.6 Variação do número de camadas

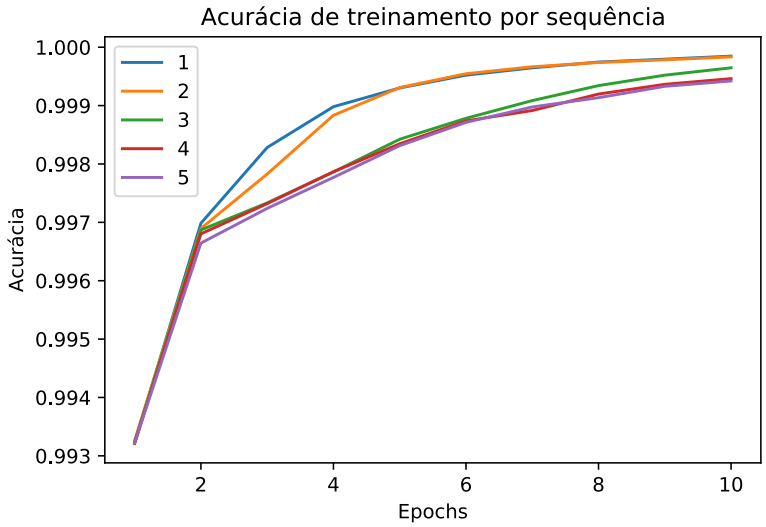
Nesse experimento foi estudado o efeito da quantidade de camadas ocultas no modelo no resultado obtido com o reconhecimento de entidades nomeadas. Para isso foi variada a quantidade de camadas do tipo BILSTM entre 1 e 5. Quanto maior o número de camadas, mais profunda a rede e mais difícil é seu treinamento, por conta do problema do *vanishing gradient* (GOLDBERG, 2016).

Esse é um problema bem conhecido na área de aprendizado profundo que é a diminuição da taxa de atualização para as camadas iniciais de uma rede profunda. Como as redes neurais são treinadas por meio do algoritmo de *backpropagation*, o gradiente do erro, usado para atualizar os pesos da rede, primeiro é aplicado a última camada, a seguir para a penúltima e assim por diante até a primeira camada. Contudo, como as funções de ativação costumam restringir as saídas das camadas internas, o

gradiente tende a diminuir as longo do processo de *backpropagation*, fazendo com que as atualizações sejam menores a cada camada da rede, dificultando o treinamento de redes profundas.

Os cinco modelos foram treinados com as mesmas configurações e sobre os mesmos dados de treinamento. As configurações de treinamento e hiper-parâmetros do modelo são os mesmos utilizados nos experimentos anteriores. As curvas de acurácia de treinamento dos modelos foram plotadas no mesmo gráfico na Figura 18 para facilitar a comparação. O mesmo foi feito para a curva de erro mostrada na Figura 19.

Figura 18 – Gráfico de acurácia sobre o conjunto de treinamento variando o número de camadas do *dataset* de 300 mil palavras

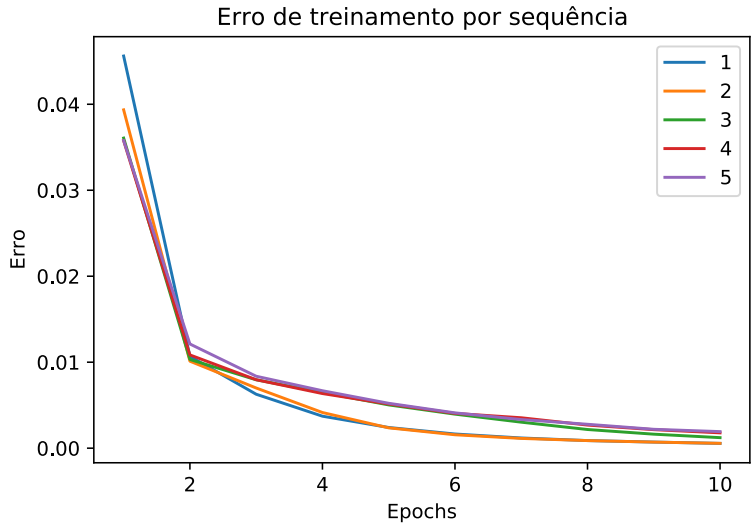


Fonte: os autores

Pode-se observar que os modelos com menos camadas convergem mais rapidamente, a curva de acurácia sobe mais rapidamente e a curva de erro decai mais rapidamente para os modelos com menos camadas. Esse é o comportamento esperado, pois redes com menos camadas são mais facilmente treinadas e convergem a estabilidade mais rapidamente, mas é possível que o melhor resultado no conjunto de validação não seja obtida por essas redes. Quando a tarefa é complexa e existem muitos dados disponíveis, é sabido por experiências práticas que aumentar o número de camadas nas redes produz melhores resultados.

Em uma aplicação que necessite de redes neurais para NER ou outra tarefa, o número de camadas deve ser escolhido em função dos recursos computacionais disponíveis, desempenho na tarefa específica de interesse e quantidade de dados disponíveis para treinamento. Nesse experimento coletamos a taxa de acerto por token na validação, para avaliar o desempenho, e tempo de treinamento de todos os modelos, como uma medida do custo computacional do treinamento dos modelos. Os resultados

Figura 19 – Gráfico de erro sobre o conjunto de treinamento variando o número de camadas do *dataset* de 300 mil palavras



Fonte: os autores

obtidos são apresentados na Tabela 2.

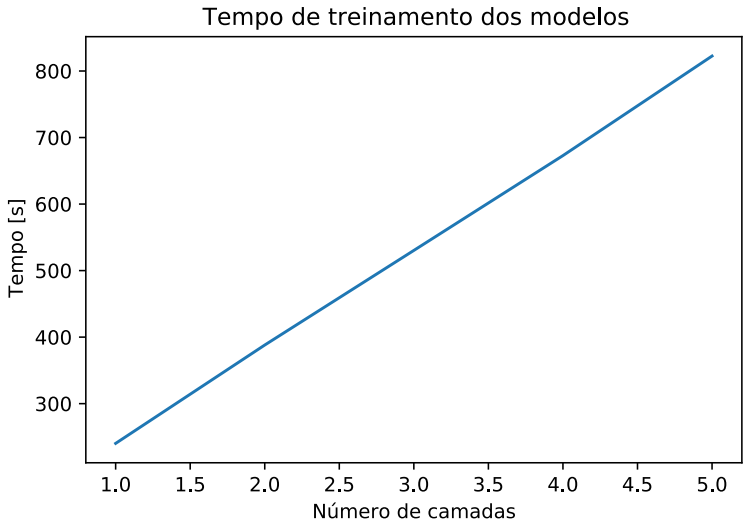
Tabela 2 – Resultados da variação da quantidade de camadas BILSTM do *dataset* de 300 mil palavras

Número de camadas	Taxa de acerto por token	Tempo de treinamento
1	94,87%	442 s
2	95,07%	644 s
3	94,44%	826 s
4	94,10%	1022 s
5	93,72%	1220 s

O modelo que obteve o melhor resultado nesse experimento foi a rede com duas camadas ocultas, que apresentou uma taxa de acerto por token de 95,07%. É esperado que exista um ponto ótimo em que aumentar o número de camadas não produz melhores resultados, mas isso é condicionado a quantidade de dados disponível, então caso sejam usados mais dados é possível que o ponto ótimo se altere.

O tempo de treinamento dos modelos aumenta de forma linear com o número de camadas, como mostrado na Figura 20. Para a configuração do treinamento utilizado e o número de dados o tempo gasto é baixo para todos os modelos, principalmente quando comparado com os valores reportados no estado da arte que são de horas a dias. Então, não deverá ser problema aumentar o número de camadas futuramente quando mais dados estiverem disponíveis.

Figura 20 – Gráfico do tempo de treinamento dos modelos do *dataset* de 300 mil palavras



Fonte: os autores

6.4 Dados de Notícia expandidos

Esta seção relata os experimentos com um *dataset* posterior ao utilizado para os experimentos iniciais retratados na seção anterior, com 612 mil tokens classificados, mais do que dobrando o tamanho da base anterior de 304 mil palavras. Este experimento foi feito como um meio de comparação inicial entre o *dataset* inicial e o final, com aproximadamente 1 milhão de palavras, possibilitando a visualização da influência do tamanho da base de dados no aprendizado da rede durante o treinamento. Embora não seja garantido, é fortemente esperado que a utilização de mais dados para o treinamento dos modelos produza melhores resultados, isso é um dos princípios fundamentais do aprendizado profundo, a melhoria contínua com o aumento do *dataset* enquanto modelos de inteligência artificial mais simples podem atingir um patamar onde o aumento do número de dados não se traduz em melhores resultados de predição.

6.4.1 *Baseline*

Embora o *dataset* utilizado nos experimentos apresentados a seguir seja uma ampliação do anterior, é importante re-estabelecer o *baseline* para cada alteração no conjunto de dados utilizado. Embora a natureza dos dados sejam a mesma, os dados usados para validação e treinamento são diferentes e podemos utilizar o *baseline* para determinar se a tarefa se tornou mais fácil ou difícil em função da mudança de distribuição das classes. Dito isso, o novo *baseline* para o *dataset* de notícias ampliados que será apresentado nos experimentos a seguir é de 89,74%. O *baseline* obtido para esse conjunto de dados foi muito semelhante ao *baseline* anterior conforme esperado,

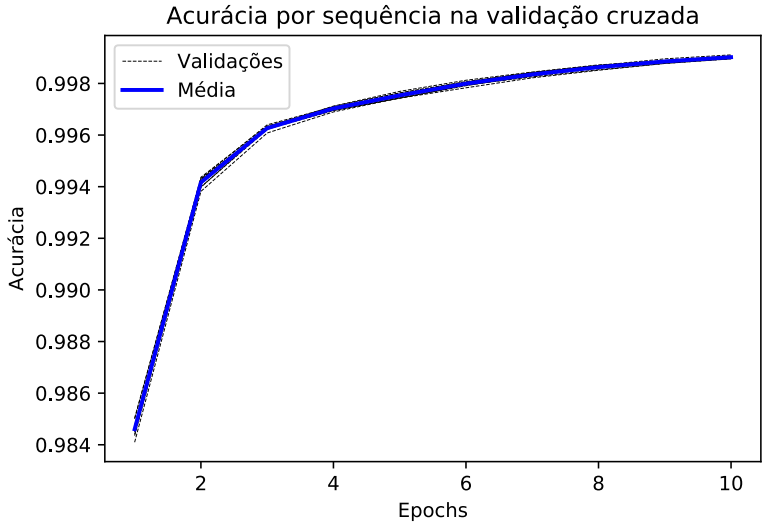
mas deve ser considerada as medidas obtidas no mesmo conjunto na avaliação dos modelos.

6.4.2 Rede GRU com validação cruzada

Repetiu-se o experimento feito com o *dataset* de aproximadamente 300 mil palavras com este novo *dataset*, com 612 mil. O treinamento GRU com o novo *dataset* apresentou 96,27% de acurácia na validação, que se comparado à taxa do experimento anterior com GRU de 93,20%, apresenta um aumento de 3,07% que pode ser atribuído à maior base de dados disponível para o processo de treinamento. Para facilitar a comparação com os experimentos realizados com outros *datasets*, não foram realizadas mudanças nos parâmetros da rede neural para este experimento.

As Figura 21 e Figura 22 mostram os gráficos de acurácia e erro de treinamento pelos epochs, e a Figura 23 mostra a matriz de confusão obtida com este treinamento durante a validação.

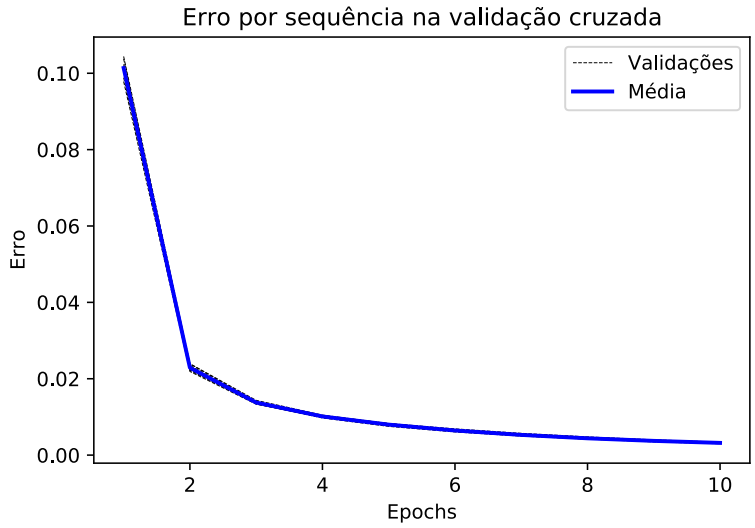
Figura 21 – Gráfico de acurácia sobre o conjunto de treinamento da rede GRU com validação cruzada do *dataset* de 600 mil palavras



Fonte: os autores

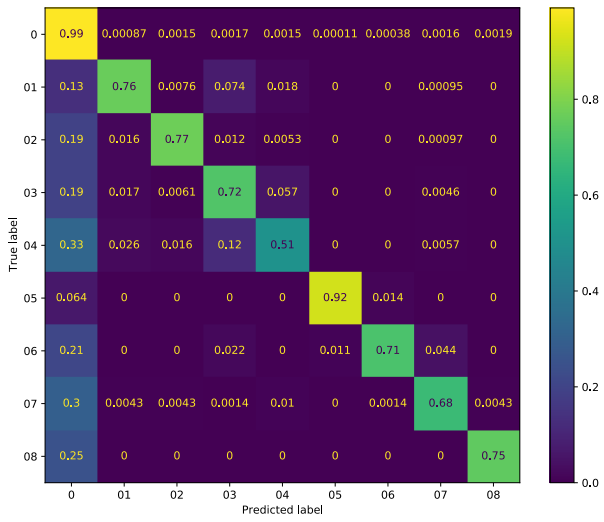
A matriz de confusão do novo modelo aponta grandes mudanças favoráveis na classificação de lugares, que foi de 49% para 76% se comparado ao modelo antigo, um salto de 27%, e na classificação de tempo, que foi de 31% para 75%, aumentando de 44%. As classificações de pessoas, organizações e porcentagens também tiveram flutuações positivas, coincidentemente todas de 6%, de 71% para 77%, de 66% para 72% e de 65% para 71%, respectivamente, enquanto miscelânea apresentou uma variação negativa de 3 pontos percentuais, de 54% para 51%. O restante das classificações apresentou apenas pequenas mudanças, como a perda de

Figura 22 – Gráfico de erro sobre o conjunto de treinamento da rede GRU com validação cruzada do *dataset* de 600 mil palavras



Fonte: os autores

Figura 23 – Matriz de confusão de classificação sobre o conjunto de validação da rede GRU com validação cruzada do *dataset* de 600 mil palavras



Fonte: os autores

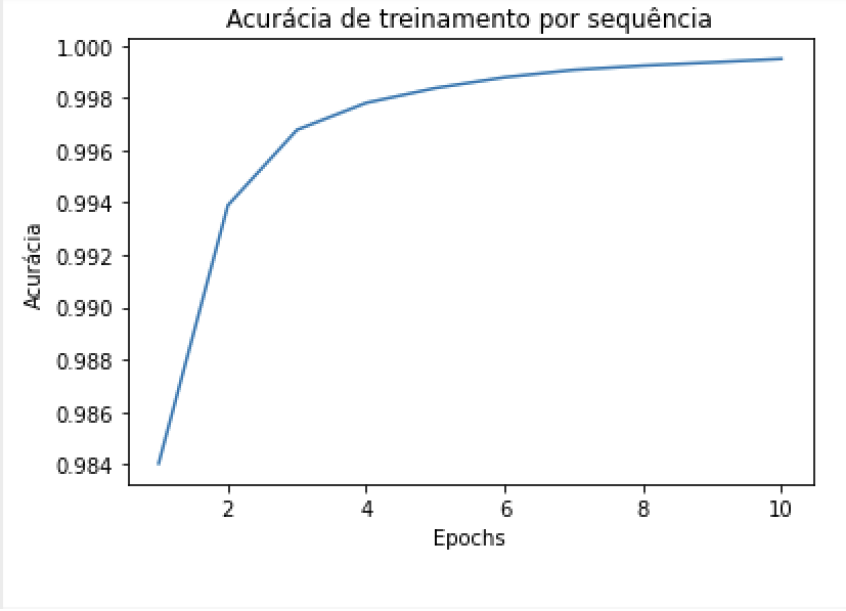
1%, de 93% para 92%, da classificação de valor monetário, ou o ganho de 2%, de 66% para 68%, que a classificação de data obteve.

6.4.3 Rede LSTM bidirecional

O experimento foi replicado com o novo *dataset* de 612 mil palavras para verificar novamente o melhor desempenho das redes LSTM bidirecionais na classificação de entidades nomeadas, obtendo desta vez uma taxa de acerto de 96,36%, superior a tanto o experimento rodado com GRU sobre o mesmo dataset, que apresentou acurácia de 96,27%, ainda que apenas marginalmente, e apresentando uma melhora mais acentuada se comparada à taxa de acerto apresentada pelo modelo treinado pela mesma rede porém com o dataset menor de aproximadamente 300 mil palavras, que foi de 94,90%, fazendo com que este novo treinamento tivesse um aumento de 1,46% que podem ser atribuídos ao aumento da base de dados utilizada.

Ao utilizar o novo *dataset*, não foram modificados nenhum parâmetro do modelo, a fim de criar uma comparação mais fácil com os outros resultados. Os gráficos de acurácia e de erro se encontram nas Figura 24 e Figura 25, e a matriz de confusão na Figura 26

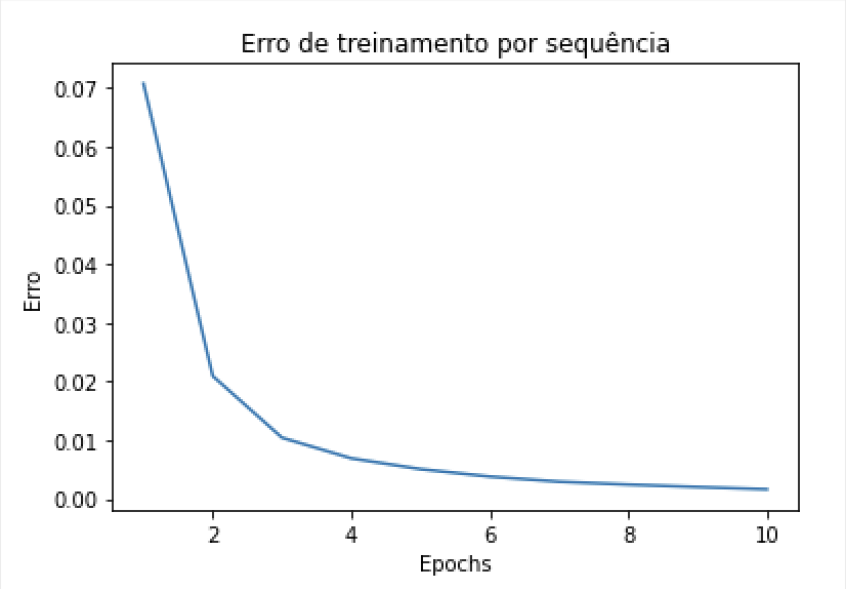
Figura 24 – Gráfico de acurácia sobre o conjunto de treinamento da rede BILSTM do *dataset* de 600 mil palavras



Fonte: os autores

Pela a matriz de confusão, é possível observar melhoras significativas na classificação de entidades de todas as categorias que o modelo busca classificar, com a classificação de lugares passando para 83% em comparação com o valor inicial de 55%, um aumento de 28%; pessoas passando a ser identificadas 80% das vezes se comparada com a taxa anterior de 68%, aumento de 12%; organizações passou de 66% para 73%, um dos menores aumentos na comparação entre os *datasets*, de 7%; miscelânea passou a ser capaz de classificar 61% comparado ao 30% da anterior,

Figura 25 – Gráfico de erro sobre o conjunto de treinamento da rede BILSTM do *dataset* de 600 mil palavras



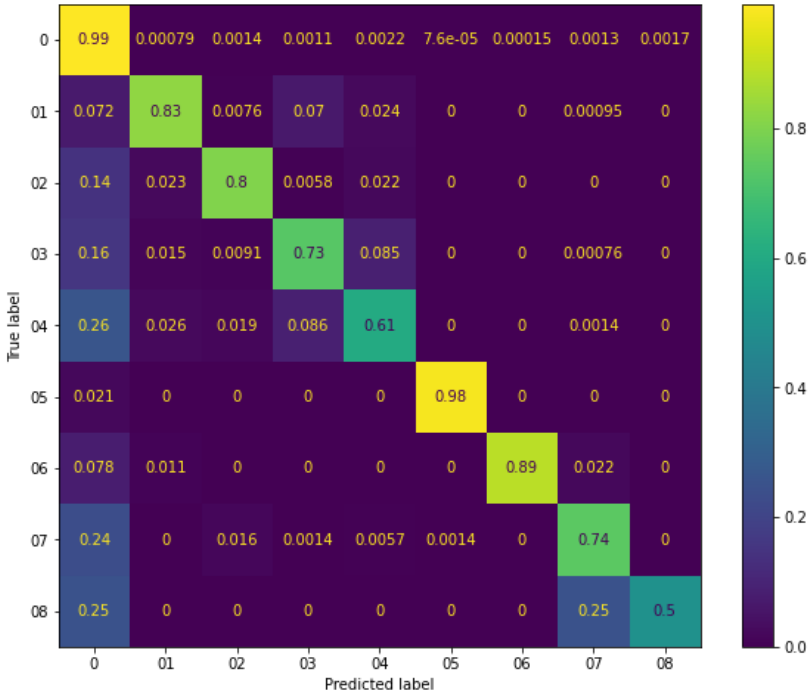
Fonte: os autores

com um salto de 31%; valor monetário observou ganho de 13% ao comparar o novo resultado de 98% ao anterior de 85%; porcentagens foram corretamente classificadas 89% das vezes com o novo *dataset* se comparadas com a taxa anterior de 49%, apresentando o maior ganho entre os dois experimentos, de 40%; data observou uma taxa de 74% que se comparada com a anterior de 54% mostra um crescimento de 20%; e tempo apresentou um ganho considerável de 36% comparando o inicial de 14% e o com o *dataset* expandido, atingindo 50%.

Ao se comparar com o treinamento através da rede neural GRU, relatado anteriormente, a maioria das categorias apresentam ganho quando treinadas pela rede LSTM bidirecional: Lugares apresenta ganho de 7% se comparado à taxa de 76% do GRU; pessoas ganha 3% se comparado à 77% do rede anterior; organizações apresenta um ganho de 1% na comparação com 72%; miscelânea apresenta um ganho mais expressivo de 10% se comparado ao 51% obtido pela GRU; valor monetário obteve ganho de 6% na comparação com o 92% obtido anteriormente; porcentagens teve o maior ganho, 18%, se comparado com 71%; e datas tiveram um ganho de 6% na comparação com o 68% obtido com uma rede GRU. Porém, a classe de tempo obteve um valor estranhamente alto na rede GRU que não foi replicado na rede BILSTM, resultando em uma queda do valor de 75% obtido anteriormente para um valor de 50%, perda de 25%.

Notou-se neste experimento, via a matriz de confusão proporcionada, que apesar do aumento sensível na detecção e classificação da tag 08, que representa as entidades de tempo, que era muito frequente a confusão desta com entidades pertencentes à tag

Figura 26 – Matriz de confusão de classificação sobre o conjunto de validação da rede BILSTM do *dataset* de 600 mil palavras



Fonte: os autores

07, que deve ser atribuída quando a entidade detectada trata-se de uma data. Como ambas tem linguagem e vocabulário comum entre si e semântica similar, decidiu-se realizar um experimento separado no qual as tags 7 e 8 eram consideradas idênticas, resultando na matriz de confusão mostrada na Figura 27

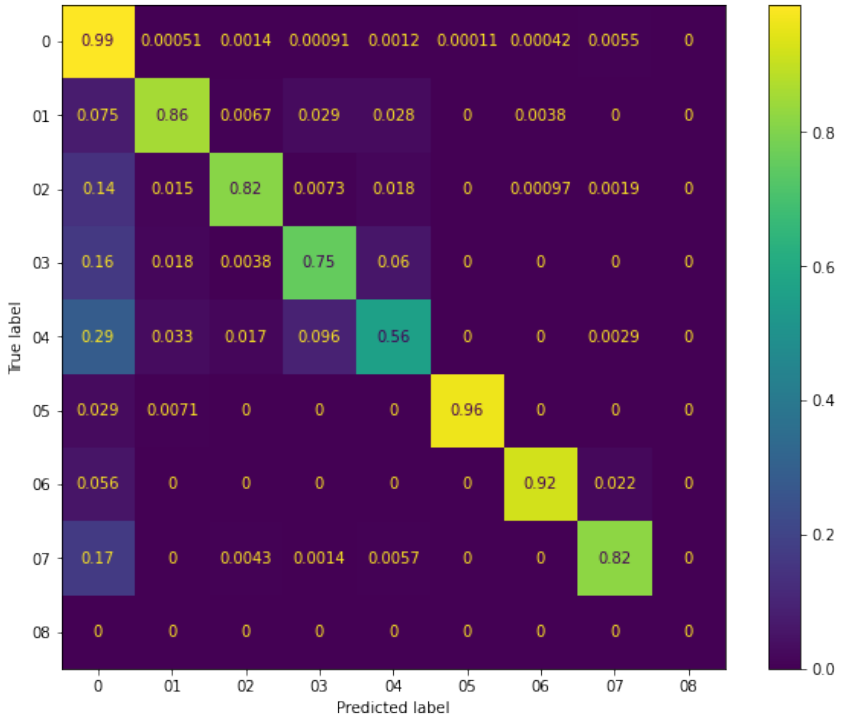
Como pode ser observado, isso causou pequenas flutuações na classificações dos outros tipos de entidades, entre 1% e 5%, mas garantiu um valor significativamente melhor dos tipos que a confusão entre o experimento tinha o objetivo de resolver, resultando em uma classificação com taxa de acurácia de 82%, se comparado com os valores anteriores de 74% e 50% para data e tempo, respectivamente.

6.4.4 Embeddings de palavras pré-treinadas

Novamente, os experimentos com *embeddings* pré-treinados também foram realizados para o *dataset* intermediário de 612 mil palavras, e seus resultados comparados com a versão realizada com o *dataset* anterior, de 304 mil palavras, bem como com o treinamento feito com a rede BILSTM para o mesmo *dataset*, a fim de averiguar tanto a eficácia da rede BILSTM para uma solução de NER e explicitar a importância de bases de dados maiores para soluções de PLN.

O modelo baseado nos *embeddings* pré-treinados apresentou uma melhora de 2,02% na taxa de acerto se comparado ao modelo baseado na base de dados anterior,

Figura 27 – Matriz de confusão de classificação sobre o conjunto de validação da rede BILSTM do *dataset* de 600 mil palavras considerando 7 e 8 como iguais



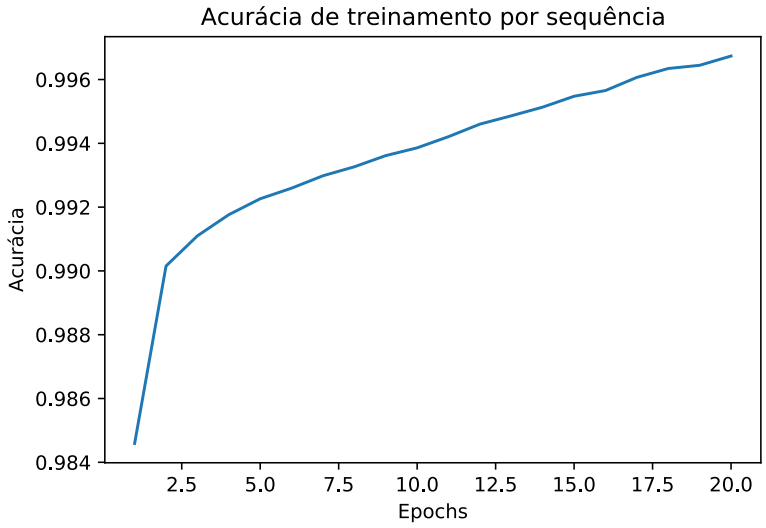
Fonte: os autores

atingindo 95,92% de acurácia nas suas classificações, se comparado com a taxa de 93,90% apresentada pelo modelo realizado anteriormente. Porém, os *embeddings* pré-treinados se mostraram inferiores ao treinamento com a rede BILSTM, que atingiu uma acurácia de 96,36% sobre a mesma base de dados.

Os gráficos de acurácia e do erro podem ser vistos nas Figura 28 e Figura 29, respectivamente, e a matriz de confusão resultante do treinamento pode ser visualizada na Figura 30.

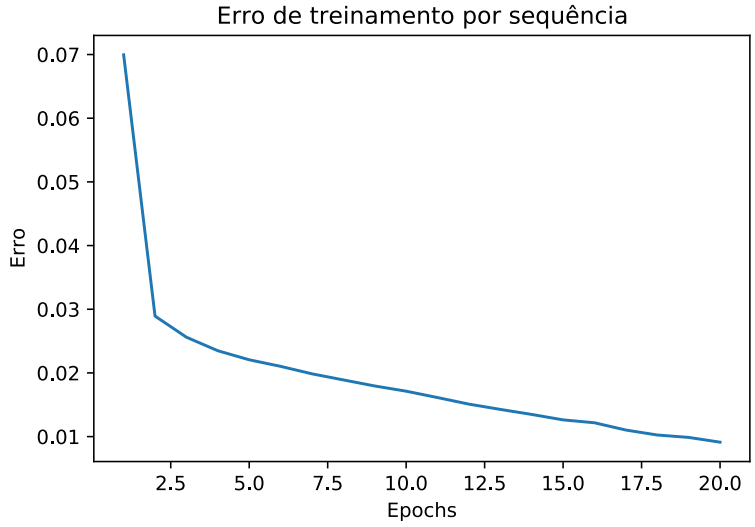
Como pode ser observado na matriz de confusão, a maioria das categorias obtiveram aumentos expressivos na sua taxa de acerto na classificação se comparados com o modelo baseado no *dataset* menor. Lugares teve um aumento de 27% se compararmos a taxa de 32% obtida com o menor *dataset* com a nova de 59%; pessoas cresceu de 60% para 83%, obtendo um crescimento de 23%; miscelânea saiu de apenas 5,1% para 43%, aumento de quase 38%; valor monetário passou de 71% ara 96%, crescendo 25%; porcentagens passaram a ser classificadas 82% das vezes, que se comparadas com a taxa anteior de 58% pode-se observar um crescimento de 24%; datas chegaram a uma taxa de acerto de 78%, crescimento de 31% na comparação com com a taxa anterior de 47%; e tempo obteve o maior aumento relativo, de quase 43% entre o 50% obtido pelo novo modelo e o 7,3% do anterior.

Figura 28 – Gráfico de acurácia sobre o conjunto com *embeddings* pré-treinados do *dataset* de 600 mil palavras



Fonte: os autores

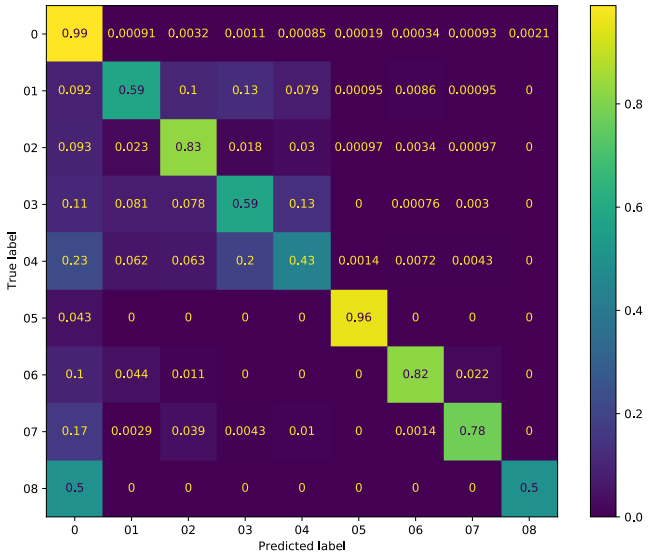
Figura 29 – Gráfico de erro sobre o conjunto com *embeddings* pré-treinados do *dataset* de 600 mil palavras



Fonte: os autores

Estranhamente, porém, a categoria que tinha obtido a maior taxa de acerto no modelo anterior, organizações, com 73%, acabou caindo consideravelmente neste novo modelo para 59%, por razões que não ficaram muito claras. Foi discutido que talvez seja a presença de organizações menos convencionais no dataset maior que acabou influenciando negativamente no modelo, porém não se chegou em uma conclusão definitiva.

Figura 30 – Matriz de confusão de classificação sobre o conjunto de validação com *embeddings* pré-treinados do *dataset* de 600 mil palavras



Fonte: os autores

6.4.5 Variação do número de camadas

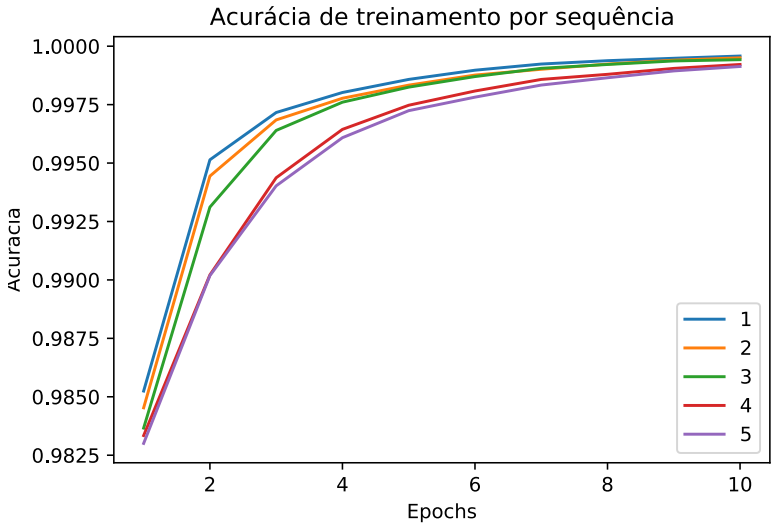
Os experimentos de variação no número de camadas também foram repetidos com o *dataset* expandido, e como nos outros não houve mudança nos demais parâmetros do treinamento do modelo para permitir uma comparação mais precisa entre os resultados.

Os gráficos que mostram a acurácia e o erro dos modelos de diferentes números de camadas, variando de 1 a 5, podem ser observados nas Figura 31 e Figura 32, respectivamente.

O experimento de variação de camadas teve como seu melhor resultado o modelo BILSTM com 2 camadas, que apresentou uma taxa de acerto de 96,88%, a maior entre os 5 modelos apresentados, como pode ser visto na Tabela 3, que apresenta os resultados gerais do experimento. Ainda que para tarefas complexas, seja esperado que os modelos mais profundos sejam mais eficazes, como o *dataset* ainda é relativamente pequeno se comparado ao corpus de um modelo mais avançado, como os existentes para a língua inglesa, é possível que com o crescimento da base de dados os modelos mais profundos começariam a demonstrar melhores resultados.

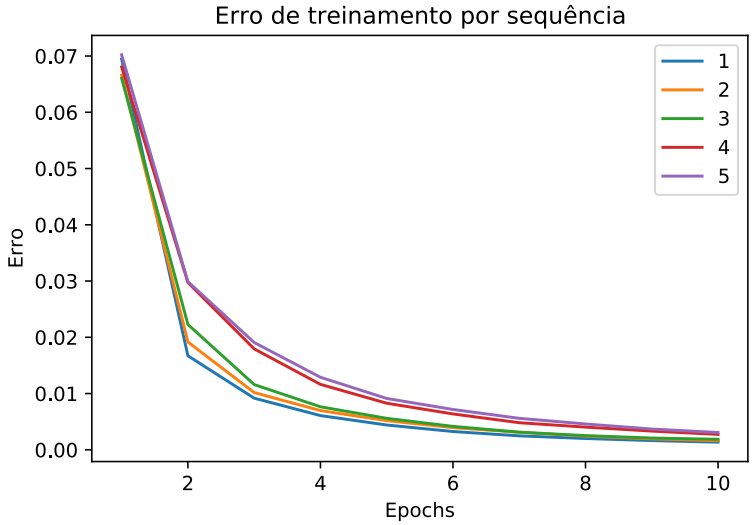
Também vale ressaltar que apesar do modelo de 2 camadas ter apresentado o melhor resultado, a diferença total da acurácia dos modelos é relativamente baixa, com o modelo de 2 camadas tendo uma taxa de acerto apenas 0,3% maior do que o pior

Figura 31 – Gráfico de acurácia sobre o conjunto de treinamento variando o número de camadas do *dataset* de 600 mil palavras



Fonte: os autores

Figura 32 – Gráfico de erro sobre o conjunto de treinamento variando o número de camadas do *dataset* de 600 mil palavras



Fonte: os autores

modelo deste experimento, o de 4 camadas, com 96,58%.

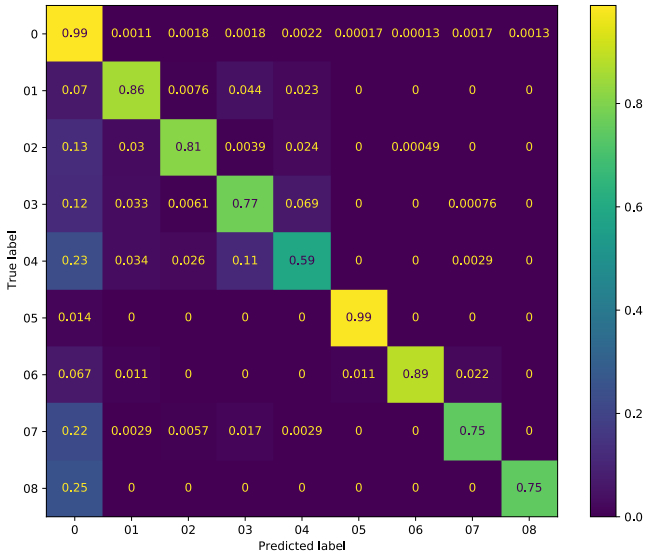
Abaixo, também pode-se visualizar a matriz de confusão do modelo que apresentou o melhor desempenho neste experimento na Figura 33.

Este modelo teve um bom aproveitamento geral, com quase todas as classes observando desempenho de 75% ou superior, com a exceção de miscelânea, que ficou com apenas 59%.

Tabela 3 – Resultados da variação da quantidade de camadas BILSTM do *dataset* de 600 mil palavras

Número de camadas	Taxa de acerto por token	Tempo de treinamento
1	96,84%	853 s
2	96,88%	1011 s
3	96,64%	1188 s
4	96,58%	1270 s
5	96,66%	1411 s

Figura 33 – Matriz de confusão de classificação sobre o conjunto de validação do modelo com 3 camadas ocultas do *dataset* de 600 mil palavras



Fonte: os autores

6.5 Dados de Notícia com expansão final

Esta seção repete os experimentos anteriores com um *dataset* de notícias anotadas ainda maior. Enquanto o conjunto anterior possuia 612 mil palavras, o novo conjunto expandido possui um total de 1.000.509 palavras.

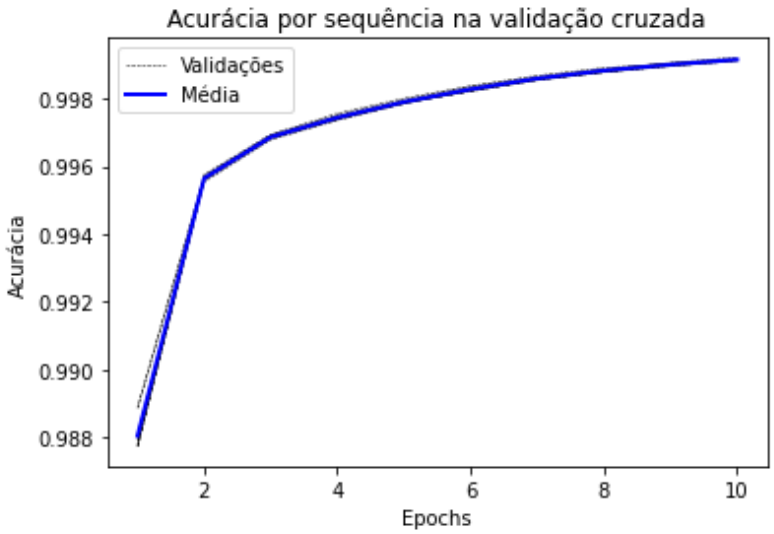
6.5.1 *Baseline*

Novamente, é importante re-estabelecer o *baseline* para cada alteração no conjunto de dados utilizado. O novo *baseline* para o *dataset* de notícias ampliado pela última vez que será apresentado nos experimentos a seguir foi de 88,77%. O *baseline* obtido para esse conjunto de dados foi muito semelhante aos *baselines* anteriores conforme esperado, mas deve ser considerada as medidas obtidas no mesmo conjunto na avaliação dos modelos.

6.5.2 Rede GRU com validação cruzada

O experimento da rede GRU foi repetido para o conjunto de dados expandido sem alterar nenhum dos parâmetros da rede para verificar a possibilidade de reutilizar o modelo apresentado quando existem mais dados disponíveis. O gráfico de acurácia no treinamento mostrado na Figura 34 e o de erro mostrado na Figura 35 indicam uma convergência mais rápida em número de epochs, para uma mesma epoch o modelo treinado com mais dados possui maior acurácia e menor erro. Dado que existem mais informação possíveis de serem aprendidas em uma epoch, pois existem mais dados, é esperado que para poucos epochs o modelo com mais dados disponíveis apresente um resultado melhor. Dessa maneira, é obtido melhores índices durante o treinamento para todos as epochs.

Figura 34 – Gráfico de acurácia sobre o conjunto de treinamento da rede GRU com validação cruzada do *dataset* de 1 milhão de palavras

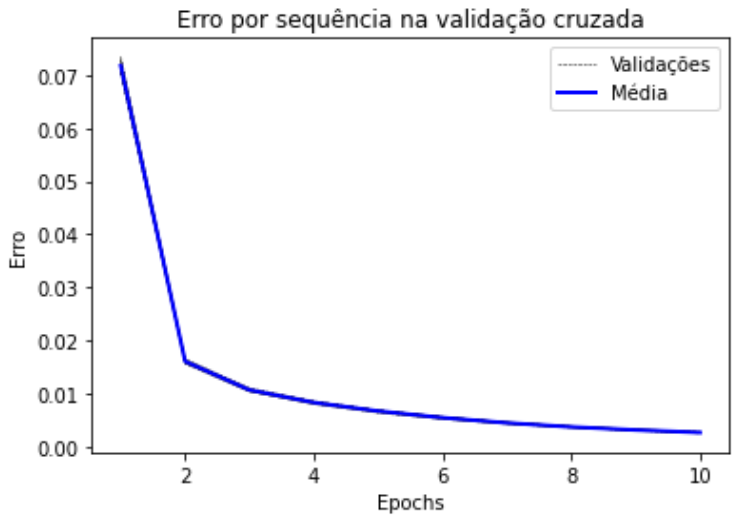


Fonte: os autores

Outra característica interessante observada é que as variações nas curvas da validação cruzada forem menores para o modelo com o maior *dataset*. Isso pode ser atribuído a maior instabilidade do aprendizado quando existem menos dados disponíveis, que provoca maiores variações na caminho percorrido durante o treinamento e maior variação dos resultados obtidos. Assim, a validação cruzada é um elemento mais importante para os *datasets* menores cujos modelos possuem maior variabilidade durante o treinamento e de resultados obtidos.

Essas indicações de melhores resultados foram confirmadas pela taxa de acerto por token obtida na validação de 96,42%. Portanto o modelo GRU apresentado pode aproveitar a maior quantidade de dados disponíveis para aprimorar seu resultado, uma vez que a taxa de acerto obtida de 96,42% foi maior do que a anterior de 96,27%,

Figura 35 – Gráfico de erro sobre o conjunto de treinamento da rede GRU com validação cruzada do *dataset* de 1 milhão de palavras



Fonte: os autores

sem uma variação apreciável do *baseline* que indicasse alteração na tarefa realizada, e apresenta uma diferença ainda mais apreciável com o dataset inicial de 304 mil palavras, que teve uma taxa de acerto de 93,20% com o mesmo tipo de treinamento.

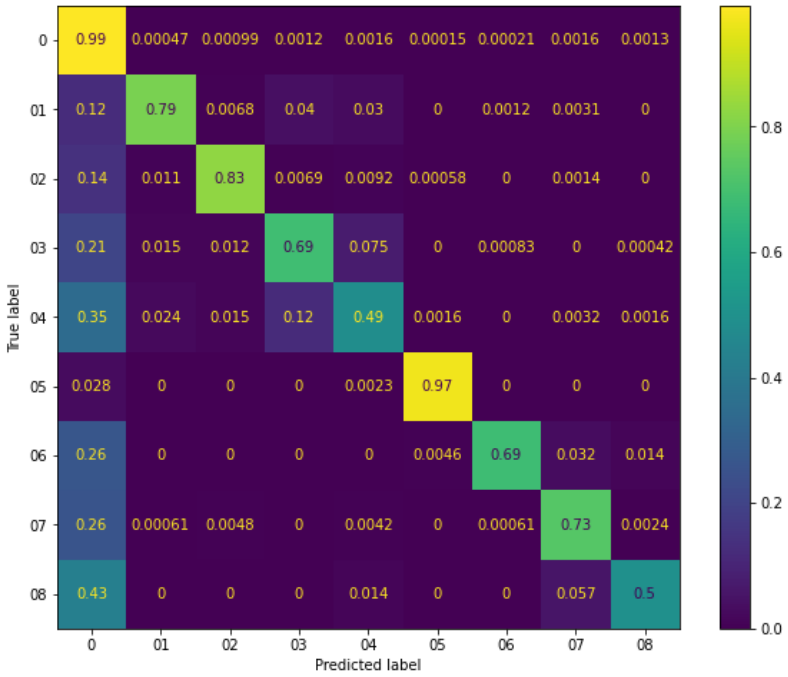
Observando a matriz de confusão desse modelo apresentada na Figura 36 é mostrado que a melhoria dos resultados é traduzida em maior acerto na classificação das entidades. Praticamente todas as classes consideradas tiveram melhoria em suas classificações, com exceção apenas da classe organização (03) que teve uma pequena baixa de 2%. Esse modelo apresentou ótimos resultados para a classificação de valores monetários (97%) e pessoas (83%).

6.5.3 Rede LSTM bidirecional

Da mesma forma que os experimentos anteriores, o experimento da rede BILSTM foi replicado para o *dataset* de 1 milhão de palavras para verificar se melhores resultados são obtidos com esse modelo caso mais dados estejam disponíveis. Para facilitar a comparação não foram alterados os parâmetros do modelo e treinamento. Os gráficos de acurácia é mostrado na Figura 37 e o de erro na Figura 38, observa-se o comportamento esperado de convergência mais rápida, quando comparado aos mesmos modelos com menos dados.

Foi obtida uma taxa de acerto na validação de 97,08% confirmando os melhores resultados para todos os modelos com conjuntos de dados maiores. Assim, pode-se afirmar que o modelo BILSTM apresentado pode aproveitar a maior quantidade disponíveis para melhorar seu aprendizado e obter classificações de entidades nomeadas

Figura 36 – Matriz de confusão de classificação sobre o conjunto de validação da rede GRU com validação cruzada do *dataset* de 1 milhão de palavras



Fonte: os autores

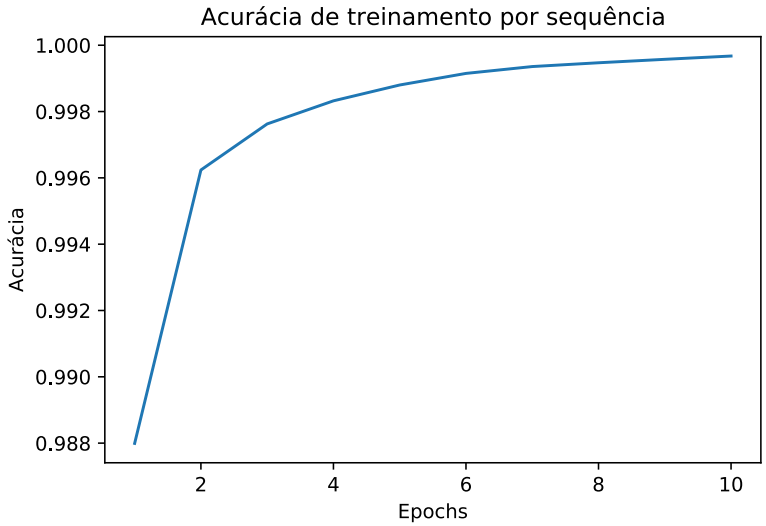
melhores.

A matriz de confusão para este experimento é mostrada na Figura 39. Com este experimento foi obtido uma ótima classificação para as classes 05 com 98% e 06 com 89%, mesmo a classe de miscelânea, que foi uma das mais difíceis em todos os modelos, obteve um dos melhores resultados encontrados no trabalho. Assim, esse modelo BILSTM com o *dataset* de um milhão de palavras foi uma das melhores configurações encontradas no trabalho para reconhecimentos de entidades nomeadas na língua portuguesa.

6.5.4 Embeddings de palavras pré-treinadas

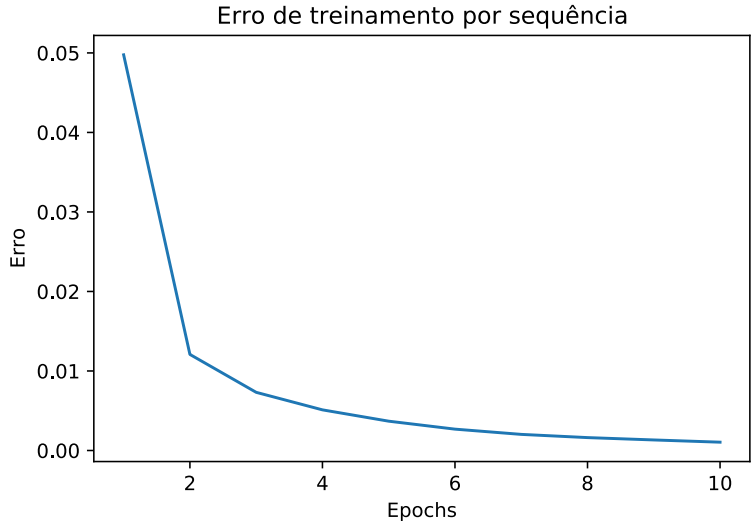
Mais uma vez o experimento foi repetido para o *dataset* com um milhão de palavras sem alterações nos parâmetros do modelo e treinamento. Como o conjunto de dados possui mais palavras, agora com 45.385 palavras únicas, é importante verificar a cobertura do vocabulário dos *embeddings* utilizados. Foram encontrados 41.076 palavras com vetorização disponível e 4.311 palavras cuja vetorização eram inexistentes e tiveram de ser representadas como o vetor nulo. Ou seja, 9,50% das palavras do vocabulário extraído das notícias não tinham vetorizações disponíveis, esse valor é maior do que o 6,4% obtidos com o menor *dataset*. Isso significa que conforme aumentamos o conjunto de dados uma proporção maior de palavras do vocabulário

Figura 37 – Gráfico de acurácia sobre o conjunto de treinamento da rede BILSTM do *dataset* de 1 milhão de palavras



Fonte: os autores

Figura 38 – Gráfico de erro sobre o conjunto de treinamento da rede BILSTM do *dataset* de 1 milhão de palavras



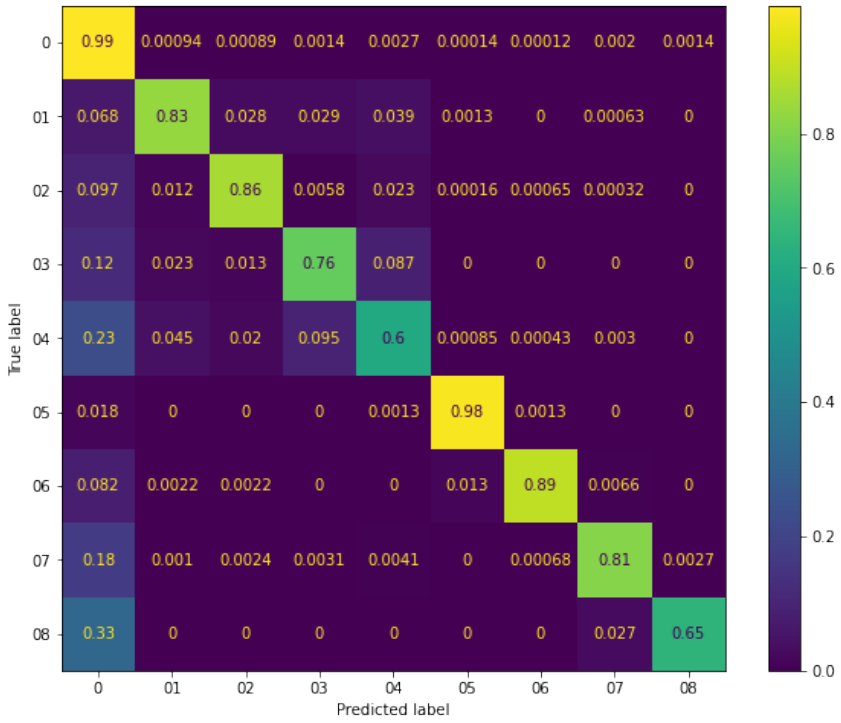
Fonte: os autores

não possuem representação própria.

Os gráficos de acurácia mostrado na Figura 40 e erro mostrado na Figura 41 revelam uma variação menos brusca na primeira epoch em comparação com o mesmo modelo com o menor *dataset*.

Durante a validação desse modelo foi obtida uma taxa de acerto de 96,41%, melhorando consideravelmente o resultado anterior, mas ainda com resultados inferi-

Figura 39 – Matriz de confusão de classificação sobre o conjunto de validação da rede BILSTM do *dataset* de 1 milhão de palavras



Fonte: os autores

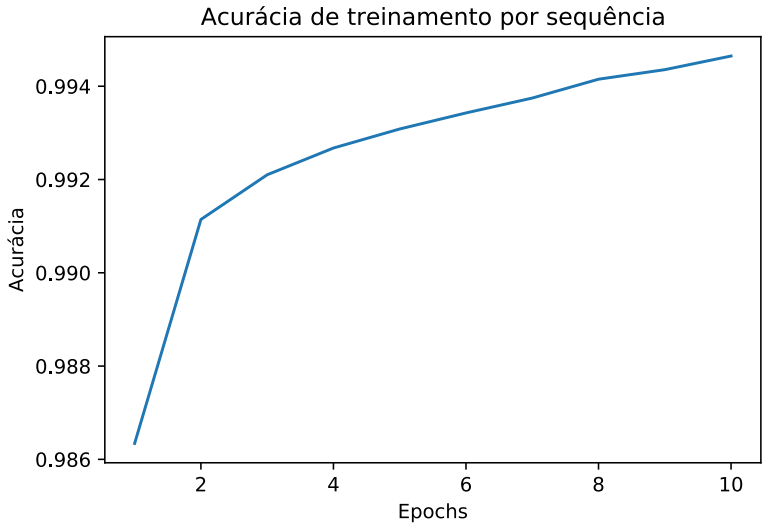
ores ao experimento que aprende o *embedding* durante o treinamento apresentados na subseção 6.5.3. Observando a matriz de confusão na Figura 42 notamos grandes melhorias em diversas categorias, principalmente na categoria de tempo (08) que atingiu uma acurácia de 89%. Mas existem algumas categorias como pessoas (01) e miscelânia (04) que possuem resultados abaixo dos demais modelos.

A partir desses resultados podemos afirmar que para uso efetivo de *embeddings* pré-treinados para reconhecimento de entidades nomeadas é preciso ter um corpus de tamanho considerável, pois pequenos conjuntos de textos não aproveitam bem o uso dos *embeddings* existentes. Como foram usados vetores de 300 dimensões para representar as palavras, é possível que nos experimentos anteriores não seja vantajoso o uso dos *embeddings* por existir muita informação nos vetores. É possível que utilizado *embeddings* menores seja mais fácil para os modelos compreenderem melhor as representações das palavras e com isso realizar classificações melhores.

6.5.5 Variação do número de camadas

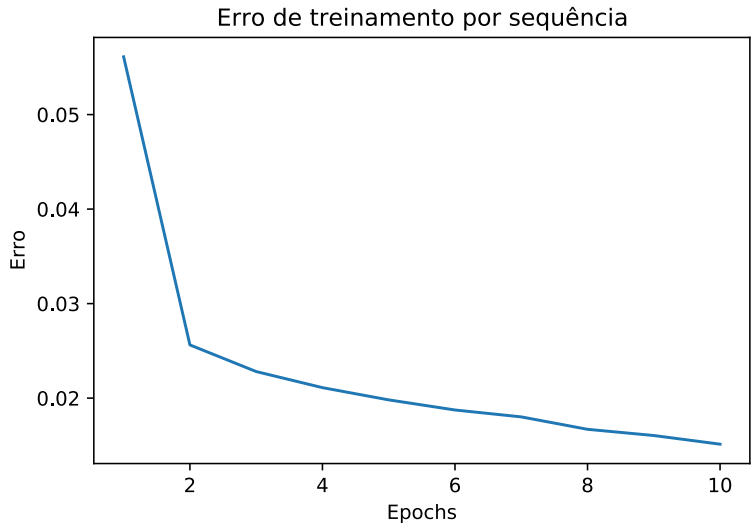
Para avaliar o efeito do maior *dataset* na quantidade de camadas ocultas da rede neural o experimento da variação do número de camadas foi repetido. Novamente não foram realizadas alterações nos demais parâmetros, a única variável é o tamanho do conjunto de dados disponível para treinamento. Os gráficos de acurácia e erro

Figura 40 – Gráfico de acurácia sobre o conjunto de treinamento com *embeddings* pré-treinados do *dataset* de 1 milhão de palavras



Fonte: os autores

Figura 41 – Gráfico de erro sobre o conjunto de treinamento com *embeddings* pré-treinados do *dataset* de 1 milhão de palavras

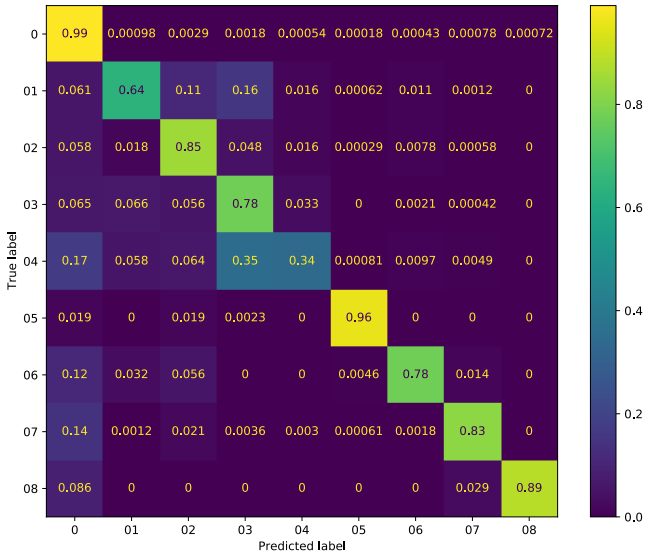


Fonte: os autores

obtidos são mostrados na Figura 43 e Figura 44 respectivamente.

O comportamento dos modelos com menos camadas terem a convergência mais rápida é mantida, como esperado pela maior facilidade de treinamento desses modelos. Algo interessante que foi observado é a maior separação entre as curvas dos modelos em ambos os gráficos. Esse é o comportamento esperado e, provavelmente, no experimento anterior a quantidade de dados disponível não era suficiente para

Figura 42 – Matriz de confusão de classificação sobre o conjunto de validação com *embeddings* pré-treinados do *dataset* de 1 milhão de palavras



Fonte: os autores

aflorar a diferença entre os modelos, principalmente na primeira epoch em que todos os modelos praticamente tiveram a mesmas taxas de acurácia e erro.

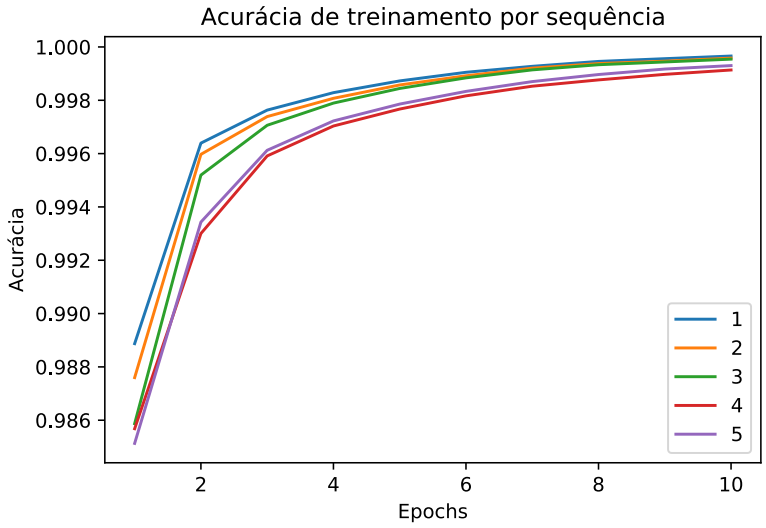
Os resultados obtidos na validação para este experimento são mostrados na Tabela 4. Nota-se melhores resultados na taxa de acerto por token em todos os modelos, conforme já notado pelos experimentos anteriores. O melhor modelo para o maior *dataset* foi o com 3 camadas ocultas que obteve 97,21%, diferentemente do experimento anterior onde o melhor modelo havia sido o de 2 camadas, mas neste experimento esses dois modelos tiveram resultados muito próximos com uma diferença de apenas 0,01%.

Tabela 4 – Resultados da variação da quantidade de camadas BILSTM do *dataset* de 1 milhão de palavras

Número de camadas	Taxa de acerto por token	Tempo de treinamento
1	97,13%	1724 s
2	97,20%	1973 s
3	97,21%	2129 s
4	96,59%	2374 s
5	97,02%	2367 s

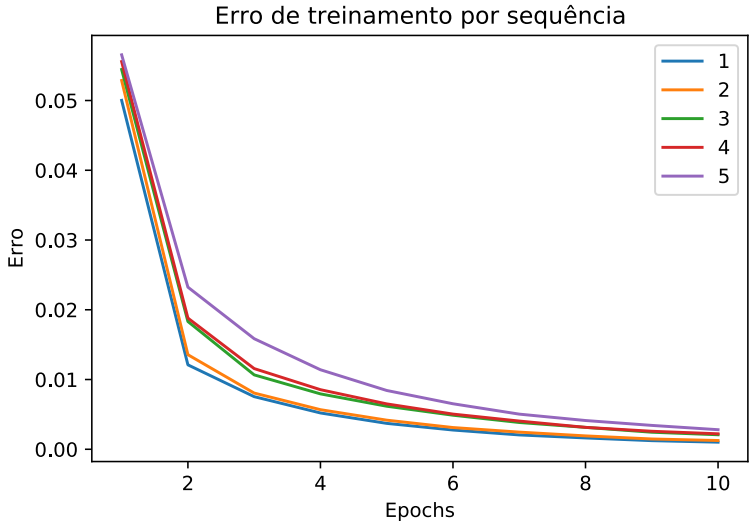
Contudo, algo que é numericamente confiável é o melhor desempenho de modelos mais profundos, que se aproximaram dos demais, sendo que o modelo com 5 camadas obteve a taxa de acerto por token de 97,02% ficando apenas 0,19% abaixo

Figura 43 – Gráfico de acurácia sobre o conjunto de treinamento variando o número de camadas do *dataset* de 1 milhão de palavras



Fonte: os autores

Figura 44 – Gráfico de erro sobre o conjunto de treinamento variando o número de camadas do *dataset* de 1 milhão de palavras



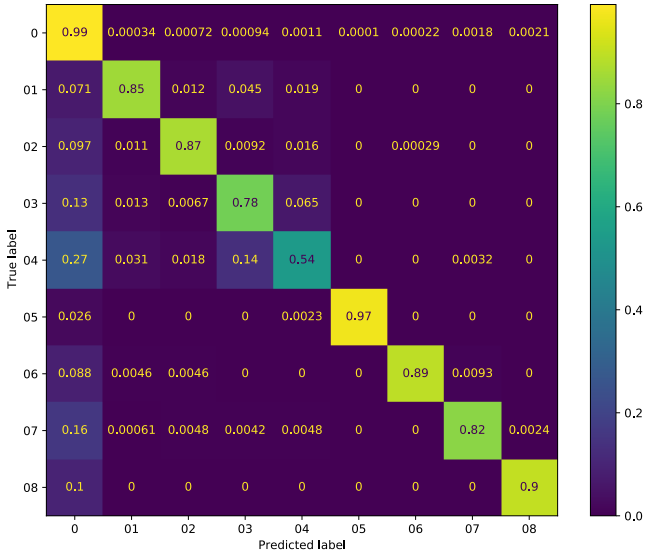
Fonte: os autores

do melhor modelo. O esperado é que os modelos mais profundos obtenham melhores resultados quando existam muitos dados e a tarefa seja complexa, como é a tarefa de NER. Especula-se que caso o *dataset* usado fosse ainda maior, esses modelos mais profundos superassem os modelos menores.

A matriz de confusão para o modelo de 3 camadas que foi o de melhor resultado neste experimento é mostrada na Figura 45. Nessa matriz os elementos estão bem

concentrados na diagonal principal, indicando um bom reconhecimento de todas as entidades. Obteve-se uma taxa de acerto maior do que 78% para todas as entidades exceto a miscelânea (04) que é uma das classes mais difíceis para todos os modelos experimentados.

Figura 45 – Matriz de confusão de classificação sobre o conjunto de validação do modelo com 3 camadas ocultas do *dataset* de 1 milhão de palavras



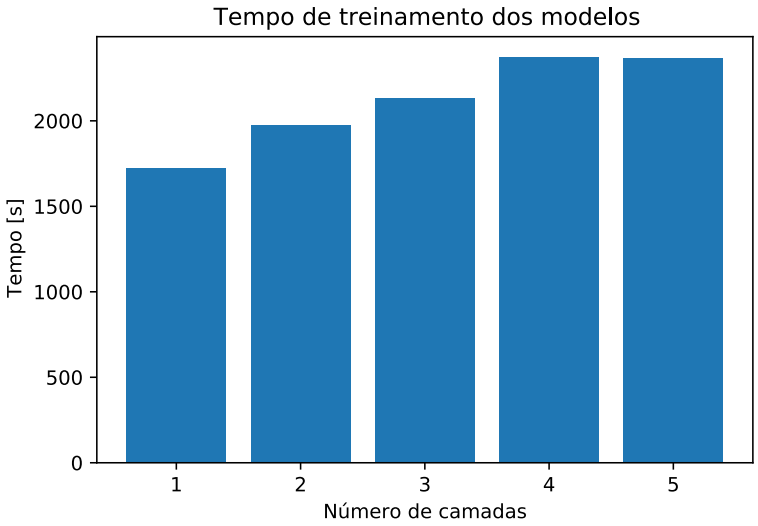
Fonte: os autores

O tempo execução dos modelos aumentou consideravelmente para todos os modelos, em função do aumento do *dataset*. Os tempos foram plotados na Figura 46, onde é observada a tendência linear de crescimento do tempo em função do número de camadas. Houve algo inesperado que é o tempo para o modelo com 5 camadas, que não aumentou conforme a tendência e se manteve praticamente igual ao modelo de 4 camadas, suspeita-se que isso possa ser atribuído ao estado do ambiente de execução em nuvem que pode ter realizado um alocamento de recursos no momento dos experimentos.

6.6 Análise dos resultados

Analisando todos os resultados das diversas experimentações realizadas puderam ser extraídas uma série de informações a respeito da tarefa de reconhecimento de entidades nomeadas para português, que serão explicadas a seguir. Primeiramente, um sumário dos resultados de taxa de acerto de token na validação para todos os modelos e *datasets* apresentados é mostrado na Tabela 5.

Figura 46 – Gráfico do tempo de treinamento dos modelos do *dataset* de 1 milhão de palavras



Fonte: os autores

Modelo	300k	600k	1M
baseline	88,43%	89,74%	88,77%
GRU	93,20%	96,27%	96,42%
BILSTM	94,90%	96,36%	97,08%
embeddings pré-treinados	93,90%	95,92%	96,41%
1 BILSTM	94,87%	96,84%	97,13%
2 BILSTM	**95,07%**	**96,88%**	97,20%
3 BILSTM	94,44%	96,64%	**97,21%**
4 BILSTM	94,10%	96,58%	96,59%
5 BILSTM	93,72%	96.66%	97,02%

Tabela 5 – Sumário de resultados dos experimentos

De forma geral, todos os modelos apresentaram melhorias com o aumento do conjunto de dados de treinamento, indicando que a limitação para a tarefa proposta é a disponibilidade de corpus e não limitações nos modelos de aprendizado de máquina. Como já mencionado anteriormente, o *dataset* para NER na língua portuguesa desenvolvido neste trabalho servirá para complementar pesquisas futuras visto que os resultados indicam que quanto maior o corpus de treinamento utilizado melhores os resultados finais.

Ainda, é possível afirmar que o modelo BILSTM obteve o melhor resultado em todos os cenários, superando o modelo GRU para os 3 conjuntos de dados apresentados. Também, o uso de *embeddings* pré-treinados não aprimorou os resultados do modelo BILSTM, em todos os casos aprender o *embedding* juntamente com o treinamento obteve melhores classificações. Porém, o uso desses *embeddings* se apresentou-se

melhor quando o tamanho do corpus aumentou e consequentemente seu vocabulário. Assim, é possível que expandido ainda mais o *dataset* coletado o uso desses vetores de palavras pré-treinados sejam vantajosos e ultrapassem os modelos BILSTM simples.

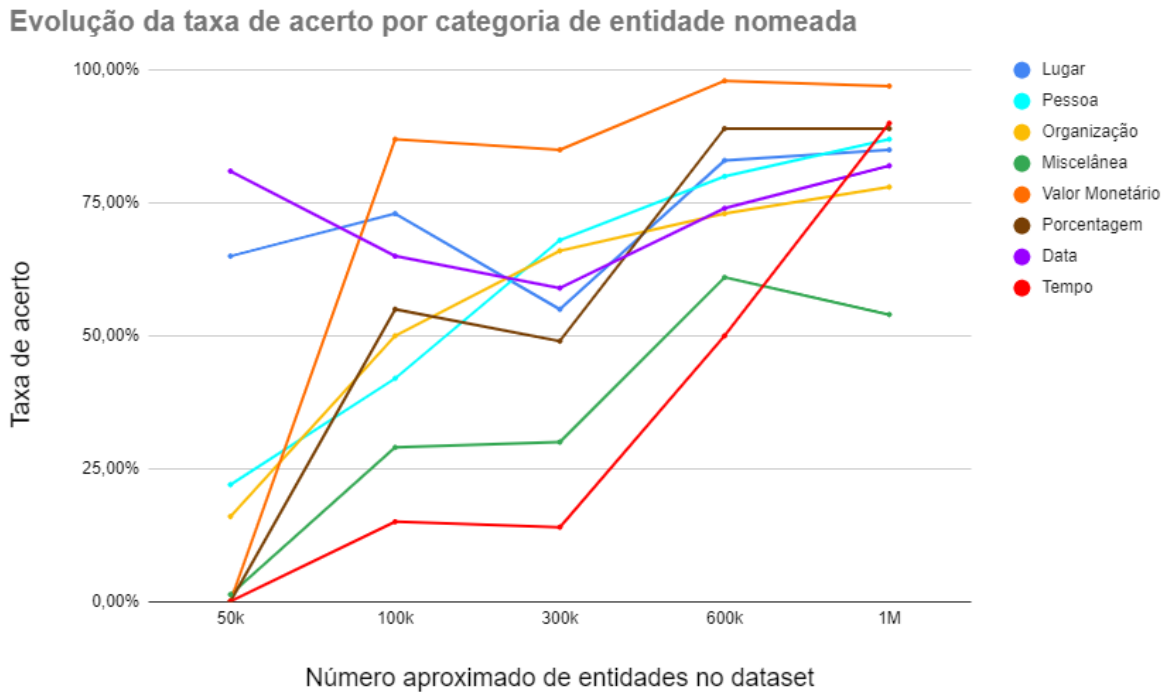
Quanto a quantidade de camadas ocultas recorrentes usadas na rede foi percebida uma tendência clara: modelos profundos não devem ser aplicados para pequenos *datasets*, mas podem ser aplicados para grandes *datasets*. Mesmo que os modelos mais profundos não tenham tido o melhor desempenho em nenhum dos casos, é certo que para o maior conjunto de dados, o modelo com 3 camadas superou o modelo com 2 camadas, que anteriormente foi o melhor modelo. Também, a diferença entre o maior e melhor modelo diminui à medida em que o tamanho do corpus foi aumentado, indicando que seria possível que os modelos maiores tenham o melhor resultado para maiores *datasets*.

Quanto as classes de entidades foram percebidas algumas tendências entre todos os modelos experimentados. Uma delas é a facilidade de classificação da classe de valor monetário, a qual diversos modelos tiveram acerto maior do que 90% na classificação. Isso pode ser atribuído pela uniformidade com que essas informações são apresentadas, comumente existe um indicador da moeda utilizada, como ‘R$’ ou ‘U$’ seguido de numerais e complementados com unidades de grandeza como ‘mil’, ‘milhões’ e ‘bilhões’. Mas ainda existem variações que ocorreram nos textos e foram corretamente identificadas pelos modelos como escrever por extenso o nome das moedas e numerais.

Outra classe que apresenta uma grande uniformidade assim como o valor monetário é a porcentagem, que muitas vezes é representada como um numeral seguido do símbolo ‘%’, mas também possui variações encontradas no corpus como ‘cinco pontos percentuais’. Apesar disso, sua classificação não foi tão consistente quanto a classe valor monetário nos diversos modelos, principalmente quando usados menores conjuntos de dados.

Uma entidade que causou grandes erros de classificação para quase todos os modelos foi a classe de miscelânea, que por sua própria natureza não é algo bem definido e imediatamente reconhecível já que pode ser um conjunto diverso de elementos. Nos textos anotados essa classe foi usada para representar títulos de livros, filmes, clubes de futebol, nome de doenças e outras entidades nomeadas que não as demais consideradas. Outro motivo para tal dificuldade de classificação é a diferença entre a anotação realizada por pessoas diferentes, que é ainda mais acentuada quando a definição de miscelânea não é concreta. Uma vez que o corpus anotado foi preparado de maneira manual por diferentes indivíduos é possível que diferentes regras de anotação tenham sido aplicadas às entidades classificadas como miscelânea, dificultando assim o aprendizado de sua correta classificação pelos modelos de aprendizado de máquina.

Figura 47 – Evolução das acurácias de cada classe em função do tamanho do *dataset*



Fonte: os autores

Outra classe que foi de difícil classificação foi a classe de tempo. Acredita-se que isso seja atribuído a grande variabilidade na forma de expressar tempo na língua portuguesa e também a sua baixa frequência de ocorrência entre as entidades do corpus, quando comparado com a entidade de data por exemplo. Algo que é comumente aplicado em solução de mercado para NER é a combinação das entidades de data e tempo em uma única classe chamada de *datetime*, uma vez que elas possuem características semelhantes semanticamente e por veze ocorrem juntamente. Assim, uma possível melhoria para os modelos apresentados seria tratar essas duas entidades nomeadas como a mesma, que poderia trazer melhores resultados durante a validação e melhor aplicação na sua tarefa final.

Foi realizada uma análise sobre as taxas de acerto para cada entidade nomeada em função do tamanho do conjunto de dados utilizado. O gráfico da Figura 47 ilustra as taxas de acerto das 8 categorias de entidades nomeadas para os três tamanhos de *dataset* estudados assim como dois outros menores. Foram considerados os modelos BILSTM com duas camadas para todos os *datasets* apresentados no gráfico. Claramente é observada a tendência de crescimento de todas as categorias com o aumento do conjunto de dados.

O *dataset* de aproximadamente 50 mil palavras é realmente extremamente pequeno para o treinamento de modelos de aprendizado de máquina, uma vez que

diversas categorias tiveram taxa de acerto nulo. A partir de aproximadamente 100 mil palavras todas as classes realizam alguma classificação correta, porém metade delas ainda com taxa de acerto abaixo de 50%. Para os *datasets* apresentados já é observado um nível aceitável de taxa de acerto para várias entidades diferentes, que de forma geral todas tendem a aumentar conforme a quantidade de dados disponível aumenta. Chegando ao resultado obtido para o maior conjunto de dados em que 7 das 8 categorias de entidades possuem taxas de acerto maiores do que 75% e apenas miscelânea com apresentou acerto menor.

7 CONSIDERAÇÕES FINAIS

7.1 Conclusões do Projeto de Formatura

O projeto de Reconhecimento de Entidades Nomeadas na Língua Portuguesa alcançou medidas aceitáveis de performance. Em seus modelos finais com o *dataset* mais extenso, de aproximadamente 1 milhão de palavras, alcançou taxas de acertos altas para suas classes, com exceção da classe 04 (Miscelânea) todas ficaram por volta de 80% ou maior. As taxas de acerto gerais também se mostraram consideravelmente acima das *baselines*, porém nunca ultrapassando por muito o patamar de 97%. Com estes resultados já está sendo efetuado um reconhecimento robusto das entidades nomeadas, possibilitando seu uso para Língua Portuguesa.

Também foi possível verificar uma certa volatilidade em alguns aspectos da metodologia de classificação. Os modelos de rede neural apresentam diferenças entre si, verificadas em seus resultados distintos, porém é possível observar também que essas diferenças se expressam mais em certas classificações, como 04 (Miscelânea), 08 (Tempo) e 03 (Organização). Isso expõe aspectos naturais da língua, que são classificações que podem apresentar construções muito distintas, principalmente em Miscelânea, que coleta quaisquer construções que não se enquadram em outras categorias. Desse modo cada modelo acaba interpretando estas entidades e seus padrões de modos diferentes, gerando resultados altamente variados e, as vezes, não condizentes com a performance das outras classes. Isso é um aspecto normal do uso humano de linguagens, e não se mostra diferente para a Língua Portuguesa.

7.2 Contribuições

Durante o decorrer do projeto foi gerado um *dataset* de NER para a Língua Portuguesa com um corpus de pouco mais de um milhão de entitdades, que está disponibilizado no Github pelo endereço <https://github.com/IanLaRosa/PCS3560_SEM_2020_S15> para uso livre em outros projetos que o necessite. Esse corpus foi completamente desenvolvido pelos autores, desde a coleta dos textos, tratamentos dos dados e classificação manual das entidades de acordo com os oito tipos de entidades padronizados em pesquisas de NER.

Também é fornecido para projetos futuros uma base de metodologia e solução inicial para resolver o problema do NER para a língua portuguesa. Contemplando *baselines* para o conjunto de dados desenvolvido e resultados de classificação para o treinamento dos modelos de aprendizado de máquina apresentados. Isso possibilita que se construam futuras soluções de NER para a língua portuguesa com essa base de soluções diferentes e aprimoradas para o problema.

7.3 Perspectivas de Continuidade

Vale destacar também a importância da escolha de um *dataset* e sua classificação cuidadosa. Para esse projeto se escolheu um *dataset* baseado em textos de notícias e artigos, que utilizam de expressões e construções típicas desse tipo de texto, e também possuem frequência alta de entidades nomeadas, um dos motivos para que foi escolhido. Esses fatos se manifestam nos resultados obtidos, as redes neurais treinadas se adaptam aos padrões encontrados no texto, assim apresentam melhores resultados em textos dessa espécie ou similares. Em projetos futuros seria interessante o uso de corpus de outras origens, como livros acadêmicos ou de ficção, mensagens e postagens de redes sociais, entre outros. Estes *datasets* certamente apresentariam concentrações diferentes de entidades nomeadas, e seria recomendado revisar e alterar quais categorias de entidades que se buscaria, é possível até que outros tipos de redes neurais obtenham resultados ainda melhores.

Outra possibilidade é a criação de uma solução híbrida, fazendo uso de algoritmos tradicionais de programação junto as redes neurais apresentadas. O uso humano de linguagem é altamente dinâmico e subjetivo, porém ainda é possível detectar por meio de modos tradicionais alguns padrões mais definidos da linguagem. Por exemplo, nas classes 05 (Valor Monetário) e 06 (Porcentagem) seria possível, com o auxílio de algoritmos tradicionais, alcançar taxas de acerto ainda melhores, dado que apresentam símbolos e construções bem definidas e previsíveis, com as redes neurais apenas realizando a classificação daqueles que fogem destes padrões recorrentes.

REFERÊNCIAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Acesso em: 01 dez. 2020.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural language processing with Python: analyzing text with the natural language toolkit**. [S.l.]: "O'Reilly Media, Inc.", 2009.

BIRD STEVEN, E. L.; KLEIN, E. **Natural Language Processing with Python**. [S.l.]: O'Reilly Media Inc, 20009.

CHIU, J. P.; NICHOLS, E. Named entity recognition with bidirectional lstm-cnns. **Transactions of the Association for Computational Linguistics**, v. 4, p. 357–370, 2016.

CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv preprint arXiv:1412.3555**, 2014.

FILANNINO, M.; BARI, M. D. Gold standard vs. silver standard: the case of dependency parsing for italian. **CLiC it**, p. 141, 2015.

FIRTH, J. R. A synopsis of linguistic theory, 1930-1955. **Studies in Linguistic Analysis**, Basil Blackwell, 1957. Disponível em: <https://ci.nii.ac.jp/naid/10020680394/en/>. Acesso em: 01 dez. 2020.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. [S.l.: s.n.], 2011. p. 315–323.

GOLDBERG, Y. A primer on neural network models for natural language processing. **Journal of Artificial Intelligence Research**, v. 57, p. 345–420, 2016.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

HARTMANN, N. et al. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. **arXiv preprint arXiv:1708.06025**, 2017.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

JIA, Y. et al. Caffe: Convolutional architecture for fast feature embedding. **arXiv preprint arXiv:1408.5093**, 2014.

JURASFKY, D.; MARTIN, J. H. **Speech and Language Processing**. 3. ed. [s.n.], 2019. Disponível em: <https://web.stanford.edu/~jurafsky/slp3/>. Acesso em: 01 dez. 2020.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MILIDIÚ, R. L.; DUARTE, J. C.; CAVALCANTE, R. Machine learning algorithms for portuguese named entity recognition. **Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial**, Asociación Española para la Inteligencia Artificial, v. 11, n. 36, p. 67–75, 2007.

NOTHMAN, J. et al. Learning multilingual named entity recognition from wikipedia. **Artificial Intelligence**, Elsevier, v. 194, p. 151–175, 2013.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543.

PINNA, F. C. de A.; NÉTO, J. C.; RUGGIERO, W. V. A brazilian portuguese real-time voice recognition to deal with sensitive data. In: IEEE. **2019 53rd Asilomar Conference on Signals, Systems, and Computers**. [S.l.], 2019. p. 1872–1876.

Princeton University. **WordNet**. 2010. Disponível em: <https://wordnet.princeton.edu>. Acesso em: 01 dez. 2020.

Python Software Foundation. **Python 3.8 documentation**. 2020. Disponível em: <https://docs.python.org/3/>. Acesso em: 01 dez. 2020.

SERRANI, V. M.; UEBEL, L. F. Bancos de fala para o português brasileiro. **Linguamática**, v. 3, n. 1, p. 69–75, 2011.

Torch. **PyTorch Documentation**. 2019. Disponível em: <https://pytorch.org/docs/stable/index.html>. Acesso em: 01 dez. 2020.

WIKIPEDIA. **Wikipedia: the free encyclopedia**. 2020. Disponível em: <https://www.wikipedia.org>. Acesso em: 01 dez. 2020.