



INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

ESTG

GLOSSIFICA: Tradutor Automático de Português para Glosa da Língua Gestual Portuguesa
Vasco António Gonçalves Alves

2022



INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

GLOSSIFICA: TRADUTOR AUTOMÁTICO DE PORTUGUÊS PARA GLOSA DA LÍNGUA GESTUAL PORTUGUESA

Dissertação de Mestrado

Vasco António Gonçalves Alves

Escola Superior de Tecnológica e Gestão



INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

Vasco António Gonçalves Alves

GLOSSIFICA: TRADUTOR AUTOMÁTICO DE
PORTUGUÊS PARA A GLOSA DA LÍNGUA GESTUAL
PORTUGUESA

Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação do
Professor Doutor Luís Miguel Cabrita Romero
Professor Doutor Pedro Miguel Teixeira Faria

Coadjuvante

Professor Doutor Jorge Manuel Ferreira Barbosa Ribeiro

novembro de 2022

Este trabalho faz parte do projeto intitulado “*IVLinG: Interprete Virtual de Língua Gestual*”, financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER), através do programa Compete2020, código de projeto POCI-01-0247-FEDER-068605.



UNIÃO EUROPEIA

Fundo Europeu
de Desenvolvimento Regional

AGRADECIMENTOS

Começo por agradecer aos orientadores deste projeto, Prof. Doutor Luís Romero e Prof. Doutor Pedro Faria pela confiança depositada sobre mim para a realização deste projeto, bem como todo o suporte e orientação dada. Além destes, agradeço também ao Prof. Doutor Jorge Ribeiro, que mesmo não sendo orientador do projeto, sempre esteve disponível para ajudar no que fosse necessário. A estes três, agradeço ainda a ajuda no desenvolvimento e validação do meu primeiro artigo publicado, publicação feita na 17ª Conferência Ibérica de Sistemas e Tecnologias Informáticas (CISTI'2022) realizada em Madrid.

Agradeço também aos meus colegas e amigos, Beatriz Miranda, Bruno Ribeiro, David Verde, Duarte Dias e Tânia Silva pela companhia, suporte e restantes momentos proporcionados ao longo do desenvolvimento deste trabalho. Também agradeço aos meus colegas de turma pelo suporte entre todos que existiu ao longo do desenvolvimento das respetivas dissertações de cada um. Sem esquecer também, todos aqueles meus amigos que tinham conhecimento deste trabalho e foram mostrando o seu carinho, suporte e apoio.

Aos professores que fizeram parte de todo o caminho até aqui pela contribuição direta ou indireta que tenham tido sobre mim, e ainda especial agradecimento ao Ângelo Costa e Vânia Ferreira pela ajuda essencial no que toca à Língua Gestual Portuguesa.

Por fim, talvez mais importante, agradeço à minha família próxima, que de forma direta ou indireta sempre me foram apoiando.

Sem me esquecer de ninguém, a todos, mesmo não tendo sido anteriormente mencionados, que fizeram parte deste caminho, obrigado.

RESUMO

A Língua Gestual Portuguesa (LGP) é uma das três línguas oficiais de Portugal, sendo a utilizada pela comunidade surda. No entanto, apesar de ser uma língua oficial nacional, verifica-se que os cerca de 30 mil indivíduos que utilizam esta língua como sua língua materna demonstram uma enorme dificuldade de comunicação com a população em geral já que, dos cerca de 11 milhões de pessoas em Portugal, apenas cerca de 110 mil são utilizadores da LGP, ou seja, apenas 1% da população. Acresce ainda que a LGP segue uma estrutura e uma gramática muito diferentes da língua portuguesa, não permitindo que a grande maioria dos surdos consiga ler e interpretar texto em português. Existem diversas tecnologias que permitem a tradução entre línguas naturais diferentes, disponíveis a qualquer pessoa. No entanto, é muito inferior o número de tradutores disponíveis para a tradução entre uma língua natural e uma língua gestual, dificultando imenso a comunicação entre utilizadores e não utilizadores da LGP.

Neste contexto, o principal objetivo do trabalho, que surge no âmbito do projeto IVLinG – *Intérprete Virtual de Língua Gestual*, que aqui se descreve foi o de projetar e desenvolver um tradutor capaz de traduzir texto em português para uma representação textual escrita de uma frase da língua gestual (denominada por “glosa da LGP”). Para tal, foram realizados estudos acerca da estrutura e gramática da LGP e experimentados diversos métodos, anteriormente desenvolvidos e utilizados com diversas outras línguas. Seguidamente, foi desenvolvido um tradutor, suportado numa arquitetura híbrida, que articula a utilização de regras gramaticais e a utilização de uma rede neuronal. A ferramenta desenvolvida foi testada com traduções, obtidas da parte de uma intérprete e de um formador de LGP,

cuja língua materna deste último é a própria LGP. Os testes realizados com o tradutor demonstraram tratar-se de uma ferramenta capaz de realizar boas traduções entre as duas línguas, tendo sido alcançados resultados BLEU e METEOR de 0.722405 e 0.795578 respetivamente numa tradução baseada em regras, e BLEU e METEOR de 0.698358 e 0.752246 respetivamente para uma tradução baseada em rede neuronal.

Paralelamente ao desenvolvimento do tradutor, foi também construído um dataset (“corpus”) de traduções entre a língua portuguesa e a glosa da LGP, corpus utilizado para o treino da rede neuronal do tradutor.

Palavras-Chave: *Língua Gestual Portuguesa, Tradução Automática, Machine Learning, Tradução para Glosa*

ABSTRACT

The Portuguese Sign Language (LGP) is one of three Portugal's official languages, being the one used by the deaf community. However, despite being an official language of Portugal, it appears that the 30.000 individuals (approximately) who use this language as their mother tongue will have a lot of difficulty in communicating with the general population, since, of the approximately 11 million people in Portugal, only about 110.000 are LGP users, that is, only about 1% of the population. Furthermore, the LGP follows a very different structure and grammar from the Portuguese language, preventing most deaf people from being able to read and interpret texts written in Portuguese correctly. There are several technologies that allow the translation between different natural languages, available to anyone. However, the number of translators available for translating between a natural language and a sign language is much smaller, making communication between LGP users and non-users immensely difficult.

In this context, the main objective the work, that arises within the scope of the IVLinG project - *Intérprete Virtual de Língua Gestual*, described here was to design and develop a translator capable of translating Portuguese into LGP gloss. For this, studies were carried out on the structure and grammar of LGP and various methods, previously developed and used with other languages, were tested. Next, a translator was developed, supported by a hybrid architecture, which articulates the use of grammatical rules and the use of a neural network. The developed tool was tested with translations, obtained from an interpreter and an LGP trainer, whose native language of the latter is LGP itself. The tests carried out showed that it is a tool capable of carrying out good translations between the two languages, archiving BLEU

and METEOR scores of 0.722405 and 0.795578 respectively for a rule-based approach, and BLEU and METEOR scores of 0.698358 and 0.752246 for a neural-translation approach.

Parallel to the development of the translator, a corpus of translations between the Portuguese language and the LGP gloss was also built, corpus used to train the translator's neural network.

Keywords: *Portuguese Sign Language, Machine Translation, Machine Learning, Translation from Portuguese to Portuguese Sign Language Glosses*

ÍNDICE

Índice	xi
ÍNDICE DE FIGURAS.....	xv
ÍNDICE DE TABELAS.....	xvii
GLOSSÁRIO, SIGLAS e ACRÓNIMOS.....	xviii
1. Introdução.....	1
1.1. Contextualização	1
1.2. Objetivos.....	1
1.3. Metodologia Utilizada	2
1.4. Estrutura do Documento	4
2. Enquadramento Teórico	6
2.1. Língua Gestual Portuguesa	6
2.1.1. Características da Língua Gestual Portuguesa (LGP).....	6
2.1.2. Glosa	7
2.1.3. Características Gramaticais e Estruturais da LGP.....	8
2.2. Machine Learning e Processamento de Linguagem Natural.....	11
2.2.1. Redes Neurais Artificiais	12
2.2.2. Recurrent Neural Networks	13
2.2.3. Long-Short Term Memory	14
2.2.4. Gated Recurrent Unit	15
2.2.5. Encoder-Decoder.....	16
2.2.6. Embeddings	17
2.2.7. BLEU - BiLingual Evaluation Understudy	18

2.2.8.	METEOR - Metric for Evaluation of Translation with Explicit ORdering	19
2.2.9.	Natural Language Processing	19
2.3.	Tradução Automática	20
2.3.1.	RBMT – Rule Based Machine Translation	20
2.3.2.	SMT – Statistical Machine Translation	20
2.3.3.	NMT – Neural Machine Translation	21
2.3.4.	HMT – Hybrid Machine Translation	21
3.	Estado da Arte	22
3.1.	RBMT e SMT – Tradução Baseada em Regras e Modelos Estatísticos	22
3.1.1.	Conclusões RBMT e SMT	31
3.2.	NMT – Tradução Baseada em Redes Neurais	32
3.2.1.	Apreciação NMT	38
3.3.	Conclusões	40
4.	Arquitetura do Sistema	41
4.1.	Enquadramento	41
4.2.	Arquitetura do Módulo de Tradução de Português para LGP	42
4.3.	Conversão de Áudio em Texto	45
4.3.1.	Ferramentas Speech to-Text	45
4.3.2.	Avaliação e Resultados	46
4.4.	Arquitetura do Tradutor de Português para a Glosa da LGP	51
4.5.	Seleção de Rede Neuronal	53
4.5.1.	Corpora Utilizada	53
4.5.2.	Resultados Alcançados	54
4.6.	Conclusões	55
5.	Implementação do Sistema	57

5.1.	Características Computacionais.....	57
5.2.	Tecnologias e Ferramentas Utilizadas	58
5.2.1.	Python	58
5.2.2.	spaCy	58
5.2.3.	Keras.....	58
5.2.4.	Sockets e Websockets	58
5.3.	Módulo de Tradução de Português para LGP e Integração no Sistema Completo	59
5.4.	Tradutor de Português para Glosa da LGP	60
5.4.1.	Conversão de SVO para OSV e Aplicação de Regras Gramaticais	61
5.4.2.	Construção e Utilização de Rede Neuronal.....	69
5.4.3.	Tradutor Híbrido.....	73
5.4.4.	Exemplo de Utilização do Glossifica.....	74
5.5.	Exemplo de Utilização de Tradutor para Glosa	76
5.6.	Conclusões	78
6.	Análise de Resultados.....	80
6.1.	Qualidade de Tradução	80
6.2.	Escalabilidade	82
6.3.	Corpus de Traduções de Português para Glosa da LGP	83
6.4.	Análise Comparativa.....	85
6.5.	Conclusões	87
7.	Conclusões e Trabalho Futuro	89
7.1.	Conclusões	89
7.2.	Trabalho Futuro	91
8.	Referências.....	93
9.	Anexos / Apêndices	101

9.1. Artigos Publicados em Conferências Internacionais	101
9.1.1. Neural Machine Translation Approach in Automatic Translations between Portuguese Language and Portuguese Sign Language Glosses	101
9.2. Experimentação e Comparação de Tradutor Neuronal entre Línguas com e sem a Utilização de Categorias Gramaticais Para a Tradução	102
9.3. Código	106
9.3.1. Script de Obtenção de Títulos e Corpos de Notícias	106
9.3.2. Tradutor Híbrido	107
9.3.3. Excerto de Código – Tradutor Baseado Em Regras	109

ÍNDICE DE FIGURAS

Figura 1 - Exemplo Rede Neuronal [15].....	12
Figura 2 – Visão geral de uma RNN (Adaptada de [19]).....	13
Figura 3 – Células da RNN simples e LSTM [23].	14
Figura 4 - Células da LSTM e GRU [23].	15
Figura 5 - Representação de Encoder-Decoder básica [30].	16
Figura 6 - Representação de um Encoder-Decoder com mecanismo de atenção [31].	17
Figura 7 - Escala de avaliação BLEU [33].	18
Figura 8 - Algoritmo de [40].....	23
Figura 9 - Exemplos de traduções de [42].	24
Figura 10 - Algoritmo de [43].	25
Figura 11 - Algoritmo original de [44].	26
Figura 12 - Algoritmo de [43].	26
Figura 13 - Visão geral do sistema de [46].	28
Figura 14 - Arquitetura geral de [47].....	28
Figura 15 - Algoritmo de procura de raízes e gestos de [48].	29
Figura 16 - Algoritmo de [50].	30
Figura 17 – Arquitetura de [52].....	35
Figura 18 - Arquitetura do tradutor RBMT de [54].	35
Figura 19 - Processo de treino de [54].	36
Figura 20 - Processo de teste de [54].	36
Figura 21 - Arquitetura do sistema de [55].	36
Figura 22 - Processo de Tradução de Texto.	44
Figura 23 - Arquitetura Geral do Módulo de Tradução PT -> LGP.	44
Figura 24 – Fluxograma representativo script de obtenção de transcrições.	49
Figura 25 - Fluxograma representativo script de avaliação das transcrições.	50
Figura 26 - Diagrama de Atividades do Módulo Tradução PT para Glosa LGP.	53
Figura 27 - PT->EN LSTM.....	54
Figura 28 - PT->EN LSTM Bidirecional.	54

Figura 29 - PT->Libras LSTM.	55
Figura 30 - PT->Libras LSTM Bidirecional.	55
Figura 31 - Diagrama de Estados Representativo do Módulo de Tradução de Português para Animação LGP.	60
Figura 32 - Árvore de Dependências Frase B.....	62
Figura 33 - Exemplo de Conversão SVO -> OSV.....	65
Figura 34 - Modelo Encoder-Decoder Definido.	72
Figura 35 - Exemplo de Teste de Tradução realizado no Postman.	75
Figura 36 - Envio de Mensagem a partir do Sistema Geral.	77
Figura 37 - Traduções feitas pelos tradutores.....	77
Figura 38 - Receção do Vídeo pelo Sistema Geral.....	78
Figura 39 – Tempo em segundos para X pedidos simultâneos	83
Figura 40 - BLEU dados treino PT-EN sem modificação.	104
Figura 41 - BLEU dados treino PT-EN com modificação para categorias gramaticais.	104
Figura 42 - BLEU dados teste PT-EN sem modificação.....	105
Figura 43 - BLEU dados teste PT-EN com modificação para categorias gramaticais.	105

ÍNDICE DE TABELAS

Tabela 1 - Características / Componentes utilizados em cada trabalho RBMT e SMT.	32
Tabela 2 - Arquiteturas e Principais Conclusões de Cada Trabalho NMT.	39
Tabela 3 – Exemplos de Transcrições Obtidas usando as ferramentas Google Recognition e Wit.AI.	50
Tabela 4 - Resultados das Métricas para as Transcrições do Dataset e Quantidade de Palavras Corretas e Erros.	51
Tabela 5 - Divisões pelo método .sents do spaCy.	62
Tabela 6 - Regras de modificação da estrutura das frases.	68
Tabela 7 - Regras de modificação a nível da palavra das frases.	69
Tabela 8 – Hiper parâmetros utilizados na Rede Neuronal.	72
Tabela 9 - Exemplos de Traduções Conseguídas pelo Tradutor Híbrido.	75
Tabela 10 - Traduções esperadas e Métricas de Avaliação.	76
Tabela 11 - Exemplos de Traduções 81	81
Tabela 12 - Exemplos de Traduções conseguidas pelo tradutor híbrido 81	81
Tabela 13 - Resultados BLEU e METEOR para as traduções do Tradutor Construído	81
Tabela 14 - Tempo de resposta em segundos a vários pedidos simultâneos 82	82
Tabela 15 - Exemplo de Traduções no Corpus 84	84
Tabela 16 - Funcionalidades do Glossifica e outros tradutores 85	85
Tabela 17 - Comparação Vantagens e Desvantagens Glossifica vs Outros Tradutores.	86
Tabela 18 - Exemplos de Traduções das Redes Neurais Modificada e Não Modificada.	105

GLOSSÁRIO, SIGLAS E ACRÓNIMOS

Lematização: Conversão de palavras para a sua forma canónica;

Stemização: Conversão de palavras flexionadas para a sua forma base, para a sua raiz;

Tokenização: Divisão de frase a palavras individuais;

LGP: Língua Gestual Portuguesa;

Glosa: Representação textual escrita dos gestos de uma frase LGP;

ML: Machine Learning;

RNA: Redes Neurais Artificiais;

RNN: Recurrent Neural Networks;

LSTM: Long-Short Term Memory;

BLEU: BiLingual Evaluation Understudy;

METEOR: Metric for Evaluation of Translation with Explicit ORdering;

NLP: Natural Language Processing;

RBMT: Rule Based Machine Translation;

NMT: Neural Machine Translation;

HMT: Hybrid Machine Translation;

XMPP: Extensible Messaging and Presence Protocol;

JSON: JavaScript Object Notation;

Híperparâmetros: Parâmetros definidos para o treino de uma rede neuronal;

1. INTRODUÇÃO

1.1. Contextualização

A Língua Gestual Portuguesa é uma das três línguas oficiais nacionais além do português e mirandês, sendo a utilizada pela comunidade surda para comunicar. Estima-se que existem cerca de 30 mil indivíduos pertencentes a esta comunidade em Portugal [1]. Contabilizando-se ainda toda a comunidade envolvente, tais como familiares, professores, técnicos, etc., estima-se que existam cerca de 110 mil utilizadores totais da língua gestual portuguesa, o que representa apenas 1% da população nacional, tendo em conta os valores dos Censos de 2021 [2]. Olhando-se para esses valores, consegue-se facilmente perceber que qualquer surdo terá imensas dificuldades em comunicar com a população geral, sendo que na maioria dos casos, essa comunicação poderá ser impossível de se realizar, já que muita pouca gente tem conhecimento da língua gestual ou de como a utilizar. No entanto, nos bolsos de quase todos pode estar uma solução para mitigar essas dificuldades. Da mesma forma que já é perfeitamente normal utilizar os smartphones (e não só) para traduções entre a língua natural portuguesa e outras línguas, questiona-se porque não os utilizar também para a tradução entre a língua natural e a língua gestual.

1.2. Objetivos

O principal objetivo deste trabalho prende-se com a construção de um tradutor automático de texto ou fala em português para a glosa da Língua Gestual Portuguesa. Este tradutor automático estará integrado numa aplicação móvel a desenvolver para a comunicação bidirecional entre surdos e não surdos. No entanto, tanto a aplicação como a tradução no sentido Língua Gestual Portuguesa para texto em português não fazem parte do âmbito deste trabalho. Para alcançar o principal objetivo, podem ser definidos os seguintes objetivos mais pequenos:

- **Estudo Comparativo de Ferramentas de Speech-to-Text:** O tradutor automático a ser desenvolvido terá a capacidade de receber, não só texto,

mas também áudio, áudio este que necessitará de ser transcrito de forma a ser processado e traduzido para a glosa da Língua Gestual Portuguesa;

- **Estudo Comparativo e Escolha de Técnicas de Tradução Automática:** Será necessário realizar um estudo para se conhecer o atual estado da arte de ferramentas e técnicas de tradução automática, de forma a determinar-se qual(is) a(s) abordagem(s) a ser(em) utilizada(s);
- **Determinação de Regras Gramaticais e Estrutura da LGP:** Realização de um estudo de bibliografia e outros trabalhos que envolvam a LGP, de forma a determinar características estruturais e gramaticais, e importância dessas características na LGP;
- **Implementação de um Tradutor de Português para a Glosa da LGP, composto por:**
 - Tradutor Automático que utilize as Características Gramaticais e Estruturais da LGP;
 - Tradutor Automático que utilize Redes Neurais;
 - Tradutor Híbrido que junte os dois anteriores;
- **Criação de um Corpus com Traduções entre Texto em Português e Glosa da LGP:** Não foi encontrado um corpus público disponível, sendo necessária a sua criação, de modo a ser utilizado no treino de redes neurais para a realização da tarefa de tradução;

1.3. Metodologia Utilizada

Na realização do trabalho que aqui se descreve foi utilizada a metodologia *Design Science Research* (DSR), metodologia essa que se baseia na criação e avaliação de artefactos para a resolução de problemas, envolvendo processos bem definidos como: o desenho e a avaliação dos artefactos produzidos, a produção de contribuições científicas e a apresentação dos resultados alcançados [3]. O autor de [4], após realização de um estudo acerca de vários trabalhos publicados sobre DSR, identificou quais os pontos comuns em todos eles, de forma a propor um processo de DSR para o desenvolvimento e avaliação de investigação em sistemas da informação. As atividades

do processo proposto são a Identificação do Problema e a Motivação, a Definição dos Objetivos para a Solução, o Desenho e Desenvolvimento, a Demonstração, a Avaliação e a Comunicação e Difusão do Trabalho:

- **Identificação do Problema e Motivação:** Definir o problema, obter um bom conhecimento do estado da arte e estabelecer a importância da solução. Neste trabalho esta atividade está descrita nas secções 1.1. Contextualização e 3. Estado da Arte;
- **Definição dos Objetivos para a Solução:** Estabelecer os objetivos para a solução. Envolve determinar o que é possível e exequível. Envolve também conhecer atuais soluções para o problema. Os objetivos podem ser quantitativos ou qualitativos. Os objetivos quantitativos demonstram como a solução proposta será melhor do que outras, o que pode envolver a definição e utilização de métricas para a comparação com soluções anteriores, ou objetivos qualitativos, que descrevem como os artefactos permitirão alcançar soluções para o problema. Atividade descrita na secção 1.2. Objetivos;
- **Desenho e Desenvolvimento:** Determinar a funcionalidade e arquitetura dos artefactos e posterior criação dos mesmos. Esta atividade está descrita na secção 4. Arquitetura do Sistema;
- **Demonstração:** Demonstrar de que modo o artefacto desenvolvido resolve o problema (ou problemas) identificados. Esta demonstração pode ser conseguida através de experimentação, simulação, realização de casos de estudo, entre outras. Esta atividade está descrita na secção 5.4.4. Exemplo de Utilização do Glossifica, 5.5. Exemplo de Utilização de Tradutor para Glosa e 6. Análise de Resultados;
- **Avaliação:** Verificar e medir o comportamento do artefacto desenvolvido para resolver o problema, de forma a determinar se os artefactos apresentam resultados esperados, tendo em consideração os objetivos estabelecidos. Esta atividade está descrita na secção 6. Análise de Resultados;
- **Comunicação e Difusão:** Partilhar o problema, a solução desenvolvida e os resultados da mesma, o que pode envolver a publicação de artigos em

conferências ou revistas. Esta atividade está apresentada nos apêndices deste documento;

1.4. Estrutura do Documento

Neste subcapítulo apresenta-se de forma resumida a organização deste trabalho, em capítulos, indicando o conteúdo de em cada um deles. Este trabalho está organizado nos seguintes 9 capítulos:

- **1 – Introdução:** Neste capítulo é apresentado o contexto, razões e objetivos para a importância da realização deste trabalho. É ainda apresentada a metodologia utilizada no desenvolvimento do mesmo e o modo como este se encontra estruturado;
- **2 – Enquadramento Teórico:** O segundo capítulo incide sobre conceitos relevantes acerca das principais tecnologias utilizadas no desenvolvimento do trabalho, modelos e conceitos associados a redes neuronais, ferramentas utilizadas, bem como alguns conceitos sobre a Língua Gestual Portuguesa (LGP) e a sua gramática;
- **3 – Estudo do Estado da Arte:** O terceiro capítulo foca-se na apresentação de trabalhos já realizados, relacionados com a tradução automática entre linguagens naturais e língua gestual. É um capítulo dividido em duas partes principais, onde na primeira são apresentados trabalhos que utilizam abordagens ao problema utilizando técnicas de tradução baseada em regras ou baseada em estatística, enquanto na segunda parte apresentam-se trabalhos relacionados com o problema, mas que utilizam abordagens baseadas em redes neuronais. Em cada uma das partes escreve-se uma conclusão, detalhando-se os principais tópicos e componentes de cada uma delas e, no final, apresenta-se uma conclusão resumo, indicando as principais ideias obtidas do estudo do estado da arte;
- **4 – Arquitetura:** Neste capítulo apresentam-se as arquiteturas para o desenvolvimento do sistema e dos componentes do mesmo. Adicionalmente, descreve-se o estudo realizado sobre algumas das

principais ferramentas de transcrição de áudio para texto gratuitas, efetuando-se também uma análise comparativa de várias métricas de avaliação da transcrição de um corpus, disponibilizado gratuitamente para o efeito de desenvolvimento de sistemas de transcrição de áudio para texto. É também aqui descrito um estudo comparativo de redes neurais, realizado no âmbito da tarefa de tradução automática.

- **5 – Implementação do Sistema:** Neste capítulo apresentam-se as implementações construídas para o sistema, com base nas arquiteturas definidas no capítulo 4. São detalhadas as implementações e as escolhas realizadas durante as mesmas. Após concluídas as implementações, são apresentados alguns exemplos que demonstram o funcionamento das mesmas;
- **6 – Análise de Resultados:** Neste capítulo efetua-se uma análise aos resultados obtidos acerca do sistema implementado. Estes resultados focam-se na qualidade das traduções produzidas pelo sistema;
- **7 – Conclusões e Trabalho Futuro:** Neste capítulo é apresentado um resumo do trabalho realizado, discutindo resultados alcançados e contribuições produzidas, sugerindo-se possíveis continuações deste trabalho;
- **8 – Referências:** Neste capítulo apresenta-se a lista das referências utilizadas no desenvolvimento do trabalho descrito neste documento;
- **9 – Anexos ou Apêndices:** Neste capítulo encontram-se documentos complementares ou de apoio ao desenvolvimento deste trabalho, bem como artigos publicados em conferências internacionais e código desenvolvido.

Em termos da organização deste documento, é também de referir que se optou por reduzir a quantidade de código a apresentar no corpo do texto, sendo este substituído por representações gráficas, através de algoritmos ou diagramas, de modo a manter o documento o mais uniforme possível. O código que se considera mais importante para as componentes e funcionalidades desenvolvidas está apresentado nos apêndices, sendo referenciado no local a que esse código se refere.

2. ENQUADRAMENTO TEÓRICO

2.1. Língua Gestual Portuguesa

A Língua Gestual Portuguesa (LGP) é uma das línguas oficiais de Portugal sendo a utilizada pela comunidade surda. De acordo com a Associação Portuguesa de Surdos [1], existem 30.000 surdos utilizadores da LGP. Considerando docentes, familiares, amigos e outros utilizadores, esse número de utilizadores da LGP sobe para uma estimativa de 110.000 utilizadores. Tendo em conta os resultados dos Censos de 2021, toda a comunidade utilizadora de LGP representa apenas cerca de 1% da população de Portugal [2]. Tal como qualquer língua natural, as línguas gestuais também são bastante complexas, já que existem muitos gestos, e o significado desses gestos é alterado com pequenas modificações da posição das mãos, movimento da mesmas e contexto da conversa. As línguas gestuais também não são iguais a línguas naturais, não seguindo, por exemplo, regras nem gramática da língua natural [5]. A representação gestual, ou gesto, possui um conjunto de características bastante importante para a correta comunicação, principalmente a orientação, configuração e movimentos realizados pelas mãos, ponto de articulação do movimento e expressões faciais e corporais. A modificação, por mais pequena que seja, de qualquer uma destas características, poderá resultar em gestos com significados diferentes daqueles que se pretende expressar, ou até mesmo em gestos sem qualquer significado. Tal como nas línguas naturais, as línguas gestuais também variam de país para país, não existindo uma única língua global. Por exemplo, devido às diferenças existentes entre a LGP e a língua portuguesa, nada garante que um utilizador da LGP consiga comunicar através da escrita de português natural.

2.1.1. Características da Língua Gestual Portuguesa (LGP)

A Língua Gestual Portuguesa (LGP), à semelhança de outras línguas gestuais, possui cinco características principais, designadas por *queremas*, são eles [6] e [7]:

- **Configuração da Mão:** Representa a configuração que a mão dominante ou que cada uma das mãos assume durante a execução do gesto. Na LGP, as configurações são baseadas no alfabeto manual português e na numeração;
- **Orientação:** Para onde a palma da mão está virada durante a execução do gesto. A inversão da orientação pode apresentar a ideia de oposição, contrário ou concordância de número;
- **Expressão Facial e Corporal:** A expressão facial é muito importante pois dá sentimento ao que está a ser representado. Através da expressão fácil, consegue-se distinguir se é interrogação, exclamação, negação, afirmação, admiração, medo, entre outras. Alguns gestos apenas são diferenciados pela expressão facial. A postura corporal pode ser utilizada para especificar os referentes (sujeitos ou objetos) e pode também ser utilizado como ponto de referência.
- **Local de Articulação:** Posição onde a mão configurada executa o gesto. Esta posição divide-se em duas formas de utilização. Na primeira, não existe contacto da mão com o corpo, sendo o gesto executado numa área imaginária que abrange a frente do tronco e da cabeça. A segunda, envolve o contacto com partes do próprio corpo (por exemplo, orelhas, pescoço, ombros, entre outros);
- **Movimento:** Depois de configuradas, as mãos podem mover-se pelos locais de articulação, no entanto, existem alguns gestos que não envolvem movimento. O gesto pode resultar de um movimento isolado ou a combinação de vários movimentos. Adicionalmente, a ação do movimento pode ser diferenciada por movimentos rápidos ou lentos, longos ou curtos, suaves ou tensos.

2.1.2. Glosa

A glosa é uma representação textual escrita de uma frase da língua gestual. A glosa é constituída por termos, sendo cada termo um dos gestos presentes na frase. A glosa costuma ser escrita considerando-se as seguintes regras [6]:

- Usar apenas letras maiúsculas;
- Usar o sinal '+' como separador dos gestos;

- Usar ‘/’ como indicação de pausa curta;
- Usar ‘//’ como indicação de pausa longa;
- Usar o sinal ‘-’ como separador das letras em palavras soletradas, e como separador dos dígitos em números cardinais;
- Escrever números ordinais e números que representam quantidades por extenso;
- Escrever verbos sempre no infinitivo.

Um exemplo de uma frase e respetiva glosa: “O João chutou a bola.”, “BOLA+J-O-A-O-~+CHUTAR”.

2.1.3. Características Gramaticais e Estruturais da LGP

As características gramaticais e estruturais da LGP que se indicam neste trabalho foram obtidas através da análise de alguns trabalhos relacionados com a LGP, tais como [7], [8], [9], [10], [11] e ainda em resultado de reuniões com intérpretes e utilizadores de LGP. Dos trabalhos citados, quase todos indicam a obtenção da informação relativa às características da LGP de um livro [7] sendo, talvez, este livro o principal trabalho publicado relacionado com a gramática da Língua Gestual Portuguesa. No entanto, verifica-se uma grande dificuldade em encontrar-se mais informação relacionada com a gramática, sendo que à data da realização deste trabalho, o livro anteriormente indicado já não é impresso nem vendido.

2.1.3.1. Estrutura Frásica

A LGP segue a estrutura OSV (Objeto-Sujeito-Verbo). Esta estrutura distingue-se do português natural, que segue a estrutura SVO (Sujeito-Verbo-Objeto). No entanto, neste tópico existe alguma discordância, já que alguns, tais como [9] e [11], afirmam que a LGP segue a estrutura SOV (Sujeito-Objeto-Verbo). No entanto, na maioria dos casos, a frase seguindo OSV ou SOV será perceptível pelos utilizadores da LGP. A interrogação, negação e a exclamação são muitas vezes representadas em expressões faciais ao longo da frase. No entanto, a negação, pode também ser representada pela inclusão do NÃO no fim da frase, e a interrogação, pode também ser construída através da utilização de pronomes interrogativos (por exemplo: quem, quando, quantos).

2.1.3.2. *Nomes*

Na LGP, na grande maioria dos casos, não existe distinção do género, sendo apenas indicado quando necessário. Nos casos em que conhecer o género é relevante, a indicação é feita com a introdução do mesmo antes do nome, como se fosse um prefixo (ex: MULHER + «nome»). Sendo omitido, quando relevante, é considerado o género masculino. Em alguns casos, a distinção é feita por gestos diferentes, que representam cada um dos géneros. No caso da necessidade da utilização do plural, existem algumas formas diferentes de o indicar. A primeira forma é a incorporação do número após o nome, em caso de quantidades pequenas (exemplo: CAO-TRÊS), ou incorporação de quantificadores (CAO-MUITO). Outra forma é a repetição do movimento, em que o movimento é repetido, da mesma forma, várias vezes seguidas, num movimento regular e contínuo. Ainda uma outra forma, designada por rebordo, consiste em repetir o gesto realizado na mão dominante pela mão não dominante.

Em termos de nomes próprios, algumas pessoas (geralmente as mais importantes/conhecidas) possuem um nome gestual, no entanto, regra geral, os nomes são soletrados.

2.1.3.3. *Iconicidade*

As línguas gestuais contêm alguns gestos que tentam criar uma mímica do objeto do mundo real, sendo perceptíveis mesmo por pessoas que nunca tenham tido contacto com a LGP. Estes gestos, que apresentam semelhanças com um objeto real, são designados por gestos icónicos. Existem ainda gestos referenciais, que envolvem apontar diretamente para a realidade ou para um espaço que a representa, e ainda os gestos arbitrários, gestos que não apresentam qualquer relação com o objeto real.

2.1.3.4. *Adjetivos*

Na LGP, os adjetivos são colocados após o nome, por exemplo, *gato pequeno* ficaria “GATO + PEQUENO”.

2.1.3.5. *Graus Substantivos*

Na LGP, a representação dos graus diminutivos e aumentativos é realizada através da composição de gestos, sendo adicionado um gesto representativo, normalmente PEQUENO ou GRANDE, respetivamente, após o substantivo. Por exemplo, a palavra *gatinha* seria representada por “MULHER + GATO + PEQUENO”.

2.1.3.6. Verbos

Normalmente, na LGP é utilizada a forma infinitiva dos verbos. Os verbos podem ser marcados temporalmente, sendo essa marcação conseguida através de expressões faciais, durante a realização do gesto na forma infinitiva, ou através da adição de advérbios temporais (como amanhã, ontem, ...) caso façam sentido na frase, ou através dos gestos “PASSADO” e “FOI”, para representação do passado em frases negativas e positivas respetivamente, e “BREVE” para representação de algo no futuro ([7] indica o gesto “FUTURO” para marcação do futuro, no entanto o formador e intérprete de LGP que estão a contribuir neste projeto indicam que “BREVE” é mais indicado. Não existem flexões em termos de número de género. O verbo *ser* e o verbo *estar* não são representados.

2.1.3.7. Números

Dependendo do contexto, a forma de representação dos números é diferente. Para quantidades, durações e repetições, são utilizados gestos que representam a ordem decimal (10, 100, 1000, ...), por exemplo, o número 2000 seria representado pela combinação “2 + MIL”. Noutros casos, os números são representados individualmente, numa espécie de soletração. Para este último caso, o 2000 seria representado pela combinação de “2 + 0 + 0 + 0”. Para quantidades, o número é representado depois do objeto.

2.1.3.8. Gestos Compostos

Os gestos compostos são gestos que foram criados pela junção de dois ou mais gestos que já existiam anteriormente para a criação de um novo termo. Um exemplo de um gesto composto é o gesto para *vegetariano*, representado por “PLANTA + COMER”.

2.1.3.9. Gestos Variáveis

Os gestos variáveis são gestos que são realizados de forma diferente dependendo do contexto em que são utilizados. Um exemplo é o gesto para *alto*, que quando se refere a pessoas ou animais é representado de uma forma, enquanto quando se refere a objetos é representado de outra forma.

2.1.3.10. Gestos Específicos

Os gestos específicos são gestos que normalmente representam um conjunto de palavras em português escrito, cuja representação individual de palavra a palavra

poderá em certos casos comprometer o significado da frase. Por exemplo, “Abrir a porta” possui uma representação específica diferente da representação individual de “*PORTA + ABRIR*”.

2.1.3.11. Artigos Definidos e Indefinidos

Os artigos definidos e indefinidos não existem na LGP.

2.1.3.12. Não Existência de Gesto Para uma Palavra

No caso em que não exista um gesto para uma palavra específica (na tradução de português para os termos da LGP), deve-se tentar procurar utilizar outros termos que consigam representar o que se pretende, ou seja, procurar obter um sinónimo para esse termo inexistente. Em último caso, não sendo possível construir a frase original de modo a permitir a utilização de um sinónimo ou algo equivalente, soletra-se a palavra em questão.

2.2. Machine Learning e Processamento de Linguagem Natural

O Machine Learning (ML) é considerado um dos ramos da Inteligência Artificial (IA) que se foca na utilização de dados e construção de algoritmos que consigam replicar a forma de como os seres racionais (ex. humanos) aprendem, melhorando constantemente ao longo do tempo o seu processo de aprendizagem [12]. Existem vários algoritmos de ML estruturados no conceito de aprendizagem supervisionada, não supervisionada e por reforço, sendo que não existe um único que sirva para todos os objetivos da IA, como a classificação, previsão, otimização, segmentação, reconhecimento, etc. Ou seja, certos algoritmos de ML terão melhores desempenhos sobre outros consoante vários fatores, como as características dos datasets e os objetivos que se pretendem atingir com a aplicabilidade das abordagens de ML [13]. No âmbito de aplicabilidade deste trabalho, embora não abordando todos os temas, seguidamente, abordam-se alguns termos relacionados com Machine Learning importantes e utilizados neste trabalho.

2.2.1. Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA) são algoritmos associados ao paradigma conexionista da aplicabilidade da Inteligência Artificial, sendo capazes de reconhecer relações entre dados, automaticamente, através de processos que se assemelham à forma como o cérebro humano processa a informação. São compostas por vários nós, separados por várias camadas e ligados entre si de modo a partilharem dados entre eles, representando os neurónios e as sinapses do cérebro humano, respetivamente. As RNA podem ser englobadas nas abordagens de ML tipicamente para processos de classificação, previsão (aprendizagem supervisionada) e descoberta de padrões e aglomeração de dados (aprendizagem não supervisionada). Tipicamente as RNA representam-se utilizando nós e pesos, que representam a ligação entre os nós. A Figura 1 representa uma rede neuronal do tipo *feed-forward*, o tipo básico de redes neuronais onde os dados apenas se movem numa direção, do input para o output, sem que o resultado de um dos nós de uma camada possa afetar os resultados de nós de camadas anteriores ou da mesma camada, apenas das seguintes [12] [14]. Este tipo de redes não possui memória, isto é, as suas operações são independentes umas das outras.

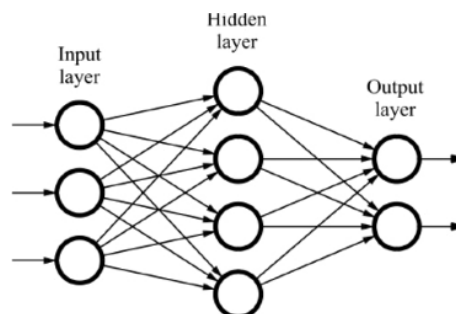


Figura 1 - Exemplo Rede Neuronal [15].

As redes neuronais possuem a capacidade de se adaptarem consoante os dados que lhes são fornecidos e os que são esperados dela. Esta capacidade de adaptação é conseguida através do ajuste automático dos pesos das ligações de um dado nó da rede, de forma que a combinação de todas essas ligações e nós forneça o resultado esperado para os valores de entrada, sem que outros valores já aprendidos pela rede sejam perdidos. Isto é conseguido através de algoritmos de otimização, como por exemplo, o *Backpropagation* [16] e o *Gradient Descent* [17].

2.2.2. Recurrent Neural Networks

As Recurrent Neural Networks (RNN) [18] são um tipo de rede neuronal mais complexa, caracterizada por possuir uma “memória interna”, permitindo que seja mantida informação temporal em memória e assim considerar essa informação nos resultados de iterações seguintes. Esta memória é conseguida através da ligação de nós a si próprios. Esta capacidade de manter informação em memória permite que este tipo de redes trabalhe com dados sequenciais e temporais, sendo o tipo de rede neuronal indicado para trabalhar com processamento de linguagem natural e problemas de sequência-para-sequência, onde se encontra a tradução de texto entre línguas. A Figura 2 apresenta uma visão geral de uma RNN, sendo essa visão geral depois desdobrada, de forma a representar os valores que passaram pelas várias camadas da rede (representada pelo ‘H’) anteriormente considerados na obtenção do resultado da iteração seguinte da rede.

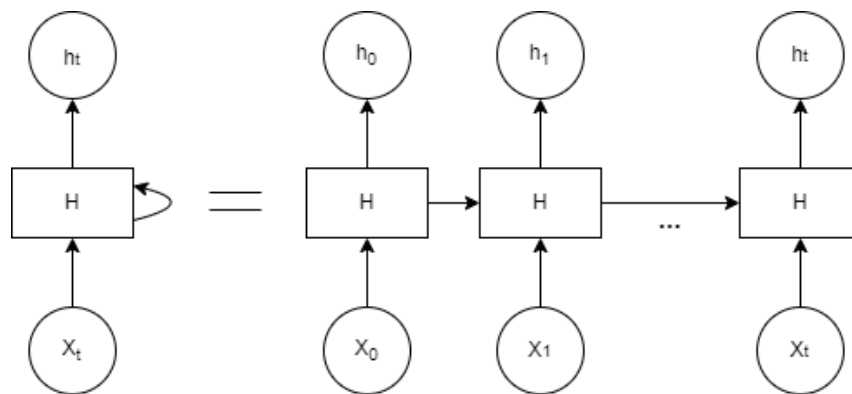


Figura 2 – Visão geral de uma RNN (Adaptada de [19]).

As redes neurais podem ser utilizadas para problemas de *um-para-um*, *um-para-muitos*, *muitos-para-um*, *muitos-para-muitos*. *Um-para-um* significa que a rede possui um único input e um único output, *um-para-muitos* significa que a rede possui um input e pode gerar vários outputs (exemplo: geração automática de legendas para imagens – input: 1 imagem, output: uma ou mais palavras), *muitos-para-um* significa que a rede pode possuir vários inputs mas apenas terá um output (exemplo: análise de sentimentos, input: frase, output: bom/mau/...), e por fim, *muitos-para-muitos* que significa que a rede pode receber vários valores como input e produzir vários valores como output (exemplo, tradução de texto). No entanto, RNNs simples sofrem de *vanishing* e *exploding gradient*. Estes problemas surgem quando os valores dos ajustes realizados pelo mecanismo de otimização e correção dos pesos das ligações da rede

tornam-se muito baixos, não afetando o peso das ligações entre os nós, impossibilitando a aprendizagem de relações temporais entre inputs distantes, ou quando se torna muito alto, causando alterações muito grandes aos pesos [20].

2.2.3. Long-Short Term Memory

A Long-Short Term Memory (LSTM) [21] é um tipo de RNN mais complexo que corrige os principais problemas das RNN simples, evitando assim o *vanishing gradient* e o *exploding gradient*. A LSTM possui nós mais complexos que, além de evitar os problemas referidos, melhoram os resultados da rede. Numa RNN simples, as iterações seguintes têm acesso ao valor do input e ao estado (*hidden state anterior*) das iterações anteriores. A aplicação da combinação destes dois valores sobre uma função de ativação (normalmente tanh para RNN simples) resulta no novo estado (*hidden state atual*), que será utilizado na iteração seguinte da rede, ou será o resultado final da aplicação da rede para um dado problema. Numa LSTM, não existe apenas a aplicação de uma função de ativação à combinação de valores anteriores e do atual, mas sim uma combinação de vários blocos designados por *gates* que controlam qual a informação a ser esquecida ou a manter (*forget gate*, *input gate* e *output gate*) durante o treino da rede. Neste tipo de rede não é passado apenas o *hidden state* entre as várias iterações, mas também um outro estado, designado por *cell state*. A diferença é que o *hidden state* mantém informação relativa a iterações anteriores, sendo que esta informação pode ser reescrita ou perdida a qualquer momento, enquanto o *cell state* possui o objetivo de manter conhecimento sobre os valores mais importantes que já passaram pelas iterações anteriores da rede, sem os perder tão facilmente [22]. A Figura 3 mostra a diferença entre as células das RNN e LSTM.

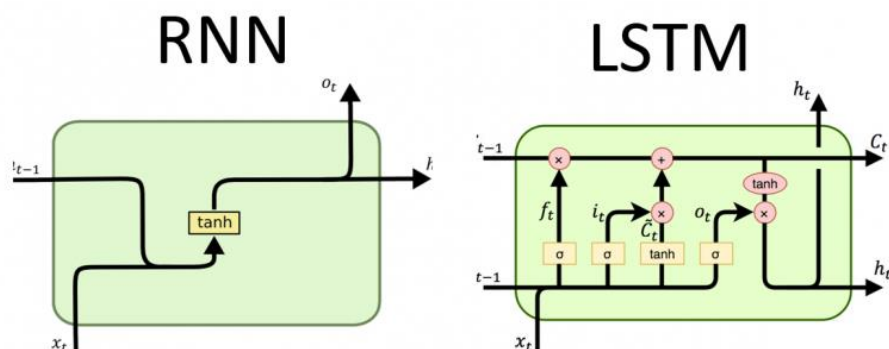


Figura 3 – Células da RNN simples e LSTM [23].

2.2.4. Gated Recurrent Unit

A Gated Recurrent Unit (GRU) [24] é um outro tipo de RNN que, tal como as LSTM, possuem o objetivo de eliminar os problemas com as RNN simples. As GRUs possuem células mais simples, quando comparadas com as LSTM, já que apenas possuem duas *gates*, uma designada por *update gate* e a outra por *reset gate*, ao contrário das três *gates* das últimas. A *update gate* das GRU é equivalente à *input gate* das LSTM e a *reset gate* é equivalente à *forget gate* das LSTM. Comparada às LSTM, a *output gate* desaparece, visto ser passada entre as várias iterações apenas um *hidden state*. No entanto, as duas *gates* das GRU permitem determinar qual a informação a manter (mais relevante) e qual a informação a esquecer (menos relevante). A Figura 4 apresenta as células das LSTM e das GRU.

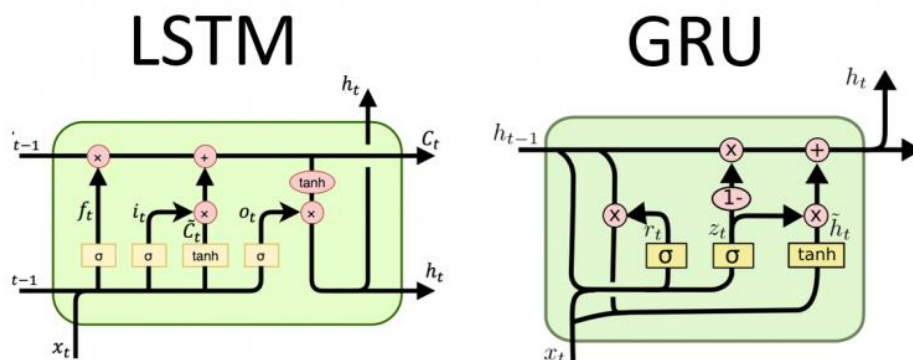


Figura 4 - Células da LSTM e GRU [23].

Analisando as duas, consegue-se facilmente determinar que as GRU deverão ser mais eficientes que as LSTM, já que utilizam menos recursos computacionais, devido a possuírem apenas duas *gates*. Em [25], os autores realizam uma comparação entre as LSTM e as GRU para a tarefa de tradução (de árabe para inglês) numa arquitetura Encoder-Decoder. Através dos resultados que obtiveram, determinaram que as LSTM alcançaram um resultado BLEU superior, enquanto as GRU demoraram menos tempo a serem treinadas. No entanto, em termos de BLEU entre as LSTM e as GRU, a vantagem das LSTM foi apenas ligeiramente superior. Já [26], testaram as LSTM e as GRU para *Sequence Modeling*, tendo determinado que estas são bastante superiores às RNN simples, mas que entre elas, não conseguiram encontrar resultados substanciais, ao ponto de conseguirem determinar a melhor.

2.2.5. Encoder-Decoder

A Encoder-Decoder [27] corresponde a uma arquitetura composta por duas redes neurais recorrentes (RNN), uma designada por Encoder, que converte uma sequência de símbolos para um vetor de tamanho fixo, e a segunda designada por Decoder, que converte esse vetor noutra sequência de símbolos diferente. Ambas as redes são treinadas ao mesmo tempo, já que o input da segunda será o output da primeira. Os autores da Encoder-Decoder, ao utilizarem a arquitetura proposta, composta por RNNs que possuem *reset* e *update gates* (GRU), para computar probabilidades estatísticas sobre as frases utilizadas, conseguiram melhorar os resultados dos sistemas, apenas baseados em estatísticas. Em [28], os autores utilizaram uma Encoder-Decoder na tradução direta de uma sequência para outra, em vez de recalcularem novos valores a utilizar em sistemas baseados em estatísticas. Estes últimos utilizaram LSTMs como Encoder e Decoder do seu sistema e, mesmo com um sistema pouco otimizado, alcançaram resultados superiores aos sistemas baseados em estatísticas. No entanto, a conversão, no Encoder, de uma sequência de tamanho variável para um vetor de tamanho fixo pode ser problemática, principalmente em sequências muito grandes, pois pode existir a perda de informação importante na passagem do Encoder para o Decoder. Isto é demonstrado em [29], onde se apresenta uma solução que ficou conhecida como mecanismos de atenção (*attention*). Este mecanismo de atenção é proposto como um método para alinhar e traduzir frases. O alinhamento consiste em determinar que partes da frase de input são importantes para cada palavra do output, e a tradução consiste em utilizar a informação relevante para produzir o output. Adicionalmente, utilizaram uma RNN GRU bidirecional no input, e concluem ter alcançado melhores resultados que propostas anteriores, principalmente para frases mais longas.

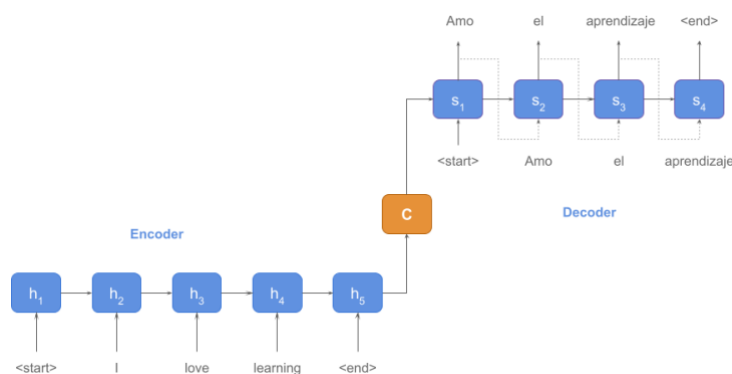


Figura 5 - Representação de Encoder-Decoder básica [30].

A Figura 5 apresenta uma visão gráfica da arquitetura Encoder-Decoder simples. O Encoder recebe uma palavra de cada vez, passando informação para iterações seguintes. No final, gera um vetor (representado por C), vetor esse usado como input do Decoder, que tendo em consideração os valores do vetor, e o output anterior (se houver) gera o próximo output da rede. Um Encoder-Decoder com mecanismo de atenção guarda informação a cada iteração do Encoder, e utiliza essa informação no Decoder, como representado na Figura 6.

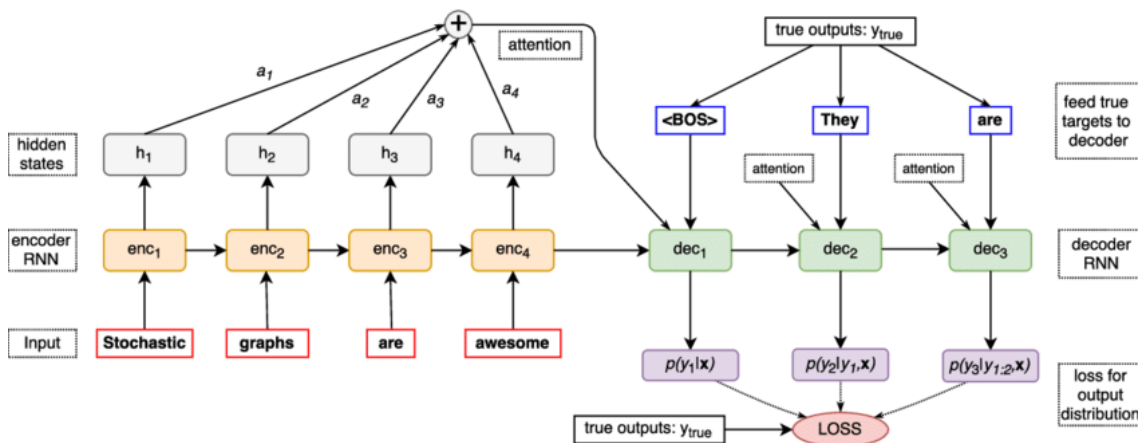


Figura 6 - Representação de um Encoder-Decoder com mecanismo de atenção [31].

Em [25], os autores testaram a arquitetura Encoder-Decoder com uma variação de LSTM, GRU e as equivalentes bidirecionais, com mecanismo, tendo alcançado melhores resultados, em termos de resultado BLEU, utilizando LSTM como Encoder e Decoder. Considerando o tempo de treino, a GRU bidirecional como Encoder e LSTM como Decoder foi a mais rápida a ser treinada. Em termos médios de resultado BLEU e tempo de treino, GRU bidirecional como Encoder e LSTM bidirecional como Decoder alcançaram melhores resultados.

2.2.6. Embeddings

As redes neurais podem ser utilizadas para vários problemas, sejam eles problemas que utilizem números ou palavras, como é o caso da tradução entre línguas. No entanto, apesar de poderem ser utilizadas para resolver problemas desse tipo, as redes neurais em si apenas compreendem, isto é, apenas trabalham com números. As *Embeddings* são uma forma de conversão de palavras para vetores numéricos de tamanho fixo (com tamanho definido), permitindo assim que a rede 'compreenda' as palavras e consiga trabalhar com elas, indiretamente. Uma outra abordagem utilizada

para converter palavras para vetores numéricos representativos é o método *One-Hot Encoding*, no entanto, o método *One-Hot Encoding*, apesar de fácil compreensão e utilização, apresenta um problema relacionado com a quantidade de palavras que se pretende codificar através do mesmo. O tamanho do vetor gerado pelo método *One-Hot* é, obrigatoriamente, igual à quantidade de palavras a codificar. Se for necessário codificar 10.000 palavras, cada uma será codificada num vetor de tamanho 10.000, preenchido com zeros (0) e um único um (1) que representa qual é a palavra. As *Embeddings* resolvem isto, pois convertem as palavras para vetores de números reais em vez de apenas zeros e uns.

2.2.7. BLEU - BiLingual Evaluation Understudy

A BiLingual Evaluation Understudy (BLEU) em [32] é uma das primeiras e principais métricas de avaliação de sistemas de tradução automática. Esta métrica alcança o seu resultado através da contagem da quantidade de sequências de palavras, designadas por *n-grams*, numa dada frase, que se encontram na ou nas frases que eram esperadas, no caso de a tradução automática comparar a quantidade de sequências de palavras que existem na frase traduzida pelo sistema e nas frases que representam as traduções corretas. Quanto maior a quantidade de sequências de palavras que se encontram, tanto na frase traduzida como na ou nas esperadas, melhor é a tradução dada pelo sistema, como demonstrado pelos criadores do BLEU, em [32]. Esta métrica apresenta resultados entre 0 e 1, 0 para más traduções, 1 para traduções idênticas às traduções esperadas. A Figura 7 mostra uma escala que representa as avaliações BLEU. Acima de 30%, as traduções costumam ser aceitáveis, acima de 50% já são consideradas traduções bastante boas e fluentes. Deve-se ter em conta que nem as traduções manuais de um conjunto de frases realizadas por humanos conseguirão chegar ao resultado BLEU 1, já que diferentes pessoas poderão traduzir a frase de forma diferente, não se conseguindo chegar a um consenso do que se entende por tradução perfeita.

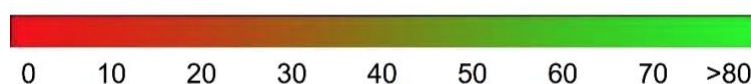


Figura 7 - Escala de avaliação BLEU [33].

Sempre que a frase traduzida é menor que a frase esperada, atribui-se uma penalização, conhecida como *Brevity Penalty*, ao resultado BLEU, já que uma frase mais

curta poderia alcançar 1 como resultado, sem possuir todas as palavras necessárias. Um exemplo poderia ser a seguinte frase esperada – *O gato está deitado no tapete.* – frase traduzida – *O gato está deitado.* – O BLEU desta tradução, sem a penalização, seria 1, já que as sequências de palavras da frase traduzida, utilizadas pelo BLEU para avaliação, são sequências que existem na frase esperada. Esta penalização é calculada tendo em conta todo o corpus a ser avaliado, já que para frases curtas, a penalização seria muito elevada.

2.2.8. METEOR - Metric for Evaluation of Translation with Explicit ORdering

A Metric for Evaluation of Translation with Explicit ORdering – Meteor [34] é outra métrica de avaliação de sistemas de tradução automática, que avalia através da procura de correspondências de palavras entre a frase traduzida e a frase esperada, em vez de utilizar sequências de palavras como o BLEU. Além disso, no BLEU, quando existem várias frases esperadas para uma dada tradução, o resultado será obtido pela procura das sequências de palavras no conjunto de todas as frases esperadas, em vez de comparar cada uma dessas frases do conjunto de frases esperadas individualmente e obter o melhor resultado. Na Meteor isso não acontece, já que quando existe mais que uma frase que representa a tradução esperada, as frases desse conjunto serão individualmente comparadas com a frase traduzida pelo sistema, e o melhor resultado será o considerado. Os autores comparam o METEOR com o BLEU, concluindo que o METEOR mostra maior correlação com traduções humanas. No entanto, em [35] consideram o contrário, com o BLEU a mostrar melhores resultados na tradução de árabe para inglês, no entanto, indicam que esta discrepância não era de todo esperada, culpando a versão do METEOR utilizada, dizendo que não é totalmente compatível com árabe.

2.2.9. Natural Language Processing

A Natural Language Processing - NLP [36] é uma subárea da Inteligência Artificial que se dedica ao estudo da Linguagem Natural e como a processar computacionalmente para tarefas como a de tradução automática, reconhecimento de fala, análise de sentimentos, entre outras. De forma a processar texto ou fala, a NLP utiliza uma

combinação de modelos linguísticos baseados em regras, modelos estatísticos e modelos de machine learning e de deep learning.

2.3. Tradução Automática

A Tradução Automática (*Machine Translation - MT*) é uma das áreas mais trabalhadas que utiliza técnicas de NLP [37]. É a área que procura utilizar software para a tradução de texto ou áudio de uma língua para outra. Existem várias abordagens para a tradução automática, tais como as abordagens RBMT (*Rule Based Machine Translation*), SMT (*Statistical Machine Translation*) e NMT (*Neural Machine Translation*), sendo também comum utilizarem-se abordagens híbridas, que consistem na utilização de mais de um tipo de abordagem, de modo a procurar congregar as vantagens das várias abordagens utilizadas e mitigar as desvantagens da utilização de apenas uma abordagem [38].

2.3.1. RBMT – Rule Based Machine Translation

A tradução baseada em regras (*Rule Based Machine Translation*) [38] é um tipo de tradução que se baseia na utilização de regras gramaticais que necessitam de ser manualmente definidas, para que seja efetuada a tradução de uma língua para outra. O grande problema desta abordagem é que tudo o que é necessário para a correta tradução entre as línguas deve ser explicitamente definido.

2.3.2. SMT – Statistical Machine Translation

A tradução baseada em estatística (*Statistical Machine Translation*) [38] é um método que se baseia na utilização de machine learning para a construção de modelos estatísticos que avaliam a relação entre palavras. Estes modelos são gerados a partir de corpus que possuam um número muito grande de traduções, sendo esta uma grande vantagem e também desvantagem deste método. Se por um lado, não existe a preocupação em implementar e procurar todas as regras gramaticais necessárias, quando comparado com as RBMTs, por outro lado é necessário que o corpus exista para que o modelo consiga estimar corretamente os seus parâmetros.

2.3.3. NMT – Neural Machine Translation

A tradução baseada em redes neurais (*Neural Machine Translation*) [38] é o método que consiste na utilização de redes neurais para a realização das traduções. A grande vantagem de utilização de redes neurais para a realização da tradução prende-se com o facto deste tipo de redes possuir a capacidade de considerar todas (ou quase todas) as características das partes já analisadas de uma frase, ao inferir o resultado na língua de destino, conseguindo resultados superiores quando comparado com as anteriores. No entanto, é necessária a existência de um bom corpus de traduções, tal como nas abordagens SMT.

2.3.4. HMT – Hybrid Machine Translation

A abordagem híbrida (*Hybrid Machine Translation*) [39] é uma abordagem que consiste na utilização de várias abordagens únicas na construção do sistema de tradução automática. Esta abordagem procura combinar as vantagens das várias abordagens únicas e, conseqüentemente, mitigar as desvantagens das mesmas. Uma das formas mais comuns da construção de uma abordagem híbrida é na utilização de um método RBMT e noutro que seja construído a partir de um corpus, SMT ou NMT. Por exemplo, o tradutor RBMT poderia ser utilizado para a construção de uma tradução inicial, e um tradutor NMT ou SMT utilizado para melhorar essa primeira tradução, ou, em sentido contrário, utilizar regras para processar os dados enviados para a NMT ou SMT, bem como processar os dados resultantes dessas últimas.

3. ESTADO DA ARTE

Este capítulo apresenta o estudo do estado da arte de aplicações e algoritmos utilizados na tradução de linguagem natural em língua gestual, encontrando-se divididos em duas subsecções, dependentes do estilo de abordagem utilizada nos trabalhos estudados. A primeira subsecção apresenta trabalhos que se baseiam em regras gramaticais (RBMT – *Rule-Based Machine Translation*) que se focam na utilização de características morfológicas, sintáticas, semânticas, estrutura e significado da frase, e ainda trabalhos que se baseiam em análise estatística (STM – *Statistical Machine Translation*) que se focam na criação de modelos estatísticos, criados a partir de traduções já existentes. A segunda subsecção apresenta trabalhos que foram desenvolvidos com recurso a redes neuronais (NMT – *Neural Machine Translation*) para a realização da tradução da língua natural a termos da língua gestual.

3.1. RBMT e SMT – Tradução Baseada em Regras e Modelos Estatísticos

Nas últimas décadas vários trabalhos têm vindo a ser apresentados na tradução baseada em regras e modelos estatísticos, como os que serão de seguida apresentados.

Alrikabi e V. Hafner [40], propõe um sistema para a tradução de árabe para a Língua Gestual Americana (ASL - *American Sign Language*). O sistema proposto começa por verificar se existem símbolos desconhecidos no texto, realizando uma análise morfológica, palavra a palavra, para determinar a raiz da palavra e as características acerca da mesma, através de padrões morfológicos. De seguida, é realizada uma análise sintática do resultado da análise morfológica, para preencher características morfológicas incompletas e determinar as características incertas. Adicionalmente, são eliminadas as características erradas, e no final da análise, é construída uma árvore de análise sintática. Essa árvore de análise sintática é depois convertida numa árvore equivalente da ASL. Após a execução do algoritmo, caso os termos da árvore ASL existiam no dicionário de gestos do sistema, um robô representa esses gestos. Caso

contrário, é realizada uma análise às características do termo inexistente, de modo a tentar encontrar uma forma alternativa de representar o termo. A Figura 8 apresenta uma visão geral da arquitetura do sistema.

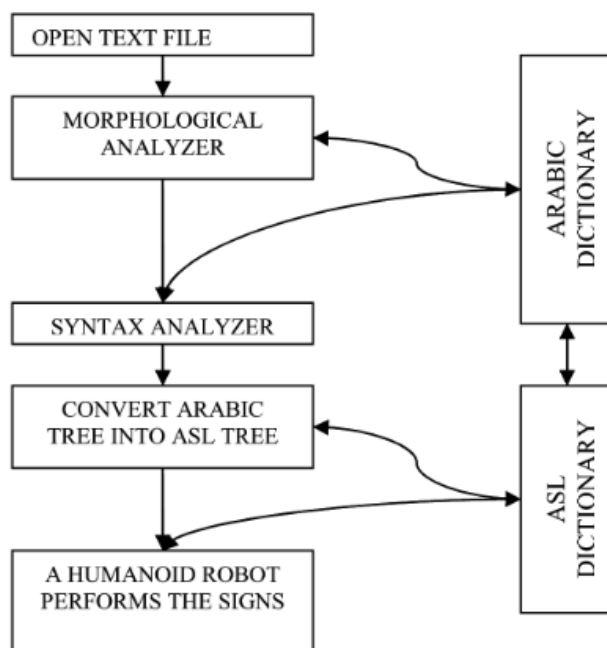


Figura 8 - Algoritmo de [40].

M. Grif e Y. Manueva [41], apresentaram um sistema para a tradução automática de russo para a língua gestual Russa. O seu algoritmo começa por realizar uma análise ao texto, que consiste em realizar a análise morfológica, análise sintática e análise semântica. Neste trabalho, os autores focam-se na análise semântica, que consiste em descobrir as alternativas para cada palavra da frase, processamento da fraseologia, processamento de preposições, confirmação do significado real da palavra e procura dos termos correspondentes às palavras. O primeiro passo inicia-se pela descoberta de todas as traduções possíveis para cada palavra, e significados dessas traduções, sendo que no passo seguinte, o de processamento da fraseologia, procuram eliminar traduções e significados que não façam sentido, tendo em consideração a frase original. No passo de processamento de preposições, pretende-se descobrir o significado das preposições utilizadas na frase e no passo da confirmação do significado, procura-se confirmar o significado real da palavra (caso esta possua vários significados diferentes). Quando todas as palavras tiverem um único significado lexical, procuram-se os termos associados a cada palavra.

K. Saija, S. Sangeetha e V. Shah [42] apresentam um tradutor de voz inglesa para a língua gestual indiana (ISL – Indian Sign Language).

No.	Spoken Sentence	Model Generated ISL gloss	True ISL gloss
1	Hey, what is your name?	HELLO WHAT YOUR NAME	HELLO WHAT YOUR NAME
2	This is a huge campus.	THIS BIG GROUND	THIS BIG SCHOOL or THIS BIG GROUND
3	India is a neat and clean country.	INDIA IS A CLEAN AND CLEAN COUNTRY	INDIA IS A CLEAN COUNTRY
4	I prefer bread and butter in breakfast.	*I LIKE _ AND _ IN MEAL	*I LIKE _ AND _ IN MEAL

*Underscore () denotes no replacement found.

Figura 9 - Exemplos de traduções de [42].

Este trabalho possui uma componente de transcrição de texto baseada no método estatístico “Hidden Markov Model” para a realização do reconhecimento da voz. Depois de possuírem a transcrição de uma dada frase, realizam um tratamento de texto a essa transcrição antes de a converterem para ISL. Esta fase de tratamento de texto começa pela remoção de pontuação, remoção de “stop words” (palavras vazias) que não alterem o significado da frase (por exemplo “a”, “o”, “em”, etc.). De seguida, é feita a tokenização do texto, sendo que as palavras resultantes da tokenização passam ainda pela fase de stemização, que é o processo de diminuir as palavras à sua raiz (ex: {Ferro, Ferreiro, Ferrão} -> “Ferr”). Terminado o tratamento inicial do texto, é realizado o mapeamento das palavras ao termo correspondente. Se a base de dados de termos do sistema não possuir o termo para uma dada palavra, o WordNet, dicionário inglês online, é utilizado para procurar palavras sinónimas da palavra que não possui termo associado na base de dados. Dos sinónimos obtidos, é selecionado o que tenha o significado mais próximo da frase original e que exista no dicionário de termos ISL do sistema. A Figura 9 apresenta alguns exemplos de traduções deste sistema.

Em [43], T. Dasgupta e A. Basu também propõem um sistema para a tradução de inglês para ISL. O sistema proposto possui quatro componentes principais, sendo elas, o processamento do texto, a representação da estrutura gramatical, a geração da frase ISL e a verificação da síntese ISL.

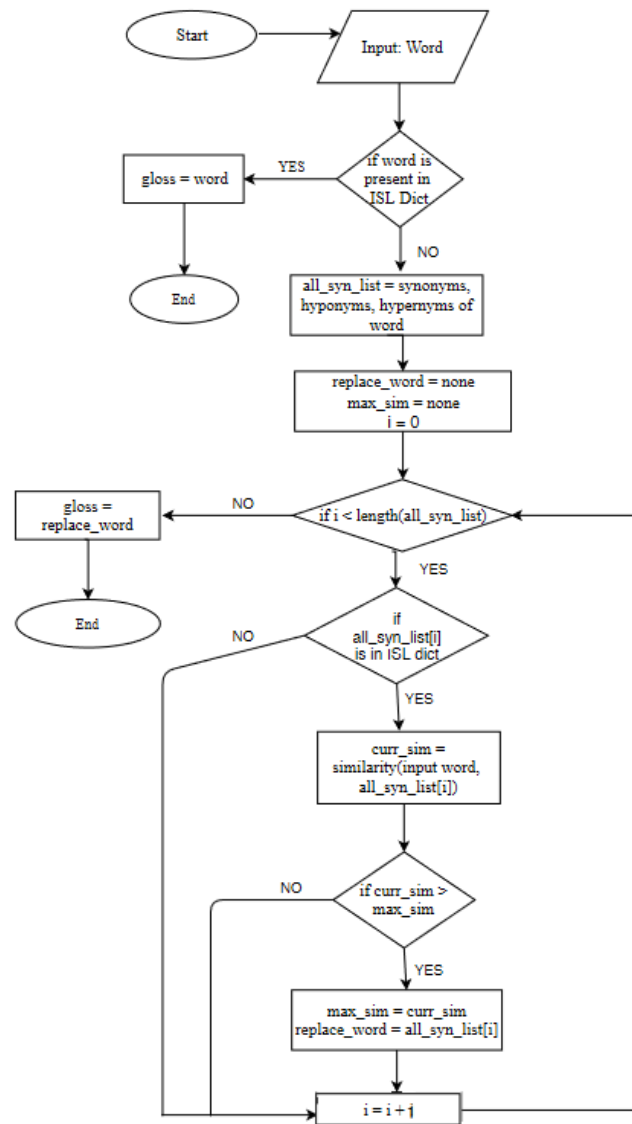


Figura 10 - Algoritmo de [43].

O processamento do texto inicia-se pela identificação de expressões temporais e dialetos específicos da linguagem e que apenas existem na mesma. De seguida, a frase é representada numa estrutura gramatical onde se identificam características sobre a mesma, obtidas através de uma análise morfológica. Por fim, a estrutura gerada da frase original é convertida para uma estrutura representativa da frase em ISL. Neste passo são realizados o mapeamento de palavra a termo e a reordenação desses termos, de modo que a frase final cumpra com a estrutura da ISL. A Figura 12 apresenta a arquitetura básica do sistema deste trabalho.

Santos [44] apresentam um sistema de tradução baseado em regras de língua textual portuguesa para a língua gestual portuguesa (LGP). Este trabalho foca-se no

desenvolvimento de uma segunda versão do algoritmo utilizado pelo sistema. A Figura 11 apresenta o algoritmo original do sistema, sendo que cada parte deste algoritmo encontra-se explicada na seguinte lista:

- Error Correcting + Normalization: correção de erros e formatação do texto (colocar tudo em minúsculas, por exemplo);
- Tokenizer: divisão da frase em palavras;
- POS-Tagger: classificação de cada palavra (se é nome, verbo, ...);
- Name Entity Extraction: identifica nomes que são entidades (utiliza uma lista com nomes);
- Stemmer: deteta prefixos e sufixos, e adiciona as palavras para representar esses prefixos ou sufixos, e obtém a raiz das palavras;
- Lexical Transfer: converte a lista de palavras para os termos da LGP, através de um dicionário;
- Structure Transfer: reordena as palavras para cumprir com a estrutura da LGP;
- Lookup: lista de palavras e gloss equivalente;

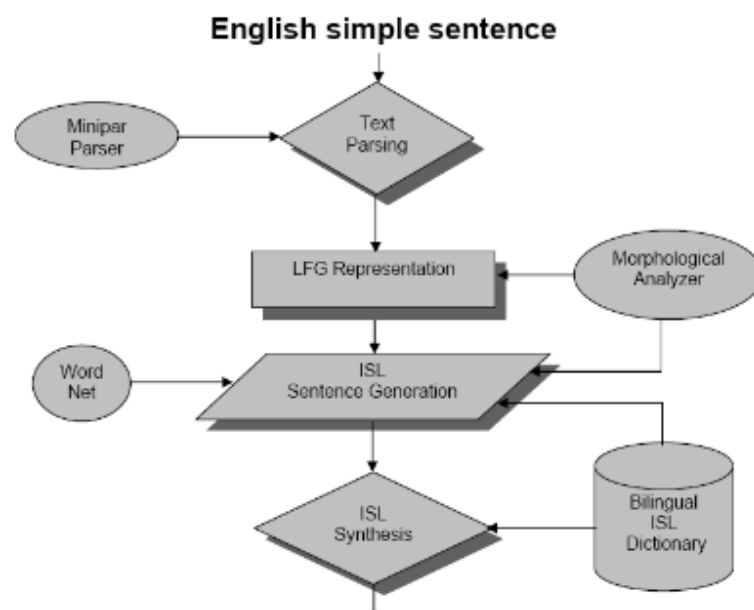


Figura 12 - Algoritmo de [43].

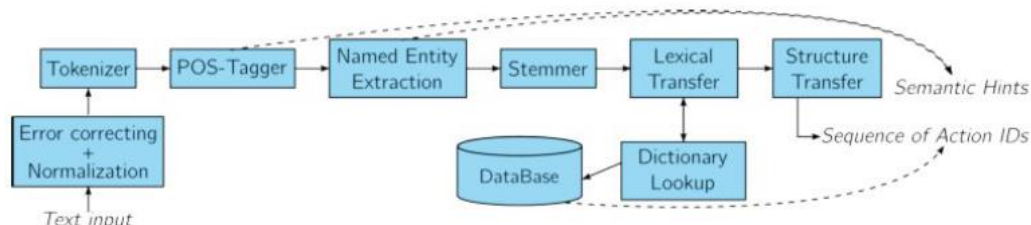


Figura 11 - Algoritmo original de [44].

O algoritmo mais recente é baseado no original, sendo que neste, a frase começa por ser tokenizada, isto é, dividida em palavras, onde cada palavra é colocada num

objeto. Para cada objeto, a acentuação e a pontuação são removidas, e as palavras são convertidas em letras minúsculas, de modo a que todas as palavras sigam o mesmo formato. De seguida, são analisadas as terminações das palavras (por exemplo, uma palavra acabada em “inha” indica algo “pequeno” e “feminino”. Na LGP é necessário adicionar essas duas palavras à frase), e por fim são filtradas as “stop words” (palavras que não são relevantes), principalmente artigos (não são utilizados na LGP). Seguidamente, é efetuada uma análise morfológica a cada palavra, de modo a obterem-se características da mesma, e por fim as palavras são reordenadas de modo a cumprirem com a estrutura da LGP, e para cada objeto representativo da palavra é adicionada a sua raiz. As palavras resultantes são depois mapeadas, através de um dicionário, ao respetivo termo que será depois representado por um avatar. Mais recentemente, no seguimento deste trabalho, em [45] é apresentada uma ferramenta para a criação dos gestos representados pelo avatar do sistema.

T. Vichyaloetsiri e P. Wuttidittachotti [46], os autores apresentam um sistema web que disponibiliza um web service que quando recebe uma frase, traduz a mesma para língua gestual e devolve um ficheiro .GIF animado, com a representação dos gestos da respetiva frase. Numa visão geral do sistema, o mesmo é composto por 5 partes:

1. Web Service: responsável por receber o texto e passá-lo para a secção seguinte;
2. Processamento do Texto (que comunica com o Dicionário): responsável por manipular a frase recebida, de acordo com as regras definidas no sistema;
3. Dicionário;
4. Procura: responsável por procurar conjuntos de imagens que sejam a representação em língua gestual das palavras a traduzir;
5. Produção de Imagens: responsável por compilar as imagens num ficheiro .GIF, ficheiro esse que será depois devolvido ao requisitante pelo web service.

A Figura 13 apresenta uma visão geral do sistema.

N. Aouiti [47] propõem um sistema para a tradução de árabe para a língua gestual árabe (ASL). O algoritmo utilizado neste sistema começa por realizar o pré-tratamento da frase, onde esta é dividida em palavras, e cada palavra é convertida para a sua forma canónica, e no final é efetuada a análise morfológica de cada palavra. De seguida, o

sistema realiza uma análise sintática, de modo a verificar se a frase está estruturalmente “correta”, sendo depois realizada uma análise semântica e uma análise pragmática. Depois deste tratamento, é realizado o mapeamento das palavras nos respetivos termos, que serão representados por um avatar. A Figura 14 oferece uma representação do sistema em questão.

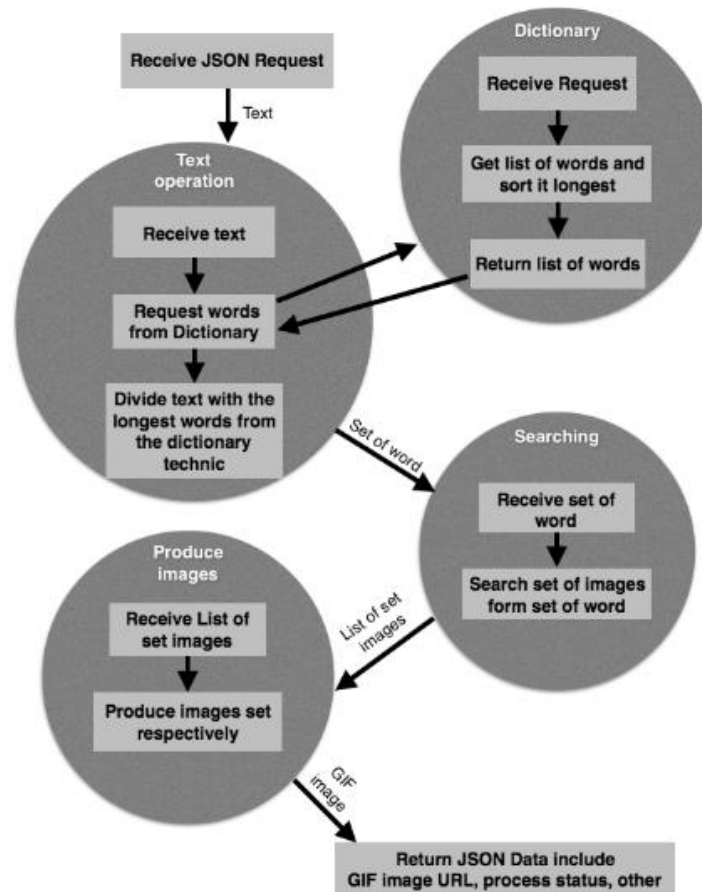


Figura 13 - Visão geral do sistema de [46].

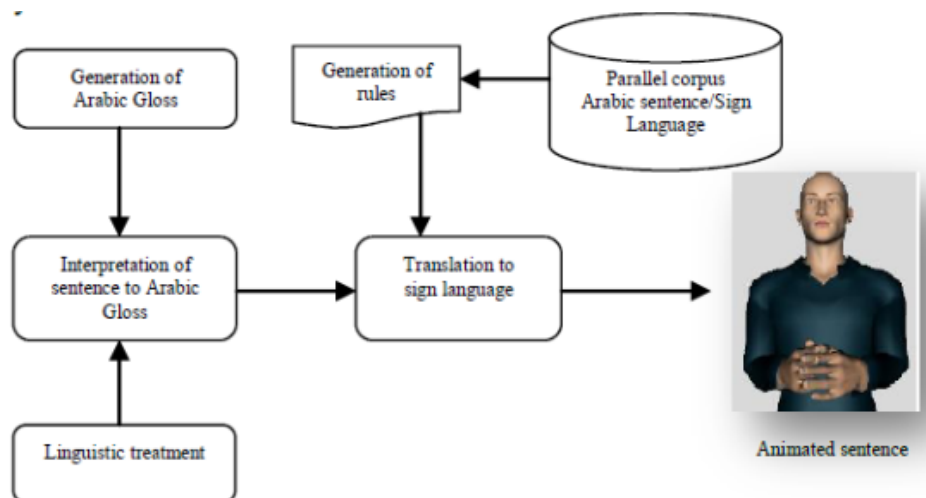


Figura 14 - Arquitetura geral de [47].

Em [48], H. S. Al-Khalifa apresenta um sistema para a tradução de árabe para língua gestual árabe em sistemas móveis. Neste sistema, os autores apresentam um algoritmo que começa por realizar a tokenização da frase, e para cada palavra resultante da tokenização, é obtida a sua raiz. A raiz da palavra é obtida através de um ficheiro de “raízes”, que possui o mapeamento entre palavras e a respetiva raiz da mesma. Se a palavra existir no ficheiro de raízes é considerada processada e o gesto da raiz utilizado na representação final do texto em língua gestual. Caso contrário, são aplicadas regras gramaticais sobre a palavra, de forma a tentar encontrar-se a raiz da mesma, utilizando variações desta. Se mesmo após a aplicação das regras, nenhuma das variações da palavra existir no ficheiro de raízes, a palavra será soletrada. Por fim, os gestos identificados necessários representar, são representados por um avatar. A Figura 15 apresenta o algoritmo de procura da raiz de cada palavra da frase introduzida.

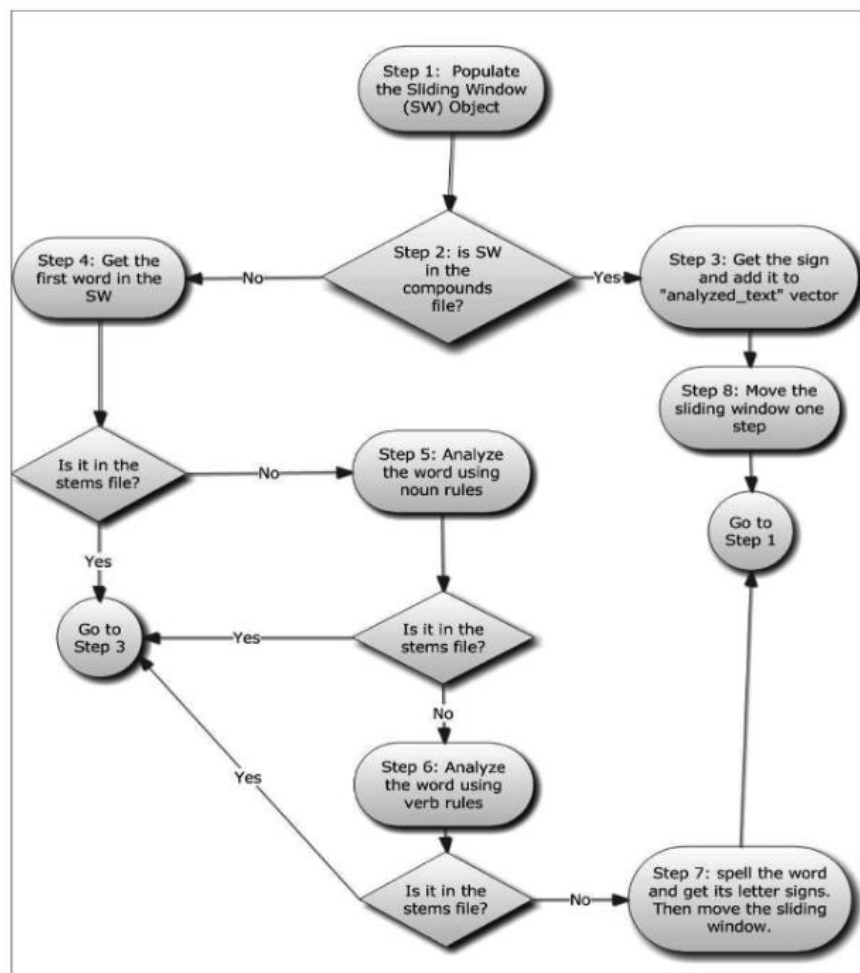


Figura 15 - Algoritmo de procura de raízes e gestos de [48].

B. Sarkar, K. Datta, *et al.* [49] apresentam uma solução para a conversão de Bangla para língua gestual de Bangla (BaSL). Neste sistema, o texto é processado através de regras gramaticais, e, se necessário, é reordenado para que cumpra com as necessidades da BaSL. Depois desse processamento, é procurado numa base de dados de vídeos, o vídeo que corresponde às palavras a serem representadas. Nesta língua gestual, as palavras da linguagem natural podem representar duas palavras na língua gestual.

D. Kouremenos, *et al.* [50], os autores apresentam um sistema para a conversão da linguagem natural grega para a língua gestual grega. O sistema apresentado é composto por três módulos principais, o módulo de análise e tratamento de texto, o módulo de mapeamento entre a linguagem natural e a língua gestual, e por fim o módulo de representação dos termos. O módulo de análise e tratamento do texto realiza a análise morfológica ao texto e a lematização das palavras. O módulo de mapeamento entre linguagem natural e língua gestual funciona à base de regras, regras que são aplicadas ao resultado do módulo de análise e tratamento para fazer com que a frase cumpra com as regras da língua gestual grega. A Figura 16 apresenta o algoritmo deste sistema.

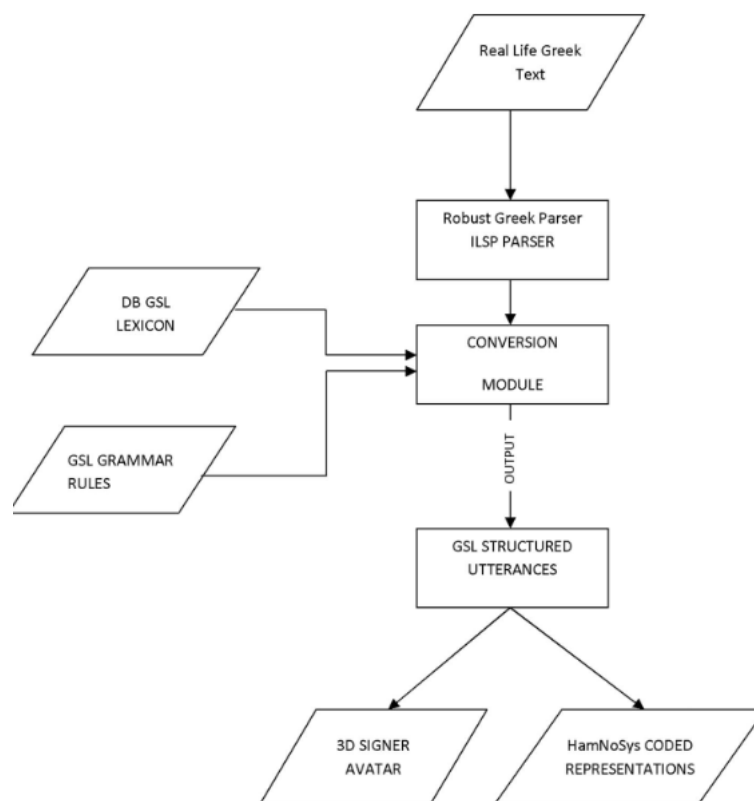


Figura 16 - Algoritmo de [50].

P. Escudeiro, *et al.* [51], os autores apresentam uma aplicação bidirecional para a tradução, com uma parte que traduz de LGP para Texto, através da captura dos movimentos através de uma câmara e posterior classificação por parte de uma rede neuronal, e a outra parte do tradutor, a tradução de português para a respetiva Glosa LGP, sendo esta representada via avatar. Focando apenas na segunda parte, de tradução de português para a glosa LGP, não existe qualquer referência à aplicação de algum tipo de método para tradução, isto é, não existe referência à aplicação de regras, à utilização de redes neuronais, ou outros métodos. Utilizando e testando o avatar de demonstração disponibilizado no website deste trabalho¹, verifica-se que a tradução aparenta ser “palavra-por-palavra”, não sendo aplicada qualquer regra que ajuste o texto original às características gramaticais da glosa da LGP.

3.1.1. Conclusões RBMT e SMT

Durante a análise dos trabalhos anteriormente descritos, foi-se construindo uma tabela comparativa dos sistemas para ser possível comparar as várias componentes que cada sistema possui. Os dados recolhidos estão apresentados na Tabela 1. Analisando a tabela, é possível determinar que quase todos os trabalhos realizam algum tipo de pré-tratamento e normalização do texto (remover acentuação, pontuação, tokenizar frases, lematizar palavras, etc.) e reordenação das palavras de acordo com regras gramaticais da língua gestual. Todos usam regras da língua gestual para realizarem o mapeamento da linguagem natural para os termos da língua gestual, já que são algoritmos muito baseados nas regras da língua gestual, sendo que os sistemas que no final apresentam ao utilizador o resultado de forma gráfica, mapeiam as palavras ou termos resultantes da aplicação das várias componentes utilizadas nos seus algoritmos às respetivas animações armazenadas no sistema. Por outro lado, poucos sistemas procuram encontrar um termo opcional para palavras desconhecidas ou uma representação opcional para termos desconhecidos do sistema, alguns optando por soletrar as palavras/termos desconhecidos. Verifica-se ainda que quase todos procuram características das palavras, pois quase todos realizam a análise morfológica, contrastando com as análises semânticas e sintáticas, poucas vezes utilizadas.

¹ <https://demo.virtualsign.com/> - DemoSite VirtualSign

Característica	Trabalho Analisado									
	[40]	[41]	[42]	[43]	[44]	[46]	[47]	[48]	[49]	[50]
Remoção de símbolos desconhecidos	X									
Remoção de Stop-Words / palavras irrelevantes			X		X	X				
Normalização do texto (remoção de pontuação, acentuação e tokenização, lematização)				X	X	X	X	X	X	X
Análise morfológica	X	X	X	X	X	X	X	X		X
Análise Sintática	X	X					X			
Análise Semântica		X					X			
Reordenação de palavras de acordo com regras gramaticais da língua gestual	X		X	X	X		X		X	X
Mapeamento de palavras conhecidas pelo sistema ao respetivo termo/gesto	X	X	X	X	X	X	X	X	X	X
Tentar descobrir o significado das palavras através das suas características	X									
Tentar encontrar alternativas para palavras desconhecidas pelo sistema		X	X					X		

Tabela 1 - Características / Componentes utilizados em cada trabalho RBMT e SMT.

3.2. NMT – Tradução Baseada em Redes Neurais

A abordagem utilizando NMT difere das duas abordagens anteriores (RBMT e SMT) devido à sua capacidade de aprender e melhorar à medida que vai sendo treinada com traduções. As abordagens anteriores requerem que regras e características necessárias para a tradução sejam explicitamente utilizadas (para as RBMT) ou análise extensiva de traduções para a criação de modelos estatísticos (para as SMT), enquanto as NMT possuem a capacidade de descobrir padrões automaticamente.

M. Brour e A. Benabbou [52] apresentam um tradutor de linguagem natural árabe para a língua gestual árabe, a qual utiliza uma rede neuronal do tipo Encoder-Decoder para a realização da tradução. O algoritmo desenvolvido para o sistema começa por realizar uma análise morfológica ao texto a traduzir. Esta análise morfológica é realizada para se encontrarem características acerca de cada palavra da frase, características

essas que são agrupadas num objeto, juntamente com a palavra a que se referem. O objeto de características é composto pela palavra, a raiz da palavra, o padrão, o tipo, o género, número, prefixo e sufixo. As características que não possam ser determinadas para uma dada palavra ficam com valor nulo. De seguida, os objetos obtidos na análise morfológica são convertidos em vetores numéricos, no passo de Encoding, já que a rede neuronal apenas trabalha com valores numéricos como input. O vetor codificado segue o seguinte formato: VO[353] = (Prefix [80], Pattern[20], NameType[110], pronoun[30], Particle[15], VerbTense[3], Gender[2], Number [3], Suffix[90]). O tamanho do vetor é dado pela soma do tamanho de todos os campos e o tamanho de cada um desses campos representa a quantidade de elementos diferentes que existem na base de dados do sistema, para cada campo. Por exemplo, existem, no máximo, 80 prefixos na base de dados, logo o tamanho desse campo é 80. A codificação de cada um dos campos do vetor é realizada utilizando um de dois algoritmos. A maioria dos campos é codificado utilizando o método *one-hot*, método esse que atribui uma representação binária a cada valor possível para o campo, sendo essa representação binária composta por um '1', colocado numa posição específica que representa o valor em questão, e as restantes posições a '0'. Utilizando o campo *Gender* como exemplo, que possui tamanho 2, e assumindo a ordem de género Feminino e depois Masculino, e que o campo só aceitará esses dois valores, cada valor codificado será representado por dois números binários. Para o valor feminino, o primeiro dígito será '1' e os restantes '0', já que esse género é o primeiro (pela ordem exemplificada). Da mesma forma, para o valor masculino, o dígito que possuirá o '1' é o segundo. Numa abstração, a codificação de um dado campo realizada através do método *one-hot* gerará uma sequência de N dígitos binários (N = quantidade de valores diferentes que esse campo pode possuir), sequência essa que apenas terá um único '1', na posição que representa o valor que o campo possui, sendo os restantes dígitos da sequência colocados a '0'. Este método é vantajoso quando comparado com o método de codificação numérica, que consiste em atribuir um número diferente a cada valor possível do campo (1 ao primeiro, 2 ao segundo, etc.), pois com este último, a rede poderá erradamente assumir relações entre a sequência de valores numéricos atribuídos, quando os valores que o campo pode assumir não possuírem qualquer relação entre eles [53]. O campo *Pattern* do objeto é o único que segue o outro algoritmo de codificação, em que cada letra do valor que o campo possui

é codificada com um valor obtido pela sua posição no alfabeto sobre 100. Estes dados codificados são depois utilizados como input de uma rede neuronal do tipo “feed-forward back-propagation” a qual gera um vetor com valores binários baseados nos pesos do modelo treinado. O vetor gerado por esta última deve ser decodificado para se obter a frase traduzida. O vetor resultante segue um formato semelhante ao dos dados do input, em que cada termo da língua gestual árabe é representado por $V_{gi}[173] = (\text{interrogation}[1], \text{pronoun}[30], \text{pattern}[20], \text{name}[80], \text{number}[2], \text{particle}[5], \text{gender}[1], \text{pronoun}[30], \text{tense}[4])$. O tamanho máximo do vetor gerado é 173, que representa o tamanho da base de dados de língua gestual árabe utilizada. À semelhança da codificação do vetor para as palavras da frase a ser traduzida, os tamanhos dos campos do vetor da frase final também representam o número máximo de valores diferentes que esse campo pode assumir. A decodificação deste vetor para os termos da língua gestual é realizado através de uma camada Decoder, que faz o processo inverso à Encoder, transformando os valores numéricos texto. A arquitetura Encoder-Decoder é uma arquitetura construída tendo RNNs (Recurrent Neural Networks) como base. No fim, os termos são utilizados para gerar uma animação representada por um avatar. A Figura 17 apresenta uma visão geral da arquitetura deste sistema. ~

Oliveira, *et al.* [54] propõem a criação de um tradutor de português-brasileiro para Libras (língua gestual brasileira) NMT através da utilização de um corpus criado a partir de um outro tradutor, sendo este último do tipo RBMT, ou seja, baseado em regras. Os autores começam por construir o seu tradutor, baseado em regras, para a criação do corpus a utilizar na criação do tradutor do tipo NMT. O tradutor baseado em regras possui as etapas de remoção de símbolos não representados na língua gestual (aspas, acentos, etc.), análise morfológica, conversão de números escritos por extenso em números, uniformização do texto, e por fim, lematização das palavras, sendo este resultado depois traduzido para os termos da Libras, através das suas regras gramaticais e de tradução. O resultado é combinado num ficheiro com a frase original tratada, ficheiro esse que será o corpus da rede neuronal para a tradução de português-brasileiro para libras. A Figura 18 mostra a arquitetura do tradutor RBMT.

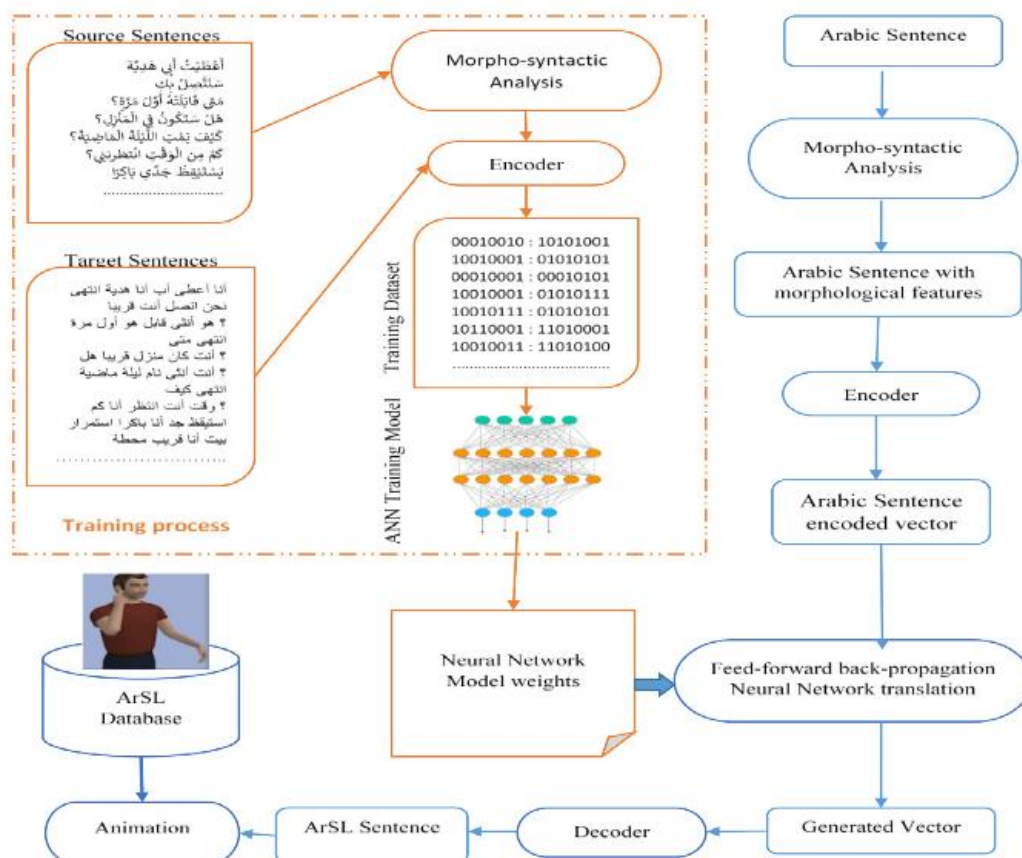


Figura 17 – Arquitetura de [52].

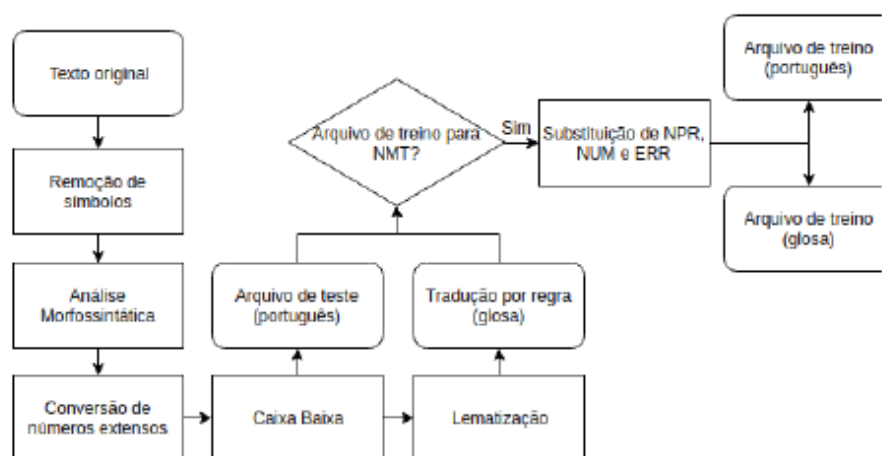


Figura 18 - Arquitetura do tradutor RBMT de [54].

O pré-tratamento dos dados para o tradutor neuronal é semelhante aos passos do tradutor baseado em regras, com a diferença de que para o tradutor neuronal não é realizada a lematização. Além disso, apesar dos passos serem iguais para ambas as abordagens, estes são necessários por razões diferentes. Por exemplo, no tradutor

neuronal procuram reduzir a complexidade das frases, enquanto no tradutor baseado em regras procuram uniformizar o texto. Foram criados dois tipos de pré-processamento diferentes. Um para os ficheiros de teste, e outro para os ficheiros de treino. A diferença é a existência de uma etapa de substituição de nomes, números e erros ortográficos quando os ficheiros de treino são gerados. Para o reconhecimento de nomes foi desenvolvida uma abordagem híbrida, que tem em consideração a análise morfológica e um ficheiro que possui uma série de nomes.

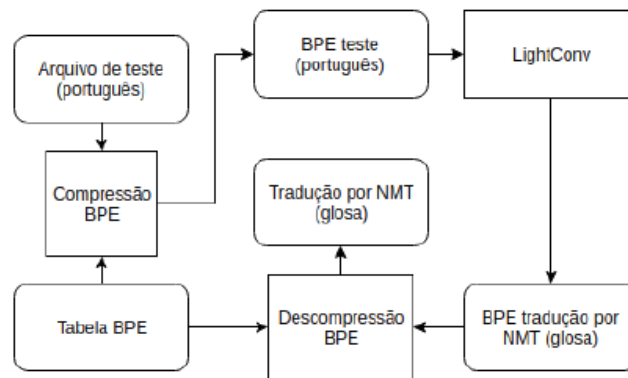


Figura 19 - Processo de treino de [54].

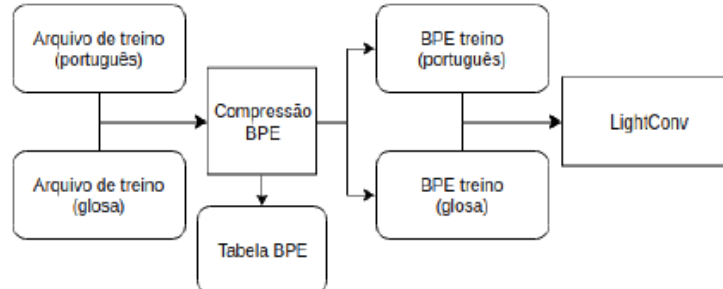


Figura 20 - Processo de teste de [54].

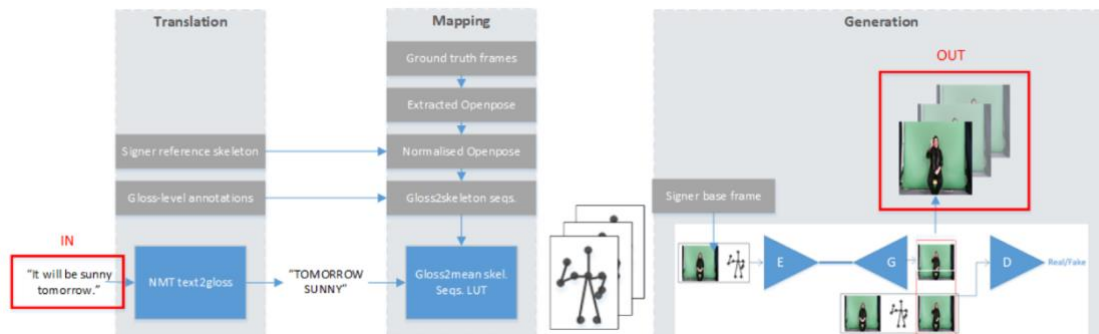


Figura 21 - Arquitetura do sistema de [55].

S. Stol, *et al.* [55], os autores apresentam uma abordagem onde utilizam a tradução automática neuronal para a tradução de linguagem natural alemã para língua gestual alemã. O tradutor de texto deste sistema para língua gestual consiste em três componentes: uma rede neuronal para a tradução de texto-para-termos, uma tabela de pesquisa para mapear termos a sequências de gestos, e por fim possui uma rede para a geração de vídeo. A tradução de texto para língua gestual é realizada através de redes neurais de arquitetura Encoder-Decoder com mecanismo de atenção. Ao contrário do que é mais comum, esta Encoder-Decoder foi construída com GRUs (Gated Recurrent Units) em vez de LSTMs (Long-Short Term Memory). A Figura 21 figura apresenta uma visão geral desse sistema.

Veríssimo, *et al.* [56] realizaram um estudo de várias arquiteturas de redes neurais utilizadas para a tradução automática de texto para a língua gestual, tendo aplicado e testado essas arquiteturas para a tradução de português-brasileiro para língua gestual brasileira (Libras). Neste estudo, a grande utilização da arquitetura *Encoder-Decoder* é destacada. O Encoder recebe uma frase de tamanho variável e converte-a num vetor numérico de tamanho fixo que representa a frase inserida, enquanto o Decoder, recebe o output do Encoder, um vetor numérico, e converte esse vetor numa sequência de palavras, tendo em consideração os valores do vetor que foi usado como input. A tradução de texto é um problema tipo *sequence-to-sequence*, onde uma sequência de dados é mapeada noutra sequência. No entanto, neste mapeamento, é necessário ter-se em conta outros fatores que podem modificar o significado da frase (tais como dependências temporais, já que o significado de uma palavra pode ser modificado por palavras que tenham aparecido anteriormente, ou a palavra atual pode alterar o significado de palavras futuras). Se estes fatores aparecerem muito distanciados uns dos outros, a arquitetura Encoder-Decoder simples, com utilização de RNNs simples apenas, não os irá considerar corretamente, dando mais importância às últimas palavras a aparecer e pouca às primeiras. Para resolver este problema, surgiram as Encoder-Decoder que utilizam LSTM (Long-short Term Memory), que permite manter partes importantes da frase e considerá-las quando as últimas palavras estão a ser codificadas no Encoder. No entanto, estas redes começam a falhar para frases muito longas, já que mesmo utilizando LSTM, poderá não ser possível manter todas as partes importantes da frase e considerá-las ao avaliar partes ainda não codificadas. Para tal,

podem-se utilizar redes bidirecionais, ou introduzir mecanismos de atenção. O mecanismo de atenção permite manter os resultados intermédios da codificação das palavras no Encoder, e utilizarem-se esses resultados no Decoder, de forma a determinar melhor o contexto e a que é que algo se refere. A evolução desta última é a arquitetura *Transformer*, modelo este que apenas utiliza a ideia do mecanismo de atenção, removendo as RNNs e consequentes LSTMs e GRUs [57]. A remoção da recorrência das RNNs resulta em menos operações a serem realizadas, e, consequentemente, o treino da rede é realizado num menor intervalo de tempo. Surgiu também ainda um modelo chamado de *LightConv*, baseado em CNNs (Convolutional Neural Network) em vez de RNNs. Diferentemente da *Transformer*, a *LightConv* reutiliza os pesos da rede para os elementos de contexto, independentemente do tempo atual. Este último modelo destaca-se por ser computacionalmente leve e ter resultados equivalentes às RNNs. Depois desta análise de abordagens, realizam um estudo mais aprofundado à utilização da *Transformer* e da *LightConv* para a tradução de português-brasileiro para Libras. No seu estudo, determinam que a *LightConv* alcançou resultados ligeiramente superiores à *Transformer*. No entanto, determinam ainda que o tamanho da base de dados utilizada foi o maior fator de impacto na precisão dos modelos.

3.2.1. Apreciação NMT

Tendo em consideração os vários trabalhos analisados e os dados constantes na Tabela 2, verifica-se que um dos trabalhos, além de apresentar o tradutor baseado em redes neuronais para traduzir da linguagem natural para língua gestual, apresenta também um tradutor baseado em regras, utilizado para a construção de um corpus, a ser utilizado no treino do tradutor NMT. O tradutor baseado em regras foi desenvolvido devido à inexistência de corpora com traduções de português-brasileiro para Libras. As traduções geradas pelo tradutor baseado em regras foram devidamente validadas através de um utilizador da linguagem natural e da língua gestual em questão, de modo que o tradutor baseado em redes neuronais fosse treinado com dados de boa qualidade. Os restantes trabalhos utilizaram corpora já existentes para as suas linguagens, apesar de serem corpora bastante limitados, como apontado por alguns autores. Os resultados BLEU apresentados na Tabela 2 podem não ter significados muito importantes, já que são resultados que foram obtidos de corpora diferentes e maioritariamente pequenos

e de baixa qualidade. Por exemplo, um dos trabalhos apresenta resultado BLEU de .80 em 1.00, enquanto o trabalho 3 apresenta cerca de .50. No entanto, o trabalho 3 utiliza mecanismos de atenção, algo que provavelmente resultaria em resultados superiores aos apresentados, caso usassem os mesmos dados utilizados no trabalho 1. Como todos utilizaram corpora diferentes, é quase impossível, ou mesmo impossível, compararem-se os resultados obtidos e dizer-se que um é melhor que o outro, para um dado caso de uso, sem todos serem testados. Além disso, todos utilizaram corpora bastante pequenos, visto não existirem grandes corpora disponíveis para a tradução de linguagem natural para língua gestual. Também pela análise dos trabalhos, verifica-se que as arquiteturas Transformer e LightConv são as mais recentes, no que concerne à tradução entre linguagens naturais, e que ainda não tiveram muita utilização na tradução entre linguagem natural e língua gestual. No entanto, à semelhança das traduções entre linguagem natural, estas deverão também apresentar resultados semelhantes na tradução entre linguagem natural e língua gestual, já que ambas são problemas seq2seq (sequence-to-sequence), quando criadas com bons corpora.

Trabalho	Arquitetura da Rede	Resultados e conclusões obtidos pelos autores
[52]	Encoder-Decoder	Indicam BLEU médio de .80 para o corpora utilizado.
[54]	LightConv	Indicam BLEU de 0.55, tanto para o tradutor RBMT e para o NMT. Indicam ainda que o seu tradutor alcançou melhores resultados que o VLibras (ferramenta do governo brasileiro).
[55]	Encoder-Decoder com GRUs e com mecanismo de atenção	Apresentam BLEU perto de .50, apresentando ainda resultados superiores à ferramenta que realiza a operação inversa (Termos para texto natural). Referem também da qualidade dos dados que possuem não é a melhor.
[56]	Transformer e LightConv	Indicam que a LightConv alcançou resultados ligeiramente superiores para os dados utilizados, alcançado um BLEU de 0.62.

Tabela 2 - Arquiteturas e Principais Conclusões de Cada Trabalho NMT.

3.3. Conclusões

Neste capítulo realizou-se o estudo de vários trabalhos desenvolvidos com o propósito da tradução de linguagens naturais para línguas gestuais, trabalhos com tradutores baseados em regras (RBMT) ou estatísticas (SMT) e ainda trabalhos com tradutores baseados em redes neuronais (NMT). Inicialmente, foram estudados trabalhos com as abordagens mais clássicas, mas baseadas em regras (RBMT) e baseadas em estatísticas (SMT), tendo-se verificado que a maioria dos trabalhos aplica tratamentos de texto aos dados, de modo a consegui-los normalizar o mais possível, de acordo com regras implementadas. Depois, serão traduzidos para a língua gestual correspondente, através do mapeamento da palavra ao respetivo termo, através de dicionários. No entanto, em contraste com a grande utilização de certos componentes de tratamento de texto, verifica-se que poucos procuram obter termos alternativos para mapeamentos palavra-termo, que não existam nas suas bases de dados, alguns optando por soletrar as palavras. Após a análise dos trabalhos referidos, foram analisados outros trabalhos focados na tradução de linguagem natural para língua gestual, mas que utilizam redes neuronais para a tarefa de tradução (NMT). Nestes, destacam-se as redes neuronais que seguem a arquitetura Encoder-Decoder, com a utilização de LSTMs ou GRUs e mecanismos de atenção. No entanto, é de destacar ainda os modelos Transformer e LightConv, modelos mais recentes e que parecem apresentar resultados promissores na tarefa de tradução, apresentando resultados semelhantes às Encoder-Decoder, utilizando menos recursos computacionais para o treino da rede, já que não utilizam a recorrência das RNNs. As redes neuronais apresentam melhores resultados que a utilização de sistemas baseados em regras e em estatísticas [58], já que possuem a capacidade de determinar relações entre palavras e padrões durante o seu treino, melhorando a cada tradução. No entanto, apenas são viáveis com a existência de um grande corpus de qualidade, que possua o texto natural e a respetiva tradução em termos da língua gestual, para que a rede neuronal treinada possa apresentar bons resultados. Um método para a criação desse corpus pode ser através da utilização de um sistema baseado em regras, ou em estatísticas, para se obterem traduções da linguagem natural para a língua gestual, sendo as traduções verificadas e corrigidas, se necessário, por um utilizador tanto da linguagem natural como da língua gestual.

4. ARQUITETURA DO SISTEMA

Neste capítulo apresenta-se a arquitetura geral do sistema com os diversos componentes que o integram. Inicialmente, será apresentada uma visão geral sobre a arquitetura do sistema completo, sendo depois apresentada a arquitetura do módulo de tradução de português para língua gestual portuguesa, e ainda a arquitetura do tradutor de português para a glosa língua gestual, principal foco deste trabalho. Foi também realizada uma análise comparativa de ferramentas de speech-to-text gratuitas, para seguidamente escolher-se uma a utilizar na tarefa de transcrição dos áudios recebidos. Esta ferramenta permite realizar a tradução para a glosa da língua gestual e posterior representação dessa glosa. Adicionalmente, apresenta-se uma análise comparativa de redes neuronais e resultados obtidos nas tarefas de tradução entre diferentes línguas, de modo a procurar-se determinar qual das redes poderá alcançar melhores resultados na implementação final.

4.1. Enquadramento

O trabalho desenvolvido e apresentado neste documento faz parte de um projeto maior, designado por IVLinG² (Intérprete Virtual de Língua Gestual), que tem como principal objetivo a construção de um tradutor móvel bidirecional de português e LGP, focado em ambiente médico/hospitalar. No processo de tradução de português para LGP, existe um módulo capaz de receber áudio ou texto, e traduzir, através de redes neuronais, esse áudio ou texto para os termos da língua gestual a serem representados. Esses termos são depois apresentados ao utilizador da aplicação móvel sob a forma de um avatar 3D. A tradução inversa, foca-se na captura da representação dos gestos através da câmara do telemóvel, e através dessa captura, utilizando técnicas de reconhecimento de imagem, extraem-se os termos representados, que depois serão convertidos para o respetivo texto em português. A tradução de LGP para português e o desenvolvimento da aplicação móvel são tarefas a desenvolver por outros parceiros associados ao projeto. A aplicação IVLinG e respetivos servidores serão, neste projeto,

² IVLinG – Intérprete Virtual de Língua Gestual – https://tech.ipvc.pt/projeto.php?id_projeto=184

referidos como *Sistema Geral*, sendo que a comunicação com este segue APIs específicas. O trabalho que aqui se descreve foca-se na componente que utiliza Tradução Baseada em Regras e Machine Learning para a tradução de português para a glosa da LGP.

4.2. Arquitetura do Módulo de Tradução de Português para LGP

O módulo de tradução proposto, de português para LGP, baseia-se na disponibilização de um serviço XMPP [59], acessível via websockets. Este serviço recebe uma mensagem em formato JSON [60] que, dependendo do tipo de dados a traduzir, texto ou áudio, inclui o tipo de mensagem (variável consoante seja texto ou áudio), código da sala de onde a mensagem teve origem ou um *id*, no caso de não ser originária de uma sala de chamada, o texto a traduzir ou o áudio em base64, formatos de mensagem apresentados nos “Excerto de Código 1” e “Excerto de Código 2”, respetivamente. Este processo é gerido por um cliente XMPP central, que encaminha toda a informação, desde que recebe uma mensagem de tradução, até ao momento em que o resultado é enviado a quem solicitou a tradução.

```
{
  "textId": "ID",
  "messageType": "interpretText",
  "text": "[(Nome ficheiro) || Texto inserido];data:text/plain;base64,[texto convertido para base64]"
}
```

Excerto de Código 1 - Pedido Tradução de Texto em formato JSON.

```
{
  "audioid": "ID",
  "messageType": "interpretAudio",
  "audio": "[(Nome ficheiro) || Audio gravado];data:text/plain;base64,[áudio convertido para base64]"
}
```

Excerto de Código 2 - Pedido de Tradução de Áudio em formato JSON.

Recebida a mensagem, o gestor central verifica o tipo da mesma. Caso seja do tipo “*interpretAudio*” (mensagem de áudio) será necessário extrair-se o áudio da respetiva chave no objeto json recebido. Essa extração é efetuada através da divisão do valor textual presente na chave “*audio*” no carater ‘;’, considerando-se o conteúdo à direita desse carater. Concluída a divisão, descodifica-se o conteúdo, que se encontra em formato Base64, e guarda-se num ficheiro de áudio. Esse ficheiro de áudio é

seguidamente enviado para um serviço de transcrição de áudio para texto, serviço esse que devolve o texto a ser traduzido para a glosa da LGP. Sendo a mensagem recebida do tipo *"interpretText"*, que representa uma mensagem de texto, é necessário extrair-se o texto a traduzir do objeto json recebido, tal como se realiza com as mensagens de áudio, com a diferença de que a chave do objeto é *"text"*. Tal como nas mensagens de áudio, extrai-se o texto através da divisão do valor textual presente na chave *"text"* do objeto recebido, divisão realizada no carater *'*, sendo considerado o valor à sua direita, que será o texto a traduzir. Possuindo-se o texto a ser traduzido para os termos da língua gestual, esse texto é então enviado para o tradutor neuronal. Nesse tradutor neuronal são aplicadas algumas regras e é realizada a normalização desse texto, antes do mesmo ser fornecido à rede neuronal para ser traduzido. O resultado da rede neuronal é a glosa da língua gestual para a respetiva frase. O funcionamento do tradutor neuronal é detalhado mais adiante neste trabalho. Possuindo-se a glosa, a mesma é enviada para o cliente de animação. Esse cliente de animação pesquisa numa base de dados o gesto correspondente a cada um dos termos presentes na glosa, e encadeia-os de modo a criar um vídeo com a representação da frase, realizada por um avatar 3D. Esse vídeo é guardado numa localização temporária, e a localização é devolvida ao gestor central. Este obtém o vídeo dessa localização, convertendo-o para Base64, e adicionando o resultado da codificação Base64 à resposta. Tal como a mensagem de pedido, a mensagem de resposta é enviada dentro de um objeto json, sob a forma de objetos com os formatos apresentados em *"Excerto de Código 3"* e *"Excerto de Código 4"*, para a mensagem de texto e de áudio, respetivamente.

```
{
  "textId": "ID",
  "messageType": "textInterpretation",
  "video": "[Nome ficheiro];data:[MIME TYPE
(video/webm)];base64,[video convertido para base64]"
}
```

Excerto de Código 3 - Resposta Tradução de Texto em formato JSON.

```
{
  "audioId": "ID",
  "messageType": "audioInterpretation",
  "video": "[Nome ficheiro];data:[MIME TYPE
(video/webm)];base64,[video convertido para base64]"
}
```

Excerto de Código 4 - Resposta Tradução de Áudio em formato JSON.

A Figura 23 apresenta uma visão geral da arquitetura deste módulo de tradução de português para língua gestual e respetiva representação e a Figura 22 apresenta de forma simplificada esse processo.

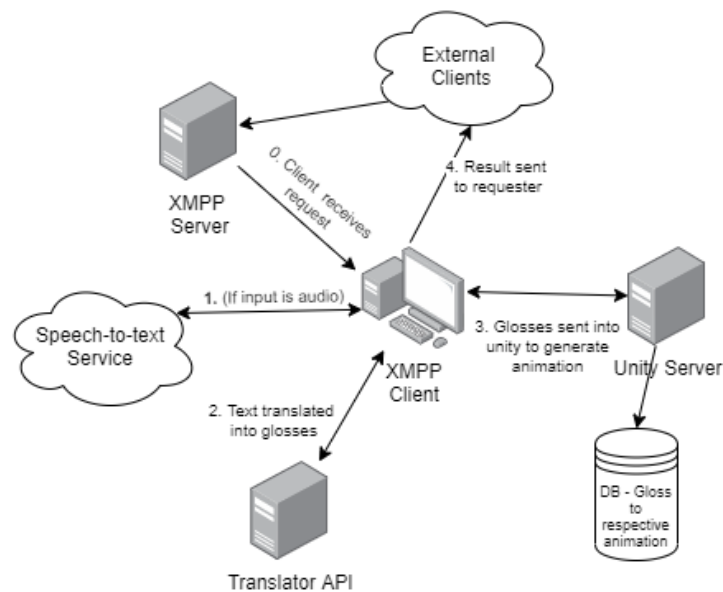


Figura 23 - Arquitetura Geral do Módulo de Tradução PT -> LGP.

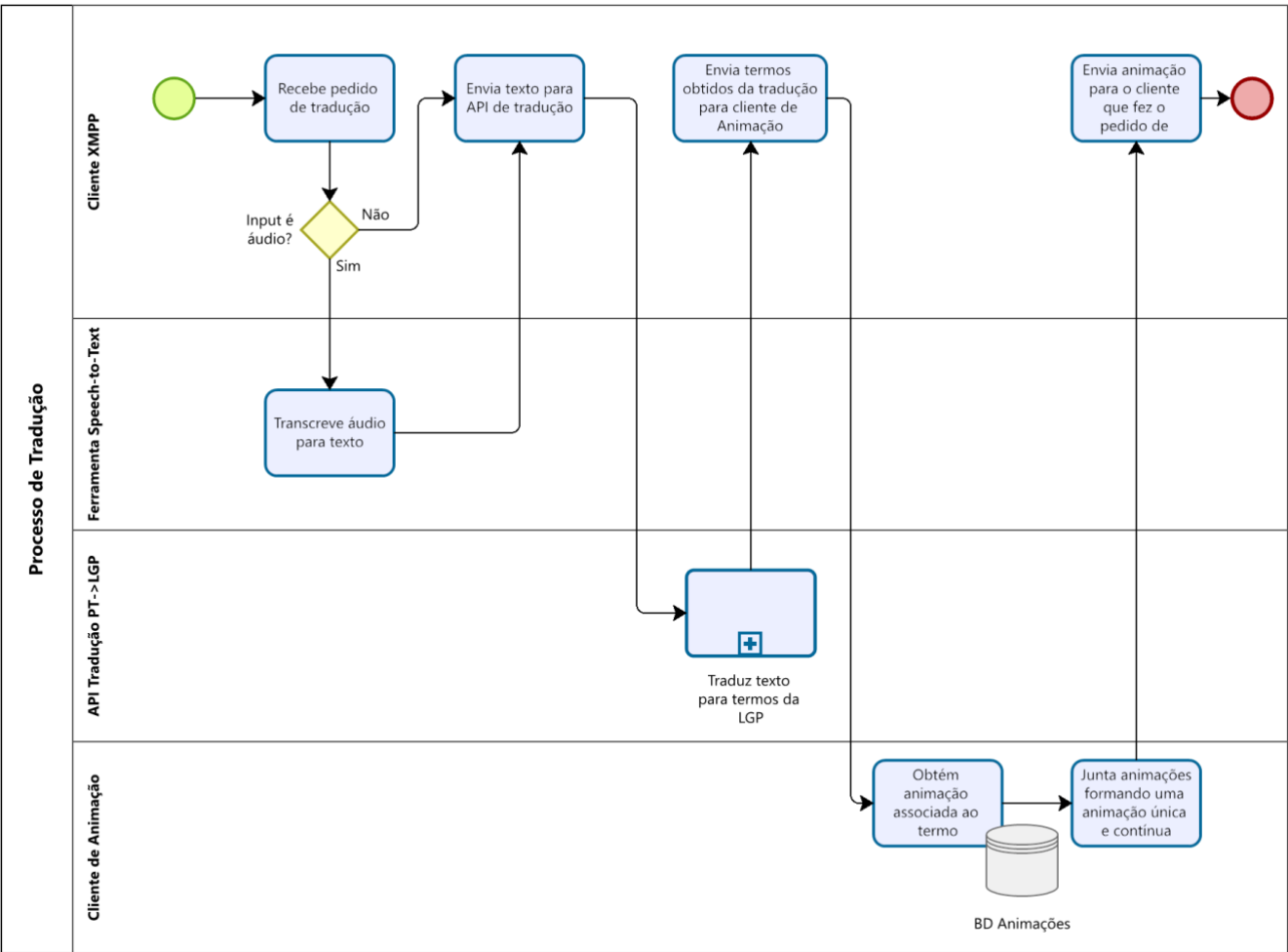


Figura 22 - Processo de Tradução de Texto.

4.3. Conversão de Áudio em Texto

O módulo de tradução de português para a glosa da LGP, tal como apresentado na arquitetura da solução, pode, além de texto, receber áudio para ser traduzido para a glosa da língua gestual. De modo a efetuar a tradução do áudio para a glosa da língua gestual, é necessário transcrever esse áudio, para se obter a sua representação textual, representação essa que é depois submetida ao tradutor de português para LGP.

Optou-se por obter a transcrição do áudio através de ferramentas já existentes e gratuitas de Automatic Speech Recognition (ASR), também conhecidas como ferramentas de Speech-to-Text (STT), visto não ser a transcrição de áudio para texto o conteúdo principal deste trabalho.

4.3.1. Ferramentas Speech to-Text

Existem diversas ferramentas e serviços disponíveis para a realização de transcrições de texto para áudio, no entanto, a grande maioria é paga. Em termos de ferramentas gratuitas, que são possíveis de serem utilizadas através de scripts python, encontraram-se e utilizaram-se a Wit.ai e a Google Speech Recognition.

4.3.1.1. *Google Speech Recognition*

A Google Speech Recognition é uma ferramenta gratuita disponibilizada pela Google nos serviços da Speech API para a realização de transcrições. Esta ferramenta é nativamente suportada pela biblioteca SpeechRecognition do python, biblioteca essa utilizada neste trabalho.

A Google Speech Recognition [61] não é a mesma ferramenta que a Google Cloud Speech-to-Text, sendo esta última paga. A Google Speech Recognition aparenta estar a deixar ser utilizada, em favorecimento da Speech-to-Text da Google Cloud.

4.3.1.2. *Wit.ai*

A Wit.ai [62] é uma ferramenta gratuita para a realização de transcrições, fazendo atualmente parte da coleção de serviços da Meta (Facebook). Tal como a ferramenta anterior, também esta é nativamente suportada pela biblioteca SpeechRecognition do python.

A Wit.ai, apesar de ser uma ferramenta criada para ser utilizada em chat bots, a sua parte de Speech-to-Text pode ser aplicada a outros contextos, tal como as necessidades deste trabalho.

4.3.2. Avaliação e Resultados

Para a avaliação das ferramentas indicadas anteriormente, utilizaram-se métricas de avaliação de ASR.

4.3.2.1. Métricas de Avaliação de Automatic Speech Recognition

Foram utilizadas as seguintes métricas para a avaliação das ferramentas de transcrição:

4.3.2.1.1. WER - Word Error Rate

A *Word Error Rate* (WER) [63] é uma métrica derivada da “Distância de Levenshtein”, que permite determinar a taxa de diferenças entre duas frases através das inserções, remoções ou substituições realizadas na conversão de uma frase para a outra. Em Speech Recognition, esta métrica permite determinar a taxa de erros (inserções, remoções e substituições) entre a frase transcrita e a frase esperada, através da fórmula:

$$WER = \frac{S + R + I}{N}$$

Em que:

- S - Quantidade de **S**ubstituições;
- R – Quantidade de **R**emoções;
- I – Quantidade de **I**nserções;
- N – Quantidade de Palavras da Frase Inicial.

No entanto, esta fórmula não indica o quão bom um sistema é, apenas permite comparar qual sistema é o melhor (para a mesma frase / mesmo conjunto de frases), pois não possui um limite superior, podendo exceder os 100%.

4.3.2.1.2. MER - Match Error Rate

A *Match Error Rate* (MER) é uma métrica que indica a probabilidade de uma palavra na frase estar errada [64]. A fórmula para o MER é:

$$MER = \frac{S + R + I}{S + R + I + C}$$

Em que:

- S - Quantidade de Substituições;
- R – Quantidade de Remoções;
- I – Quantidade de Inserções;
- C – Quantidade de Palavras Corretas.

4.3.2.1.3. RIL – Relative Information Lost

A *Relative Information Lost* (RIL) [65] é uma métrica baseada em informação mútua que permite determinar-se a dependência estatística entre as palavras da frase transcrita (X) e da frase esperada (Y). Ao contrário da WER, esta métrica tem em conta a distribuição dos erros na frase, e é calculada utilizando-se a entropia de Shannon através de:

$$RIL = \frac{H(Y|X)}{H(Y)}$$

onde:

$$H(Y) = - \sum_{i=1}^n P(y_i) \log P(y_i)$$

e:

$$H(X|Y) = - \sum_{i,j} P(x_i, y_j) \log P(x_i, y_j)$$

No entanto, esta métrica nunca foi muito adotada por ser difícil de aplicar e medir 0 em mapeamentos um-para-um [66].

4.3.2.1.4. WIL – Word Information Lost

A *Word Information Lost* (WIL) [66] é uma métrica baseada na RIL que utiliza a quantidade de erros (inserções, remoções, substituições) e a quantidade de palavras corretas. Esta métrica corrige problemas associados à RIL, sendo mais fácil de se aplicar [63]. A WIL segue a fórmula:

$$WIL = 1 - \frac{C^2}{(C + S + R)(C + S + I)}$$

Em que:

- S – Quantidade de Substituições;
- R – Quantidade de Remoções;
- I – Quantidade de Inserções;
- C – Quantidade de Palavras Corretas.

4.3.2.1.5. WIP – Word Information Preseved

A *Word Information Preserved* (WIP) representa uma aproximação à proporção de informação que é retida na frase transcrita, face à frase esperada [63]. É o contrário da WIL, podendo ser obtida através de:

$$WIP = 1 - WIL$$

4.3.2.2. *Avaliação*

Para realizar a avaliação das duas ferramentas, utilizando as métricas indicadas anteriormente, desenvolveram-se dois scripts Python. O primeiro, que para um determinado dataset, realiza a transcrição de todos os áudios, compara as transcrições obtidas pelas ferramentas com as esperadas, utilizando as métricas WER, MER, WIL e WIP. O dataset utilizado foi o “Multilingual LibriSpeech (MLS) dataset” e está disponível no website OpenSLR (<https://www.openslr.org/94/>) [67]. Este dataset possui uma série de áudios distribuídos por várias pastas, e um ficheiro com as transcrições para cada um dos respetivos ficheiros de áudio. As transcrições estão identificadas com um código que segue o formato X_Y_Z, em que X e Y são conjuntos de 4 ou 5 dígitos, que representam, respetivamente, a pasta e subpasta do ficheiro e Z o número do ficheiro. Os nomes dos ficheiros também seguem o mesmo formato X_Y_Z (exemplo: código 11247_10229_000030 | caminho ./11247/10229/11247_10229_000030.flac).

O primeiro script Python desenvolvido (representação algorítmica na Figura 24) lê cada linha do ficheiro de transcrições, e para cada uma delas, começa por separar o nome do ficheiro da transcrição (separados por \t). Depois disso, utilizando o nome extraído, separa-o, tendo em consideração o formato explicado anteriormente, e obtém o ficheiro de áudio. Esse áudio é então enviado às ferramentas de speech-to-text, selecionadas através da biblioteca Python “SpeechRecognition” (biblioteca que implementa o acesso a várias ferramentas de transcrição pagas e não pagas, incluindo a

Google Recognition e a Wit.AI). Recebidas as transcrições, as mesmas são guardadas num ficheiro separado por vírgulas (csv) juntamente com a transcrição esperada, para serem posteriormente analisadas e avaliadas.

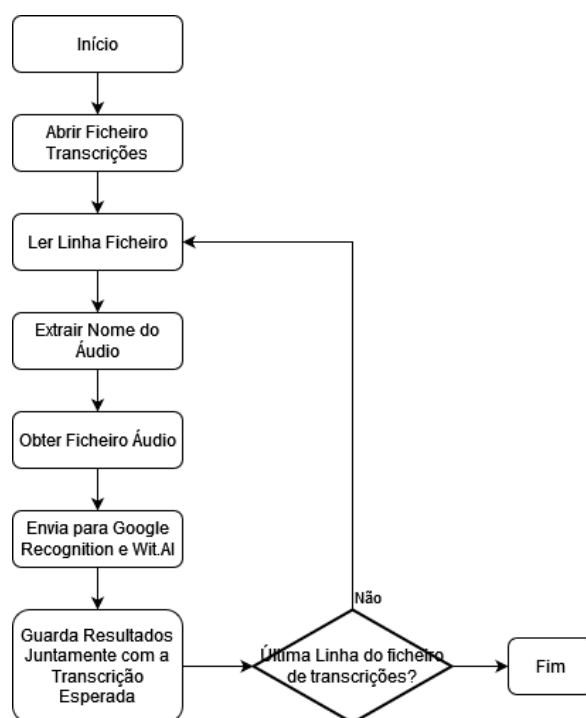


Figura 24 – Fluxograma representativo script de obtenção de transcrições.

Alguns exemplos de transcrições obtidas para o dataset escolhido e armazenadas no ficheiro pelo script anterior estão apresentadas na Tabela 3.

#	Original	Google Recognition	Wit.AI
1	Fim da parte 11.	Fim da parte 11	Fim da parte onze.
2	Pelas ladeiras das terras mais altas.	Pelas Ladeiras das serras mais altas	ladeiras das terras mais altas
3	Pode tratar dele quando quiser.	Pode tratar dele quando quiser	Pode tratar dele quando quiser.
4	O que tens meu amigo? Perguntei-lhe com uma voz que esforçava por tornar calma.	O que tens meu amigo perguntei-lhe como avós que esforçar por tornar calma	ou que tens meu amigo? Perguntei-lhe com uma voz que esforçava por tornar a calma
5	A praia estava deserta. Caia um chuveiro, ventos frios sopravam do mar.	estava deserta um chuveiro ventos frios Surprise	A praia estava deserta, caía um chuveiro, ventos frios sopravam do mar.

Tabela 3 – Exemplos de Transcrições Obtidas usando as ferramentas Google Recognition e Wit.AI.

Como as transcrições obtidas estão a ser armazenadas num ficheiro, é possível assim avaliarem-se todas as transcrições de uma só vez, em vez de serem avaliadas uma a uma. Esta avaliação é realizada no segundo script (representado de forma algorítmica na Figura 25) com recurso à biblioteca “jiwer” (biblioteca que possui métricas de avaliação de ASR já definidas, incluindo a WER, MER, WIL e WIP). Inicialmente, o script começa por ler todas as linhas do ficheiro csv, gerado pelo script que realiza as transcrições, dividindo cada coluna dessas linhas em três arrays, um com as transcrições esperadas, outro com as transcrições obtidas da Google Recognition e outro com as transcrições obtidas da Wit.AI. De seguida esses três arrays recebem um pré-tratamento, de modo a normalizar todos os dados, sendo então avaliados, com recurso à biblioteca jiwer, e os resultados obtidos apresentados.

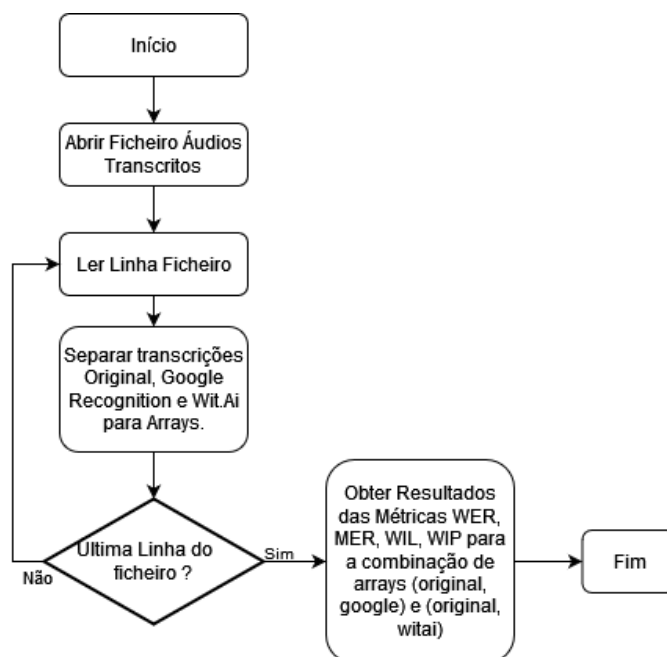


Figura 25 - Fluxograma representativo script de avaliação das transcrições.

A Tabela 4 apresenta os resultados das métricas e as quantidades de palavras transcritas corretamente (C), substituições (S), remoções (R) e inserções (I) para o corpus utilizado.

Ferramenta	WER (%)	MER (%)	WIL (%)	WIP (%)	C (N.º)	S (N.º)	R (N.º)	I (N.º)
Google Recognition	0.31814	0.30909	0.46744	0.53256	398372	117184	44625	16405
Wit.AI	0.15813	0.15510	0.24484	0.75516	482556	56951	20674	10959

Tabela 4 - Resultados das Métricas para as Transcrições do Dataset e Quantidade de Palavras Corretas e Erros.

Analisando os resultados da tabela, consegue-se determinar que, para o dataset utilizado, a Wit.AI alcançou resultados bastante superiores, quando comparada com a Google Recognition. É possível também determinar-se, não ao analisar os resultados das métricas, mas sim ao analisar as transcrições obtidas por cada ferramenta (exemplos da Tabela 3), que, ao contrário da Google Recognition, a Wit.AI produz transcrições com acentuação e pontuação, resultando em transcrições mais próximas do esperado.

Tendo em conta estes resultados, a Wit.AI foi a ferramenta de ASR selecionada para a realização das transcrições para este tradutor.

4.4. Arquitetura do Tradutor de Português para a Glosa da LGP

O tradutor de português para a glosa da Língua Gestual Portuguesa é uma componente do módulo de tradução de português para LGP, sendo esta a componente mais explorada no desenvolvimento deste trabalho. Esta componente é composta pela conversão da estrutura frásica, onde são aplicadas regras de forma a converter uma frase do tipo SVO (Sujeito-Verbo-Objeto – estrutura das frases em português) para OSV (Objeto-Sujeito-Verbo – estrutura utilizada na LGP) e onde são aplicadas ainda outras regras relativas à estrutura da frase, tal como a colocação de advérbios temporais no início da frase ou a colocação de determinantes possessivos e números após o substantivo. Posteriormente à aplicação das regras que envolvem a modificação da estrutura frásica e a reordenação de partes da frase, passa-se para a aplicação de regras gramaticais, que apenas realizam modificações a nível da palavra (removendo-a, modificando-a e/ou adicionando outras palavras) sem alteração na ordem das componentes da estrutura frásica, como são por exemplo, a remoção de palavras não representadas na LGP (artigos definidos e indefinidos são exemplos de palavras não representadas na LGP), lematização (conversão para forma canónica) de verbos e tratamento de graus substantivos, estes últimos com a inclusão de novas palavras representantes do grau diminutivo e aumentativo respetivamente, entre outras regras. As regras implementadas nestas duas partes iniciais do tradutor de português para a glosa da LGP baseiam-se nas regras descritas na secção 2.1.3. (Características Gramaticais e Estruturais da LGP) e na utilização de técnicas e ferramentas de

Processamento de Linguagem Natural, para a obtenção de características gramaticais das palavras da frase a ser traduzida. Estas duas partes resultam numa primeira tradução, sendo esta tradução baseada unicamente em regras, representando um tradutor RBMT. Esta primeira tradução é depois enviada a uma rede neuronal do tipo Encoder-Decoder que procura melhorar essa tradução, de acordo com os padrões encontrados pela rede no seu momento de treino. O grande objetivo da rede neuronal, além de melhorar traduções, é identificar padrões que representem regras gramaticais, que não foram encontradas e implementadas nas fases anteriores, já que a inexistência de literatura sobre a gramática da LGP impossibilita a determinação de mais regras sobre a mesma. A combinação da rede com o tradutor baseado em regras permitirá ainda disponibilizar uma tradução alternativa *melhorada*, nos casos em que o resultado da rede neuronal seja diferente do resultado da tradução baseada em regras. A Figura 26 apresenta as atividades que fazem parte desta componente e ligações entre as mesmas.

A combinação de uma tradução baseada em regras e a rede neuronal para expandir essa primeira tradução baseada em regras é, neste momento, obrigatória, já que devido à inexistência de corpus grandes e de qualidade para a tradução de português para a glosa da LGP impossibilita o treino de uma rede neuronal que sozinha seja capaz de obter boas traduções, e ainda, com a inexistência de gramática detalhada da LGP é impossível determinar e implementar tudo o que é necessário para traduções, sempre totalmente corretas através de regras. Assim, esta ferramenta poderá também vir a ser uma fonte de dados para a criação de um corpus de grandes dimensões, para o treino de uma rede neuronal para um sistema que apenas utilize a rede para a obtenção de traduções. Independentemente da forma de como a rede neuronal será utilizada, foram realizados testes a arquiteturas diferentes de redes neuronais, de modo a obter-se uma perspectiva de qual a arquitetura e hiperparâmetros que poderão possuir um melhor desempenho para a tarefa de tradução, sendo esses testes e resultados apresentados no próximo capítulo - 4.5. Seleção de Rede Neuronal.

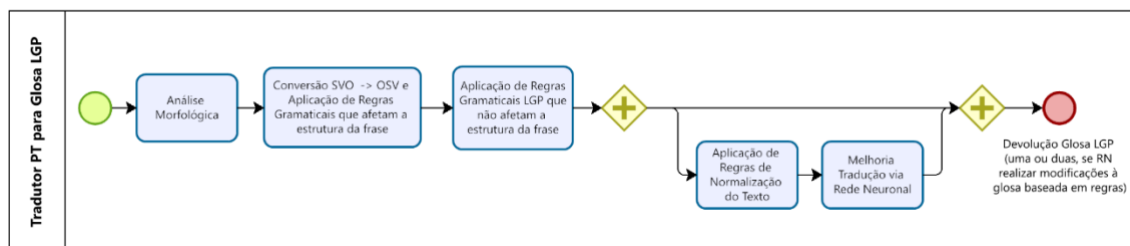


Figura 26 - Diagrama de Atividades do Módulo Tradução PT para Glosa LGP.

4.5. Seleção de Rede Neuronal

A seleção da arquitetura a ser utilizada na rede neuronal mostra ser bastante importante, uma vez que algumas arquiteturas irão alcançar melhores resultados que outras. De modo a obter-se uma ideia sobre qual arquitetura que irá comportar-se melhor na tradução de português para a glosa da LGP, e devido à inexistência de corpus suficientemente grandes (ou mesmo pequenos) para a criação de uma rede neuronal capaz de realizar a tradução para as línguas importantes para este trabalho, optou-se pela realização de experimentações entre português – inglês e português – LIBRAS (Língua Brasileira de Sinais) . Para cada uma destas traduções foram implementadas redes neurais do tipo Encoder-Decoder e Encoder-Decoder Bidirecionais, com ambas a utilizarem LSTM.

4.5.1. Corpora Utilizada

Para a experimentação das duas arquiteturas para a tradução de português para inglês, utilizou-se pelo corpus disponibilizado pelo projeto *Tatoeba*³ (*Multilingual Corpus of Sentence Equivalents*), projeto este que consiste na criação de traduções entre várias línguas, traduções feitas voluntariamente por quem queira colaborar no crescimento do projeto. O corpus é constituído por cerca de 170 mil traduções com tamanhos variados. Para a tradução de português para LIBRAS, foi necessário criar-se o corpus e nesta criação utilizou-se a ferramenta vLibras, ferramenta que devolve a glosa libras para uma determinada frase indicada. As frases em português utilizadas e enviadas à vLibras foram as mesmas frases existentes no corpus de português-inglês do *Tatoeba*.

³ <https://tatoeba.org/en/> - Tatoeba: Collection of Sentences and Translations

Após a obtenção dos corpora, aplicaram-se sobre ambos algumas regras de tratamento de texto e normalização do mesmo. Nestas regras constam, por exemplo, remoção de pontuação e caracteres estranhos, modificação de capitalização, tokenização, entre outras. A aplicação destas regras permite assegurar que os dados estão normalizados e que podem ser trabalhados de igual modo nos passos seguintes. Das 170 mil traduções dos corpora, 85% foram selecionadas aleatoriamente para serem utilizadas como dados de treino das redes, e as restantes 15% como dados de teste.

4.5.2. Resultados Alcançados

As experimentações foram realizadas com uma série de combinações de parâmetros diferentes, variando valores de *batch size*, número de *epochs*, entre outros, para ambas as arquiteturas e línguas utilizadas. Estas experimentações apresentaram resultados bastante promissores, como se verifica nas avaliações BLEU apresentadas nas Figura 27 e Figura 28, para português-inglês, e Figura 29 e Figura 30 para português-Libras. É de realçar que para cada combinação de parâmetros e arquitetura e línguas, a divisão de dados foi sempre a mesma, de modo a conseguir-se realizar uma comparação mais correta, uma vez que deste modo, foram utilizados os mesmos dados, evitando-se assim a influência de possíveis bias causadas pelas divisões dos dados nos resultados finais obtidos. As figuras Figura 27 à Figura 30 apresentam os resultados BLEU obtidos na avaliação das traduções conseguidas pelas redes para os dados de teste, já que estes dados de teste são utilizados na realização de traduções com frases que as redes neuronais nunca tinham “visto” anteriormente, ao contrário dos dados de treino, que possui frases que já foram trabalhadas pela rede, enquanto esta era treinada. Para estes últimos, os resultados BLEU foram superiores, como era esperado.

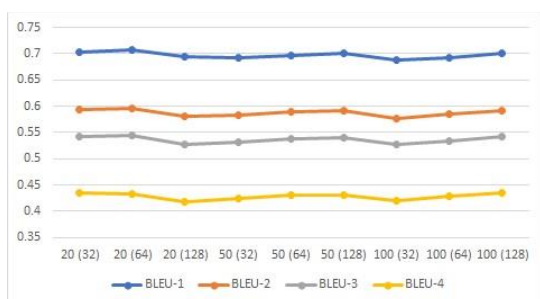


Figura 27 - PT->EN LSTM.

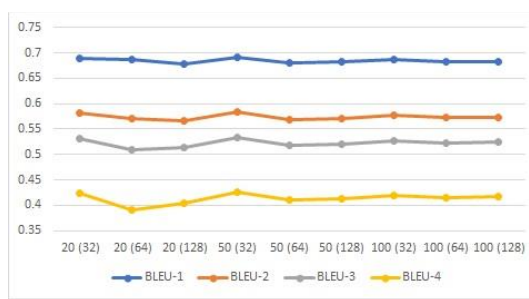


Figura 28 - PT->EN LSTM Bidirecional.

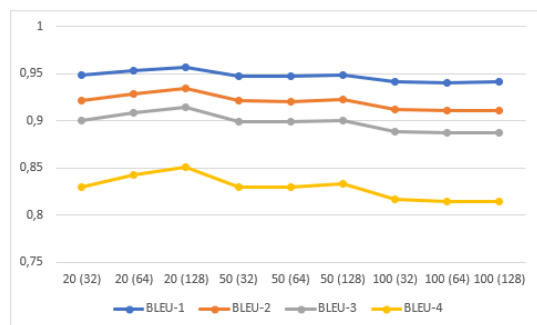


Figura 29 - PT->Libras LSTM.

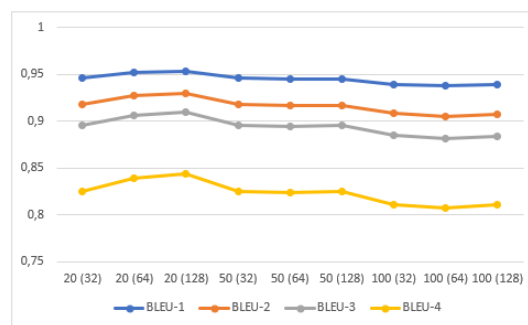


Figura 30 - PT->Libras LSTM Bidirecional.

Analizando os resultados obtidos, pode-se determinar que ambas as arquiteturas conseguiram obter resultados bastante altos para a tarefa de tradução para ambas as combinações de línguas testadas. Nota-se ainda que a existência da rede LSTM bidirecional não afeta em muito os resultados, quando comparando à rede equivalente sem camadas bidirecionais.

4.6. Conclusões

Neste capítulo foram apresentadas as arquiteturas gerais do sistema e de componentes mais específicas do mesmo, sendo mais detalhadas aquelas que apresentam maior importância para a realização do trabalho aqui descrito, sem prestar atenção a detalhes de implementação. Foram ainda apresentados estudos e resultados obtidos acerca de ferramentas de transcrição de áudio para texto, de modo a ser selecionada uma ferramenta para a tarefa de transcrição no produto final, e ainda experimentações de arquiteturas de redes neurais para se estudar qual poderá apresentar melhores resultados na implementação final. Para a ferramenta de transcrição, a Wit.AI alcançou melhores resultados, e para a arquitetura de rede neuronal, a Encoder-Decoder com camadas LSTM não bidirecionais, apresentaram resultados ligeiramente superiores. As implementações desenvolvidas neste trabalho para os diversos componentes mais pequenos e subsequentes junções, de modo a formar o módulo de tradução de português para língua gestual e integração com o sistema como um todo serão apresentadas no próximo capítulo. É também importante de realçar que, apesar da inexistência atual de corpus para a tradução de português para LGP, um será construído durante e após o término deste trabalho, permitindo que em

trabalhos futuros na área exista uma base sólida que possa ser utilizada. A rede neuronal final implementada para a melhoria das traduções conseguidas pela utilização das regras gramaticais irá utilizar uma versão inicial, mas bastante pequena, do corpus que se pretende ir construindo.

5. IMPLEMENTAÇÃO DO SISTEMA

Neste capítulo serão apresentadas as implementações desenvolvidas no decorrer deste trabalho, para a criação das várias componentes e a comunicação entre as mesmas e ainda a implementação de integrações para a comunicação com sistemas externos. Estas implementações serão detalhadas, identificando-se abordagens utilizadas e considerações tomadas.

Iniciar-se-á pela apresentação das características computacionais da máquina de desenvolvimento do trabalho, tanto a nível de Hardware como de Software. De seguida, será apresentada a implementação do módulo proposto de tradução de português para a animação representativa da glosa da LGP e respetiva integração do mesmo com as ferramentas e protocolos estabelecidos para a comunicação do módulo com o sistema geral. Por fim, será detalhada a construção da componente de tradução de português para a glosa da LGP deste módulo, módulo que recebeu o nome de Glossifica (tornar algo em glosa). O avatar 3D utilizado não faz parte do âmbito deste trabalho, sendo apenas mencionado de forma breve neste capítulo, uma vez que foi utilizado apenas para a apresentação de alguns exemplos.

5.1. Características Computacionais

Todas as componentes desenvolvidas neste trabalho utilizam a linguagem Python, na sua versão 3.9.9 64-bit para Windows, e respetivas bibliotecas disponíveis nesta linguagem para o Sistema Operativo, Windows 11 Education, versão 21H2. Esta linguagem de programação foi selecionada devido à existência de imensos recursos, com especial atenção aos focados em *Machine Learning* e *Processamento de Linguagem Natural*. Em termos de hardware, a máquina de desenvolvimento conta com um processador Intel i7-10700K (8-core, 16 threads, 3.8GHz), placa gráfica NVIDIA GeForce RTX 3060 12GB, 64GB de RAM, e 4TB de armazenamento (2TB SSD + 2TB HDD). Todas as implementações aqui apresentadas seguem as arquiteturas apresentadas anteriormente no capítulo 4. (Arquitetura do Sistema).

5.2. Tecnologias e Ferramentas Utilizadas

5.2.1. Python

O Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos. É uma linguagem muito utilizada na área da Inteligência Artificial e subáreas, devido a fatores como a consistência e a simplicidade do código, a grande variedade de bibliotecas e frameworks, muitas delas orientadas para tarefas comuns de Inteligência Artificial, a existência de uma grande comunidade de programadores, e ser independente da plataforma onde é executado o código.

5.2.2. spaCy

O spaCy [68] é uma biblioteca open-source para Processamento de Linguagem Natural (NLP) em Python. O spaCy foi desenhado para ser utilizado na produção de aplicações que necessitem de tratar grandes volumes de texto. As principais funcionalidades do spaCy são a Part-of-speech (POS) Tagging, que consiste em atribuir características morfológicas a palavras, Análise de Dependências, Lematização, Classificação de Texto, Detecção de Frases, entre outras. Além de ser utilizado para a extração de informação, pode ainda ser utilizado para a realização de pré-processamento de texto, na sua utilização em deep learning.

5.2.3. Keras

O Keras [69] é uma biblioteca open-source escrita em Python que permite um desenvolvimento fácil e rápido de redes neurais sobre frameworks, tal como a TensorFlow [70], Theano [71], Microsoft Cognitive Toolkit [72], entre outras. O Keras disponibiliza uma API simples e consistente e bastante abstrata, de modo a facilitar a implementação de redes neurais nas várias frameworks compatíveis.

5.2.4. Sockets e Websockets

Os Sockets e os Websockets conceptualmente possuem funcionamentos semelhantes, no sentido em que são canais que se mantêm abertos para a troca de dados entre duas máquinas. No entanto, enquanto os Sockets representam endpoints criados em cada uma das máquinas, sobre o protocolo TCP – *Transmission Control*

Protocol, os WebSockets são um protocolo de camada da aplicação do modelo TCP/IP. Ambas as tecnologias foram utilizadas neste trabalho para a comunicação de dados com sistemas remotos.

5.3. Módulo de Tradução de Português para LGP e Integração no Sistema Completo

O módulo de tradução de português para LGP representa o módulo completo, desde a receção de uma mensagem de texto ou áudio, até ao envio, do vídeo com a representação de cada termo da glosa pelo avatar 3D, ao utilizador que fez o pedido de tradução. Este módulo integra-se com o sistema geral, através de websockets, sobre o qual será obtida a receção da mensagem ou áudio a traduzir, e ainda o envio do vídeo com a representação em formato Base64. Inicialmente, é realizado um pedido via HTTP do tipo POST ao serviço de autenticação oauth. Este pedido HTTP é realizado através da utilização da biblioteca *Requests*. Seguidamente, fornecidos o nome de utilizador, a palavra-passe e os restantes parâmetros, este serviço devolve uma *access token*, token esta que deve ser incluída como parâmetro na url do websocket, de forma modo a obter-se a ligação com sucesso ao mesmo. Para a obtenção da *access token*, foi desenvolvido um script com uma função específica para a obtenção da token, dados os parâmetros necessários. A colocação desta função num script à parte permite a reutilização do mesmo, sempre que necessário, através da sua importação sempre que necessário. Além da conexão ao websocket do sistema geral, é instanciado um servidor socket para que o cliente de animação, Unity, se conecte, estabelecendo-se um caminho que ficará responsável por enviar a glosa para o cliente de animação e receber a notificação de término da animação e gravação em ficheiro de vídeo.

Este módulo, mal é iniciado, conecta-se ao websocket do sistema geral. Após esta ligação, o módulo fica num estado de espera, estado este que indica que o módulo aguarda receção de uma mensagem a traduzir. Caso receba áudio, entra num estado de transcrição, estado este que representa o envio do áudio para o serviço de transcrição de áudio para texto, e respetiva receção de uma resposta. A resposta do serviço de transcrição, ou a receção de uma mensagem de texto, passam o sistema para o estado de tradução, estado que representa a utilização do Tradutor de Português para Glosa da

LGP. A receção da resposta deste tradutor, transita o estado do sistema para o estado de animação, estado onde a glosa traduzida é submetida pelo socket ao cliente de animação, desde que este último se tenha ligado ao socket. A ligação do cliente de animação não é forçada, mas resultará num erro durante o processo, sendo dada uma resposta apenas com a glosa. A receção de resposta favorável do cliente de animação, indica a finalização da gravação da animação, estando o vídeo pronto para ser enviado. Após a passagem pelo estado de envio de vídeo, onde o vídeo é submetido ao sistema geral, através da ligação websocket ao mesmo, este módulo volta ao estado original, aguardando novas mensagens. A solução atual prevê a utilização de threads para a realização de várias traduções em simultâneo, sendo um thread instanciado no momento da receção de uma mensagem e terminado após o envio da resposta. No entanto, o número de threads disponíveis para a realização de tradução é limitado pela quantidade de clientes de animação disponíveis, sendo que a escalabilidade dos clientes de animação está limitada aos recursos computacionais disponíveis, sendo uma escalabilidade muito custosa.

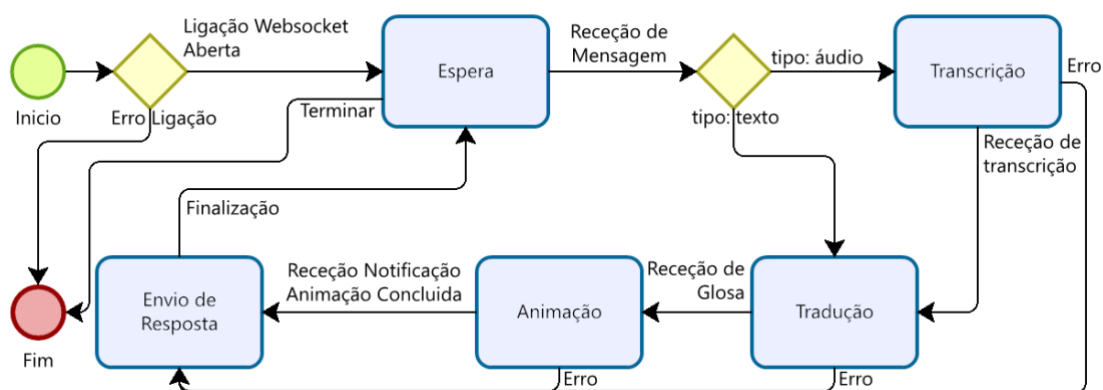


Figura 31 - Diagrama de Estados Representativo do Módulo de Tradução de Português para Animação LGP.

5.4. Tradutor de Português para Glosa da LGP

Tal como referido anteriormente, esta componente do módulo de tradução é a mais explorada e desenvolvida no trabalho que aqui se descreve. Foi dado um grande foco à implementação desta componente, pelo facto de ser a responsável por converter o português natural para a estrutura da glosa da LGP. Esta conversão é muito importante, uma vez que a maioria dos surdos terá dificuldade em conseguir compreender a frase rapidamente, ou até poderá não conseguir compreender a frase totalmente, devido às

grandes diferenças estruturais e gramaticais entre ambas. Do mesmo modo que uma pessoa falante poderá demorar a compreender uma frase de uma língua estrangeira, quando ainda se encontra em fase de aprendizagem da mesma, também de modo análogo se aplica à leitura por parte de um surdo do texto em português. Tendo em consideração estas questões, consegue-se perceber a grande necessidade de implementação de uma boa conversão da gramática do português para a gramática da LGP. No entanto, a existência de muito pouca bibliografia acerca desta gramática, torna muito difícil a implementação de uma conversão totalmente correta, devido à necessidade de inferência de regras gramaticais que possam não estar apresentadas na bibliografia utilizada e estudada, ou até devido à possibilidade de existência de algum erro no detalhe dessas regras. Adicionalmente à utilização de bibliografia e outros trabalhos na área, na procura e determinação de regras gramaticais, é também mantida, neste trabalho, uma comunicação ativa com um formador (surdo) e uma intérprete de LGP.

5.4.1. Conversão de SVO para OSV e Aplicação de Regras Gramaticais

A primeira subcomponente do tradutor de português para a glosa da LGP envolve, inicialmente, a realização de modificações à estrutura da frase original, convertendo-a para a estrutura da frase final, neste caso de SVO (Sujeito-Verbo-Objeto), em português, para OSV (Objeto-Sujeito-Verbo), na LGP. Também nesta parte que envolve modificar a estrutura frásica, além da modificação de SVO para OSV, existem outras regras que modificam parte da estrutura da frase. A segunda parte envolve a aplicação de outras regras que já não modifiquem a estrutura da frase, como por exemplo inserir ou remover palavras.

No passo da modificação da estrutura frásica, procura-se, inicialmente, determinar em que locais uma frase complexa pode ser dividida em frases simples. Considera-se uma frase simples, uma frase que possua apenas as componentes SVO, e uma frase complexa, uma frase que possua uma das componentes a repetir-se mais do que uma vez. Por exemplo, nas frases apresentadas em Exemplo 1, a frase A é considerada simples, enquanto a B e C seriam consideradas complexas. A frase A segue a estrutura SVO, enquanto a frase B e C seguem SVO+SVO. Outras estruturas podem ser encontradas, não só a combinação de frases SVO.

O João chutou a bola. (A)

O João chuta a bola e a Maria corre para casa. (B)

O João chutou a bola mas a Maria foi apanhar o gato. (C)

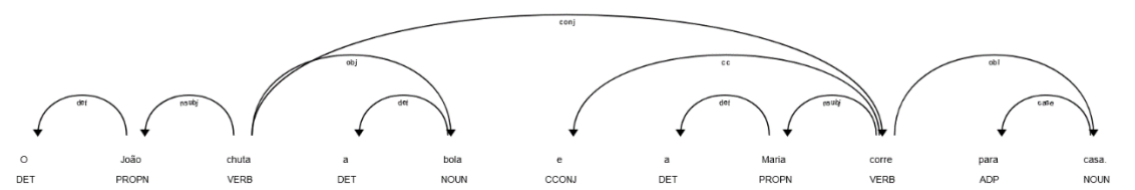
Exemplo 1 - Frases Simples e Complexas.

O primeiro método de determinação de frases simples envolve a utilização do método específico da biblioteca spaCy para o efeito. O «objeto spacy».sents fornece o acesso a uma lista de frases simples, determinadas pelos modelos existentes no spaCy. No entanto, este método nem sempre determina todas as frases simples a partir de uma frase complexa, muitas vezes resultando em sub-frases complexas, mas mais curtas que a original. No entanto, aparenta realizar a divisão em locais corretos, sejam as frases resultantes simples ou complexas. Este comportamento está apresentado Tabela 5, que são as divisões conseguidas automaticamente pelo spaCy sobre as frases do Exemplo 1. A frase B, que deveria de estar dividida no ‘e’, não foi dividida pelo spaCy.

Frases	Divisão
A	F1: O João chutou a bola.
B	F1: O João chuta a bola e a Maria corre para casa.
C	F1: O João chutou a bola F2: a Maria foi apanhar o gato

Tabela 5 - Divisões pelo método .sents do spaCy.

Para frases como a B, uma outra regra de divisão foi implementada. Analisando a árvore de dependências da frase B, apresentada na Figura 32, e de frases semelhantes, separadas por conjunções coordenadas, optou-se por realizar a divisão na conjunção coordenada (CCONJ), desde que esta tenha um verbo como ‘cabeça’. No entanto, esta regra depende, tal como outras, que a análise gramatical e morfológica efetuada pelo spaCy esteja correta. Um fator encontrado para uma melhor performance nesta tarefa



é a utilização de frases corretamente escritas. Frases mal escritas poderão resultar em

Figura 32 - Árvore de Dependências Frase B.

classificações erradas, e consequentes problemas na obtenção da tradução. Após a aplicação destas regras, que procuram numa primeira fase separarem-se as frases simples em orações coordenadas (orações independentes das outras) que existam em frases complexas, segue-se um passo que se designou por '*Hierarquização Frásica*', passo este que procura criar algo semelhante a uma árvore hierárquica com divisões de SV (Sujeito-Verbo) e O (Objeto). Se o O dividido possuir a estrutura SV ou SVO, será criado um nível na hierarquia para o tratamento dessa 'segunda' frase SV ou SVO, e assim repetitivamente até que não seja possível continuar a criar níveis. Cada nível é representado pelo seguinte objeto json:

```
obj_nivel_hierarquia = {  
  frase: texto, // frase a ser trabalhada neste nível  
  nivel_atual: [palavra, palavra, ...], // palavras que fazem parte do S e V do nível  
  atual  
  ligacao: [palavra, palavra, ...], // conector entre nível atual e próximo  
  obj: [palavra, palavra, ...], // palavras que fazem parte do objeto deste nível  
  seguinte: (obj_nivel_hierarquia), // objeto para o nível / frase seguinte  
  (geralmente resto da frase após objeto da atual)  
  negacao: {True/False}, // se frase atual apresenta negação  
  interrogacao: {True/False}, // se frase atual apresenta interrogação  
  exclamacao: {True/False}, // se frase atual apresenta exclamação  
}
```

Excerto de Código 5 - Objeto da hierarquia frásica.

Nesta hierarquização frásica, para cada palavra de cada nível, começa-se por ignorar artigos definidos (e, a, uma, uns, etc..) e palavras auxiliares, já que estas não fazem parte da LGP. De seguida, são aplicadas regras que procuram colocar números e determinantes possessivos após o substantivo. Para tal, as palavras que coincidem com estas regras são colocadas em arrays temporários, para serem colocadas no local correto mais à frente. No caso dos números, verifica-se ainda se o número possui uma dependência do tipo 'nummod' (modificador de número) para a palavra que é a sua 'cabeça', e se essa palavra está presente na lista de medidas (por exemplo, metro, centímetro, cm, euro, €, entre outras). Se sim, essa palavra também é adicionada ao array temporário de números a serem novamente inseridos na frase. Se a palavra a ser analisada é tratada por alguma destas regras, passa-se para a iteração seguinte (para a palavra seguinte). A regra seguinte verifica se a palavra a ser tratada é um advérbio temporal. Para tal, é verificado se a palavra é do tipo 'ADV' e se esta se encontra numa lista de advérbios temporais. Se sim, a mesma é colocada no início do objeto (obj)

do nível atual, e passa-se para a iteração seguinte. A próxima regra verifica se a palavra a ser tratada é um advérbio de negação. Se sim, o parâmetro 'negacao' é colocado a True, e passa-se para a próxima palavra. Para as seguintes regras, se o objeto da frase ainda **não** tiver sido encontrado, se a palavra a ser tratada for uma conjunção subordinada (SCONJ) e esta depender de uma palavra verbo ou auxiliar (VERB ou AUX), encontrou-se a ligação entre orações subordinadas. Logo, essa conjunção subordinada é removida da frase que está a ser tratada e é inserida no parâmetro 'ligacao' do nível atual da hierarquia frásica e passa-se para a palavra seguinte. De seguida, se a palavra for um verbo, encontrou-se o verbo do nível atual. Este verbo é colocado no parâmetro 'nivel_atual' do nível atual, parâmetro este onde se encontram colocadas todas as palavras do sujeito e do verbo. Para as restantes palavras, se o verbo ainda não tiver sido encontrado, a palavra é adicionada no parâmetro 'nivel atual', caso contrário, será colocada no parâmetro 'obj' do nível atual. Durante a análise das palavras que não coincidam com nenhuma das regras anteriores, e se o verbo já tiver sido encontrado, caso exista uma dependência do tipo 'obj' na palavra que está a ser analisada, a mesma é colocada no nível atual, mas considera-se que foi encontrado o objeto deste mesmo nível. Com o objeto já encontrado, se a palavra que está a ser analisada é um verbo, e ainda existirem palavras após esse verbo, cria-se um nível novo na hierarquia. Este novo nível, com formato igual ao Excerto de Código 5, ficará com o resto da frase que ainda não foi tratada, sendo que a palavra que originou a criação do novo nível fará parte desse nível que acabou de ser criado. Neste momento, as palavras que se encontravam nos arrays temporários de números e determinantes possessivos são colocados no final do objeto do nível que acabou de ser analisado, e passa-se para o tratamento do novo nível da hierarquia. Isto repete-se até que deixem de existir níveis 'seguintes' a tratar. Deixando de existir níveis a tratar, se os arrays temporários de determinantes possessivos ou números possuírem palavras, essas palavras serão colocadas no parâmetro 'obj' do último nível que foi tratado. Após a realização desta etapa de iterações, os níveis da hierarquia são combinados segundo o Algoritmo 1, resultando na primeira tradução baseada em regras do Tradutor de Português para a Glosa da LGP. A Figura 33 apresenta uma conversão realizada e legendada, consoante o seu tipo de componente da estrutura frásica (Sujeito, Verbo ou Objeto). No topo da imagem, está apresentada a frase original, SVO, anotada, a verde (o Sujeito), a azul (o Verbo) e a roxo

(os Objetos). E, no final da imagem, a frase final em OSV, também anotada com as mesmas cores e legenda, mostrando que a estrutura é, de facto, OSV.

```
array_palavras_ordenadas = []
nivel_atual = hierarquia_frasica (primeiro nível da hierarquia)
Enquanto nivel_atual['seguinte'] existir: (ainda existem níveis a serem tratados)
    Se nivel_atual['negacao']:
        Adicionar Palavra 'Não' a array_palavras_ordenadas
        Adicionar palavras em nivel_atual['ligacao'] a array_palavras_ordenadas
        Adicionar palavras em nivel_atual['nivel_atual'] a array_palavras_ordenadas
        Adicionar palavras em nivel_atual['obj'] a array_palavras_ordenadas
    nivel_atual = nivel_atual['seguinte']
```

Inverter array_palavras_ordenadas

frase_SVO_para_OSV = **Concatenar** array_palavras_ordenadas

Algoritmo 1 - Junção dos níveis da hierarquia frásica.

```
[>] Frase original: A Ana e o Miguel chutaram a bola, sendo que o João foi apanhar o gato
[>] Frases individuais determinadas
    > A Ana e o Miguel chutaram a bola, sendo que o João foi apanhar o gato
        Sujeito F1      Verbo F1      Objeto F1      Sujeito F2      Verbo F2      Objeto F2
[>] Hierarquização de subfrases
    > A iniciar criação da hierarquia para conversão de SVO para OSV...
        - Frase: A Ana e o Miguel chutaram a bola, sendo que o João foi apanhar o gato
            {
                'frase': 'A Ana e o Miguel chutaram a bola, sendo que o João foi
                'obj': [bola, ,],
                'ligacao': [],
                'nivel_atual': [Ana, e, Miguel, chutaram],
                'seguinte': {
                    'frase': o João foi apanhar o gato,
                    'ligacao': [],
                    'obj': [gato],
                    'nivel_atual': [João, apanhar],
                    'seguinte': None,
                    'negacao': False,
                    'interrogacao': False,
                    'exclamacao': False
                },
                'negacao': False,
                'interrogacao': False,
                'exclamacao': False
            }
        - Array níveis Juntos: [[gato], [João, apanhar], [], [bola], [Ana, e, Miguel,
        - Conversão SVO -> OSV: gato João apanhar bola Ana e Miguel chutaram
                                Suj F2  Suj F2  Suj F2  Suj F1  Suj F1  Suj F1
```

Figura 33 - Exemplo de Conversão SVO -> OSV.

Após a realização do tratamento ao nível da estrutura da frase, passa-se para um tratamento ao nível da palavra, que não afete a estrutura e ordem das palavras, ficando-se pela remoção, modificação e inserção de palavras. Este segundo conjunto de regras implementado, funciona do modo que a seguir se descreve. Para cada uma das palavras, sendo um nome próprio, esse mesmo nome será separado em letras e adicionado ao array final, e passa-se à palavra seguinte. Se a palavra for numérica, inicialmente verifica-se se o número inteiro é 10, 100 ou 1000. Para estes, são associados os gestos

específicos para esses números. Para outros números e casas decimais, todos os números são separados “um-a-um”, semelhante a uma soletração do número. A regra seguinte a ser analisada trata nomes não próprios femininos. Se a palavra é nome não próprio feminino, para a maioria destes casos é necessário adicionar a palavra ‘MULHER’ antes do nome. No entanto, no caso de alguns nomes isso não acontece, como por exemplo ‘bola’, que não possui equivalente no gênero masculino, ou outros como ‘Galo’ e ‘Galinha’ que possuem nomes diferentes. Para estes casos, foi utilizado e modificado o LABEL_LEX_SW⁴, um ficheiro com cerca de 1 milhão de palavras flexionadas e a cada uma das palavras foi associada uma série de características morfológicas como, por exemplo, se a palavra é aumentativa ou diminutiva. A modificação efetuada ao ficheiro do LABEL_LEX_SW é a inclusão de um novo parâmetro manualmente *-AF (Apenas Feminino)* após as palavras que não possuam equivalência masculina. Deste modo, sempre que um nome feminino está a ser verificado, procura-se o mesmo no LABEL_LEX_SW modificado. Se esse nome não possuir o modificador *-AF*, deve-se adicionar a palavra MULHER antes do nome para marcar o gênero feminino, senão, mantém-se apenas a palavra. Ainda no processo de tratamento dos nomes femininos, e utilizando o LABEL_LEX_LW, verifica-se se o nome é diminutivo (possui o parâmetro ‘:D’ no LABEL_LEX_SW) ou aumentativo (possui o parâmetro ‘:S’), sendo, para estes, adicionadas, antes do nome, as palavras ‘PEQUENA’ ou ‘GRANDE’, respetivamente. Após o tratamento dos nomes não próprios femininos, passa-se para a palavra seguinte. A próxima regra faz a remoção de determinantes artigos, caso não tenham sido removidos no momento de conversão de SVO para OSV. Também se removem aposições. Se a palavra for afetada por alguma destas regras, passa-se à iteração seguinte. A regra seguinte aplica-se a verbos e no caso da palavra a ser tratada ser um verbo, procura-se substituir o verbo pelo seu infinitivo. Esta substituição é também efetuada com recurso ao LABEL_LEX_SW. Se o verbo ou a flexão do verbo não existirem no LABEL_LEX_SW, mantém-se a forma como está escrito. Sendo um verbo afetado, passa-se à próxima palavra. A última regra existente verifica diminutivos ou aumentativos, sendo esta verificação realizada da mesma forma como quando os graus substantivos são verificados nos nomes não próprios femininos, com recurso ao LABEL_LEX_SW. Se uma

⁴ http://label.ist.utl.pt/pt/labellex_pt.php - LaBEL - Laboratório de Engenharia da Linguagem

palavra não for afetada por nenhuma destas regras, a mesma é mantida sem qualquer modificação. No final, a concatenação de tudo resulta na primeira tradução para a glosa da LGP.

Todas as regras utilizadas, tanto para a conversão da estrutura da frase, como aquelas aplicadas ao nível da palavra, foram determinadas através das características apresentadas no capítulo 2.1. Língua Gestual Portuguesa, e pela análise das relações e anotações atribuídas pelo spaCy à frase, com ajuda da visualização gráfica de árvores de dependência para várias frases de teste, como a exemplificada na Figura 32. As regras implementadas e descritas anteriormente neste tópico encontram-se sumariadas nas tabelas Tabela 6 e Tabela 7, regras aplicadas na modificação da estrutura frásica e regras aplicadas ao nível da palavra, respetivamente.

Regra #	Descrição
1	Divisão de frases independentes (orações coordenadas)
2	Para cada palavra de cada frase independente:
2.1	Ignorar determinantes artigos
2.2	Ignorar palavras auxiliares (exceto se representarem futuro ou passado)
2.2.1	Se auxiliar apresenta significado de futuro, adicionar palavra 'BREVE' (ao objeto se este já tiver sido encontrado, caso contrário ao nível_atual que representa sujeito e verbo)
2.2.2	Se auxiliar apresenta significado de passado, adicionar palavra 'PASSADO' (ao objeto se este já tiver sido encontrado, caso contrário ao nível_atual que representa sujeito e verbo)
2.3	Passar determinantes possessivos para depois do substantivo
2.4	Passar números (e medidas) para depois do substantivo
2.5	Passar advérbios de tempo para o início do objeto
2.6	Determinar se existe negação, interrogação ou exclamação
2.7	Se objeto já tiver sido encontrado, e palavra atual for verbo, criar nível da árvore hierárquica
2.8	Se palavra analisada for conjunção subordinada, e cabeça dessa palavra verbo ou auxiliar, palavra de "ligação" de orações encontrada
2.9	Se palavra analisada for verbo, verbo foi encontrado, indicando divisão entre SV e O (Sujeito-Verbo e Objeto). Palavra mantida no parâmetro 'nível_atual' (que representa o SV)

2.10	Se verbo já encontrado, palavra analisada adicionada ao parâmetro 'obj' (objeto da frase que está a ser analisada), desde que não tenha sido tratada por outra regra anteriormente. Se dependência da palavra é 'obj', encontrou-se o objeto da frase
2.11	Se verbo ainda não encontrado, e palavra ainda não tratada por outra regra, adicionar palavra ao parâmetro 'nivel_atual' (que representa o SV)
3	Reconstruir frase a partir da hierarquia frásica criada na regra 2 e sub-regras. Para cada nível da hierarquia frásica:
3.1	Se negação, interrogação ou exclamação determinadas, inserir palavra/símbolo correspondente (NÃO, ?, !)
3.2	Inserir palavras de ligação de orações (se existirem)
3.3	Inserir palavras do nível atual (palavras do Sujeito e Verbo) (se existirem)
3.4	Inserir palavras do objeto (se existirem)
4	Após junção, inverter e concatenar frases independentes

Tabela 6 - Regras de modificação da estrutura das frases.

Regra #	Descrição
1	Para cada uma das frases independentes reestruturadas:
1.1	Se palavra é nome próprio
1.1.1	Verificar acentuações (acentos devem ser separados das letras a soletrar)
1.1.2	Soletrar nome
1.2	Se palavra é número, se for 10, 100 ou 1000, adicionar termo equivalente, senão, "soletrar" número a número, incluindo casas decimais (se existirem)
1.3	Se palavra é nome não próprio feminino, verificar se existe equivalente masculino. Se sim, adicionar termo 'MULHER' antes do nome (salvo casos como galo ou galinha que possuem gestos diferentes). Verificar também se é aumentativo ou diminutivo e adicionar termo 'GRANDE' ou 'PEQUENO' respetivamente
1.4	Se palavra é determinante artigo, ignorar
1.5	Se palavra é adposição, ignorar
1.6	Se palavra é verbo, substituir pela forma infinitiva do verbo (se conseguir, senão, manter)
1.6.1	Se verbo representa passado, adicionar palavra 'PASSADO'
1.6.2	Se verbo representa futuro, adicionar palavra 'BREVE'

1.7	Se palavra é aumentativo ou diminutivo, adicionar termos 'GRANDE' ou 'PEQUENO' respetivamente
1.8	Se palavra não tratada por nenhuma regra, manter
2	Concatenar todas os termos finais das frases independentes
3	Concatenar todas as frases independentes, com indicação de pausa longa

Tabela 7 - Regras de modificação a nível da palavra das frases.

Tal como referido anteriormente, a tradução final da combinação destas regras todas resulta na primeira tradução de texto da língua portuguesa para a glosa da LGP, tendo sido criado um tradutor baseado em regras. No entanto, como as regras são muito específicas, deve-se ter em consideração que o bom funcionamento dependerá muito do quão bem escrita a frase original está, principalmente se esta for complexa. Para mitigar este problema, e melhorar traduções de frases corretamente escritas, utilizar-se-á uma rede neuronal treinada para o efeito, cuja implementação será detalhada no próximo tópico.

5.4.2. Construção e Utilização de Rede Neuronal

A segunda subcomponente do tradutor de português para a glosa da LGP envolve a utilização de uma Rede Neuronal para a melhoria das traduções alcançadas pela primeira subcomponente. Considerando os resultados dos testes realizados em 4.5. Seleção de Rede Neuronal, optou-se pela utilização de uma rede neuronal do tipo Encoder-Decoder com LSTM, não bidirecional, devido a esta ter apresentado resultados ligeiramente superiores, e da não existência de tradução “em direto”, isto é, à medida que o texto está a ser escrito (uma rede com camada bidirecional alcançaria melhores resultados para este último caso, devido a conseguir inferir o que poderia ser o resto da frase ao analisar a mesma, tanto do início para o fim, como do fim para o início).

O processo de criação da rede neuronal é composto por quatro etapas principais, Limpeza/Normalização dos Dados, Preparação dos Dados, Treino e Avaliação, cada um assumindo tarefas diferentes.

- **Limpeza/Normalização do Texto:** Neste passo todos os dados originais do corpus são tratados de modo que todos possuam o mesmo formato e estrutura. Fazem parte desta etapa as tarefas como a remoção de pontuação (ignorada na LGP, exceto a interrogação e a exclamação),

conversão de capitalização, remoção de caracteres inválidos, divisão da frase em palavras separadas, e ainda conversão de todos os nomes e números para uma notação especial. Esta última conversão é realizada com o objetivo de deixar a rede mais flexível quando aparecem nomes ou números. Os Exemplo 2 e Exemplo 3 pretendem exemplificar o porquê desta substituição dos nomes e números para notações especiais.

*(frase treino original) O **Pedro** pediu **2.5kg** de frango e **100g** de fiambre.*

*(frase convertida) O **NOME1** pediu **NUM1kg** de frango e **NUM2g** de fiambre.*

Exemplo 2 – Frases para treino da rede neuronal.

Considere-se o treino de dois modelos, um para frases originais e outros para frases convertidas. Ao inferir a tradução para uma nova frase, se fossem fornecidas as seguintes frases às respetivas redes neuronais:

(frase teste original) O Ana pediu 1kg de frango e 250g de fiambre.

*(frase convertida) O **NOME1** pediu **NUM1** de frango e **NUM2** de fiambre.*

Exemplo 3 – Frases para teste da rede neuronal.

A rede neuronal treinada com as frases originais pode ter maior dificuldade em inferir que Ana é um nome como Pedro era nas frases de treino, bem como os números 2.5 e 100 representam o ‘mesmo’ que os 1 e 250 na frase de treino, podendo mais facilmente resultar em maiores erros na tradução inferida pela rede, principalmente com corpus de treino pequenos, já que não conseguirá com poucos dados apurar corretamente muitas regras. Já a rede neuronal para as frases convertidas, teria maior facilidade em fornecer a resposta, já que a frase convertida a inferir é igual a uma já vista anteriormente (ou seguiria um padrão muito semelhante), conseguindo-se, mais facilmente, inferir o resultado esperado (o resultado também deve possuir os nomes e números convertidos para a notação utilizada). No fim, para se voltar a possuir os nomes e números originais, basta substituí-los no resultado da inferência da rede neuronal. O Apêndice 9.2. (Experimentação e Comparação de Tradutor Neuronal entre Línguas com e sem a Utilização de Categorias Gramaticais Para a Tradução) apresenta alguns resultados de um curto teste da realização desta conversão, provando que a substituição destes valores apresenta bons resultados. Também [73] e [74] realizaram

testes à utilização das categorias gramaticais para o treino de redes neurais para tradução. Estes últimos, concluem que a utilização de categorias gramaticais ajuda a melhorar as traduções, sendo que o primeiro indica ainda melhorias substanciais para poucos dados (corpus pequenos), embora, utilizem as categorias gramaticais para todas as palavras, em vez de para apenas algumas;

- **Preparação dos Dados:** Nesta etapa são definidas as percentagens sobre o total de dados do corpus de dados que serão utilizados como dados para treino e dados para teste. Além da divisão, realizada de modo aleatório (os dados originais são reordenados aleatoriamente antes da divisão), após essa divisão, são criados ficheiros com os dados divididos, prontos a serem utilizados no treino e avaliação da rede;
- **Treino:** Nesta etapa, são definidos os parâmetros a serem utilizados e o tipo de rede neuronal e respetivas camadas para a criação da rede neuronal para os dados divididos na etapa anterior. O modelo desta rede é da classe sequencial, o que significa que cada uma das camadas adicionadas ao modelo será aplicada sequencialmente, pela ordem em que foram adicionadas. O modelo, para o lado do Encoder, é composto por uma camada Embedding, e uma camada LSTM. A Embedding é uma camada que é utilizada para converter palavras para vetores numéricos, que conseguem ser utilizados pela rede neuronal. De seguida, os dados são passados à primeira camada LSTM, camada responsável por receber o valor atual a converter, e fornecer o resultado, considerando as conversões anteriores. Aqui termina a Encoder, já que a LSTM fornece os valores codificados. De seguida, para o lado do Decoder, aplica-se a camada RepeatVector. Esta camada basicamente adiciona uma dimensão a um vetor, sendo necessária, já que o resultado da LSTM do Encoder é um vetor de duas dimensões, mas a LSTM do Decoder necessita, como entrada, de um vetor de três dimensões. Após a RepeatVector, vem a camada LSTM do Decoder, que tendo em consideração os valores resultantes da Encoder, vai fornecendo valores, como resultado, a cada passo da sua execução. De seguida, existe uma camada TimeDistributed, que basicamente é uma camada que serve

para aplicar outra camada qualquer em todos os passos da execução da rede neuronal. Neste caso, a TimeDistributed aplica uma camada Dense a cada iteração da LSTM, sendo que esta camada Dense é a responsável, neste problema, por converter os valores numéricos para a palavra resultante esperada. Esta arquitetura da Encoder-Decoder está apresentada na Figura 34.

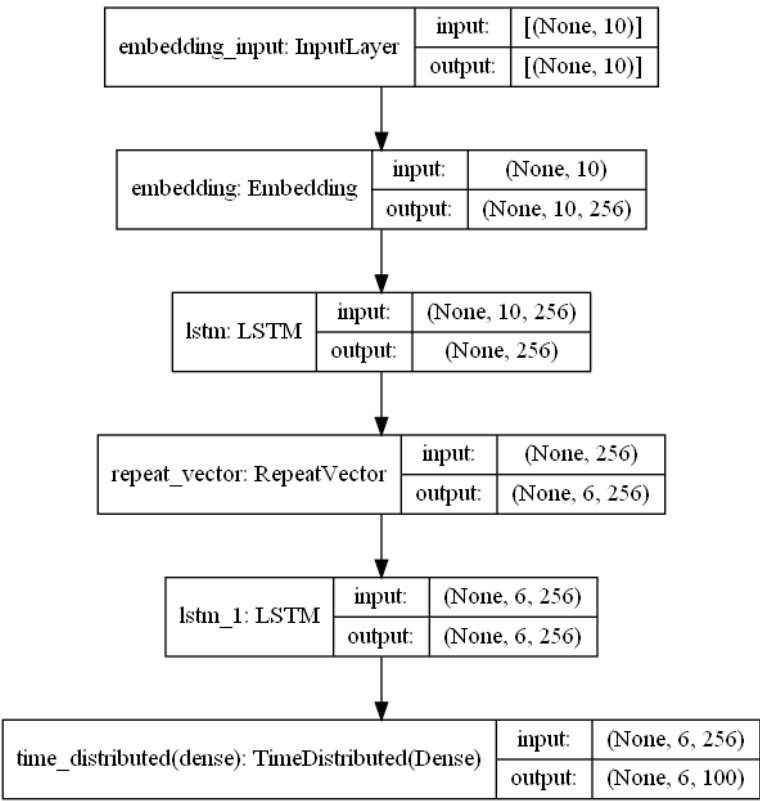


Figura 34 - Modelo Encoder-Decoder Definido.

Em termos de hiperparâmetros definidos, os mesmos estão apresentados na Tabela 8.

Hiperparâmetro	Valor
Optimizer	Adam
Loss function	Categorical crossentropy
Activation funcion	Softmax
Embedding size	256
Metrics	Accuracy
Batch Size	64
Epochs	200

Tabela 8 – Hiper parâmetros utilizados na Rede Neuronal.

O resultado desta etapa é uma rede neuronal, neste caso para a tradução de português para glosa da LGP, treinada e pronta a ser testada e avaliada, avaliação que será realizada na etapa seguinte.

- **Avaliação:** Nesta etapa é avaliada a rede neuronal criada na etapa anterior. A avaliação realiza-se através dos dados de teste, e a comparação dos resultados esperados que estão nesses dados de teste, com os resultados fornecidos pela rede neuronal. Para esta comparação, são utilizadas métricas como a BLEU. O modelo criado apresentou uma precisão de 97%. A avaliação do modelo com a métrica BLEU apresentou os seguintes resultados: BLEU-1: 0.994528 | BLEU-2: 0.989542 | BLEU-3: 0.930256 | BLEU-4: 0.693745. No entanto, deve-se ter em consideração o tamanho muito pequeno do corpus utilizado (aprox. 2200 traduções), que afetará negativamente a utilização do modelo, para inferir traduções para frases fora do contexto das frases utilizadas para o treino do mesmo.

Após a aplicação de todos estes passos, considera-se ter a rede neuronal pronta para ser utilizada no tradutor híbrido de português para a glosa da LGP.

5.4.3. Tradutor Híbrido

O tradutor híbrido é o resultado da combinação dos dois tradutores desenvolvidos anteriormente, tendo sido nomeado de Glossifica. Este tradutor, acessível via método HTTP GET, recebe o texto a ser traduzido, e devolve um array, em formato json, com a tradução conseguida pelo tradutor baseado em regras, e a tradução melhorada pela rede neuronal, de modo que possam ser apresentadas traduções alternativas ao utilizador final.

Em termos de comportamento, o tradutor recebe uma frase a traduzir, via parâmetros GET de um pedido HTTP. Essa frase é então enviada ao tradutor baseado em regras, para a realização da primeira tradução baseada em regras. Dentro deste tradutor baseado em regras, a frase começa por ser morfológicamente analisada, passando para a reestruturação das mesmas, de SVO para OSV, e após a reestruturação, são aplicadas outras regras da LGP que não afetam a estrutura da frase, regras estas que apenas eliminam, substituem ou adicionam palavras. Da aplicação destes conjuntos de regras resulta a tradução baseada em regras para a frase, sendo uma das traduções

alternativas enviada de volta ao utilizador que fez o pedido de tradução. Uma cópia desta frase é também enviada ao tradutor baseado em redes neuronais, para que este último possa melhorar a tradução conseguida pelo tradutor baseado em regras. Inicialmente, este tradutor neuronal aplica o mesmo pré-tratamento de dados que realizou aos dados de treino, para que seja mantida consistência entre a forma como foi treinado e a forma de como será utilizado. Após o pré-tratamento, é então realizada a tradução por parte da rede neuronal. O resultado é também incluído nas traduções devolvidas ao utilizador. Após esta etapa, as duas traduções são combinadas num array, que depois de convertido para formato json, é devolvido ao pedido HTTP que iniciou o processo de tradução. Atualmente, devido ao pequeno tamanho do corpus utilizado para o treino, a tradução baseada em regras é considerada a tradução principal, sendo a dada pela rede neuronal a secundária, que, em certos casos estará, de facto, melhorada, e noutros casos, bastante pior. O servidor HTTP utilizado para a disponibilização do serviço de tradução para a Glosa foi criado através da biblioteca Flask. O código desenvolvido e utilizado para o tradutor híbrido e um excerto do código para o tradutor baseado em regras estão apresentados nos apêndices 9.3.2. Tradutor Híbrido e 9.3.3. Excerto de Código – Tradutor Baseado Em Regras. De forma a integrar este tradutor híbrido com a ligação ao sistema geral, que é conseguida através de websockets, construiu-se um cliente que quando recebe um pedido de tradução via websocket, cria um pedido HTTP para enviar ao tradutor híbrido. A resposta deste último é depois recebida pelo cliente específico para ligação websocket ao sistema geral. Esta separação do cliente websocket do tradutor híbrido foi implementada e considerada para futuras modificações à arquitetura do sistema, permitindo facilmente aceder e realizar-se traduções para a glosa, mesmo que a comunicação entre o sistema geral e a tradução para LGP deixe de ser realizada via websockets.

5.4.4. Exemplo de Utilização do Glossifica

Neste tópico serão apresentados alguns exemplos da utilização do tradutor híbrido (Glossifica) desenvolvido. De modo a realizar os pedidos HTTP GET necessários para o acesso ao tradutor híbrido, foi utilizada a ferramenta Postman, uma ferramenta desenvolvida para testar endpoints de APIs HTTP. O pedido é realizado à url: *«ip_servidor»:8081/traduz?texto=«texto a traduzir»*, substituindo o *«ip_servidor»* pelo

ip da máquina onde o serviço está a correr, neste caso, *localhost*, e substituindo o «texto a traduzir» pelo texto que se pretende traduzir.

A Figura 35 apresenta uma tradução exemplo realizada através do Postman [75]. Alguns exemplos de frases e traduções resultantes estão apresentados na Tabela 9, com o resultado esperado para cada uma das frases e avaliação BLEU e METEOR das traduções alcançadas na Tabela 10.

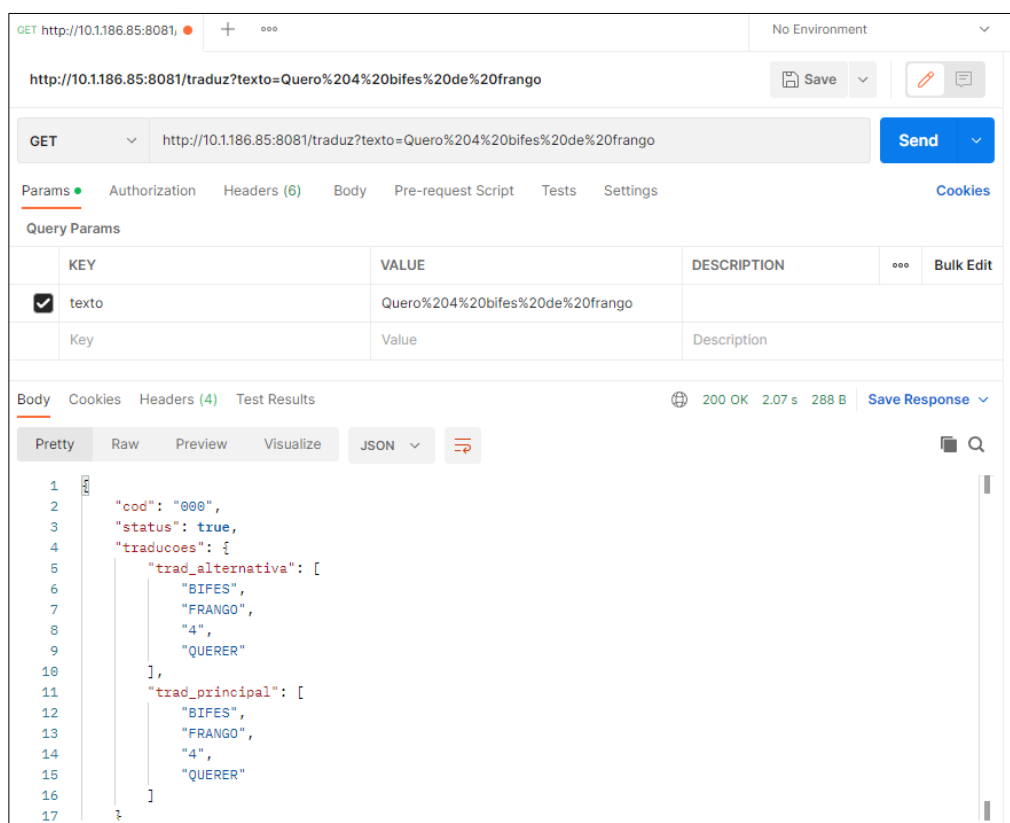


Figura 35 - Exemplo de Teste de Tradução realizado no Postman.

Número Frase	Frase Original	Tradução Baseada em Regras	Tradução Melhorada pela Rede Neuronal
1	O João passou as calças.	Calça J O A O ~ passar	J O ã O calças passarferro
2	Corte um pouco mais finos por favor.	Corte pouco mais fino favor.	Por favor maisfino.
3	O João é meu professor.	J O A O ~ professor meu.	J O ã O entregar
4	Quero 4 coxas de frango.	Coxa de frango 4 querer.	Coxas de frango 4 quero.
5	Quero 1 kg de bifes de picanha.	Bifes picanha 1 kg querer	Ola bifes picanha picanha 1 quilograma

Tabela 9 - Exemplos de Traduções Conseguidas pelo Tradutor Híbrido.

Número Frase	Tradução Esperada	BLEU score*	METEOR score*
1	J O A O ~ CALÇAS PASSARFERRO	0.5081 / 1.0000	0.8066 / 1.0000
2	POR FAVOR MAISFINO	0.0000 / 1.0000	0.0000 / 1.0000
3	J O A O ~ PROFESSOR MEU	1.0000 / 0.5475	0.9976 / 0.6726
4	COXAS DE FRANGO 4 QUERO	0.6687 / 1.0000	0.7937 / 0.9960
5	BIFES PICANHA 1 KG QUERER	1.0000 / NaN	0.9960 / 0.5010

Tabela 10 - Traduções esperadas e Métricas de Avaliação

* - Avaliação Tradução Baseada em Regras / Avaliação Tradução melhorada pela Rede Neuronal

De acordo com a informação constante nas Tabelas Tabela 9 e Tabela 10 verifica-se que apesar de por vezes a rede neuronal melhorar as traduções, noutras a tradução baseada em regras é superior. Este comportamento já era esperado, devido à quantidade muito baixa de frases no corpus utilizado.

5.5. Exemplo de Utilização de Tradutor para Glosa

O exemplo aqui apresentado pretende demonstrar o funcionamento da integração deste tradutor para glosa com o Sistema Geral. O pedido de tradução será realizado a partir do Sistema Geral, sendo esperado que este receba um vídeo representativo da tradução para a glosa realizada pelo tradutor desenvolvido neste trabalho. Este exemplo apresenta-se nas Figura 36, Figura 37, Figura 38. A Figura 37 apresenta as traduções que foram obtidas pelo tradutor baseado em regras e pelo tradutor neuronal, antes de serem enviadas para animação. Consegue-se ver o funcionamento da integração da aplicação móvel do sistema geral com o tradutor desenvolvido e apresentado.

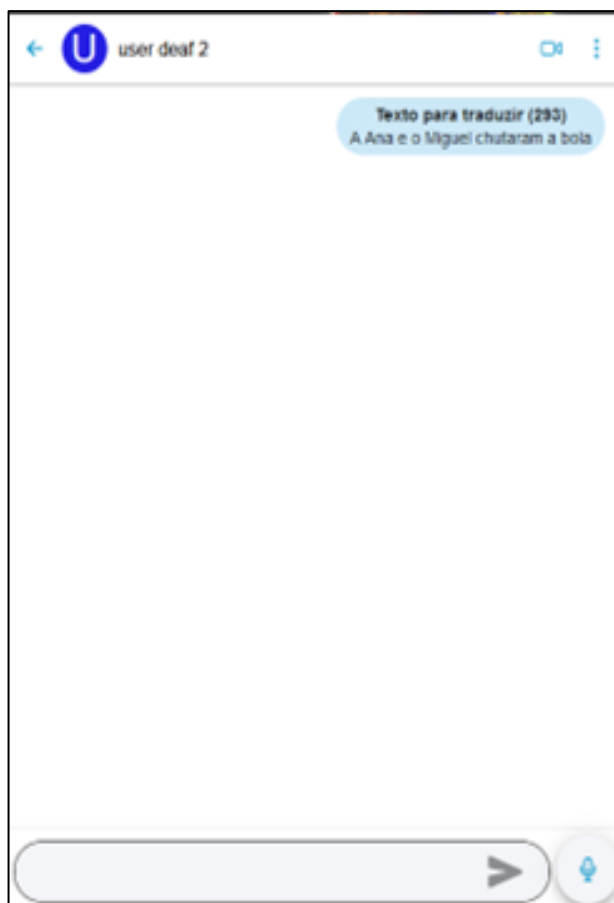


Figura 36 - Envio de Mensagem a partir do Sistema Geral.

```
A traduzir... A Ana e o Miguel chutaram a bola
Baseada em Regras... ['BOLA', 'A', 'N', 'A', 'E', 'M', 'I', 'G', 'U', 'E', 'L', 'CHUTAR']
Neuronal... ['nome1', 'entregar']
10.1.186.85 - - [07/Nov/2022 17:58:44] "GET /traduz?texto=A%20Ana%20e%20o%20Miguel%20chutaram%20a%20bola HTTP/1.1" 200 -
```

Figura 37 - Traduções feitas pelos tradutores.

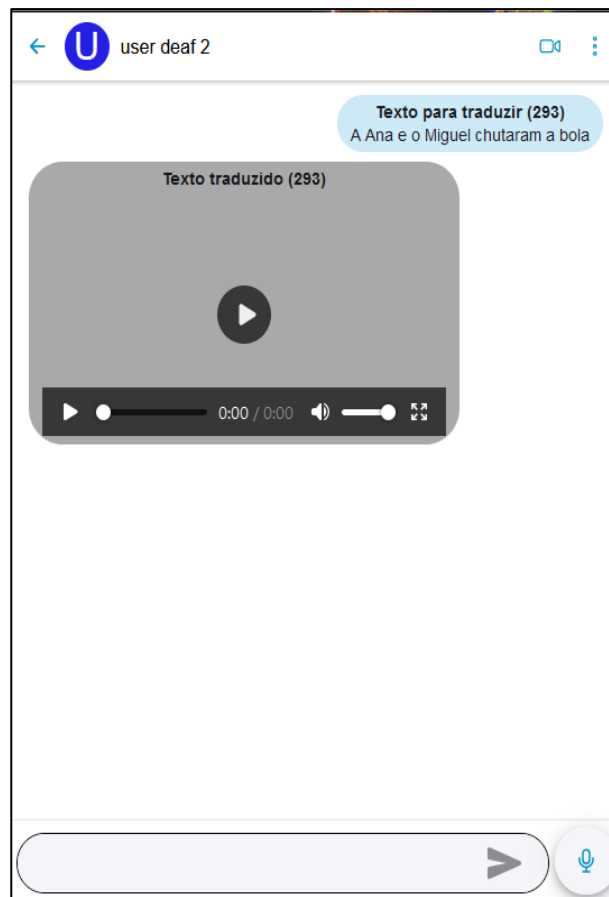


Figura 38 - Receção do Vídeo pelo Sistema Geral.

Para o texto utilizado como exemplo, “A Ana e o Miguel chutaram a bola”, verifica-se que o tradutor baseado em regras traduziu corretamente para “BOLA A-N-A E M-I-G-U-E-L CHUTAR” (Ana e Miguel soletrados, por serem nomes), enquanto que a rede neuronal ajustou essa tradução erradamente para “nome1 entregar” (nome1 posteriormente substituído para o primeiro nome, neste caso, Ana), com este erro a acontecer devido a ser uma frase fora do contexto das frases utilizadas para o treino da rede neuronal.

5.6. Conclusões

Neste capítulo foram apresentadas, de modo detalhado, as implementações desenvolvidas e diversos componentes relacionados com o projeto. Mais importante ainda, foi apresentada a implementação de um tradutor híbrido, que junta um tradutor baseado em regras e um tradutor baseado em redes neurais, para a tradução de português para a glosa da LGP. No final, foram apresentados alguns exemplos de

traduções conseguidas pelo tradutor implementado, bem como um exemplo da utilização integrada com o sistema geral, desde a receção de uma mensagem a partir do mesmo, até ao envio de um vídeo representativo da tradução obtida pelo tradutor implementado.

6. ANÁLISE DE RESULTADOS

Neste capítulo será realizada a análise aos resultados obtidos na utilização do tradutor construído para a realização de traduções. Para a realização desta análise foram solicitadas e utilizadas traduções para glosa, a um formador da LGP e a uma intérprete. Estas traduções permitirão facilmente verificar o quão boas as traduções dadas pelo tradutor desenvolvido são no seu estado atual, e determinar possíveis ajustes que possam vir a ser feitos. Além destes testes de qualidade de tradução, realizou-se uma análise a como o sistema se poderá comportar a nível de escalabilidade em termos de performance. Apresenta-se ainda o estado atual (bastante inicial) do corpus que está a ser desenvolvido conjuntamente com a intérprete e pelo formador de LGP.

6.1. Qualidade de Tradução

Esta análise realizou-se através da comparação de traduções conseguidas pelo tradutor desenvolvido com traduções realizadas por uma intérprete e por um formador utilizador nativo da LGP. Estas comparações realizaram-se pelo envio do texto original e verificação da glosa devolvida contra a glosa construída pelos utilizadores da LGP. Para tal, utilizou-se diretamente o serviço HTTP criado para o acesso ao tradutor híbrido, registando-se todos os resultados para comparação e avaliação posterior. As frases enviadas são frases para as quais se possui a tradução para glosa validada por utilizadores da LGP, e algumas frases ligeiramente modificadas (por exemplo, modificação de nomes, números) sem afetar a estrutura da frase, mantendo a validade da glosa adaptada, mas não verificada pelos utilizadores da LGP. As Tabela 11 e Tabela 12 apresentam alguns exemplos de traduções conseguidas pelo tradutor, mostrando a tradução conseguida pela parte baseada em regras numa coluna, e depois a tradução melhorada pela rede neuronal noutra coluna.

Número de Frase	Frase Original	Frase Esperada
1	A Joana quer 4 coxas de frango	COXA FRANGO 4 J-O-A-N-A QUERER
2	Corte um pouco mais finos por favor	POR FAVOR MAISFINO
3	A Ana e o Miguel chutaram a bola vermelha	BOLA VERMELHA A-N-A / M-I-G-U-E-L CHUTAR
4	Este Opel é especial	OPEL ESTE ESPECIAL
5	Os brócolos salvam vidas	BROCULO VIDA SALVAR
6	O João é jogador de futebol e a Clara é professora	J-O-A-O JOGADOR FUTEBOL / C-L-A-R-A MULHER PROFESSOR

Tabela 11 - Exemplos de Traduções

Número de Frase	Tradução da Parte Baseada em Regras	Tradução Melhorada pela Rede Neuronal
1	COXA FRANGO 4 J-O-A-N-A QUERER	ELA BIFES PICANHA 4 EUQUERER
2	CORTE POUCO MAIS FINO FAVOR	POR FAVOR MAISFINO
3	BOLA VERMELHA A-N-A E M-I-G-U-E-L CHUTAR	M-I-G-U-E-L ENTREGAR
4	OPEL ESTE ESPECIAL	O-P-E-L ENTREGAR
5	VIDA BROCOLOS SALVAR	DE COXAS COXAS COXAS
6	J-O-A-O~ JOGADOR FUTEBOL E C-L-A-R-A MULHER PROFESSOR	PREFERIR BIFES CALCAS MEU

Tabela 12 - Exemplos de Traduções conseguidas pelo tradutor híbrido

Em termos de resultados BLEU e METEOR sobre todas as traduções conseguidas pelo tradutor e as manualmente traduzidas, conseguiram-se os seguintes valores:

	BLEU	METEOR
Tradução Baseada em Regras	0.722405	0.795578
Tradução Melhorada pela Rede neuronal	0.698358	0.752246

Tabela 13 - Resultados BLEU e METEOR para as traduções do Tradutor Construído

Com a análise das Tabela 11, Tabela 12 e Tabela 13, consegue-se verificar que atualmente, para a maioria dos casos, a parte baseada em regras do tradutor híbrido alcança traduções bastante superiores que as melhoradas pela rede neuronal, no entanto, nos casos em que a rede neuronal melhora a tradução, consegue melhorá-la para uma tradução praticamente perfeita, como é o caso da frase 2. Esta discrepância

entre as duas partes do tradutor híbrido deve-se ao pequeno tamanho atual do corpus com o qual a rede neuronal foi treinada (atualmente aproximadamente apenas 2200), sendo que ao longo do tempo, à medida que o corpus for crescendo, os resultados alcançados pela rede neuronal também se tornarão bastante superiores. É esperado que este corpus continue em crescimento. Verifica-se também que o METEOR alcançou melhores resultados, visto ser uma métrica que não é negativamente afetada por frases curtas, ao contrário do que acontece com o BLEU por não conseguir encontrar padrões de tamanho 4, além da METEOR conseguir trabalhar corretamente com caracteres especiais e acentuados.

6.2. Escalabilidade

Para a análise da possível escalabilidade da utilização deste tradutor e como se poderá esperar que a performance do servidor se mantenha, optou-se por se realizar um teste de stress, conseguido pelo envio de vários pedidos de tradução em simultâneo, e verificar o quão negativamente a performance e velocidade do tradutor é afetada. A Tabela 14 mostra o número de pedidos simultâneos e o tempo para finalização de todos os pedidos, e a Figura 39 mostra esses dados de forma gráfica.

Número de pedidos em simultâneo	Tempo em segundos
10	2,490648
20	2,971813
30	3,238391
40	3,65137
50	4,93963
100	6,168377
200	10,197506
500	22,404799
1000	42,767387
5000	141,069924

Tabela 14 - Tempo de resposta em segundos a vários pedidos simultâneos

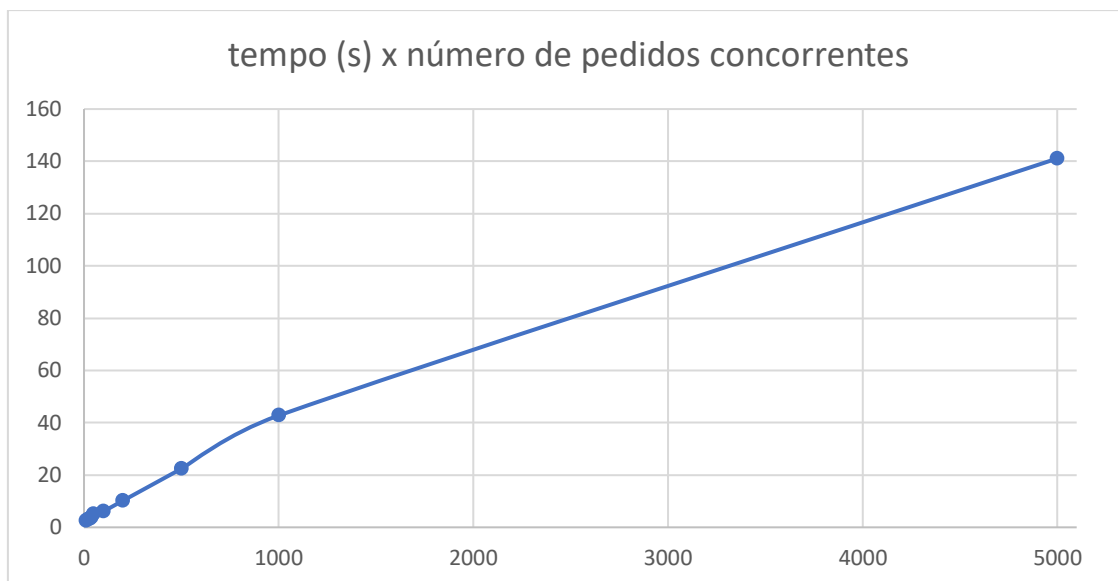


Figura 39 – Tempo em segundos para X pedidos simultâneos

Consegue-se verificar que o tempo para resposta a todos os pedidos em simultâneo relaciona-se com a quantidade de pedidos de forma linear, com a equação estimada de *tempo para resposta a todos pedidos* = $0.0277 * num_pedidos + 4.7076$, um aumento de cerca de 0.02 segundos por cada pedido extra. O valor fixo para quantidades mais pequenas de pedidos em simultâneo também é inferior, já que para apenas 10 pedidos em simultâneo, o tempo total é de cerca de 2.49 segundos, menor que os fixos 4.70 segundos da equação estimada. Se a performance, na utilização real, deste tradutor começar a reduzir-se com um aumento muito grande de utilizadores, facilmente poder-se-á escalar horizontalmente o sistema, incluindo mais servidores disponíveis para este serviço. Consequentemente, com vários servidores disponíveis para a mesma coisa, seria também necessária a existência de um balanceador de carga (*load balancer*) para que os pedidos de tradução fossem distribuídos ao longo dos servidores e não irem sempre parar ao mesmo. No entanto, com os resultados apresentados anteriormente neste tópico, não se espera que isso venha ser necessário, pelo menos num futuro próximo.

6.3. Corpus de Traduções de Português para Glosa da LGP

O corpus que foi e continuará a ser desenvolvido para o treino da rede neuronal do tradutor híbrido é também um resultado da realização deste trabalho, pois não existe nada parecido que esteja disponível, ou que sequer exista. Visto isto, foi necessária a

construção de raiz de um corpus de traduções entre português e glosa da LGP. Este corpus é composto por frases portuguesas de outros corpora disponíveis gratuitamente, e ainda da extração de títulos e resumos de notícias de jornais e principais blogs portugueses (extração realizada pelo script apresentado no apêndice 9.3.1. Script de Obtenção de Títulos e Corpos de Notícias). Estas frases são depois traduzidas para a glosa manualmente por uma intérprete e por um formador da LGP e essa glosa é colocada na coluna adjacente à frase original. As glosas aqui colocadas seguem as regras apresentadas no tópico 2.1.2. (Glosa) e 2.1.3. (Características Gramaticais e Estruturais da LGP). A Tabela 15 apresenta um excerto de traduções que compõem o corpus que foi e está a ser continuamente desenvolvido.

Frase Original	Glosa
Os brócolos salvam vidas. Médico explica por que é que toda a gente deve comer.	BROCULO + VIDA + SALVAR / MEDICO + PESSOAS + TODO + BROCULO + COMER + DEVER
Apresentadora Sara Sousa Pinto deixou de aparecer na TVI e só explicações para o que aconteceu.	APRESENTADORA + S-A-R-A / S-O-U-S-A / P-I-N-T-O + TVI + DESAPARECER // ANTES + ACONTECER + EXPLICAR
A novidade foi revelada nas redes sociais do reality show.	NOVIDADE + CONTAR + REDES + SOCIAIS + PROGRAMA + SEU
Entre os dias 7 de novembro e 2 de dezembro, o Grupo Ageas Portugal, a Fundação Ageas, a Médis e a Europacoln Portugal promovem campanha de sensibilização para a importância dos rastreios preventivos.	DATA + 7 + NOVEMBRO + 2 + DEZEMBRO + ENTRE + GRUPO.AGEAS.PORTUGAL + FUNDACAO.AGEAS + MEDIS + EUROPACOLON.PORTUGAL + SENSIBILIZAÇÃO + PROMOVER + TEMA + ANALISAR + DOENÇA + EVITAR
Este Opel é especial. Já lá vão 75 milhões.	OPEL + ESTE + ESPECIAL // VALOR + 75 + MM

Tabela 15 - Exemplo de Traduções no Corpus

Como todas as traduções estão a ser feitas por utilizadores da LGP, um deles como língua materna, garante-se que todas as traduções estão corretas. Até ao momento, este corpus conta com +- 2200 frases traduzidas para a glosa, ainda bastante pequeno, mas que continuamente continuará a crescer e espera-se que chegue a milhares de traduções entre estas línguas. Neste momento, apenas duas pessoas estão a trabalhar na adição de novas traduções consoante a sua disponibilidade, mas esta tarefa poderá (e deverá ser) acelerada pela inclusão de mais pessoas a contribuírem para este corpus.

6.4. Análise Comparativa

Esta análise baseia-se na comparação entre os resultados aqui apresentados sobre o Glossifica e os resultados apurados no estudo de outros tradutores existentes para a tradução entre português e glosa da LGP. A apresentação dos trabalhos com os quais o Glossifica será comparado já foi previamente feita na secção 3. Estado da Arte. Esta comparação será sobre vantagens e desvantagens de cada uma das três ferramentas e sobre as funcionalidades que cada uma oferece, já que não é possível a realização de uma comparação direta das mesmas com dados, devido à impossibilidade de acesso às mesmas. Esta comparação foca-se apenas em tradução de português para LGP, e não o inverso.

Funcionalidade	Glossifica	VirtualSign [51]	PE2LGP [44] [11]
Tratamento de Texto antes da Tradução	X		X
Implementação de Regras Estruturais e Gramaticais para Tradução	X		X
Utilização de Rede Neuronal para Tradução	X		
Tradutor Híbrido (Regras + Rede Neuronal)	X		
Disponibilização de uma API para comunicação com outros sistemas	X	X	
Avatar incluído		X	X

Tabela 16 - Funcionalidades do Glossifica e outros tradutores

Ferramenta	Vantagens	Desvantagens
Glossifica	Implementa várias regras estruturais e gramaticais, para a realização da tradução. Mais regras implementadas que os outros comparados;	As frases introduzidas para tradução devem estar estruturalmente bem escritas em português, sem erros, sobre pena de serem mal avaliadas morfológicamente e consequentemente mal reordenadas

	Permite a tradução de frases simples como de frases complexas, desde que bem escritas; Implementa um tradutor híbrido, juntando a tradução baseada em regras a uma tradução neuronal; É uma ferramenta isolada, isto é, facilmente acessível por qualquer outra que pretenda obter uma tradução para glosa, via protocolo HTTP; Permite a receção de áudio para transcrição e tradução para glosa.	de SVO para OSV (principalmente em frases complexas). Corpus de treino da rede neuronal pequeno;
VirtualSign [51]	Solução completa. Recebe texto e é representado diretamente por um avatar; Já possui um elevado número de gestos animados, devido a ser um projeto com alguns anos;	Tradução para “português gestual”, isto é, não é aplicado qualquer ajuste gramatical nem estrutural à frase que está a ser traduzida para LGP, sendo representada palavra-a-palavra; Não existe separação de palavra e símbolos. Isto é, por exemplo, “OLA” e “OLA.” é tratado de forma diferente, com a segunda a ser representada como “O-L-A-PONTO”;
PE2LGP [11]	Solução completa. Recebe texto e é representado diretamente por um avatar; Realiza um tratamento ao texto a traduzir através de uma tradução baseada em regras;	Menos regras implementadas quando comparado ao Glossifica; Não aparenta ser modular, isto é, não se conseguir utilizar apenas a parte de tradução para glosa noutro trabalho; Não trata dos nomes femininos (não adiciona o gesto MULHER quando necessário); Desenhado para frases simples;

Tabela 17 - Comparação Vantagens e Desvantagens Glossifica vs Outros Tradutores.

Com as funcionalidades indicadas na Tabela 16 e os principais pontos indicados na Tabela 17, verifica-se que o Glossifica apresenta melhorias na tradução de português para a glosa da LGP. Além de suportar frases complexas, ao contrário do tradutor que realiza algum tratamento à frase original, implementa mais algumas regras. Além disso, apresenta uma abordagem híbrida que utiliza redes neurais, podendo-se tirar partido das mesmas para inferir regras não implementadas e melhorar as traduções dadas pela tradução apenas baseada em regras (algumas já poderão estar num estado ótimo apenas com a aplicação de regras). No entanto, a rede neuronal no momento de escrita deste trabalho, foi treinada com um corpus pequeno, que afetará negativamente as traduções, mas que, com o aumento do corpus, conseguir-se-á também um aumento na qualidade das traduções melhoradas pela rede neuronal do tradutor híbrido. Por fim, o Glossifica é uma ferramenta que, através da sua interface HTTP, pode ser incorporada em qualquer projeto que necessite de traduções de português para a glosa da LGP.

6.5. Conclusões

Neste capítulo foram apresentados e analisados resultados relativos à performance do Glossifica, tanto a nível das traduções conseguidas por este, bem como da escalabilidade e o quanto a performance seria afetada com o aumento elevado de utilizadores. Começou-se por analisar os resultados relativos à qualidade das traduções, avaliando-se o sistema comparando-o a traduções feitas manualmente por utilizadores da Língua Gestual Portuguesa. Esta análise mostra já resultados bastante bons, com resultados das métricas BLEU e METEOR 0.722405 e 0.795578 respetivamente para uma tradução baseada nas regras implementadas, por outro lado, também se verifica que a utilização de uma rede neuronal para melhorar traduções é promissor, mas depende em muito da qualidade e quantidade de dados com os quais a rede neuronal é treinada, tendo para já, com apenas 2200 frases de treino, alcançado um BLEU de 0.698358 e METEOR de 0.752246. Atualmente, possui-se dados de qualidade, mas em pequena quantidade, afetando negativamente as esperadas melhorias pela utilização da rede neuronal no tradutor híbrido, no entanto, em frases de contexto igual ou semelhante aos dados com os quais a rede neuronal foi treinada, verifica-se que de fato esta consegue melhorar as primeiras traduções baseadas em regras e torná-las quase ou

mesmo até iguais ao esperado. Já em termos de performance com o crescimento do número de utilizadores, testando-se no pior caso possível, que é a realização de pedidos por todos os utilizadores em simultâneo, verifica-se que o tradutor segue uma relação entre número de pedidos e quantidade em segundos direta, com o número de pedidos em simultâneo a resultar no aumento do tempo de resposta a todos esses pedidos. No entanto, a estimativa de acordo com os dados apresentados no tópico 6.2. Escalabilidade apresenta um aumento de cerca de apenas 0.02 segundos por pedido extra. No momento em que se verifique que o sistema começa a tornar-se lento, isto é, a demorar a responder a todos os pedidos de tradução, este facilmente poderá ser escalado horizontalmente para outros servidores, já que se baseia na disponibilização de um serviço HTTP. Outro resultado do desenvolvimento deste trabalho é um corpus de traduções entre português e glosa da Língua Gestual Portuguesa. Este corpus ainda é pequeno, mas é esperado que cresça bastante no futuro. Por fim, foi feita uma comparação do Glossifica com outros sistemas de tradução de português para LGP, apresentando-se principais pontos fortes e fracos de cada um, finalizando-se no contributo conseguido pelo Glossifica.

7. CONCLUSÕES E TRABALHO FUTURO

7.1. Conclusões

Atualmente, o tema da inclusão social é um tema que tem sido bastante abordado e trabalhado, por vezes focando apenas as tecnologias da informação a utilizar, as diversas aplicações informáticas a serem desenvolvidas para várias situações, como por exemplo, a mobilidade e as deficiências do ser humano. Há uma área de aplicação que, apesar de apresentar algumas melhorias ao longo dos anos, continua a ficar um pouco para trás nesta evolução do percurso de desenvolvimento de trabalhos para a inclusão social, que é a área do desenvolvimento e aplicação de sistemas para a comunicação de surdos com a comunidade ouvinte. É do senso comum desta comunidade, em particular a portuguesa, que um surdo possui a capacidade de ler texto em português. No entanto, isso não é verdade, já que a língua principal de todos os surdos (ou quase todos) é a Língua Gestual Portuguesa.

O trabalho aqui apresentado visa propor uma arquitetura para um tradutor completo entre português e Língua Gestual Portuguesa, e, principalmente, estudar, desenvolver e apresentar um tradutor automático para a tradução entre português para a glosa da Língua Gestual Portuguesa tendo em consideração as características e a gramática desta última, algo que apenas alguns dos poucos trabalhos realizados nesta área consideraram, e os que consideraram, apenas para algumas regras e frases simples, contrastando com o apresentado neste documento, que procurou também implementar regras para a tradução de frases complexas. Além da implementação de regras para a tradução, este trabalho contém ainda uma rede neuronal para melhorar essas traduções, resultando então num tradutor híbrido para a tradução de português para a Língua Gestual Portuguesa, diferindo também, neste aspeto, quando comparado com trabalhos realizados anteriormente, entre estas línguas (sendo uma abordagem já utilizada em alguns tradutores para a tradução de outras línguas naturais para a língua gestual equivalente).

Os resultados alcançados mostram que a utilização deste sistema é viável e que se conseguem alcançar boas traduções, apesar do pequeno corpus disponível até ao momento, alcançando-se resultados BLEU e METEOR de 0.722405 e 0.795578 respetivamente, para uma tradução baseada apenas em regras, e 0.698358 e 0.752246 para a tradução final após a passagem pela rede neuronal. Os resultados após a passagem pela rede neuronal são inferiores devido ao corpus de treino ser pequeno, no entanto, verifica-se que quando a rede consegue melhorar, melhora as traduções deixando-as totalmente corretas ou quase totalmente corretas, algo que não é perceptível apenas ao verificar-se o resultado BLEU ou METEOR. Além dos resultados alcançados, que envolvem o cálculo de métricas, verifica-se um feedback positivo da intérprete e do formador de LGP, apesar das constantes melhorias que será necessário continuar a realizar, para reduzir ainda mais as traduções menos bem conseguidas. No entanto, estas melhorias aparecerão, maioritariamente, com o crescimento do corpus de traduções associado ao treino da rede neuronal.

Considerando tudo isto consegue-se afirmar que o principal objetivo deste trabalho foi alcançado, e espera-se agora que o que foi aqui desenvolvido seja utilizado para futuros desenvolvimentos e criação de aplicações que permitam a boa comunicação entre ouvintes e surdos, com a mesma facilidade que aplicações do género permitem entre outras línguas não gestuais. Todas as etapas da metodologia adotada e descrita em 1.3. Metodologia Utilizada foram cumpridas ao longo das várias secções apresentadas neste trabalho, com a atividade de *Identificação do Problema e Motivação* descritas nos capítulos 1.1. Contextualização e 3. Estado da Arte, atividade da *Definição dos Objetivos para a Solução* apresentada na secção 1.2. Objetivos, atividade de *Desenho e Desenvolvimento* apresentada na secção 4. Arquitetura do Sistema, *Demonstração* apresentada na secção 5.4.4. Exemplo de Utilização do Glossifica, 5.5. Exemplo de Utilização de Tradutor para Glosa e 6. Análise de Resultados, atividade de *Avaliação* na secção 6. Análise de Resultados e, por fim, a atividade de *Comunicação e Difusão* realizada através da publicação de um artigo científico numa conferência internacional, apresentado no apêndice 9.1. Artigos Publicados em Conferências Internacionais.

7.2. Trabalho Futuro

Sendo este um trabalho que explora uma nova abordagem para a tradução automática entre português e glosa da Língua Gestual Portuguesa, e considerando ainda o atual estado relativo a bibliografia existente sobre a Língua Gestual Portuguesa e respetiva gramática, e ainda a dificuldade no acesso à pouca bibliografia sobre o tema, consegue-se facilmente apontar como melhoria afinar e aumentar o leque de regras gramaticais que foram implementadas neste trabalho, na parte do tradutor que as utiliza. Uma outra tarefa que, apesar de já estar a acontecer, terá que continuar a ser executada, é a tarefa do aumento do corpus de traduções entre português e glosa da Língua Gestual Portuguesa, visto não existir nada do género publicamente disponível, aumento este que facilmente resultaria na melhoria da rede neuronal utilizada neste tradutor, e, quem sabe, tornar esse corpus suficientemente grande para que fosse possível realizar experimentações com tradutores totalmente neuronais entre português e glosa da Língua Gestual Portuguesa. Para o tradutor de português para a glosa da Língua Gestual Portuguesa, todos os possíveis trabalhos futuros aparentam apontar apenas para a melhoria de tradução, a não ser que seja encontrada alguma limitação que envolva a reestruturação do tradutor. Algo que também poderia, em certos casos, melhorar as traduções, seria a implementação de um validador e corretor de língua portuguesa, que, seria aplicado antes do envio da frase para tradução, de modo a garantir que essas frases se encontram bem estruturadas, prevenindo possíveis problemas causados pela má construção de frases. No que toca à arquitetura proposta para um tradutor completo de português para Língua Gestual Portuguesa, certamente algumas alterações deverão ocorrer para a viabilidade dum projeto deste tipo em grande escala. Um exemplo de melhoria, seria passar o avatar completamente para o cliente e removê-lo do servidor, já que esta mudança na arquitetura resultaria na colocação de grande parte do processamento no cliente e não no servidor, facilitando a escalabilidade deste último. Por fim, haverá a necessidade de execução de testes de validação em grande escala, isto é, com várias pessoas que utilizem a LGP como a sua língua materna ou não, em vez de apenas se utilizarem as traduções e opiniões de somente duas pessoas, e ainda, o teste de comunicação para o principal cenário de

utilização de um tradutor deste género, entre utilizadores de LGP e pessoas que nunca tiveram contacto com esta língua.

8. REFERÊNCIAS

- [1] Associação Portuguesa de Surdos, “Língua Gestual Portuguesa (LGP) - APSurdos,” [Online]. Available: https://apsurdos.org.pt/?page_id=3725.
- [2] I. - I. N. d. Estatística, “Censos 2021,” 2011.
- [3] A. Hevner, A. R. S. March, S. T. Park, J. Park, Ram e Sudha, “Design Science in Information Systems Research,” *Management Information Systems Quarterly*, pp. 75, DOI: 10.2307/25148625, 2004.
- [4] K. Peffers, T. Tuunanen, M. Rothenberger e S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, pp. 45-77, DOI: 10.2753/MIS0742-1222240302, 2007.
- [5] H. Cooper, B. Holt e R. Bowden, “Sign Language Recognition,” *Visual Analysis of Humans*, pp. 539-562, DOI: 10.1007/978-0-85729-997-0_27, 2011.
- [6] I. Mesquita e S. Silva, Guia Prático de Língua Gestual Portuguesa - Ouvir o Silêncio, Nova Educação, LDA, 2007.
- [7] M. A. Amaral, A. Coutinho e M. R. D. Martins, Para Uma Gramática da Língua Gestual Portuguesa, Editorial Caminho, ISBN 9789722109819, 1994.
- [8] I. Almeida, L. Coheur e S. Candeias, “From European Portuguese to Portuguese Sign Language,” pp. 140-143, DOI: 10.18653/v1/W15-5124, 2015.
- [9] R. A. F. Rodrigues, “Compreensão da Língua Gestual Portuguesa,” 2017.
- [10] I. R. Almeida, “Exploring Challenges in Avatar-based Translation from European Portuguese to Portuguese Sign Language,” Lisboa, 2014.
- [11] M. Gonçalves, L. Coheur, H. Nicolau e A. Mineiro, “PE2LGP: tradutor de português europeu para língua gestual portuguesa em glosas,” 2017.
- [12] C. M. Bishop, “Neural networks and their applications,” *Review of Scientific Instruments*, pp. 1803-1832, DOI: 10.1063/1.1144830, 1994.

- [13] B. Mahesh, "Machine Learning Algorithms -A Review," *International Journal of Science and Research (IJSR)*, p. DOI: 10.21275/ART20203995, 2019.
- [14] A. Jain, J. Mao e K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, pp. 31-44, DOI: 10.1109/2.485891, 1996.
- [15] R. Quiza e J. Davim, "Computational Methods and Optimization," pp. DOI: 10.1007/978-1-84996-450-0.
- [16] R. J. Erb, "Introduction to Backpropagation Neural Network Computation," *Pharmaceutical Research*, p. DOI: 10.1023/A:1018966222807, 1993.
- [17] S.-i. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, pp. 185-196, DOI: 10.1016/0925-2312(93)90006-O, 1993.
- [18] A. Biswal, "Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM and More," *simplilearn.com*, 2022. [Online]. Available: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>.
- [19] IBM, "What are Recurrent Neural Networks?," [Online]. Available: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>.
- [20] Y. Bengio, P. Simard e P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, pp. 157-166, DOI: 10.1109/72.279181, 1994.
- [21] M. Morchid, "Internal Memory Gate for Recurrent Neural Networks with Application to Spoken Language Understanding," *Interspeech 2017*, pp. 3316-3319, DOI: 10.21437/Interspeech.2017-357, 2017.
- [22] S. Hochreiter e J. Schmidhuber, "Long Short-term Memory," *Neural computation*, pp. 1735-1780, DOI: 10.1162/neco.1997.9.8.1735, 1997.
- [23] T. Toharudin, R. Pontoh, R. Caraka, S. Zahroh, Y. Lee e R.-C. Chen, "Employing Long Short-Term Memory and Facebook Prophet Model in Air Temperature Forecasting," *Communication in Statistics- Simulation and Computation*, p. DOI: 10.1080/03610918.2020.1854302.

- [24] K. Cho, B. van Merriënboer, D. Bahdanau e Y. Bengio, “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,” p. DOI: 10.48550/ARXIV.1409.1259, 2014.
- [25] N. Bensalah, A. Habib, A. Abdellah e A. Farouk, “LSTM vs. GRU for Arabic Machine Translation,” pp. 156-165, DOI: 10.1007/978-3-030-73689-7_16, 2021.
- [26] J. Chung, C. Gulcehre, K. Cho e Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” p. DOI: 10.48550/ARXIV.1412.3555, 2014.
- [27] K. Cho, B. a. G. C. van Merriënboer, D. Bahdanau, F. Bougares, H. Schwenk e Y. Bengio, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” p. DOI: 10.48550/ARXIV.1406.1078, 2014.
- [28] I. Sutskever, O. Vinyals e Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” p. DOI: 10.48550/ARXIV.1409.3215, 2014.
- [29] D. Bahdanau, K. Cho e Y. Bengio, “Neural Machine Translation by Jointly Learning to Align and Translate,” p. DOI: 10.48550/ARXIV.1409.0473, 2014.
- [30] D. Ibanez, “Encoder-Decoder Models for Natural Language Processing,” baeldung.com, [Online]. Available: <https://www.baeldung.com/cs/nlp-encoder-decoder-models>.
- [31] E. Golikov, V. Zhukov e M. KretoV, “Using stochastic computation graphs formalism for optimization of sequence-to-sequence model,” *OPTML 2017: 10th NIPS Workshop on Optimization for Machine Learning (NIPS 2017)*, p. DOI: 10.48550/ARXIV.1711.07724.
- [32] K. Papineni, S. Roukos, T. Ward e W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, DOI: 10.3115/1073083.1073135, 2002.
- [33] A. Lavie, “Evaluating the Output of Machine Translation Systems,” researchgate.net, [Online]. Available: https://www.researchgate.net/publication/266096165_Evaluating_the_Output_of_Machine_Translation_Systems.

- [34] S. Banerjee e A. Lavie, "METEOR: An Automatic Metric for {MT} Evaluation with Improved Correlation with Human Judgments," *Proceedings of the {ACL} Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pp. 65-72, DOI: 10.5555/1626355.1626389, 2005.
- [35] L. S. Hadla, T. M. Hailat e M. N. Al-Kabi, "Comparative Study Between METEOR and BLEU Methods of MT: Arabic into English Translation as a Case Study," *International Journal of Advanced Computer Science and Applications*, p. DOI: 10.14569/IJACSA.2015.061128, 2015.
- [36] D. H. Maulud, S. M.Zeebaree, K. Jacksi, M. A.M.Sadeeq e K. H. Sharif, "A State of Art for Semantic Analysis of Natural Language Processing," *Qubahan Academic Report*, pp. DOI: 10.48161/issn.2709-8206.
- [37] B. J. Dorr, P. W. Jordan e J. W. Benoit, "A Survey of Current Paradigms in Machine Translation," *Advances in Computers*, pp. 1-68, DOI: 10.1016/S0065-2458(08)60282-X, 1999.
- [38] J.-X. Huang, K.-S. Lee e Y.-K. Kim, "Hybrid Translation with Classification: Revisiting Rule-Based and Neural Machine Translation," *Electronics*, p. DOI: 10.3390/electronics9020201, 2020.
- [39] M. D. Okpor, "Machine Translation Approaches: Issues and Challenges," *IJCSI International Journal of Computer Science Issues*, 2014.
- [40] S. Alrikabi e V. Hafner, "A Humanoid Robot as a Translator from Text to Sign Language," pp. 375-379, 2011.
- [41] M. Grif e Y. Manueva, "Semantic Analyses of Text to Translate to Russian Sign Language," *11th International Forum on Strategic Technology*, pp. 286-289, DOI: 10.1109/IFOST.2016.7884107, 2016.
- [42] K. Saija, S. Sangeetha e V. Shah, "WordNet Based Sign Language MachineTranslation: from English Voice to ISL Gloss," *2019 IEEE 16th India Council International Conference (INDICON)*, pp. 1-4, DOI: 10.1109/INDICON47234.2019.9029074, 2019.
- [43] T. Dasgupta e A. Basu, "Prototype machine translation system from text-to-Indian sign language," *IUI '08: Proceedings of the 13th international conference*

on Intelligent user interfaces, pp. 313 - 316, DOI: 10.1145/1378773.1378818, 2008.

- [44] R. E. R. d. Santos, "PE2LGP: Do Texto`a L´ingua Gestual (e vice-versa)," 2016.
- [45] P. Cabral, H. Nicolau, L. Coheur e R. Santos, "PE2LGP Animator: A Tool To Animate A Portuguese Sign Language Avatar," *Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives*, pp. 33-38, 2020.
- [46] T. Vichyaloetsiri e P. Wuttidittachotti, "Web Service framework to translate text into sign language," *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pp. 180 - 184, DOI: 10.1109/CITS.2017.8035336, 2017.
- [47] N. Aouiti, "Towards an automatic translation from Arabic text to sign language," *2013 Fourth International Conference on Information and Communication Technology and Accessibility (ICTA)*, pp. 1-4, DOI:10.1109/ICTA.2013.6815320, 2013.
- [48] H. S. Al-Khalifa, "On the implementation of text to Arabic SignLanguage converter on mobile phones," pp. DOI:10.3233/TAD-2011-0312, 1 05 2011.
- [49] B. Sarkar, K. Datta, C. D. Datta, D. Sarkar, S. J. Dutta, I. D. Roy, A. Paul, J. U. Molla e A. Paul, "A Translator for Bangla Text to Sign Language," *2009 Annual IEEE India Conference*, pp. 1-4, DOI: 10.1109/INDCON.2009.5409449, 2009.
- [50] D. Kouremenos, S.-E. Fotinea, E. Efthimiou e K. Ntalianis, "A prototype Greek text to Greek Sign Language conversion system," *Behaviour & Information Technology*, pp. 467-481, DOI: 10.1080/01449290903420192, 2010.
- [51] P. Escudeiro, N. Escudeiro, R. Reis, J. Lopes, M. Norberto, A. B. Baltasar, M. Barbosa e J. Bidarra, "Virtual Sign – A Real Time Bidirectional Translator of Portuguese Sign Language," *Procedia Computer Science*, pp. 252-262, DOI: 10.1016/j.procs.2015.09.269, 2015.

- [52] M. Brour e A. Benabbou, “ATLASLang NMT: Arabic text language into Arabic sign language neural machine translation,” *Journal of King Saud University - Computer and Information Sciences*, p. DOI: 10.1016/j.jksuci.2019.07.006, 2019.
- [53] J. Brownlee, “Why One-Hot Encode Data in Machine Learning?,” 28 7 2017. [Online]. Available: <https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/>.
- [54] C. C. M. d. Oliveira, T. G. d. Rêgo, M. A. C. B. Lima e T. M. U. d. Araújo, “Analysis of rule-based machine translation and neural machine translation approaches for translating portuguese to LIBRAS,” *WebMedia '19: Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, pp. 117 - 124, DOI: 10.1145/3323503.3360305, 2019.
- [55] S. Stol, N. C. Camgoz, S. Hadfield e R. Bowden, “Sign Language Production using NeuralMachine Translation and GenerativeAdversarial Networks,” em *BMVC*, 2018.
- [56] V. Veríssimo, C. Silva, V. Hanael, C. Moraes, R. Costa, T. Maritan, M. Aschoff e T. Gaudêncio, “A study on the use of sequence-to-sequence neural networks for automatic translation of brazilian portuguese to LIBRAS,” *WebMedia '19: Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, pp. 101-108, DOI: 10.1145/3323503.3360292, 2019.
- [57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser e I. Polosukhin, “Attention Is All You Need,” *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, p. DOI: 10.48550/ARXIV.1706.03762, 2017.
- [58] L. Brussel, A. Tezcan e L. Macken, “A Fine-grained Error Analysis of NMT, PBMT and RBMT Output for English- to-Dutch,” *LREC*, 1 2018.
- [59] xmpp.org, “XMPP | The universal messaging standard,” [Online]. Available: <https://xmpp.org/>.
- [60] json.org, “JSON,” [Online]. Available: <https://www.json.org/json-en.html>.
- [61] google.com, “Speech-to-Text: Automatic Speech Recognition,” [Online]. Available: <https://cloud.google.com/speech-to-text/>.

- [62] Meta, “Meta: Build Natural Language Experiences,” Meta, [Online]. Available: <https://wit.ai/>.
- [63] A. Morris, “An information theoretic measure of sequence recognition performance,” *IDIAP-Com 02-03*, p. 10, 2006.
- [64] I. Maglogiannis, L. Iliadis e E. Pimenidis, “A Benchmarking of IBM, Google and Wit Automatic Speech Recognition Systems,” *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference*, pp. 73–82, DOI: 10.1007/978-3-030-49161-1_7, 2020.
- [65] W. Lea e J. Woodard, “New measures of performance for speech recognition systems,” *CASSP '84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 9, pp. 352-355, DOI: 10.1109/ICASSP.1984.1172343, 1984.
- [66] A. Moris, V. Maier e G. Phil, “From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition,” *INTERSPEECH , ISCA*, pp. DOI: 10.21437/Interspeech.2004-668, 2004.
- [67] V. Pratap, X. Qiantong, S. Anuroop, S. Gabriel e C. Ronan, “MLS: A Large-Scale Multilingual Dataset for Speech Research,” *Interspeech 2020*, pp. DOI: 10.21437/interspeech.2020-2826, 2020.
- [68] “spaCy - Industrial-Strength Natural Language Processing in Python,” spaCy, [Online]. Available: <https://spacy.io/>.
- [69] “Keras: the Python deep learning API,” keras, [Online]. Available: <https://keras.io/>.
- [70] “TensorFlow,” Google, [Online]. Available: <https://www.tensorflow.org/>.
- [71] “Theano,” theano, [Online]. Available: <https://theano-pymc.readthedocs.io/en/latest/>.
- [72] “The Microsoft Cognitive Toolkit,” Microsoft, [Online]. Available: <https://learn.microsoft.com/en-us/cognitive-toolkit/>.
- [73] Z. Z. Hlaing, Y. K. Thu, T. Supnithi e P. Netisopakul, “Improving neural machine translation with POS-tag features for low-resource language pairs,” *Heliyon*, p. DOI: 10.1016/j.heliyon.2022.e10375, 2022.

- [74] “POS-Tagging based Neural Machine Translation System for European Languages using Transformers,” *WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS*, 31 5 2021.
- [75] “Postman API Platform,” Postman, [Online]. Available: <https://www.postman.com/>.
- [76] V. Alves, J. Ribeiro, P. Faria e L. Romero, “Neural Machine Translation Approach in Automatic Translations between Portuguese Language and Portuguese Sign Language Glosses,” *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1-7, DOI: 10.23919/CISTI54924.2022.9820212, 2022.
- [77] G. PREETHAM, S. R. BHARAT, P. ALEXANDER e G. ANDI, “POS-Tagging based Neural Machine Translation System for European Languages using Transformers,” *WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS*, p. DOI: 10.37394/23209.2021.18.5, 2021.
- [78] M. F. d. S. Bettencourt, “A ordem de palavras na Língua Gestual Portuguesa: Breve estudo comparativo com o Português e outras línguas gestuais,” Faculdade de Letras da Universidade do Porto, 2015.

9. ANEXOS / APÊNDICES

9.1. Artigos Publicados em Conferências Internacionais

9.1.1. Neural Machine Translation Approach in Automatic Translations between Portuguese Language and Portuguese Sign Language Glosses

Neural Machine Translation Approach in Automatic Translations between Portuguese Language and Portuguese Sign Language Glosses

Vasco Alves, Jorge Ribeiro, Pedro Faria, Luis Romero

ADIT-LAB, Instituto Politécnico de Viana do Castelo, Rua Escola Industrial e Comercial Nuno Álvares, 4900-347
Viana do Castelo, Portugal
[alvesvasco, jribeiro, pfaria, romero]@estg.ipv.pt

Abstract — Communication between deaf and non-deaf people is a difficult task in Portugal. Only a very small portion of the population is able to communicate with the impaired ones. Unfortunately, there aren't a lot of systems which are available to the public to help in this communication. In this paper we present a component to translate from Portuguese Natural Language to Portuguese Sign Language, using Machine Learning approach, which is a sub-component of a bigger system designed to allow communication between deaf and non-deaf via smartphones. Our approach uses a Neural Machine Translation system to provide translations between Portuguese Natural and Portuguese Sign Language Glosses to be used to animate a 3D avatar. Early testing of different machine learning architectures showed promising results of BiLingual Evaluation Understudy (BLEU) evaluation on translations. The component uses a supervised learning model and results have shown that, under normal operating conditions an accuracy in the detection to translate from Portuguese-to-Portuguese sign language glosses, achieving above 80% on testing data using BLEU performance scores.

Keywords - Machine Learning, Portuguese Sign Language, Automatic Translation to Sign Language.

1. INTRODUCTION

The Portuguese Sign Language (LGP - *Língua Gestual Portuguesa*) is one of the three official national languages of Portugal, being the one used by the deaf community. Estimations indicate that there are about 30.000 people in Portugal that are deaf or hard of hearing, according to Portuguese Deaf Association - LGP [1]. Accounting for the surrounding community, that is, teachers, relatives, and other close people, it is estimated that there are about 110.000 LGP users, which, considering the results of Portugal's Census [2], represents only 1% of Portugal's population. Taking a close look at those values, we can easily conclude that a deaf or hard of hearing person will have immense difficulty while trying to communicate with the general population, being that in most cases, that communication will not be possible, since very few people have the knowledge about LGP and capability to use it, and there is no guarantee that LGP users are able to read or interpret written Portuguese correctly. However, almost everyone has the power in their pockets to mitigate those

difficulties. It is already extremely common to use smartphone, or other devices, to translate between multiple natural languages, Portuguese being one of them, so, why not use them to translate between Portuguese natural language and LGP? The answer is simple. Besides only a little few number of works have been presented to the Portuguese context (not Brazilian) [16-18] there isn't a platform publicly accessible from everywhere for the communication between people that are deaf or hard of hearing with the general population.

The work here presented was developed under the Research and Development project named as IVLinG: "Virtual Sign Language Interpreter"¹, whose aim is to design and develop a platform to help the communication between deaf and non-deaf via smartphones, specifically in medical oriented scenarios. Therefore, the project is structured in different subcomponents to provide translations between Portuguese and Portuguese Sign Language Glosses to be used to animate a 3D avatar, using smartphone devices.

The main goals of this work are: (a) to study and explore approaches for automatic Translation from Natural Language to Sign Language, for Portuguese Natural Language; b) to study, test and explore methods, algorithms, techniques, statistic formalisms, applying Artificial Intelligence/Machine Learning approaches. We focus our attention on the problem of translations between Portuguese and Portuguese Sign Language Glosses, by proposing a system using Machine Learning approach.

After a first research review using Rule Based Machine Translation from Natural Language to Sign Language and using Neural Machine Translation, we explore the challenge of translations between Portuguese and Portuguese Sign Language Glosses. By using Machine Learning architectures Encoder-Decoder with LSTM - Long Short-Term Memory and Encoder-Decoder with Bidirectional LSTM for this system we achieve promising results which are described with more detail in next sections.

This paper is organized in five sections: the first is the current introduction, followed by the presentation of related work, in which some research and projects developed are presented.

¹ IVLinG: Virtual Sign Language Interpreter, Project website: https://tech.ipv.pt/projeto.php?id_projeto=184

V. Alves, J. Ribeiro, P. Faria and L. Romero, 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), 2022, pp. 1-7, doi: [10.23919/CISTI54924.2022.9820212](https://doi.org/10.23919/CISTI54924.2022.9820212) [76].

9.2. Experimentação e Comparação de Tradutor Neuronal entre Línguas com e sem a Utilização de Categorias Gramaticais Para a Tradução

Nomes próprios e números são dois tipos de categorias que aparecem no texto, cujos valores podem ser muito variáveis, mas que a troca por outros valores da mesma categoria gramatical não afeta a estrutura ou sentido da frase em questão. Por exemplo, imaginando-se a frase “A Ana comprou 2 gelados.”, a troca do nome “Ana” por “Joana”, ficando “A Joana comprou 2 gelados”, não resultaria em qualquer diferença ao sentido da frase, já que ambas as frases mantêm o mesmo significado de “*Alguém comprou uma quantidade de gelados*”. No entanto, a Rede Neuronal, especialmente com um corpus de treino pequeno, como é o caso da Língua Gestual Portuguesa neste trabalho, poderá erradamente inferir regras com base no valor dessas características, em vez de inferir regras com base na característica que são. Também com a verificação de trabalhos como [73] e [77] os quais indicam a obtenção de melhoria de resultados com a utilização das características gramaticais em vez das palavras em si, com [73] a indicar melhorias substanciais para corpus de treino pequeno, optou-se pela realização de uma comparação de duas redes neuronais do tipo Encoder-Decoder com LSTM, para a tradução de português para inglês, uma utilizando os dados originais do corpus, e outra com conversão de nomes próprios e números para palavras específicas que os representam. Neste teste, todos os nomes próprios foram convertidos para *NOME#*, com # a iniciar em 1, e a incrementar a cada nome encontrado. O mesmo é realizado para números, com estes a serem convertidos para *NUM#*. A utilização do # para identificar nomes e números diferentes permite depois, no momento da utilização da rede treinada para inferir traduções, mapear inicialmente os nomes próprios e números encontrados na frase para as nomenclaturas específicas, e após inferência, voltar a mapear essas nomenclaturas ao nome e número que foi originalmente modificado na posição em questão.

Para a criação das redes neuronais foi utilizado o corpus *Anki*⁵, corpus disponível com várias frases em inglês e traduzidas para português. A descoberta dos nomes

⁵ <http://www.manythings.org/anki/> - Tab-delimited Bilingual Sentence Pairs from Tatoeba Project

próprios e números nas frases para conversão para as nomenclaturas utilizadas foi realizada através do spaCy, biblioteca Python de processamento de linguagem natural.

De modo a evitar casos em que o spaCy identificou mal alguma palavra como nome ou número, ignoram-se todas as combinações de frases que não possuam a mesma quantidade de nomes e números identificados entre as respectivas traduções. O código para este mapeamento está apresentado de seguida.

```
import spacy

nlp_pt = spacy.load("pt_core_news_lg")
nlp_en = spacy.load("en_core_web_lg")

# ficheiro traduções original
fc_original = open('DATASET/por-final.txt', 'r')
Lines = fc_original.readlines()

# ficheiro traduções tratadas
fc_final = open('DATASET/en-pt-tratado.txt', 'w')

for line in Lines:
    txt_convertido_pt = ""
    txt_convertido_en = ""
    qnt_nomes_en = 0
    qnt_numeros_en = 0

    line.replace("\n", "")
    en, pt = line.split("\t")

    # mapear nomes e números das frases em inglês
    for token in nlp_en(en):
        if token.text == "\n" or token.text == "\\n":
            continue
        if token.pos_ == "PROPN":
            qnt_nomes_en += 1
            txt_convertido_en += "NOME" + str(qnt_nomes_en) + " "
            continue
        if token.pos_ == "NUM":
            qnt_numeros_en += 1
            txt_convertido_en += "NUM" + str(qnt_numeros_en) + " "
            continue
        txt_convertido_en += token.text + " "

    qnt_nomes_pt = 0
    qnt_numeros_pt = 0

    # mapear nomes e números das frases em português
    for token in nlp_pt(pt):
        if token.text == "\n" or token.text == "\\n":
            continue
        if token.pos_ == "PROPN":
            qnt_nomes_pt += 1
            txt_convertido_pt += "NOME" + str(qnt_nomes_pt) + " "
            continue
        if token.pos_ == "NUM":
            qnt_numeros_pt += 1
            txt_convertido_pt += "NUM" + str(qnt_numeros_pt) + " "
```

```
        continue
    txt_convertido_pt += token.text + " "

    # ignorar traduções onde não foram identificados as mesmas
    # quantidades de nomes e números
    if qnt_nomes_en != qnt_nomes_pt or qnt_numeros_en !=
qnt_numeros_pt:
        continue

    # adicionar par de tradução ao ficheiro tratado
    fc_final.write(txt_convertido_en + "\t" + txt_convertido_pt +
"\n")

fc_final.close()
```

Excerto de Código 6 - Conversão Corpus - Nomes Próprios e Números para Nomenclatura.

Após a realização deste mapeamento, possuem-se os dois corpora, a utilizar para cada uma das redes neurais a ser treinada, o original sem modificações para a rede que apenas utiliza o texto, o tratado para a rede que utiliza a combinação do texto original com alguns mapeamentos para categorias gramaticais.

O processo de treino destas redes neurais utilizou as seguintes etapas principais, Limpeza/Normalização dos Dados, Preparação dos Dados, Treino e Avaliação. A Limpeza/Normalização dos dados foca-se na tentativa de tornar todos os dados uniformes e consistentes, limpando valores desconhecidos ou inválidos, convertendo o texto para o mesmo formato. A preparação de dados foca-se na divisão dos dados para treino e teste. O Treino de realizar a criação e o treino de um novo modelo, e a etapa da avaliação focam-se na avaliação do modelo criado, utilizando métricas, neste caso a BLEU.

A avaliação realizada utilizando os dados divididos para teste e o treino do corpus inicial resultou nos valores apresentados nos gráficos das Figura 40 e Figura 41, para os dados de treino, e Figura 42 e Figura 43 para os dados de teste.

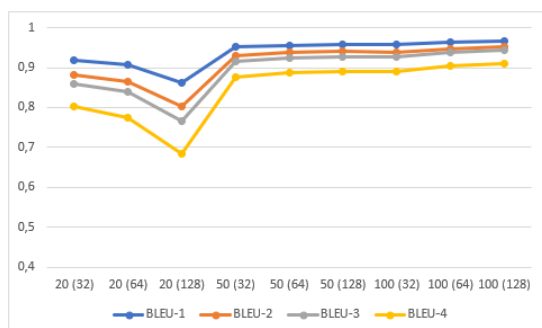


Figura 40 - BLEU dados treino PT-EN sem modificação.

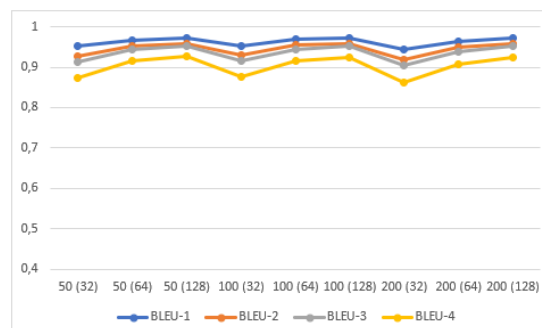


Figura 41 - BLEU dados treino PT-EN com modificação para categorias gramaticais.

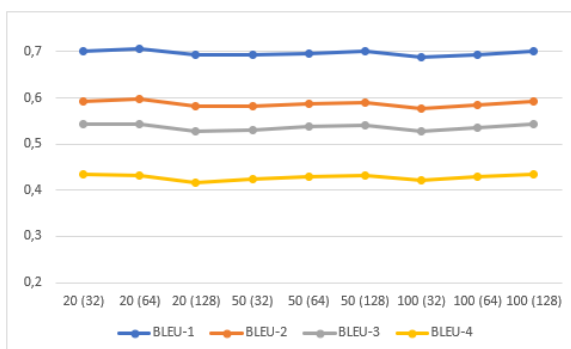


Figura 42 - BLEU dados teste PT-EN sem modificação.

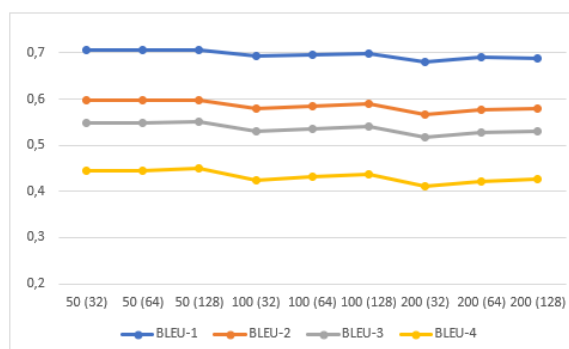


Figura 43 - BLEU dados teste PT-EN com modificação para categorias gramaticais.

Verifica-se que existiram melhorias com a utilização da rede treinada com as frases modificadas, para a substituição dos nomes próprios e números pelas nomenclaturas definidas ao nível dos dados de treino, com os dados de teste a alcançarem avaliações muito equivalentes entre os dois tipos de rede. No entanto, realizando uma avaliação utilizando cerca de 46 mil frases que possuem nomes próprios e números, verifica-se que a rede treinada com a substituição dos nomes próprios e números alcançou resultados superiores, tendo alcançado um resultado BLEU de 0.88870 e METEOR de 0.9161. A rede sem qualquer substituição apenas alcançou resultados BLEU 0.58267 e METEOR 0.48263. A análise de inferências por ambas as redes mostram que a primeira conseguiu melhores resultados para frases com nomes variados, enquanto que a segunda apresentou, numa grande quantidade dessas inferências, frases relativamente bem construídas, mas com os nomes ou os números substituídos por pronomes. A Tabela 18 mostra algumas traduções de ambas as redes para as mesmas frases.

Frase Esperada	Frase Rede Neuronal Treinada com Modificação para Categorias Gramaticais	Frase Rede Neuronal sem Modificação para Categorias Gramaticais
João used to live here	João used to live here	did used to live here
Miguel took off his mask	Miguel took off his mask	he took off his mask
Maria bought a new house	Maria bought a new house	she bought a new house
João was thirty	João was trinta years old	was was

Tabela 18 - Exemplos de Traduções das Redes Neurais Modificada e Não Modificada.

Consegue-se ainda verificar que este método de substituição de números não funcionaria para números escritos por extenso, já que a nomenclatura será depois remapeada para o valor da frase original, como é o exemplo da tradução de “O João

tinha trinta anos.” com a rede treinada com as categorias gramaticais a realizar uma tradução correta para “João was NUM1 years old”, no entanto, o remapeamento do NUM1 coloca a palavra na linguagem original. Não se procurou resolver este problema, uma vez que estes testes foram apenas realizados para verificar a capacidade da utilização de categorias gramaticais e não apenas texto para o treino da rede neuronal. Nesta avaliação foram utilizadas apenas as redes treinadas com 100 epoch e 128 batch size, já que apresentaram resultados semelhantes na avaliação realizada com os dados de treino e teste completos.

Como conclusão, foi neste apêndice apresentada uma comparação entre a tradução neuronal entre texto e a tradução neuronal mais híbrida, com texto e características gramaticais, em vez de nomes próprios e números. Com os resultados obtidos, verifica-se que existe melhoria em termos de resultados BLEU, para um conjunto de frases semelhantes, mas com nomes próprios e números diferentes entre elas, comprovando que a utilização de categorias gramaticais pode ajudar a rede neuronal a tornar-se mais abstrata na inferência destas categorias, prevenindo a criação de regras durante o treino que se associem ao valor da palavra, em vez do que a palavra é. Outro teste e comparação, que ficará como trabalho futuro, seria a criação de uma rede neuronal totalmente baseada apenas nas categorias gramaticais, ao contrário do que foi apresentado aqui, apenas algumas, sendo que estas foram escolhidas por parecerem ser das categorias gramaticais que podem assumir maior quantidade de valores diferentes sem modificar o significado da frase.

9.3. Código

Serão apenas apresentadas partes do código desenvolvido que se considere relevante para melhor compreender o descrito no tópico da implementação, ou partes do código que não se considere “expor” a ferramenta desenvolvida na sua totalidade, já que todo este trabalho faz parte de um projeto com parceiros.

9.3.1. Script de Obtenção de Títulos e Corpos de Notícias

```
10. from newsapi import NewsApiClient
11. from datetime import datetime
12. import csv
13.
14. def obterNoticiaCategoria(categoria, newsapi):
```

```
15.     top_headlines = newsapi.get_top_headlines(page_size=100,
16.         language='pt', country='pt', category=categoria)
17.
18.     for noticia in top_headlines["articles"]:
19.         data = {}
20.         data["titulo"] = noticia["title"]
21.         data["descricao"] = noticia["description"]
22.         data["origem"] = noticia["source"]["name"]
23.         data["url"] = noticia["url"]
24.         noticias.append(data)
25.
26.     return noticias
27.
28.
29. with open('headlines_'+datetime.today().strftime('%Y-%m-%d')+'.csv', 'w', newline='', encoding="utf-8") as file:
30.     writer = csv.writer(file, delimiter='\t')
31.     newsapi =
NewsApiClient(api_key='0a628492f9f84f6d9c2a4740e26fbd4d')
32.
33.     noticias_finais = []
34.     urls_tratadas = []
35.
36.     # category
37.     # business entertainment general health science sports
technology
38.     for categoria in ["sports", "health", "science",
"technology", "general", "entertainment", "business"]:
39.         noticias = obterNoticiaCategoria(categoria=categoria,
newsapi=newsapi)
40.
41.         for noticia in noticias:
42.             if noticia["url"] not in urls_tratadas:
43.                 noticias_finais.append(noticia)
44.                 urls_tratadas.append(noticia["url"])
45.
46.         for noticia in noticias_finais:
47.
writer.writerow([noticia["titulo"],noticia["descricao"],noticia["ori
gem"],noticia["url"]])
48.
49.
50.     file.close()
```

9.3.2. Tradutor Híbrido

```
from flask import Flask
from flask import request
from TradutorRegras import traduz_baseado_regras
from TradutorRedeNeuronal import traduz_neuronal, limpa_txt
app = Flask(__name__)

@app.route("/")
def base():
    return 'Serviço de Tradução de Português para Glosa da Língua
Gestual Portuguesa'

@app.route("/info")
```

```
def info():
    html = "<h1>API</h1>"
    html += "<h2>GET /traduz</h2>"
    html += "<p><b>Param: texto</b> - Texto a ser traduzido</p><br>"
    html += "<p><b><i>ou</i> Param: audio</b> - Base64 com áudio a  
ser transcrito</p><br>"
    html += "<p><b>Devolve: json_object</b> - Com estado, código de  
resposta, erro ou array com traduções <i>{'status':True/False,  
'cod':'xxx', 'erro':'descrição de erro, se existir', 'traducoes':  
'trad_principal': 'array termos tradução principal',  
'trad_alternativa': 'array termos tradução principal'}}</i></p>"
    html += "<br><br><br><br>"
    html += "Glossifica _ made by _ VA @ ADiT-Lab_IPVC 4 IVLinG"
    return html

@app.route("/traduz", methods=["GET"])
def rota_traducacao():
    base64_audio = None
    if request.args.get('texto'):
        texto_traduzir = request.args.get('texto')
    elif request.args.get('audio'):
        base64_audio = request.args.get('audio')
    else:
        return {"Erro": "Pedido mal formado", "cod": "001",  
"status": False}, 400

    if base64_audio:
        texto_traduzir = transcreve_witai(base64_audio)

    # verificar se o texto a traduzir existe
    if texto_traduzir is None or len(texto_traduzir) == 0:
        return {"Erro": "Sem frase a traduzir", "cod": "001",  
"status": False}, 400

    # enviar texto a traduzir para tradutor baseado em regras
    # > Devolve duas traduções, uma texto e outra Tokens spaCy para  
substituição dos nomes e numeros
    traducao_regras_txt, traducao_regras_tokens_spacy =  
traduz_baseado_regras(texto_traduzir)

    # converte Nomes e NUMEROS no spaCy para tokens especiais
    trad_temp_txt = ""
    qnt_nomes = 0
    qnt_numeros = 0
    arr_trocas = {}
    for token in traducao_regras_tokens_spacy:
        if token.text == "\n" or token.text == "\\n":
            continue
        if token.pos_ == "PROPN":
            qnt_nomes = 1
            trad_temp_txt += "NOME" + str(qnt_nomes) + " "
            arr_trocas["NOME" + str(qnt_nomes)] = list(token.text)
            continue
        if token.pos_ == "NUM":
            qnt_numeros += 1
            trad_temp_txt += "NUM" + str(qnt_numeros) + " "
            arr_trocas["NUM" + str(qnt_numeros)] = token.text
            continue

    trad_temp_txt += token.text + " "
```

```
# enviar texto traduzido pelo baseado em regras ao tradutor
neural
    traducao_neural =
    traduz_neural(limpa_txt(trad_temp_txt.strip()))

    # reconverter nomes e numeros
    arr_traducao_neural = []
    for idx, palavra in enumerate(traducao_neural.split()):
        if palavra.upper() in arr_trocas:

arr_traducao_neural.append(arr_trocas[palavra.upper()])
        else:
            arr_traducao_neural.append(palavra)

    print("\tA traduzir... ", texto_traduzir)
    print("\tBaseada em Regras... ", traducao_regras_txt)
    print("\tNeural... ", traducao_neural.split())

    # juntar traduções e devolver ao "pedinte", status 200
    return {"status": True, "cod": "000", "traducoes":
{"trad_principal": traducao_regras_txt, "trad_alternativa":
arr_traducao_neural}}

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8081)
```

9.3.3. Excerto de Código – Tradutor Baseado Em Regras

Este excerto de código apresenta a codificação das regras de conversão de SVO para OSV e restantes regras aplicadas à frase a traduzir.

```
#
# CRIAÇÃO HIERARQUIA FRÁSICA FRASES
#
# obj_nivel_hierarquia = {
#     frase => frase_original,
#     nivel_atual => [palavra, palavra, ...],
#     ligacao: [palavra, palavra, ...],
#     obj => [palavra, palavra, ...],
#     seguinte => ...obj_nivel_hierarquia...,
#     negacao => {True/False},
#     interrogacao => {True/False},
#     exclamacao => {True/False},
# }
# nivel_atual deverá possuir "SV", seguinte deverá possuir o "O"
# se seguinte possuir estrutura SV ou SVO, repetir, outra coisa,
adiciona a nivel_atual deste seguinte
# juntar palavras de "nivel_atual" desde o último nível para o
primeiro
    print("\n[>] Hierarquização de subfrases")
    print("\t> A iniciar criação da hierarquia para conversão de SVO
para OSV...")
    for frase in frases_utilizar:
        print("\t\t- Frase: " + frase)
        hierarquia_frases = {"frase": frase, "obj": [], "ligacao":
[], "nivel_atual": [], "seguinte": None, "negacao": False}
```

```
primeira_iteracao = True
nivel_atual = hierarquia_frases

# array para guardar temporariamente tokens determinantes
# possessivos para serem colocadas após o substantivo (após nome ou
# nome próprio). tokens removidas após utilizadas
tokens_posseivas = []
# array para guardar temporariamente tokens numéricas (e
# respectivos modificadores do número se existirem (kg, quilograma,
# ...)) para serem colocadas após o substantivo (após nome ou nome
# próprio). tokens removidas após utilizadas
tokens_numericas = []
# array para guardar temporariamente tokens adjetivos para
# serem colocadas LOGO após o substantivo (após nome ou nome próprio).
# tokens removidas após utilizadas
tokens_adje = []

while nivel_atual["seguinte"] is not None or
primeira_iteracao == True:
    primeira_iteracao = False
    verbo_encontrado = False
    obj_encontrado = False
    doc_nlp = nlp(nivel_atual["frase"])
    ignorar_ate = None # se for necessário passar iterações
    à frente até encontrar uma token específica (ignorando-a também)

    for idx, token in enumerate(doc_nlp):
        #####
        #
        # SVO para OSV e eliminação de valores a ignorar
        #
        #####
        # COISAS A IGNORAR
        # > Verificar se é necessário passar iterações à
        frente até encontrar uma token específica (ignorando-a também)
        if ignorar_ate is not None:
            if ignorar_ate.text == token.text:
                ignorar_ate = None
            continue

        # > Determinantes artigos def. e indef. (a, o, as,
        os, um, uns, uma, umas) -> eliminar
        if token.pos_ == "DET" and token.text.lower() in
        ["a", "o", "as", "os", "um", "uns", "uma", "umas"]:
            # não adicionar ao array de termos a considerar
            continue

        # > Palavras auxiliares (pos_ = AUX)
        if token.pos_ == "AUX":
            continue

        #
        #
        #

        # > Palavras adjetivos (pos_ = ADJ)
        if token.pos_ == "ADJ":
            tokens_adje.append(token)
            continue
```

```
# > Determinante possessivo (pos_ = DET | morph
PronType = Prs / Dem), guardar para colocar após substantivo (nome ou
nome próprio)
    if token.pos_ == "DET" and
token.morph.get("PronType") and token.morph.get("PronType")[0] in
["Prs", "Dem"]:
        tokens_possesivas.append(token)
        continue

# > Determinante número (pos_ = NUM), guardar para
colocar após substantivo (nome ou nome próprio)
    if token.pos_ == "NUM":
        tokens_numericas.append(token)

# verificar medidas... ALTERAR se token head da
token número possui dependencia nummod, adicionar
    if token.has_head() and token.dep_ == "nummod"
and token.head.lemma_ in LISTA_MEDIDAS_QUANTIDADES:
        tokens_numericas.append(token.head)
        # definir até token a passar à frente
        ignorar_ate = token.head

        continue

# > Advérbio de tempo, colocar no início (idx do obj
(? confirmar))
    if token.pos_ == "ADV" and token.text.lower() in
LISTA_ADV_TEMPORAIS:
        nivel_atual["obj"].insert(0, token)
        continue

# > Detecção de negação (se possuir advérbio negativo
(ADV - não, outras...?). ADV negativo desaparece e é colocado no
fim)
    if token.pos_ == "ADV" and token.text.lower() in
["não", "nao"]:
        nivel_atual["negacao"] = True
        continue

# se "obj" encontrado, verificar o momento em que a
frase passa a depender de outro verbo, e dividir
    if obj_encontrado:
        if token.head.pos_ == "VERB" and token.i <
len(doc_nlp)-1:# após V = 0, resto frase para nível seguinte,
continue ciclo níveis da hierarquia
            nivel_atual["seguinte"] = {"frase":
doc_nlp[idx+1:], "ligacao": [], "obj": [], "nivel_atual": [],
"seguinte": None, "negacao": False}

        for token_num in tokens_numericas:
            nivel_atual["obj"].append(token_num)
        for token_poss in tokens_possesivas:
            nivel_atual["obj"].append(token_poss)
        for token_adj in tokens_adje:
            nivel_atual["obj"].append(token_adj)
        tokens_numericas.clear()
        tokens_possesivas.clear()
        tokens_adje.clear()

        nivel_atual = nivel_atual["seguinte"]
```

```

        verbo_encontrado = False
        obj_encontrado = False
        continue

    # encontra conjunção, remove da frase e adiciona a
"ligacao"
    if token.pos_ == "SCONJ" and token.head.pos_ in
["AUX", "VERB"]:
        if isinstance(nivel_atual["frase"], str):
            nivel_atual["frase"] =
nivel_atual["frase"].replace(token.text, "", 1)
        else:
            nivel_atual["frase"] =
nivel_atual["frase"].text.replace(token.text, "", 1)
            nivel_atual["ligacao"].append(token)
            continue

    # SVO - encontrar o verbo para dividir SV/O
    # > se V, adicionar fim
    if token.pos_ == "VERB":
        verbo_encontrado = True

        # adicionar palavra atual
        nivel_atual["nivel_atual"].append(token)
        continue

    # se o verbo foi encontrado, procurar final do
objeto (procurar palavra com "dep_" = obj) e criar novo nível
    if verbo_encontrado:
        nivel_atual["obj"].append(token)

        if token.dep_ == "obj":
            obj_encontrado = True

    else:
        # verbo ainda não encontrado, assumir sujeito,
adicionar a "nivel_atual"
        nivel_atual["nivel_atual"].append(token)

    if obj_encontrado or verbo_encontrado:
        for token_num in tokens_numericas:
            nivel_atual["obj"].append(token_num)
        for token_poss in tokens_possesivas:
            nivel_atual["obj"].append(token_poss)
        for token_adj in tokens_adje:
            nivel_atual["obj"].append(token_adj)
    else:
        for token_num in tokens_numericas:
            nivel_atual["nivel_atual"].append(token_num)
        for token_poss in tokens_possesivas:
            nivel_atual["nivel_atual"].append(token_poss)
        for token_adj in tokens_adje:
            nivel_atual["nivel_atual"].append(token_adj)
    tokens_numericas.clear()
    tokens_possesivas.clear()
    tokens_adje.clear()

    print("\t\t\t" + str(hierarquia_frases))

    # construir frase
```

```
# adicionar "niveis_atuais" deste o último nível para o
primeiro
nivel_atual = hierarquia_frases
arr_temp_juntar = []
# - juntar num array
while nivel_atual is not None:
    if nivel_atual["negacao"]:
        arr_temp_juntar.append(nlp("Não"))
    arr_temp_juntar.append(nivel_atual["ligacao"])
    arr_temp_juntar.append(nivel_atual["nivel_atual"])
    arr_temp_juntar.append(nivel_atual["obj"])
    nivel_atual = nivel_atual["seguinte"]
# - juntar elementos do array do fim para o início para
formar a frase resultante
arr_temp_juntar.reverse()

print("\t\t\t" + str(arr_temp_juntar))

frase_apos_svo_osv = []
frase_apos_svo_osv_txt = ""
for parte in arr_temp_juntar:
    for token in parte:
        frase_apos_svo_osv.append(token)
        frase_apos_svo_osv_txt += token.text + " "

print("\t\t- Conversão SVO -> OSV: " +
frase_apos_svo_osv_txt)
frases_convertidas_osv.append(frase_apos_svo_osv)

frases_osv_termos = []
frases_osv_termos_tokens_spacy = []
for frase_osv in frases_convertidas_osv:
    arr_termos = []
    arr_tokens_spacy = []
    for token in frase_osv:
        tratado = False
        #####
        #
        #  Outras REGRAS LGP
        #
        #####

        # > Nomes próprios -> soletrar
        if token.pos_ == "PROPN":
            print("PROPN")
            arr_termos.append(list(token.text))
            arr_tokens_spacy.append(token)
            continue

        # > Números
        if token.pos_ == "NUM":
            # Número "Textual"
            if not e_numero(token.text):
                # a verificar o que fazer...
                arr_termos.append(token.text)
                arr_tokens_spacy.append(token)
            pass
        else:
            lista_termos_numeros = []
```

```
        arr_tokens_spacy.append(token)
        # Gestos específicos para 10, 100, 1000
        if float(token.text).is_integer() and
int(token.text) in [10, 100, 1000]:
            # gesto dez, cem, mil
            if int(token.text) == 10:

lista_termos_numeros.append(nlp("DEZ")[0].text)
                            if(not len(token.text.lstrip("10")) >
0):

arr_termos.append(lista_termos_numeros)
                            continue
                            token = nlp(token.text.lstrip("10"))[0]
                            if int(token.text) == 100:

lista_termos_numeros.append(nlp("CEM")[0].text)
                            if(not len(token.text.lstrip("100")) >
0):

arr_termos.append(lista_termos_numeros)
                            continue
                            token.text =
nlp(token.text.lstrip("100"))[0]
                            if int(token.text) == 1000:

lista_termos_numeros.append(nlp("MIL")[0].text)
                            if(not len(token.text.lstrip("1000")) >
0):

arr_termos.append(lista_termos_numeros)
                            continue
                            token.text =
nlp(token.text.lstrip("1000"))[0]

        # "soletrar" o que restar do número
        lista_termos_numeros = lista_termos_numeros +
list(token.text)
        arr_termos.append(lista_termos_numeros)
        continue

    # > procurar palavra na bd de flexões
    # - classes gramaticais: ADJ, ADV, CONJ, DET, INTERJ, N,
PREP, PRO, V, PFX, PREPXD, PREPXPRO, PREPXADV
    classe_gramatical = ""
    if token.pos_ in ["NOUN", "PROPN"]:
        classe_gramatical = "N"
    elif token.pos_ in ["VERB"]:
        classe_gramatical = "V"
    elif token.pos_ in ["DET"]:
        classe_gramatical = "DET"
    elif token.pos_ in ["ADJ"]:
        classe_gramatical = "ADJ"
    elif token.pos_ in ["ADV"]:
        classe_gramatical = "ADV"

    caracteristicas_flexoes = DB_FLEXOES.get(token.text +
"_" + classe_gramatical)
```

```
# > Nomes (não próprios) femininos, adicionar termo
MULHER antes do nome
if token.pos_ == "NOUN" and
token.morph.get("Gender")[0].lower() == "fem":
    # > algumas palavras apenas "são" femininas (ex:
bola, vizinha, nata, etc...)
    # > > Verificar pela lematização do ficheiro de
flexões. Se palavra = lemma, não adicionar "MULHER" ou se -AF
existir nos atributos (AF = apenas feminino, atributo adicionado
manualmente)
    # > > > APENAS VÁLIDO PORQUE O LABEL_LEX_SW FOI
MANUALMENTE MODIFICADO DE FORMA A QUE TODAS AS PALAVRAS MAPEEM AO
LEMMA MASCULINO (SE EXISTIR)
    if características_flexoes and
str(características_flexoes, "Windows-
1252")[0:str(características_flexoes, "Windows-
1252").index(".")]<code>.lower() != token.text.lower() and not b"-AF" in
características_flexoes:
    arr_termos.append(nlp("MULHER")[0].text)
    arr_tokens_spacy.append(nlp("MULHER")[0])

    # >> Diminutivo ou Aumentativo
    if características_flexoes:
        arr_termos.append(str(características_flexoes,
"Windows-1252")[0:str(características_flexoes, "Windows-
1252").index(".")]</code>

arr_tokens_spacy.append(nlp(str(características_flexoes, "Windows-
1252")[0:str(características_flexoes, "Windows-
1252").index(".")]<code>[0])

        if b":D" in características_flexoes:
            arr_termos.append(nlp("PEQUENO")[0].text)
            arr_tokens_spacy.append(nlp("PEQUENO")[0])
        if b":S" in características_flexoes:
            arr_termos.append(nlp("GRANDE")[0].text)
            arr_tokens_spacy.append(nlp("GRANDE")[0])
    else:
        arr_termos.append(token.lemma_)
        arr_tokens_spacy.append(token)

    continue

# > Determinantes artigos def. e indef. (a, o, as, os,
um, uns, uma, umas) -> eliminar
if token.pos_ == "DET" and token.text.lower() in ["a",
"o", "as", "os", "um", "uns", "uma", "umas"]:
    # não adicionar ao array de termos a considerar
    continue

# > Adposições -> eliminar
if token.pos_ == "ADP":
    # não adicionar ao array de termos a considerar
    continue

# > Pontuação -> eliminar
if token.pos_ == "PUNCT":
    # não adicionar ao array de termos a considerar
    continue
```

```
# > Verbos -> adicionar lema
if token.pos_ == "VERB":
    # >> usar lemmatização encontrada no ficheiro de
    flexões, caso contrário usar o do spacy
    if características_flexoes:
        arr_termos.append(str(características_flexoes,
"Windows-1252")[0:str(características_flexoes, "Windows-
1252").index(".")])

arr_tokens_spacy.append(nlp(str(características_flexoes, "Windows-
1252")[0:str(características_flexoes, "Windows-
1252").index(".")])[0])
    else:
        arr_termos.append(token.lemma_)
        arr_tokens_spacy.append(token)
    continue

# > Diminutivo ou Aumentativo (ainda não aplicado)
if características_flexoes:
    if b"D" in características_flexoes:
        arr_termos.append(str(características_flexoes,
"Windows-1252")[0:str(características_flexoes, "Windows-
1252").index(".")])
        arr_termos.append(nlp("PEQUENO")[0].text)

arr_tokens_spacy.append(nlp(str(características_flexoes, "Windows-
1252")[0:str(características_flexoes, "Windows-
1252").index(".")])[0])
        arr_tokens_spacy.append(nlp("PEQUENO")[0])
        continue
    if b"S" in características_flexoes:
        arr_termos.append(str(características_flexoes,
"Windows-1252")[0:str(características_flexoes, "Windows-
1252").index(".")])
        arr_termos.append(nlp("GRANDE")[0].text)

arr_tokens_spacy.append(nlp(str(características_flexoes, "Windows-
1252")[0:str(características_flexoes, "Windows-
1252").index(".")])[0])
        arr_tokens_spacy.append(nlp("GRANDE")[0])
        continue

# > Nenhuma regra aplicada, adicionar palavra
if not tratado:
    arr_termos.append(token.text)
    arr_tokens_spacy.append(token)

frases_osv_termos.append(arr_termos)
frases_osv_termos_tokens_spacy.append(arr_tokens_spacy)

arr_termos_mai = []
arr_termos_min = []
arr_termos_tokens_spacy = []

print("\n\n>>> === FINAL === >>>")
print(texto_original)
print(frases_osv_termos)
```