Universidade Federal do Pampa
Alex Malmann Becker

Uma ferramenta web para construção de Córpus Paralelo da língua falada e língua de sinais

Alegrete
2015

Alex Malmann Becker

Uma ferramenta web para construção de Córpus Paralelo da língua falada e língua de sinais

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Dr. Fábio Natanael Kepler

Alegrete

2015

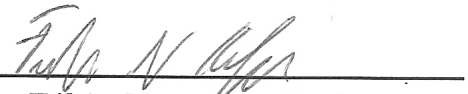
Alex Malmann Becker

Uma ferramenta web para construção de Córpus Paralelo da língua falada e língua de sinais

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

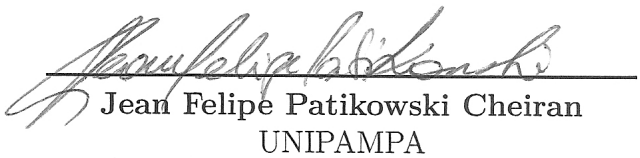
Trabalho de Conclusão de Curso defendido e aprovado em 30 de novembro de 2015

Banca examinadora:



Dr. Fábio Natanael Kepler
Orientador

Roberta dos Santos Messa
UNIPAMPA



Jean Felipe Patikowski Cheiran
UNIPAMPA

Este trabalho é dedicado primeiramente a Deus,
por ser a base fundamental em minha vida, meu guia e amigo,
ao meu pai Rogério, minha mãe Lurdes, e aos meus irmãos Arthur e Evandro,
e em especial ao meu irmão Everton, que me manda forças aonde quer que esteja,
aos amigos que estão sempre na luta comigo, e ao meu orientador que transmite seu
conhecimento sem esperar nada em troca.

Agradecimentos

Agradecemos a professora de LIBRAS Ana Paula Gomes e a intérprete Roberta Messa pela contribuição para que esse trabalho fosse desenvolvido e validado, visto que são pessoas que dominam o contexto desse trabalho e estiveram totalmente disponíveis a nos ajudar.

"A tarefa não é tanto ver aquilo que ninguém viu,
mas pensar o que ninguém ainda pensou
sobre aquilo que todo mundo vê.
(Arthur Schopenhauer)

Resumo

O principal objetivo deste trabalho é construir uma ferramenta online para anotar manualmente textos em qualquer língua falada com SignWriting em qualquer língua de sinais. Isto irá permitir a criação de córpus paralelos entre a língua falada e a de sinal, e com isso permitir o uso para a criação de ferramentas eficientes para a comunidade surda. Por exemplo, um córpus paralelo entre o Inglês e a Língua Gestual Americana poderia ser usado para treinar modelos de Aprendizado de Máquina para a tradução automática entre as duas línguas. Este sistema deve ser projetado de uma maneira que facilite a tarefa do anotador, não só por ser fácil de usar, mas também dando sugestões inteligentes conforme a anotação progride. Por construir uma ferramenta colaborativa online para a construção do córpus paralelo entre a língua falada e a de sinais, temos o objetivo de ajudar no desenvolvimento de recursos próprios para a língua de sinais, que podem ser posteriormente utilizados no estado da arte de modelos atualmente usados em ferramentas para a língua falada. Existem vários problemas e dificuldades na criação desse tipo de recurso, e nossa presente ferramenta já lida com alguns deles, como por exemplo, na representação textual de sinais e no alinhamento entre várias palavras com vários sinais.

Palavras-chave: Córpus Paralelo. SignWriting. Escrita de Sinais. LIBRAS.

Abstract

The main objective of this work is to build an online tool for manually annotating texts in any spoken language with SignWriting in any sign language. This will allow the creation of parallel corpora between spoken and sign languages that can be used to bootstrap the creation of efficient tools for the Deaf community. For example, a parallel corpus between English and American Sign Language could be used for training Machine Learning models for automatic translation between the two languages. Such system must be designed in a way that it eases the task of human annotators, not only by being easy to use but also by giving smart suggestions as the annotation progresses. By building a collaborative, online, easy to use annotation tool for building parallel corpora between spoken and sign languages we aim at helping the development of proper resources for sign languages that can then be used in state-of-the-art models currently used in tools for spoken languages. There are several issues and difficulties in creating this kind of resource, and our presented tool already deals with some of them, like adequate text representation of a sign and many to many alignments between words and signs.

Key-words: Parallel Corpora. SignWriting. Write Signals. LIBRAS.

Lista de ilustrações

Lista de siglas

ASL Língua de Sinais Americana - *American Sign Language*

DAC Comitê de Ação Surdo para SignWriting - *Deaf Action Committee for SignWriting*

DSL Língua de Sinais Dinamarquesa - *Danish Sign Language*

ELiS Sistema de Escrita das Línguas de Sinais

ELS Escrita da Língua de Sinais

LSF Língua de Sinais Francesa - *French Sign Language*

FSW Formal SignWriting

HamNoSys Sistema de Notação Hamburgo - *Hamburg Notation System*

HMM Modelo Oculto de Markov - *Hidden Markov Model*

ISL Língua de Sinais Indiana - *Indian Sign Language*

LGP Língua Gestual Portuguesa

LIBRAS Língua Brasileira de Sinais

LS Língua de Sinais

RNA Rede Neural Artificial - *Artificial Neural Network*

SGBD Sistema Gerenciador de Banco de Dados

SVM Máquina de Vetor de Suporte - *Support Vector Machine*

SW SignWriting

Sumário

1 Introdução

LS é a língua utilizada por pessoas surdas para se comunicarem entre si e com pessoas ouvintes. Segundo Paulraj et al. (2010), a LS é uma língua que é transmitida por sinal visual, ao invés de ser transmitida por som. Também, a LS possui uma semântica e sintática bem definida, como qualquer outra língua falada. No Brasil, segundo dados do Censo 2010, estima-se que existam em torno de 9.722.163 de pessoas que sofrem com algum tipo de deficiência auditiva, destes 347.481 não conseguem de modo algum ouvir (ESTATíSTICA, 2015).

A LS não é considerada uma língua universal, como já acontece na língua falada, e cada país ou região possui sua própria LS, por exemplo, no Brasil temos a Língua Brasileira de Sinais (LIBRAS), nos Estados Unidos a Língua de Sinais Americana - *American Sign Language* (ASL), e em Portugal a Língua Gestual Portuguesa (LGP). Porém, muitos surdos não conhecem uma LS, muito por falta de conhecimentos dos familiares ou até mesmo por dificuldades em encontrar escolas aptas a passarem esse conhecimento, e acabam criando sua própria comunicação entre seu círculo familiar. No Brasil com intuito de expandir e regularizar a LS, em 24 de abril de 2002 foi criada a Lei Federal Número 10.436, que oficializou a LIBRAS no Brasil, o que tornou a segunda língua oficial do país (CIVIL, 2015).

Em 1974 Valerie Sutton criou o sistema de escrita SW. BARRETO e BARRETO (2012) define SW como uma escrita visual que permite ler e escrever qualquer língua de sinais do mundo sem a necessidade de traduzir para a língua oral.

Com todo avanço que vem-se buscando conseguir para a comunidade surda, ainda muitas pessoas possuem grandes dificuldades no aprendizado quando integrada ao convívio escolar, em especial as crianças. O fator que muitas vezes prejudica a aprendizagem é o método utilizado pelas escolas, pois usam a escrita da língua falada para ensinar as primeiras palavras. Segundo BARRETO e BARRETO (2012), colocar a escrita da língua falada como meio de aprendizado inicial pode causar um confronto linguístico, principalmente com as crianças. Em contrapartida, o uso do SW ajuda o surdo a ter uma compreensão melhor da sua própria língua, além de o motivar a seguir no aprendizado, sem muitas dificuldades.

Muitos trabalhos foram desenvolvidos com o intuito de traduzir gestos da língua de sinais para a língua falada e vice-versa, porém, poucos trabalhos foram encontrados que tratam a tradução da escrita de sinais para texto. E um dos fatores que influenciam os poucos trabalhos encontrados nessa área é não possuir disponível o recurso próprio para que consiga realizar essas traduções, ou seja, não existe um córpus paralelo entre a

língua falada e a língua de sinais com SW disponível para uso.

O principal objetivo deste trabalho é construir uma ferramenta online para anotar manualmente textos em qualquer língua falada com SignWriting em qualquer língua de sinais, e com este córpus anotado permitir o uso para a criação de ferramentas eficientes para a comunidade surda.

Neste trabalho, explicamos no Capítulo 2 toda a fundamentação necessária para o entendimento da aplicação. No Capítulo 3, apresentamos os trabalhos relacionados ao nosso. Nos Capítulos 4, apresentamos a ferramenta desenvolvida. Por fim, no Capítulo 5, fazemos nossas considerações e apresentamos possíveis trabalhos que podem ser desenvolvidos com a utilização do córpus paralelo.

2 Fundamentação

2.1 Língua de Sinais

Língua de Sinais (LS) é a língua utilizada por pessoas surdas para se comunicarem entre si e com pessoas ouvintes. Segundo Paulraj et al. (2010), a LS é uma língua que é transmitida por sinal visual, ao invés de ser transmitida por som. Também, a LS possui uma semântica e sintática bem definida, como qualquer outra língua falada. No Brasil, segundo dados do Censo 2010 (ESTATíSTICA, 2015), estima-se que existam em torno de 9.722.163 pessoas que sofrem com algum tipo de deficiência auditiva, destes 347.481 não conseguem de modo algum ouvir. Com intuito de expandir e regularizar a LS, em 24 de abril de 2002 foi criada a Lei Federal Número 10.436, que oficializou a Língua Brasileira de Sinais (LIBRAS) no Brasil, o que tornou a segunda língua oficial do país (CIVIL, 2015).

A LS não é considerada uma língua universal, como por exemplo, no Brasil temos a LIBRAS, nos Estados Unidos a Língua de Sinais Americana - *American Sign Language* (ASL), e em Portugal a Língua Gestual Portuguesa (LGP). Com isso, o mesmo sinal em LIBRAS, pode possuir outro significado em ASL e também representar outro significado em LGP. Porém, muitos surdos não conhecem uma LS, muito por falta de conhecimentos dos familiares ou até mesmo por dificuldades em encontrar escolas aptas a passarem esse conhecimento, e acabam criando sua própria comunicação entre seu círculo familiar. Além disso, como qualquer língua falada no mundo, as LS também possuem diferenças entre elas. Essa diferença se dá de um país para outro ou até mesmo de região para região, dependendo de cada cultura. Segundo o autor Gesser (2009), ele explica o porquê da LS não poder ser considerada universal:

> Em qualquer lugar em que haja surdos interagindo, haverá línguas de sinais. Podemos dizer que o que é *universal* é o impulso dos indivíduos para a comunicação e, no caso dos surdos, esse impulso é **sinalizado**. A língua dos surdos não pode ser considerada universal, dado que não funciona como um decalque ou rótulo que possa ser colado e utilizado por todos os surdos de todas as sociedades de maneira uniforme e sem influências de uso. (GESSER, 2009, p. 12)

Assim como na língua falada, a LS pode ser analisada nos níveis fonético, fonológico, semântico, morfológico e sintático. Em 1960, William C. Stokoe conseguiu comprovar cientificamente que a LS é legítima, pois a mesma possui todos os critérios linguísticos, passando a partir desse momento a ser reconhecida oficialmente como uma língua. Cada LS possui um alfabeto manual para datilologia, que serve de auxilio para interação dos usuários, para soletrar palavras que não possuem um sinal ou até para soletrar algum

nome de pessoa, por exemplo. Porém, esse recurso as vezes é interpretado equivocadamente por pessoas não ligadas nesse contexto, pois faz com que muitas pessoas pensem que essa técnica seja a própria língua de sinais, o que não reflete a realidade.

2.1.1 Língua Brasileira de Sinais (LIBRAS)

A LIBRAS é a segunda língua oficial do Brasil, e é utilizada na comunicação entre os surdos. Ela faz uso principalmente das mãos para realizar a comunicação, utilizando os movimentos do corpo e expressões faciais como auxílio na realização dos sinais. Cada sinal pode ser articulado com uma ou duas mãos. Por padrão, o articulador do sinal faz uso principalmente da sua mão dominante, ou seja, usuários que são destros utilizarão mais a mão direita, enquanto os canhotos farão uso da mão esquerda. Segundo Quadros e Karnopp (2007), a LIBRAS possui cinco parâmetros fonológicos que são: a locação, o movimento, a configuração da mão, a orientação da mão e as expressões não manuais.

Segundo Quadros e Karnopp (2007), a locação (L) ou ponto de articulação (PA) é a área do corpo ou espaço no qual o sinal é executado. Na LIBRAS, o PA é realizado dentro do espaço de enunciação, o qual possui todos os pontos em um raio com alcance a todos os sinais articulados. As quatro principais áreas de articulação dos sinais são: cabeça, mão, tronco e braço.

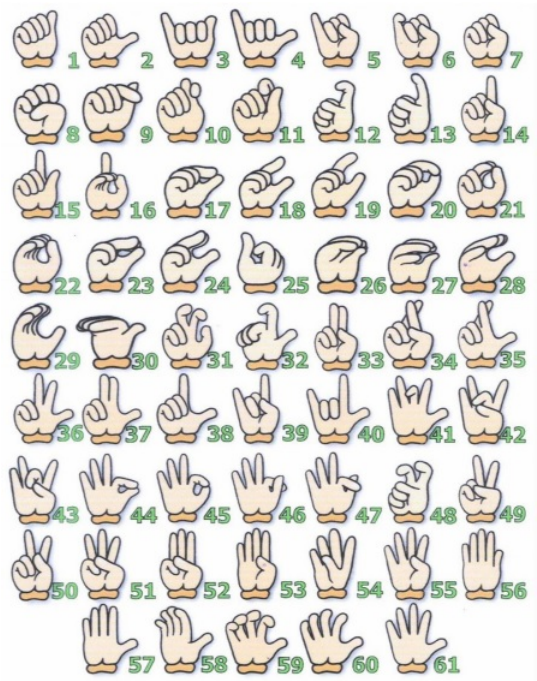
Segundo Quadros e Karnopp (2007), o movimento (M) são as várias formas e direções que podem acontecer na articulação de um sinal. São usados para diferenciar itens lexicais (nomes e verbos) ou até mesmo a direcionalidade em que o verbo ocorre. Os movimentos podem ocorrer nas mãos, pulsos e antebraço, além de poder ser movimentos unidirecionais, bidirecionais ou multidirecionais.

A configuração da mão (CM) diversas formas em que a mão pode realizar na execução dos sinais. Segundo Quadros e Karnopp (2007), a LIBRAS posui 46 CMs, porém estudos mais recentes descobriram que são na verdade 61 CMs, apresentadas na Figura 1.

Segundo Quadros e Karnopp (2007), a orientação da mão (Or) é a direção que a palma da mão aponta no momento da execução do sinal. Em LIBRAS, a Or é classificada em seis tipos: para cima, para baixo, para o corpo, para a frente, para a direita e para a esquerda.

Segundo Quadros e Karnopp (2007), expressões não manuais (ENM) são movimentos encontrados no rosto (face e olhos), na cabeça e no tronco durante a execução dos sinais. As ENM tem dois papéis na LS. O primeiro é a marcação de construção sintáticas, ou seja, permite marcar sentenças interrogativas, orações relativas, concordância, entre outros. E o segundo é a diferenciação de itens lexiais, que tem a finalidade de marcar referência pronominal, partícula negativa, advérbio, entre outros.

Figura 1 – 61 Configurações das mãos.



Fonte: Porfirio et al. (2013, p. 1)

2.2 Escritas da Língua de Sinais (ELS)

A Escrita da Língua de Sinais (ELS) é a forma de escrever uma LS, apresentando todas as possibilidades necessárias para a escrita de um sinal. Muitas vezes, a ELS tanto auxilia no entendimento da gramática da própria LS como a de outras línguas. Existem várias maneiras ou sistemas para realizar a escrita de sinais, como veremos na próxima subseção.

2.2.1 Sistemas de Escritas de Sinais

Também conhecidos como Sistemas de Notação, existem diversos sistemas que podem ser usados para escrever a LS. A seguir apresentaremos os principais sistemas e suas principais características.

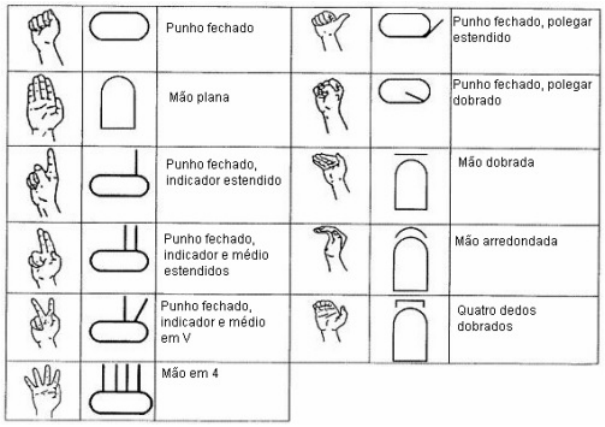
- **Notação de Stokoe:** A notação de Stokoe leva o sobrenome do seu criador, o pesquisador William C. Stokoe, o qual propôs em 1965 um sistema capaz de registrar as línguas de sinais para fins de investigação, juntamente com a publicação de um dicionário. Stokoe identificou três parâmetros dos sinais em seu sistema: a configuração da mão, a localização e o movimento (BARRETO; BARRETO, 2012). Na Figura 2 são apresentadas as configurações das mãos no sistema Stokoe.

Figura 2 – Configurações das mãos - Stokoe.

Fonte: Stumpf (2005, p. 48)

- **Sistema *HamNoSys:*** O sistema Sistema de Notação Hamburgo - *Hamburg Notation System* (HamNoSys) foi criado em 1989 na Universidade de Hamburgo (Alemanha) principalmente por Prillwitz e Vollhaber. O sistema é baseado na notação de Stokoe e possui um sistema computacional para sua escrita. Sua principal diferença se dá pela configuração de mãos, as orientações de dedos e da palma, as localizações sobre a cabeça e o tronco, os tipos de movimentos e a pontuação (STUMPF, 2005). Na Figura 3 são apresentadas as configurações das mãos no sistema HamNoSys.

Figura 3 – Configurações das mãos - HamNoSys.

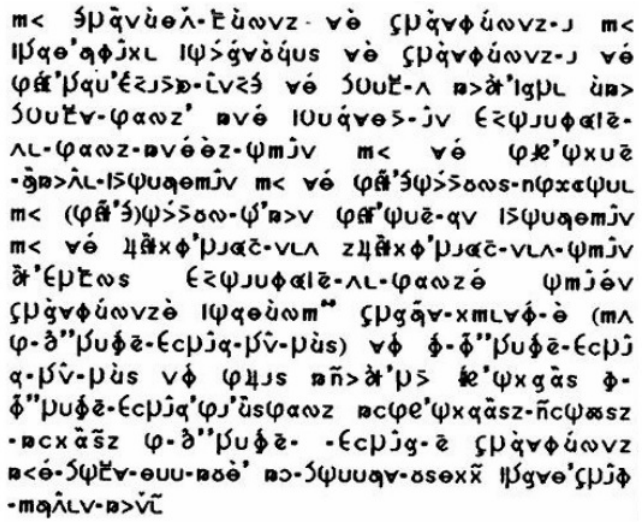


Fonte: Stumpf (2005, p. 50)

- **Sistema D'Sign:** O sistema D'Sign foi criado por Paul Jouison em meados de 1990, e continuado seus estudos pela Dr.ª Brigitte Garcia em sua tese. O sistema

D'Sign é aplicado à Língua de Sinais Francesa - *French Sign Language* (LSF) e é considerado um sistema bem elaborado capaz de transcrever frases inteiras da LSF. Suas unidades-símbolos são organizadas em: escolha dos dedos, na escolha dos braços, imagens, eixos de rotação, deslocamentos e zonas do corpo e do espaço. Na Figura 4 é apresentado um exemplo do sistema D'Sign.

Figura 4 – Exemplo do Sistema D'Sign.



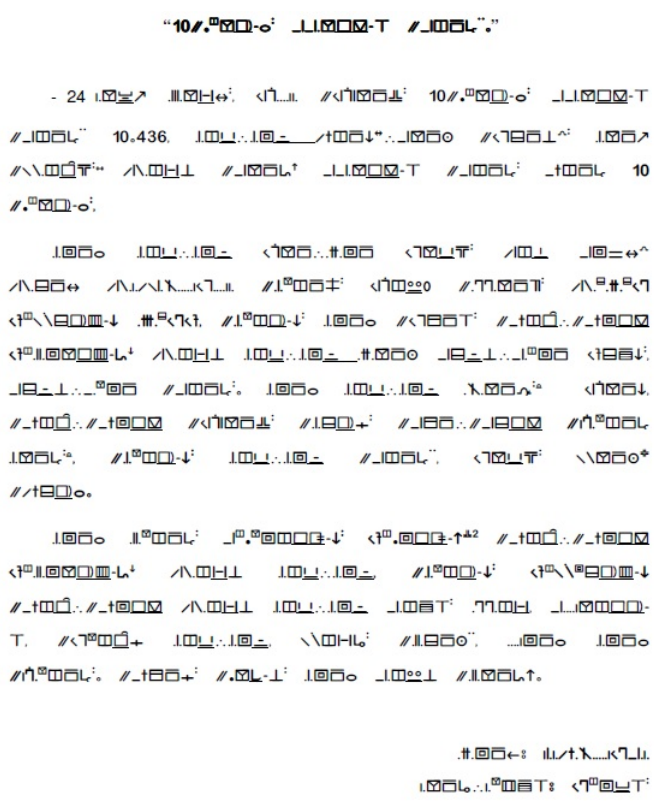
Fonte: Stumpf (2005, p. 51)

- **Sistema de Escrita das Línguas de Sinais (ELiS):** O Sistema de Escrita das Línguas de Sinais (ELiS) foi criado pela Dr.ª Mariângela Estelita Barros em 1997 e atualizado em 2008. Como parâmetros desse sistema foram adotados os três parâmetros de Stokoe e adicionados a orientação da palma e um diacrítico (orientação das pontas dos dedos), além de ser um sistema de escrita linear da esquerda para a direita. Na Figura 5 é apresentado exemplo do sistema ELiS.

- **Sistema de Escrita SignWriting (SW):** Em 1974 a norte-americana Valerie Sutton criou o sistema de escrita SW, após pesquisadores da Língua de Sinais Dinamarquesa - *Danish Sign Language* (DSL) da Universidade de Copenhagem tomarem conhecimento que a mesma havia criado um sistema para escrita de passos de dança, o DanceWriting, o que os levou a solicitarem a criação de um sistema para escrever as línguas de sinais. SW é apontando como o sistema de escrita mais utilizado no mundo e principalmente no Brasil, com usuários de mais de 40 países, além de permitir escrever qualquer LS (BARRETO; BARRETO, 2012).

  Atualmente, SW é desenvolvido pelo Comitê de Ação Surdo para SignWriting - *Deaf Action Committee for SignWriting* (DAC) o qual está ligado a uma organização sem fins lucrativos, coordenada por Sutton, a Center For Sutton movement Writing,

Figura 5 – Exemplo do Sistema ELiS.



Fonte: <http://elislibras.wix.com/home#!exemplos>

Inc. Como sistema computacional, o DAC desenvolveu uma plataforma online, o *SignPuddle*, que permite consultar, escrever e enviar e-mais com sinais em SW, entre outras funcionalidades.

Na Figura 6 é apresentado exemplo do sistema SW.

2.3 SignWriting (SW)

No Brasil não existe um sistema oficial para a escrita de sinais, porém a grande maioria de usuários, cursos de licenciatura e escolas utilizam o SW como o sistema padrão para o aprendizado da LS. Com isso, nosso trabalho dará maior ênfase ao SW e utilizará esse sistema para criação do sistema para anotar o córpus paralelo.

Quando se trabalha com SW computacionalmente é necessário trabalhar com algum formato de dados em que a máquina consiga entender e representar em forma digital. Nesse caso, textos escritos em SW possuem um formato padrão chamado de Formal SignWriting (FSW).

Veremos nas Subseções seguintes o que é FSW e a estrutura do SW.

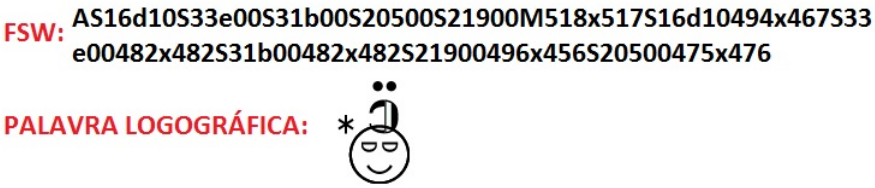
Figura 6 – Exemplo da frase “EU APRENDER LIBRAS”″ em SW

Fonte: <http://www.signbank.org/signpuddle2.0/translate.php?ui=12&sgn=46>

2.3.1 Formato de Dados Formal SignWriting (FSW)

O padrão FSW foi projetado em 2008 chegando a sua versão estável em 2012. Ele é uma definição de uma linguagem formal que permite escrever qualquer sinal de qualquer LS em uma sequência de caracteres ASCII, de tal forma que o computador consiga entender. Ou seja, é um padrão que utiliza uma string para definir palavras logográficas. A Figura 7 mostra um exemplo de um FSW e sua correspondente palavra logográfica.

Figura 7 – Exemplo FSW



Fonte: Adaptada de <http://swis.wmflabs.org>

FSW utiliza uma marcação leve (Lite Markup) que utiliza caracteres ASCII a fim de especificar a estrutura, símbolos e coordenadas. A utilização dessa marcação tem a vantagem de gerar um tamanho menor sem a necessidade de utilizar UNICODE. Cada palavra ASCII representa um sinal e é separada por um espaço em branco. FSW possui uma estrutura com expressões regulares associadas, e com isso consegue processar a marcação lite de forma rápida e eficiente. A seguir será apresentada essa estrutura (SLEVINSKI, 2015):

- **Symbol Key:**

  O Symbol Key é composto por seis caracteres, no qual o primeiro caracter S significa o início de um símbolo chave. Os três caracteres seguintes identificam o símbolo base. E por fim, os dois últimos caracteres identificam o preenchimento de rotação e modificadores. A notação para o Symbol Key é `S[123][0-9a-f]{2}[0-5][0-9a-f]`.

- **Coordinate:**

  A coordenada geral é a mais adequada para o processamento, e é definida por três números seguidos por um x seguido por mais três números. A notação para essa coordenada é `[0-9]{3}x[0-9]{3}`.

- **Explicit Coordinate:**

  O segundo tipo de coordenada é a coordenada explícita, que restringe corretamente o intervalo de números de 250 a 749. A notação para essa coordenada é `(2[5-9][0-9]|[3-6][0-9]{2}|7[0-4][0-9])x(2[5-9][0-9]|[3-6][0-9]{2}|7[0-4][0-9])`.

- **Signbox:**

  Um Signbox permite no máximo 2 coordenadas pré-processadas (baixo para direita). Este valor pré-calculado define uma caixa delimitadora em torno dos símbolos utilizados no sinal. A notação do Signbox é `[BLMR]([0-9]{3}x[0-9]{3})(S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*`.

- **Term:**

  O termo adiciona um prefixo de símbolo sequencial antes de definir um Signbox, o qual é usado para classificação. A notação do termo é `(A(S[123][0-9a-f]{2}[0-5][0-9a-f])+)[BLMR]([0-9]{3}x[0-9]{3})(S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*`.

- **Punctuation:**

  A pontuação inclui um pré processamento mínimo da coordenada. O máximo de coordenadas de uma pontuação é derivada através da transformação do número de cada coordenada separadamente, o qual é combinado os resultados. Para os valores de X e Y, o máximo = 1000 - mínimo. A notação da pontuação é `S38[7-9ab][0-5][0-9a-f][0-9]{3}x[0-9]{3}`.

- **Text:**

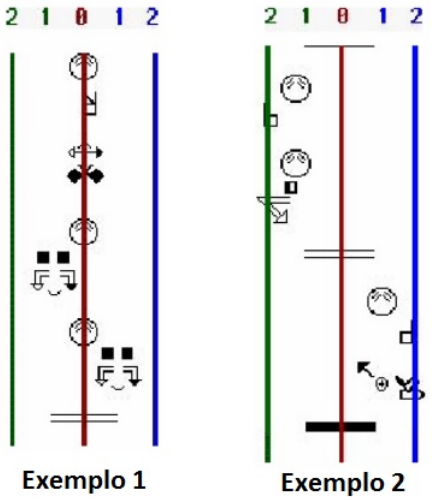
  O texto é definido como uma lista de sinais de pontuação. A notação para o texto é `((A(S[123][0-9a-f]{2}[0-5][0-9a-f])+)?[BLMR]([0-9]{3}x[0-9]{3})(S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*|S38[7-9ab][0-5][0-9a-f`

```
][0-9]{3}x[0-9]{3})( (A(S[123][0-9a-f]{2}[0-5][0-9a-f])+)?[BLMR]([0-
9]{3}x[0-9]{3})(S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*|
S38[7-9ab][0-5][0-9a-f][0-9]{3}x[0-9]{3})*.
```

2.3.2 Estrutura do SignWriting

Segundo BARRETO e BARRETO (2012), o sistema SW é escrito de cima para baixo em colunas, e os símbolos são reunidos de acordo com a estrutura do corpo humano. Cada coluna é dividida em três linhas imaginárias. Conforme é mostrado nos exemplos da Figura 8, a linha central representada pelo número 0 é responsável por identificar o centro do corpo humano, além de que as mãos, a cabeça e o tronco são escritos nela. As linhas 1 e 2 servem para representar o movimento das mãos e o corpo, tanto para esquerda como para direita. Como por exemplo, no exemplo 1 da Figura 8 a linha 1 deverá ser usada quando as mãos se movimentam para qualquer lado e a cabeça e o corpo não se movimentam, enquanto que no exemplo 2 da Figura 8 o tronco é deslocado da linha central e a cabeça deslocada para linha 1 e as mãos se movem para a linha 2. De acordo com Stumpf (2005), com a dificuldade em escrever uma língua tridimensional em um espaço plano, como é o caso do papel, a escrita em colunas visa a solucionar esse problema.

Figura 8 – Exemplos SW



Fonte: Adaptada de Stumpf (2005, p. 55)

2.4 Córpus Paralelo

Córpus paralelo é um conjunto de textos onde seus tokens (palavras) são alinhados entre uma língua fonte e uma língua alvo e são dados como entradas geralmente a um

modelo de treinamento, como por exemplo uma rede neural, para realizar a tradução entre as línguas. Caseli (2007) conceitua como:

> (...) Um *córpus* paralelo alinhado é um conjunto de exemplos (geralmente sentenças) escritos em uma língua fonte acompanhados de suas traduções na língua alvo. Esse conjunto de sentenças paralelas pode estar alinhado lexicalmente, ou seja, cada par de sentenças possui indicações de quais *tokens* (palavras, unidades multipalavras, símbolos de pontuação etc.) da sentença fonte são traduções de quais *tokens* da sentença alvo. (CASELI, 2007, p. 1)

2.5 Objetivos deste trabalho

O principal objetivo deste trabalho é construir uma ferramenta online para anotar manualmente textos em qualquer língua falada com SignWriting em qualquer língua de sinais. Isto irá permitir a criação de córpus paralelos entre a língua falada e a de sinal, e com isso permitir o uso para a criação de ferramentas eficientes para a comunidade surda. Por exemplo, um córpus paralelo entre o Português e a Língua Brasileira de Sinais poderia ser usado para treinar modelos de Aprendizado de Máquina para a tradução automática entre as duas línguas.

3 Trabalhos Relacionados

Nesta capítulo, apresentamos um levantamento da área com os principais trabalhos relacionados ao nosso e trabalhos que poderiam utilizar o córpus paralelo pela nossa ferramenta.

Zafrulla et al. (2011) analisaram o potencial do Kinect 360 no reconhecimento da Língua de Sinais (LS) e na verificação de jogos educativos para crianças surdas. Compararam um sistema baseado no protótipo do Kinect com o sistema que os autores já haviam desenvolvidos em trabalhos anteriores (sistema CopyCat), e que utiliza luvas coloridas e acelerômetros para rastrear os movimentos das mãos das crianças. Coletaram um total de 1000 frases em Língua de Sinais Americana - *American Sign Language* (ASL). Como verificação da acurácia entre o sistema CopyCat e o que faz uso do Kinect, eles chegaram ao melhor percentual quando as capturas dos sinais foram realizados em pé, com um total de 88,02% para o Kinect contra 76,12%, o que torna o Kinect uma opção viável para a verificação de sinais. Entretanto, esse trabalho deixa em aberto melhorias que devem ser feitas em relação ao reconhecimento das formas de mão como o próprio autor menciona, além de não deixar claro se o sistema utiliza algum método para análise da expressão facial, o que poderia contribuir para o jogo em questão.

Li et al. (2011) criaram um tradutor da ASL utilizando o Kinect 360 como dispositivo de entrada dos gestos. Os dados 3D das articulações capturadas pelo Kinect são analisados e combinados com uma biblioteca de sinais pré-gravadas. Os sinais combinados são transcritos a palavra ou frase, dando saída para uma interface de usuário. Entretanto os autores não apresentaram claramente os resultados obtidos quanto a taxas de acertos dos sinais.

Biswas e Basu (2011) propuseram um método para reconhecer gestos humanos utilizando o sensor Kinect 360. O método utiliza as imagens de profundidade capturadas pelo sensor, e aplica o método *auto thresholding* para separar os gestos humanos do fundo da cena. Treinaram o sistema utilizando Máquina de Vetor de Suporte - *Support Vector Machine* (SVM), com 8 classes (8 gestos diferentes). Entretanto os autores não deixaram claro o sucesso de reconhecimento do método, apenas apresentaram quantos frames foram treinados e testados em cada classe.

Oz e Leu (2011) apresentaram um sistema para reconhecimento de palavras em ASL para o Inglês, utilizando Rede Neural Artificial - *Artificial Neural Network* (RNA) para realizar essa tradução. Para detectar e extrair as características dos gestos visuais, usaram uma luva sensorial com 18 sensores (Cyberglove) e um rastreador de movimento (Flock of Birds 3-D). Utilizaram o algoritmo *backpropagation* para treinar o modelo de

RNA, contendo 50 palavras, com diferentes amostras (3, 6 e 12) por palavras. Realizaram dois testes com 120 sinais que não estavam no conjunto de treinamento. O primeiro foi com testes sequenciais e o o segundo foi com testes aleatórios, e obtiveram uma precisão de 90% no reconhecimento.

Zaki e Shaheen (2011) apresentaram um sistema para reconhecimento de gestos manuais em ASL. O sistema possui vários módulos, como por exemplo, detecção de mão, rastreamento, extração e um classificador, utilizando o Modelo Oculto de Markov - *Hidden Markov Model* (HMM). Selecionaram três características para realizar o reconhecimento: PCA (Análise de componentes principais), posição Kurtosis e o código que representa o movimento da mão. Utilizaram como entrada um banco de dados denominado *RWTH-BOSTON-50* e obtiveram uma taxa de erro no conhecimento do sinais de 10,90%.

Almasoud e Al-Khalifa (2012) apresentaram um sistema de tradução semântica do texto em árabe para SignWriting (SW) usando um analisador morfológico, aplicado as regras gramaticais, e por fim, executaram uma pesquisa semântica que substituiu cada palavra por seu símbolo SW, usando ontologia de domínio. Entretanto os autores, apesar de citarem que utilizaram especialista nos dois experimentos, não apresentaram dados que comprovam a acurácia em termos de percentuais.

Nguyen e Ranganath (2012) apresentaram dois algoritmos para reconhecimento de expressões faciais isoladas em ASL. O primeiro algoritmo usa transição dinâmicas do formato de rosto para aprender a predizer o formato do rosto no próximo instante de tempo. E o algoritmo 2, foi integrado o rastreador KLT em um esquema recursivo de Bayesian, que também utiliza formato de rostos e o modelo de transição dinâmicas. As restrições são especificadas através de um conjunto de forma de rosto que são treinadas pelo modelo PPCA. Utilizaram o HMM e uma SVM (com 6 expressões gramaticais) para avaliar e monitorar os resultados, o qual obtiveram uma precisão de 91,76% nos testes dependentes da pessoa e 87,71% independentes da pessoa, no reconhecimento de expressões faciais, e em comparação com o rastreador KLT, obtiveram uma precisão de 76%.

Kishore e Kumar (2012) apresentaram um sistema para reconhecimento da Língua de Sinais Indiana - *Indian Sign Language* (ISL) a partir de vídeos, independente da complexidade do fundo. Utilizaram o algoritmo *backpropagation* em uma RNA para classificar os sinais, os quais cada sinal do vídeo são convertido para comando de voz e texto. Treinaram a RNA com 1404 neurônios de entrada, 351 neurônios de saída e 652 neurônios na camada oculta. Avaliaram o sistema com um banco de dados composto por 351 sinais, e como resultados obtiveram uma taxa de 93% no reconhecimento dos sinais. Entretanto, apesar de usar um vetor de características, os autores não deixaram claro se o sistema consegue reconhecer expressões faciais.

Porfirio et al. (2013) apresentaram um método para reconhecer configurações de

mão em Língua Brasileira de Sinais (LIBRAS), utilizando malhas 3D. Utilizaram o sensor Kinect para obter as informações para criação de uma base de dados, composta por 610 vídeos, onde 5 atores realizaram todas as configurações de mão (61 em LIBRAS). Aplicaram o método *Spherical Harmonics*, que permite extrair recursos invariantes às condições de iluminação, tradução, rotação e escala, o qual é uma das principais vantagens em utilizar malhas 3D. Com o uso de malhas 3D, obtiveram um índice de acerto acima de 96%. Entretanto o autores não deixaram claro como realizaram a classificação de malhas similares, uma vez que podem existir configurações de mãos parecidas e que é confundida até mesmo no mundo real.

Yang e Lee (2013) propuseram um método para o reconhecimento de sinais e expressões faciais. Utilizaram o método BoostMap para verificar as formas de sinais e segmentá-las e uma máquina de vetor de suporte para reconhecer as expressões faciais. A avaliação do método atingiu uma taxa de 84% na precisão do reconhecimento dos gestos, com base em um banco de dados composto por 98 sentenças assinadas em ASL, das quais, cada sentença possui vários sinais manuais vinculados. Estes sinais foram capturados por duas câmeras, criando um vocabulário composto por 24 sinais, 17 alfabetos e 5 expressões faciais. Entretanto os autores não deixaram claro se o método consegue lidar com diferentes tipos de ambientes, como por exemplo, se teria um bom reconhecimento da expressão facial quando o assinador estiver exposto a uma baixa condição de iluminação.

Wang et al. (2013) apresentaram uma abordagem para reconhecimento de gestos a partir de uma mapa de profundidade capturados por um sensor de profundidade, Kinect 360. Propuseram um método para a segmentação do corpo humano e adotaram um mapa de distância para detectar mãos humanas. Criaram um conjunto de dados para realizar a avaliação do método, uma vez que não encontraram nenhum banco de dados para o cenário apresentado. O método proposto foi avaliado com 90% de precisão. Entretanto os autores não deixaram claro se utilizaram algum método para reconhecimento dos dedos, uma vez que o Kinect 360 não dá suporte a esse tipo de reconhecimento.

Almeida, Guimarães e Ramírez (2014) propuseram uma nova metodologia para extração de características da LIBRAS utilizando o Kinect para obtenção dos dados. Para fins de avaliar a metodologia, com o auxílio de especialistas, escolheram 34 sinais com 5 amostras para cada sinal e registraram em um banco de dados os dados recebidos do sensor. Como resultados, obtiveram por exemplo, para o sinal "person", uma acurácia entre 95% a 100%, enquanto para o sinal "prison", ficaram entre 37% a 66%, mas na maioria dos sinais obtiveram uma média acima de 80%. Entretanto os autores não deixaram claro qual o método que utilizaram para o reconhecimento dos dedos.

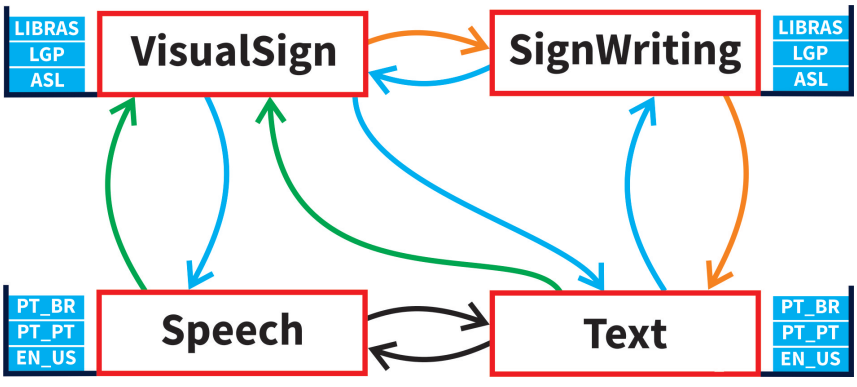
Kar e Chatterjee (2014) apresentaram um sistema para traduzir um documento escrito em SW para o idioma Inglês, com intuito de ser utilizado como uma aplicação móvel. O processo para tradução é dado pela entrada de imagens escritas em SW, as quais

são convertidas em um documento com frases em Inglês, e por fim, convertido o texto em áudio. Criaram um dicionário de SW em excel que contém todas as informações das palavras, o qual é usado na comparação das imagens na hora da tradução. A avaliação do sistema não foi realizada com usuários SW, e sim por cinco avaliadores, e obtiveram 88% das sentenças como sintaticamente corretas. Entretanto os autores não deixaram claro qual conjunto de dados usaram para realizar a avaliação e o que usaram para conversão do texto para áudio. Além disso, a proposta apresentada é limitada, uma vez que se aplica apenas a frases simples.

Sun, Zhang e Xu (2015) propuseram um algoritmo para modelar e reconhecer a LS em nível de sentença, utilizando o sensor Kinect para capturar os sinais. Aplicaram uma SVM latente para realizar a modelagem e a classificação do sinal simultaneamente. Para avaliar o método, realizaram testes com palavras avulsas e em sentenças, utilizando um banco de dados para ASL com cerca de 2000 registros para ambos os casos, e cada frase contém informações de cor, profundidade e do esqueleto. Como resultado, obtiveram uma precisão de 84,1% com o modelo proposto em nível de reconhecimento por sentença.

Os trabalhos mencionados acima, estão todos ligados com ferramentas que podem fazer uso de um córpus anotado com a nossa ferramenta. A Figura 9 apresenta um relacionamento dos principais blocos da área de tradução de uma LS, estando nosso trabalho relacionado com a utilização do córpus para a tradução dos blocos SignWriting para Text. A maioria dos trabalhos relacionados estão ligados diretamente na conversão do bloco VisualSign para o bloco Text e Speech, e vice-versa. Também, na Figura 9, as setas laranjas indicam que existem poucos trabalhos encontrados para conversão dos blocos, as setas azuis possuem muitos trabalhos encontrados, as setas verdes possuem até aplicativos móveis que realizam a tradução, como por exemplo, o aplicativo Hand Talk e o ProDeaf Libras, enquanto que as setas pretas possuem até serviços disponíveis para realizar a conversão dos blocos.

Figura 9 – Relacionamentos dos principais blocos de tradução de uma LS.



Nas buscas realizadas, não foi encontrado um córpus paralelo em SW disponível

para uso, tão pouco encontramos uma ferramenta que permitiria a criação desse córpus. A ferramenta que podia ser usada para essa criação seria a ferramenta *SignPuddle*, porém a mesma realiza traduções simples, a partir de uma consulta em um dicionário, e sua utilização como ferramenta para criação do córpus além de tornar o processo de anotação demorado e pouco flexível, seria necessário outras ferramentas externas para realizar todo o processo necessário para anotação do córpus. Com isso, optamos por desenvolver uma ferramenta colaborativa que auxiliasse o anotador apresentando sugestões inteligentes na hora da anotação, além de tornar o processo mais fácil e ágil, e assim, disponibilizar tanto o código fonte como o córpus anotado, para que outros pesquisadores consigam usufruir e desenvolver novas tecnologias e contribuir com a comunidade surda.

4 SignCorpus Annotator

O desenvolvimento desse trabalho foi dirigido pelo processo iterativo e incremental com uma abordagem ágil, que é um processo da engenharia de software que possui como foco entregar vários incrementos, onde cada incremento são executadas as atividades do processo, para assim obter um ganho caso ocorra alguma solicitação de mudança no projeto (SOMMERVILLE, 2007).

4.1 Diagrama de classe

A Figura 10 apresenta o diagrama de classe do domínio da aplicação. Os dois principais pacotes são *dictionary* e *document*. O primeiro pacote é responsável por controlar as entradas dos dicionários, além de armazenar a frequência de uso de cada sinal, com a finalidade de melhorar as sugestões de busca pelo melhor sinal na hora da anotação. O segundo pacote é responsável por manter os documentos e seus status de anotação, onde os sinais de cada sentença de um documento estão ligadas a entradas de dicionário, pelo seu Formal SignWriting (FSW). Embora cada *token* de uma sentença esteja alinhado com zero ou um sinal, a ferramenta permite anotação *multi-token*, ou seja, um único sinal que está sendo atribuído a uma frase, permite que o usuário concatene várias palavras consecutivas.

4.2 Processo para criação do córpus paralelo

A Figura 11 apresenta o processo de criação para anotar um documento. O processo começa com o usuário fazendo o *upload* de um documento (em Português, por exemplo) e executando o processo de preparação do documento, em que o sistema irá segmentar o documento em sentenças e, em seguida, em *tokens*. O usuário então pode começar a selecionar a primeira sentença para anotar. Selecionada a sentença, o sistema realiza uma busca por sinais candidatos para cada *token* e classifica-os de acordo com a semelhança e frequência de utilização. O usuário, em seguida, combina ou separa palavras e seleciona o melhor sinal para cada entrada. Como é comum em línguas de sinais, algumas palavras em línguas faladas não tem sinal. Estes são informados como palavras sem sinal. Uma vez que cada sentença do documento é anotado, o documento é marcado como anotado. Com a repetição desse processo, conseguimos ter um córpus paralelo com vários documentos anotados, e assim utilizar o córpus no treinamento de uma rede neural para tradução automática do sinal, por exemplo.

Figura 10 – Diagrama de classe do domínio da aplicação.

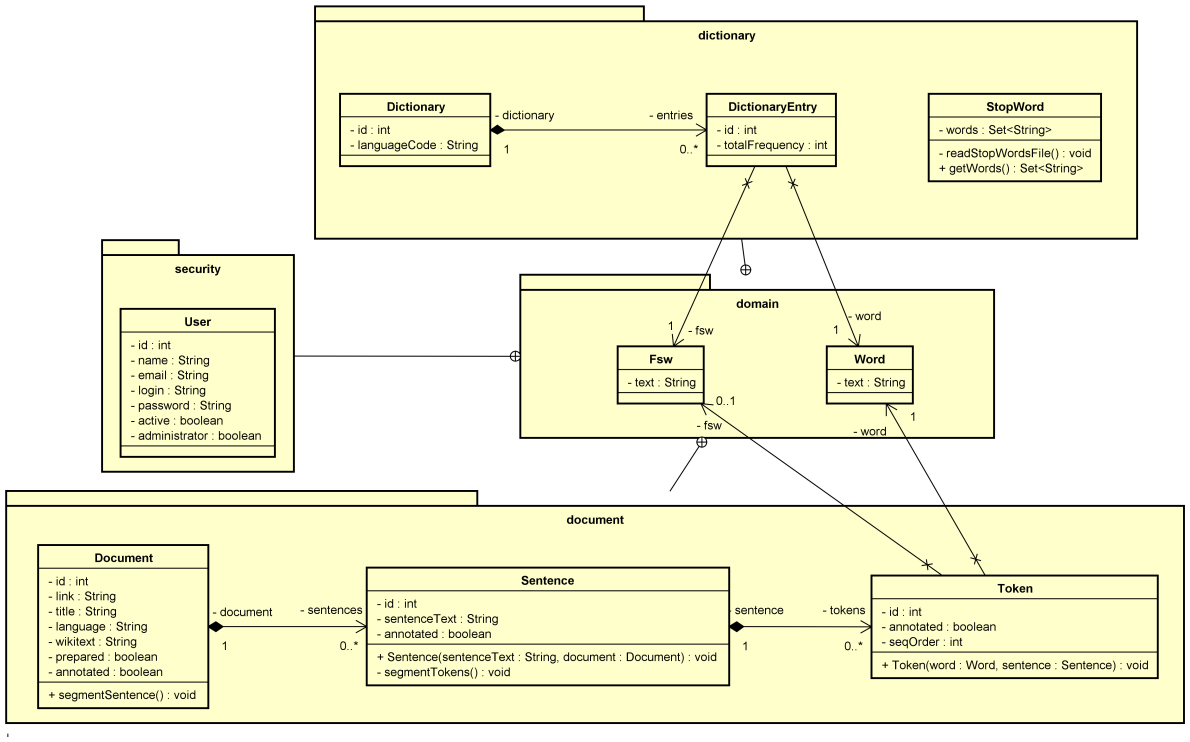
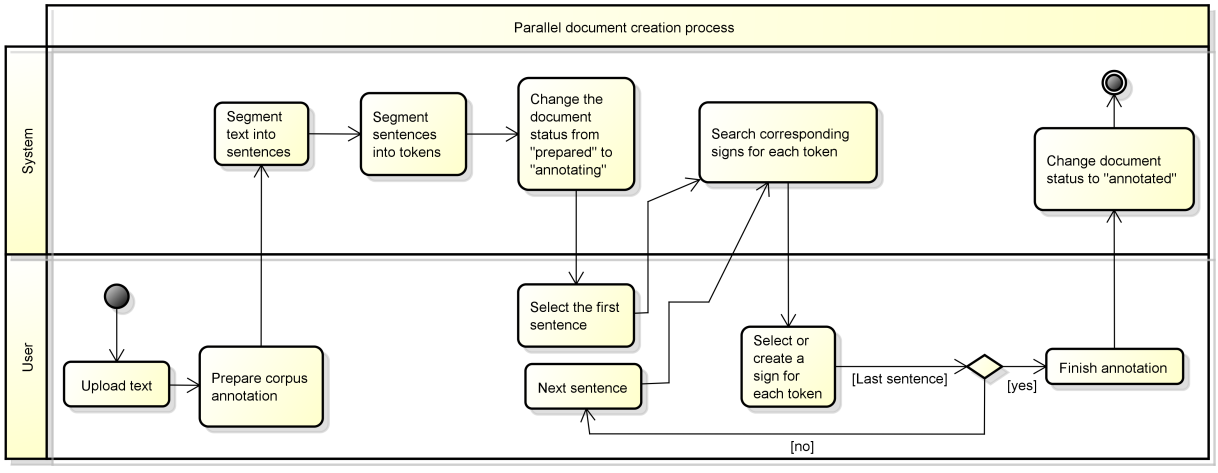


Figura 11 – Processo criação do córpus paralelo.



4.3 Implementação

A ferramenta foi implementada na plataforma Java Web usando o *framework* JSF (JavaServer Faces) com uma arquitetura MVC (Model-View-Controller). Como servidor de aplicação, foi utilizado o servidor Glassfish 4. Também, como Sistema Gerenciador de Banco de Dados (SGBD) usamos o MySql e para persistência dos dados o framework

Hibernate que possui grandes benefícios em seu uso, como desempenho e gerenciamento próprio da criação/atualização das tabelas e outros atributos conforme configurado via anotação nas classes java. Optamos por utilizar esses frameworks por serem fortemente usados pela indústria de software, além de trazer vantagens na hora do desenvolvimento e na manutenção do software.

4.4 Incrementos

Como mencionado, utilizamos um processo iterativo e incremental e com isso foi gerado um total de dez incrementos em todo o ciclo de desenvolvimento da ferramenta, os quais abordaremos nas próximas subseções. A Figura 12 apresenta resumidamente o que foi desenvolvimento em cada incremento. Nos incrementos foram utilizados tarefas e histórias de usuário. Histórias de usuário são uma forma de representar a necessidade do usuário e são bastante usados por metodologias ágeis.

Figura 12 – Lista de incrementos.

Incremento 1	
1	Configurar inicial do projeto
2	Configurar módulo de segurança
3	Como usuário desejo poder me registrar no sistema
4	Como usuário desejo poder me logar no sistema
5	Como usuário desejo poder atualizar meus dados
Incremento 2	
6	Configurar biblioteca para leitura do FSW e mostrar o sinal
7	Como usuário desejo poder importar o dicionário SignPuddle
8	Como usuário desejo poder ver as palavras do dicionário
Incremento 3	
9	Como usuário desejo poder adicionar novos documentos manualmente
10	Como usuário desejo poder importar documentos do Wikipedia
Incremento 4	
11	Como usuário desejo poder preparar o documento para anotação
Incremento 5	
12	Como usuário desejo poder unir/separar os tokens da sentença
13	Como usuário desejo poder incluir o FSW manualmente
incremento 6	
14	Implementar busca avançada do sinal
15	Como usuário desejo poder anotar um token com o sinal escolhido
Incremento 7	
16	Integrar editor SignMaker
Incremento 8	
17	Como usuário desejo poder exportar o arquivo do córpus paralelo
18	Internacionalizar sistema (Inglês e Português)
Incremento 9	
19	Validar sistema
Incremento 10	
20	Implantar sistema online para produção

4.4.1 Incremento 1

O primeiro incremento ficou responsável por realizar toda a configuração necessária para que o projeto seja executado, como por exemplo, configuração das bibliotecas via Maven, configuração dos frameworks JSF e Hibernate. Também, implementamos o módulo de segurança ao sistema, o qual utilizamos o framework Spring Security, que é um framework que provê autenticação criptografada e um rigoroso controle de acesso que é altamente configurado via XML. Além disso, implementamos algumas histórias de usuários ligadas ao acesso do sistema, bem como permitir o usuário poder realizar registro no sistema, realizar login e poder atualizar seus dados, que podem ser vistas nas Figura 13, Figura 14 e Figura 15, respectivamente.

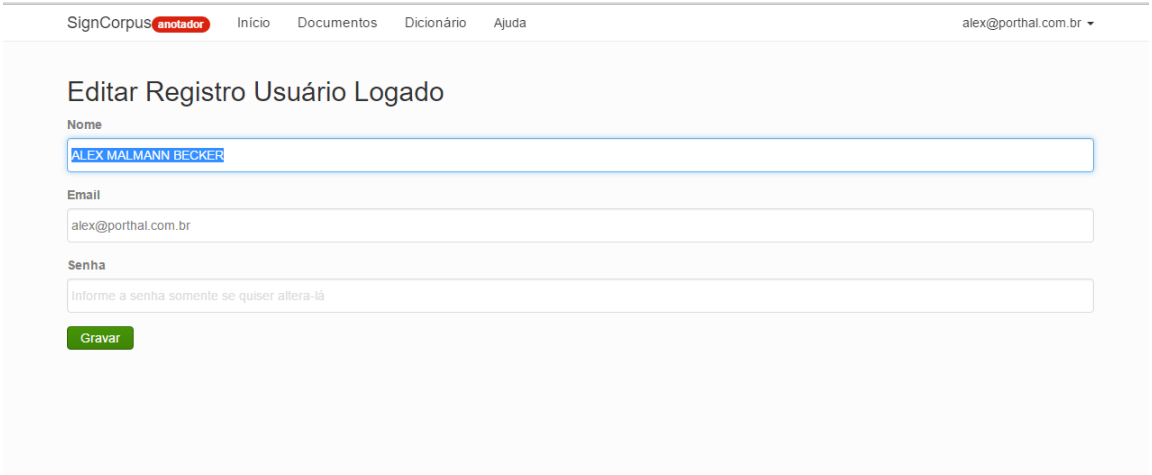
Figura 13 – Página de login no sistema.

4.4.2 Incremento 2

O segundo incremento ficou responsável por realizar a integração de um dicionário SignWriting no sistema e fazer a leitura de um FSW para mostrar o sinal ao usuário. Com isso, integramos o dicionário disponível pela ferramenta *SignPuddle*, em um formato "JSON". A leitura desse arquivo é feita utilizando a biblioteca GSON, que auxilia a leitura desse tipo de arquivo, convertendo os dados diretos para a classe passada como parâmetro. A Figura 16 apresenta o código implementado para importar o arquivo do dicionário, enquanto Figura 17 apresenta a página do dicionário com o botão para importação do dicionário.

Figura 14 – Página para se registrar no sistema.

Figura 15 – Página para editar dados usuário logado.



Para realizar a leitura de um FSW utilizamos a biblioteca SignWriting 2010, disponível em <https://github.com/Slevinski/sw10js>. Essa biblioteca é escrita em Javascript, e utiliza especificações internacionais do SignWriting (SW). Na Figura 18 apresentamos o código para realizar a leitura de um FSW e sua conversão para o formato SVG. A função FSW2SVG recebe o FSW e atributos que podem ser incorporados no sinal, como por exemplo a cor que o sinal deverá ser mostrado. Com isso, utilizamos a função da biblioteca através do método sw10.svg, retornando o sinal no formato SVG. Já o método convertFswToSvg é utilizado para desenhar no lugar específico da página. Por fim, apresentamos na Figura 19, as entradas do dicionário, com o sinal desenhado a partir de um FSW.

Figura 16 – Código para importação do dicionário *SignPuddle*.

```
public void importDictionaryJson(FileUploadEvent event) throws FileNotFoundException, UnsupportedEncodingException, Exception {
    Reader reader = new InputStreamReader(event.getFile().getInputstream(), "UTF8");

    Gson gson = new Gson();
    ImportDictionary importedDic = gson.fromJson(reader, ImportDictionary.class);

    Dictionary dic = new Dictionary();
    dic.setLanguageCode(importedDic.getLanguageCode());
    dic.setEntries(new ArrayList<DictionaryEntry>());
    for (ImportDictionaryEntry entry : importedDic.getEntries()) {
        for (String word : entry.getWords()) {
            DictionaryEntry dictionaryEntry = new DictionaryEntry();
            dictionaryEntry.setDictionary(dic);
            dictionaryEntry.setTotalFrequency(0);
            dictionaryEntry.setWord(new Word(word));
            dictionaryEntry.setFsw(new Fsw(entry.getFsw()));
            dic.getEntries().add(dictionaryEntry);
        }
    }
    boolean imported = new DictionaryDao().salvar(dic);
    if (imported) {
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO, "", "Dicionário " +
                event.getFile().getFileName() + " importado com sucesso!"));
    } else {
        FacesContext.getCurrentInstance().addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "",
                "Erro ao importar dicionário: " + event.getFile().getFileName()));
    }
    init();
}
```

Figura 17 – Página para importação do dicionário.

4.4.3 Incremento 3

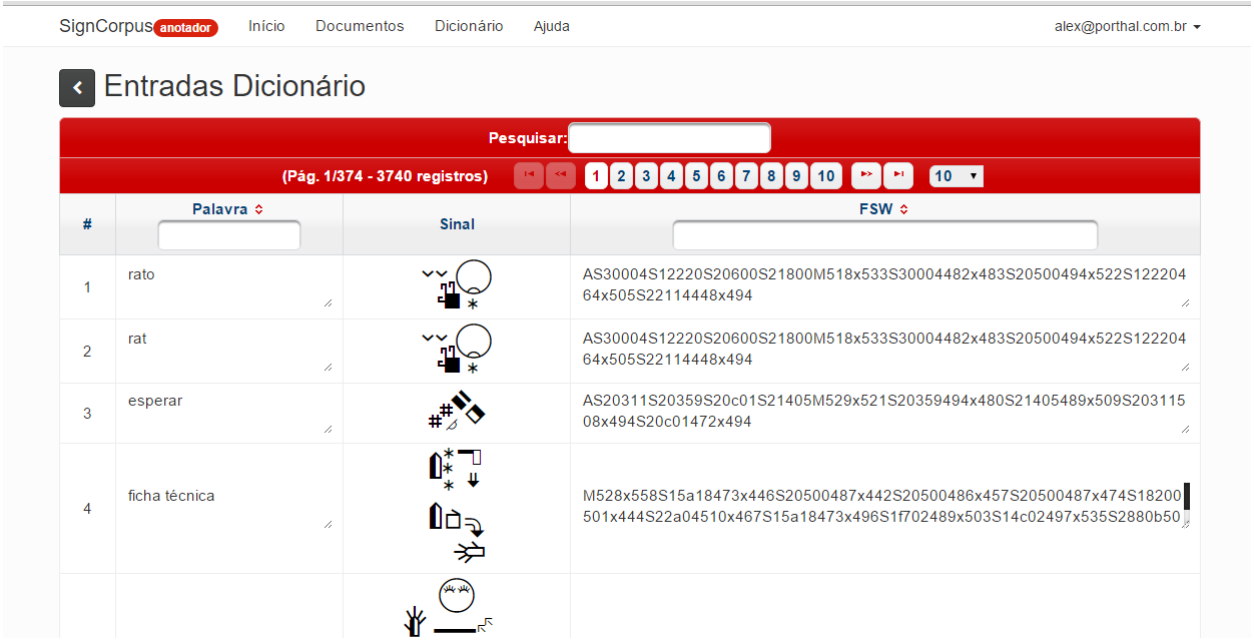
No incremento 3 foi desenvolvida a história de usuário para adicionar novos documentos, importando artigos da Wikipedia ou cadastrando manualmente. A Figura 20 apresenta a página para realizar o cadastro do documento, podendo ser informado a URL da Wikipedia e clicando no botão "Puxar Informações", preencherá automaticamente os campos abaixo. Caso deseja preencher manualmente, basta informar os campos Título e Texto do documento.

Para realizar a importação dos artigos da Wikipédia foi utilizada a biblioteca

Figura 18 – Código javascript para chamada do método de leitura do FSW.

```
<script>
    function FSW2SVG(fsw, classAttributes) {
        var options = {};
        if (classAttributes !== '') {
            options['class'] = classAttributes;
        }
        return sw10.svg(fsw, options);
    }
    ;

    function convertFswToSvg(fsw, name, classAttributes) {
        $('div[name=options_' + name + ']').each(function () {
            $(this).html(FSW2SVG(fsw, classAttributes));
        });
    }
    ;
</script>
```

Figura 19 – Página com as entradas do dicionário com a mostra do sinal.



"JSOUP", na qual consegue fazer a leitura de uma página web e obter o conteúdo de um elemento passando o id do mesmo, além de ser compatível com Java. A Figura 21 apresenta o código usado para realizar essa importação.

4.4.4 Incremento 4

O incremento 4 ficou responsável por executar o processo de preparação do documento, ou seja, preparar o documento de forma que permita que o anotador consiga

Figura 20 – Página para inclusão de documentos no sistema.

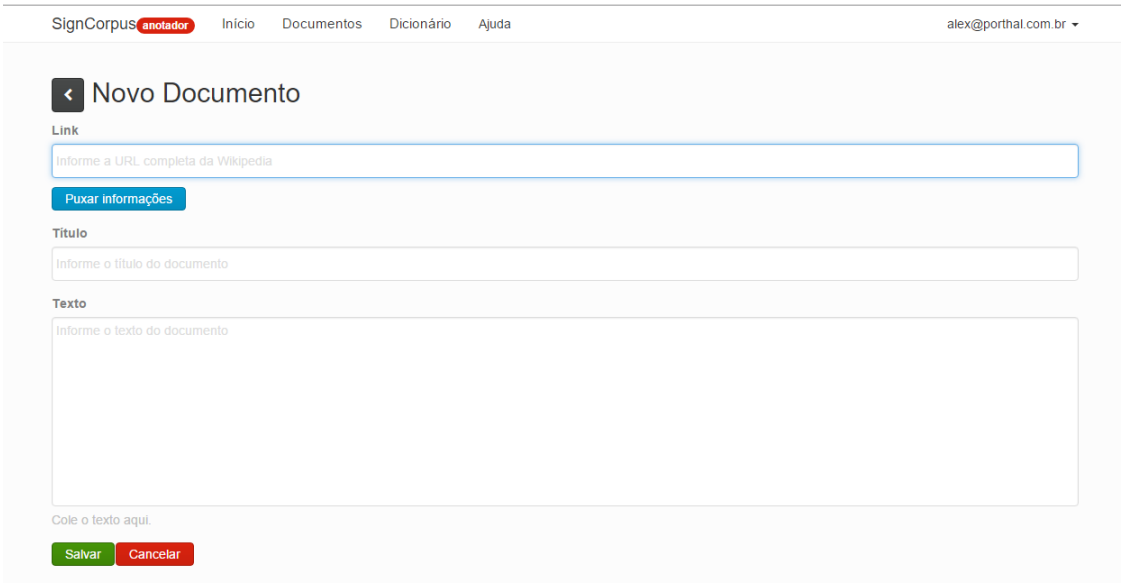


Figura 21 – Código para importação de artigo da Wikipedia.

```
public void importWikiText(String doc) throws IOException {
    try {
        org.jsoup.nodes.Document jdoc = Jsoup.connect(doc).get();
        Element title = jdoc.getElementById("firstHeading");
        this.document.setTitle(title.text());

        Element body = jdoc.getElementById("mw-content-text");
        this.document.setWikiText(body.text());
    } catch (Exception e) {
        FacesContext.getCurrentInstance().addMessage(null,
                new FacesMessage(FacesMessage.SEVERITY_ERROR, "",
                        "Erro ao puxar informações do link informado!"));
    }
}
```

anotar um sinal para cada *token*. Esse processo consiste basicamente em pegar o texto do documento a ser preparado e segmentá-lo em sentenças, após isso, cada sentença é tokenizada, gerando assim os *tokens*. Para realizar esse processo, utilizamos a biblioteca OpenNLP, que é uma biblioteca baseada em aprendizado de máquina para processar texto de linguagem natural, e que da suporte a tarefas como a segmentação e a tokenização. As Figura 22 e Figura 23 apresentam os métodos para segmentação e tokenização, respectivamente.

Figura 22 – Código para segmentar o texto em sentença.

```
public boolean segmentSentence() throws FileNotFoundException {
    boolean returnStatus = false;
    ExternalContext ec = FacesContext.getCurrentInstance().getExternalContext();
    InputStream modelIn = ec.getResourceAsStream("/resources/opennlp/en-sent.bin");

    try {
        SentenceModel model = new SentenceModel(modelIn);
        SentenceDetectorME sentenceDetector = new SentenceDetectorME(model);

        String strSentences[] = sentenceDetector.sentDetect(this.getWikiText());

        for (String strSentence : strSentences) {
            this.getSentences().add(new Sentence(strSentence, this));
        }
        returnStatus = true;
    } catch (IOException e) {
    } finally {
        if (modelIn != null) {
            try {
                modelIn.close();
            } catch (IOException e) {
            }
        }
    }
    return returnStatus;
}
```

Figura 23 – Código para tokenizar a sentença.

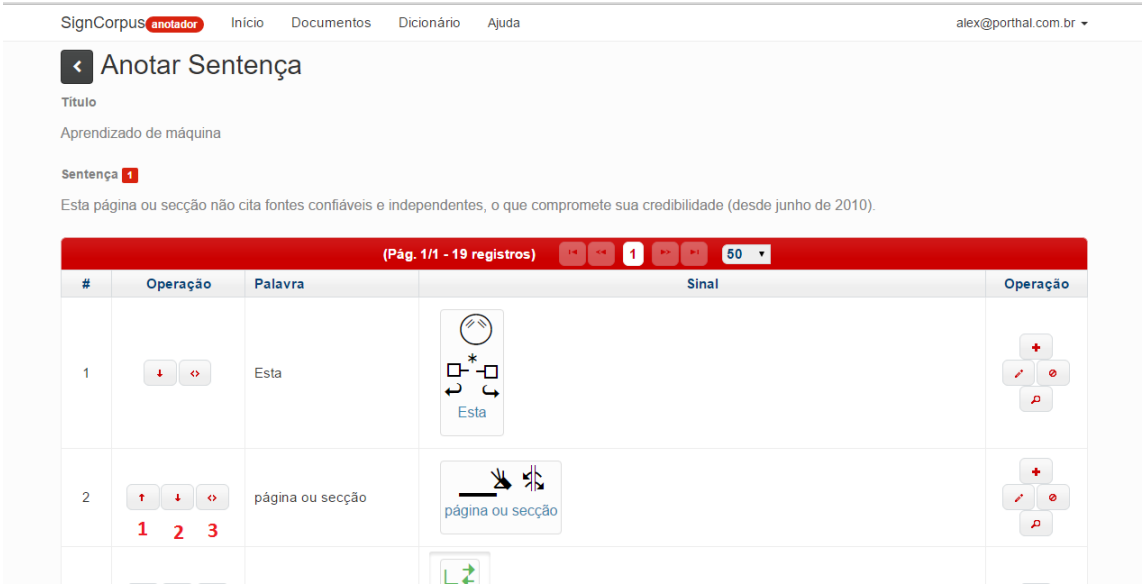
```
private void segmentWords() throws FileNotFoundException, IOException {
    ExternalContext ec = FacesContext.getCurrentInstance().getExternalContext();
    InputStream is = ec.getResourceAsStream("/resources/opennlp/en-token.bin");
    TokenizerModel model = new TokenizerModel(is);
    TokenizerME tokenizer = new TokenizerME(model);

    String strTokens[] = tokenizer.tokenize(this.getSentence().getSentenceText());

    int seqOrder = 1;
    for (String strToken : strTokens) {
        Token token = new Token(new Word(strToken), this, seqOrder++);
        this.tokens.add(token);
    }
    is.close();
}
```

4.4.5 Incremento 5

O quinto incremento foi responsável por adicionar operações às palavras, e assim resolver o problema de várias palavras ter o mesmo sinal. Essas operações são de unir palavras, tanto com a palvra de cima, como com a palavra de baixo, e também poder separar as palavras novamente. Essas três operações são representadas pelos botões 1, 2 e 3 respectivamente na Figura 24.

Além disso, implementamos uma função para poder adicionar um sinal inserindo o FSW direto. A Figura 25 apresenta o código javascript para validar um FSW, o qual tem o retorno se é válido pela chamada da função `sw10.fsw(fsw)`, da biblioteca SignWriting

Figura 24 – Página anotação com as operações da palavra.



2010. Por fim, na Figura 26 mostra o modal para inclusão de um FSW.

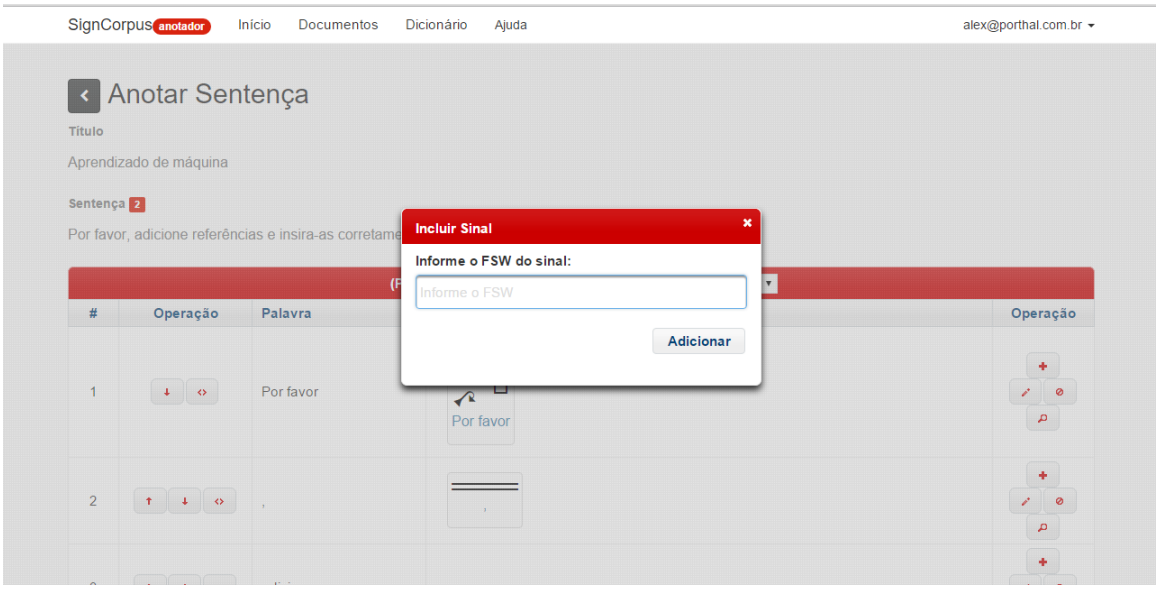
Figura 25 – Código para validar um FSW.

```
<script type="text/javascript">
    function validateFsw(fsw) {
        var isValid = sw10.fsw(fsw);
        if (isValid === '') {
            showMessageInvalidFsw();
        } else {
            newFswToken();
        }
    };
</script>
```

4.4.6 Incremento 6

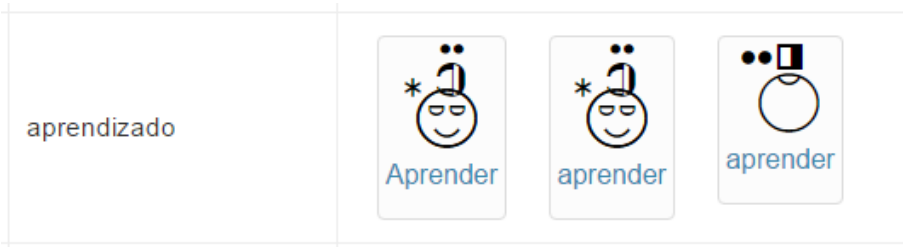
O sexto incremento ficou responsável por implementar a principal parte do sistema, que é a busca inteligente dos sinais, auxiliando assim o anotador com melhores sugestões na hora da anotação, e a parte de realizar a própria anotação do sinal. Para essa busca inteligente, implementamos alguns conceitos como por exemplo, o uso de stopWords e a lematização da palavra. StopWords são palavras que podem ser consideradas irrelevantes aplicado a um contexto, e no nosso caso na busca, pois como acontece em Língua Brasileira de Sinais (LIBRAS), palavras como artigos e pronomes, por exemplo, não são traduzidas, e com isso precisamos ignorá-las, para até mesmo ter um melhor desempe-

Figura 26 – Modal para incluir FSW na anotação.



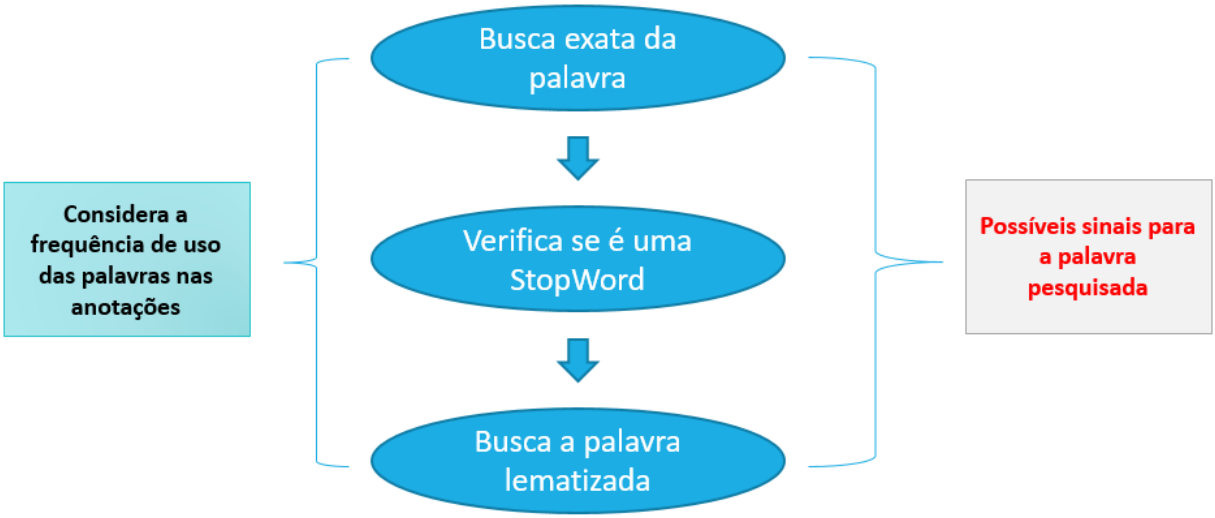
nho na busca. Já a lematização é uma técnica que buscadores de palavras utilizam para reduzir a palavra ao seu radical comum, ignorando o tempo verbal, gênero, plural, etc, pois como acontece em LIBRAS, não há conjugação de verbos, por exemplo. A Figura 27 apresenta a busca com lematização, onde para a palavra "aprendizado", o sistema busca pela palavra "aprend", mostrando sugestões mais coerentes para o anotador. Para implementar a lematização utilizamos a biblioteca PTStemmer que possui suporte à linguagem Java, é de código livre, e possui lematização para a língua portuguesa. Utilizamos a implementação do algoritmo Orengo Stemmer, com o qual obtemos a lematização da palavra através do método wordStemming. Além disso, utilizamos a frequência de uso da palavra nas anotações para auxiliar na classificação do melhor sinal. A Figura 28 apresenta o processo de busca dos sinais, o qual possui três etapas, que serão executadas uma vez que a etapa em execução não seja satisfeita.

Figura 27 – Busca com a etapa de lematização.



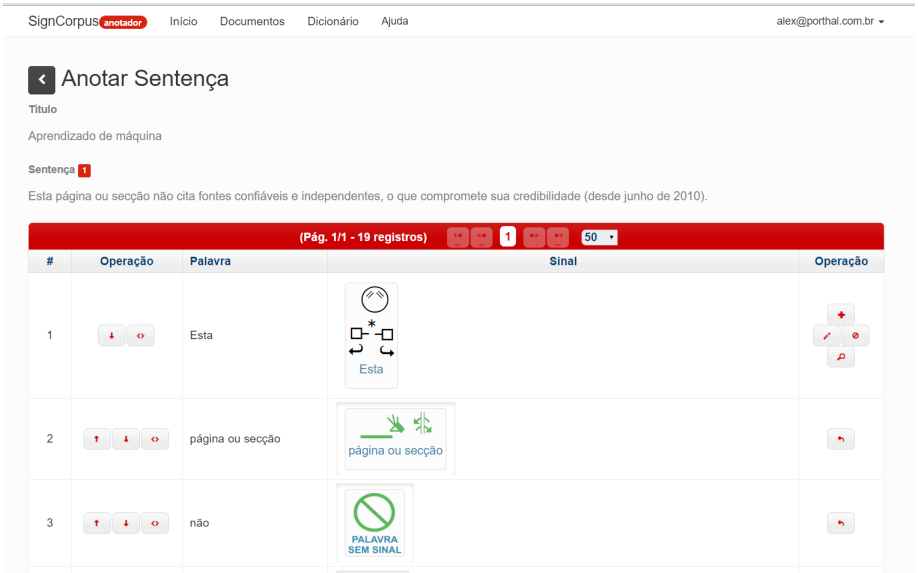
Para anotação, com base nas sugestões de sinais dadas pelo sistema, o usuário pode apenas clicar no sinal e assim realizar a anotação do mesmo, ou realizar a inclusão

Figura 28 – Processo de busca por sinais.



manual do sinal, o qual será apresentado nos incrementos seguintes. Conforme apresenta a Figura 29, o sinal em preto significa que são sugestões do sistema, sinal em verde significa que foi anotado na palavra, e caso a palavra não possui sinal, pode informar que o mesmo não possui sinal, ficando com o símbolo como mostra a palavra "não", além de poder desfazer a anotação a qualquer momento.

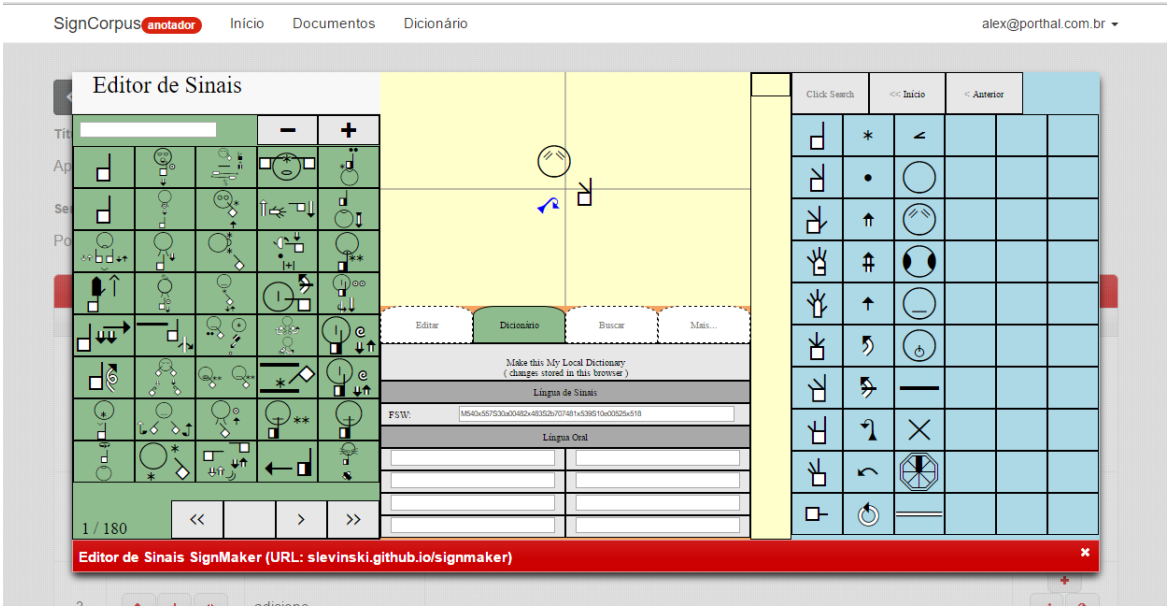
Figura 29 – Página com exemplo de uma palavra anotada e outra sem anotação.



4.4.7 Incremento 7

O incremento 7 ficou designado a integração de um editor de sinais. Para isso, integramos o editor *SignMaker*, que permite a criação de qualquer sinal e gera automaticamente o FSW do mesmo, o qual é necessário para o nosso sistema interpretá-lo, além de que seu código fonte está sob a licença MIT, o que permite seu uso nesse trabalho. O *SignMaker* é desenvolvido em JavaScript, e assim permite uma fácil integração com diversos tipos de sistema com suporte a essa linguagem. Para seu correto funcionando, é necessário realizar o download no site *SignPuddle* dos arquivos do dicionário e alfabeto, na linguagem do contexto da aplicação, e adicionar no diretório de configuração da biblioteca, além de passar como parâmetros o idioma utilizado. A Figura 30 apresenta o modal com o editor integrado ao nosso sistema, com a palavra "Por favor" desenhada, e abaixo seu FSW gerado.

Figura 30 – Modal com o editor SignWriting desenhado o sinal a palavra "Por Favor".



4.4.8 Incremento 8

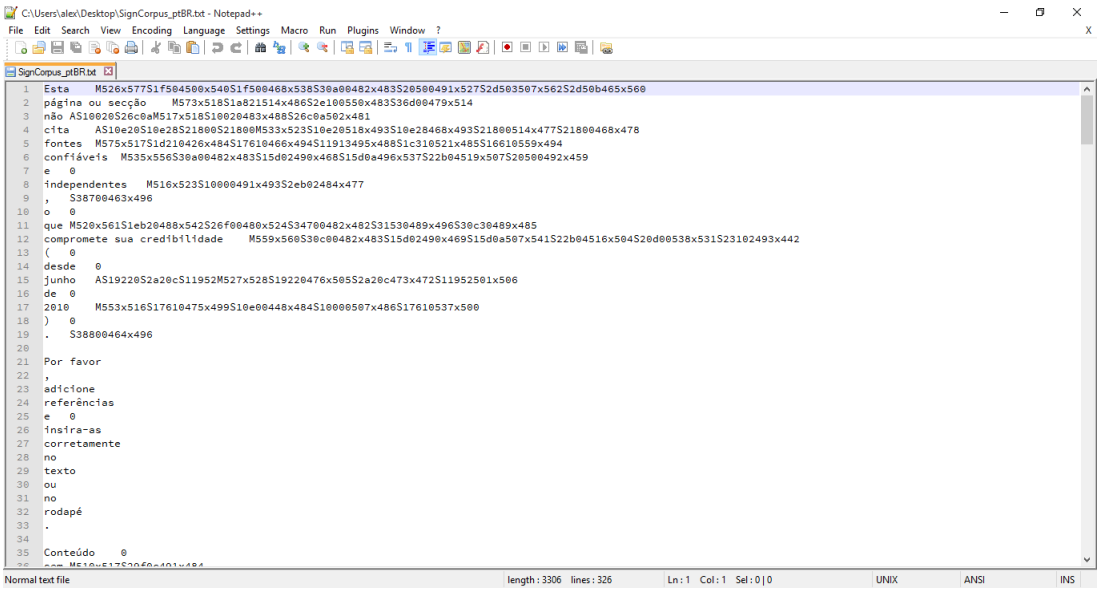
O incremento 8 ficou responsável pela implementação da exportação do córpus paralelo e a internacionalização do sistema, com suporte aos idiomas Inglês e Português. A Figura 31 mostra duas formas de gerar o córpus paralelo. Clicando no botão "Exportar documentos" no ítem 1, gerando o córpus de todos os documentos que estão na etapa de anotação, ou pode exportar apenas um documento que está na etapa de anotação, clicando no botão do ítem 2 correspondente a linha do documento desejado.

A Figura 32 apresenta uma parte do arquivo txt correspondente ao córpus paralelo. Para esse córpus, foi definido alguns padrões que se faz necessário saber para poder fazer

Figura 31 – Tela do sistema para exportação do córpus paralelo.

uso do mesmo. Primeiramente, cada palavra ou conjunto de palavras estarão separadas do FSW por uma tabulação, e caso seja uma palavra composta por exemplo, a mesma será separada por espaço. Por segundo, cada palavra que não possuir sinal estará vinculada ao código zero. Outro ponto importante é saber quando muda de sentença e documento. Com isso, uma linha em branco corresponde que mudou de sentença, e duas linhas em branco consecutivas informa que mudou de documento. Por fim, palavras que não possuem FSW significa que as mesmas ainda não foram anotadas.

Figura 32 – Arquivo do córpus paralelo.



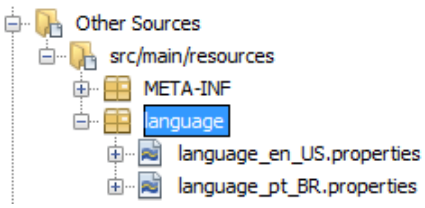
Para realizar a internacionalização do sistema, precisamos realizar algumas configurações no projeto, uma vez que o framework JSF tem suporte para essa funcionalidade, além de criar os arquivos das duas línguas as quais o sistema terá suporte. A primeira

configuração a ser feita é no arquivo faces-config.xml, que será informado quais as línguas que o sistema terá suporte e o local que ficará os arquivos contendo a tradução de cada idioma. A Figura 33 mostra a configuração desse arquivo, o qual a tag locale-config é referente as linguagens que terá suporte, enquanto a tag resource-bundle é referente ao local e nome do arquivo (base-name) e a variável (var) que será usada para leitura das mensagens nos arquivos xhtml. A Figura 34 apresenta os dois arquivos de cada idioma suportado pela aplicação (Inglês e Português), que são arquivos com a extensão properties, o qual é composta pela chave e sua tradução, de cada palavra ou sentença a ser mostrada para o usuário.

Figura 33 – Configuração do arquivo faces-config.xml.

```
<application>
    <locale-config>
        <default-locale>pt_BR</default-locale>
        <supported-locale>pt_BR</supported-locale>
        <supported-locale>en_US</supported-locale>
    </locale-config>

    <resource-bundle>
        <base-name>language.language</base-name>
        <var>language</var>
    </resource-bundle>
</application>
```

Figura 34 – Arquivo de cada idioma suportado pela aplicação.



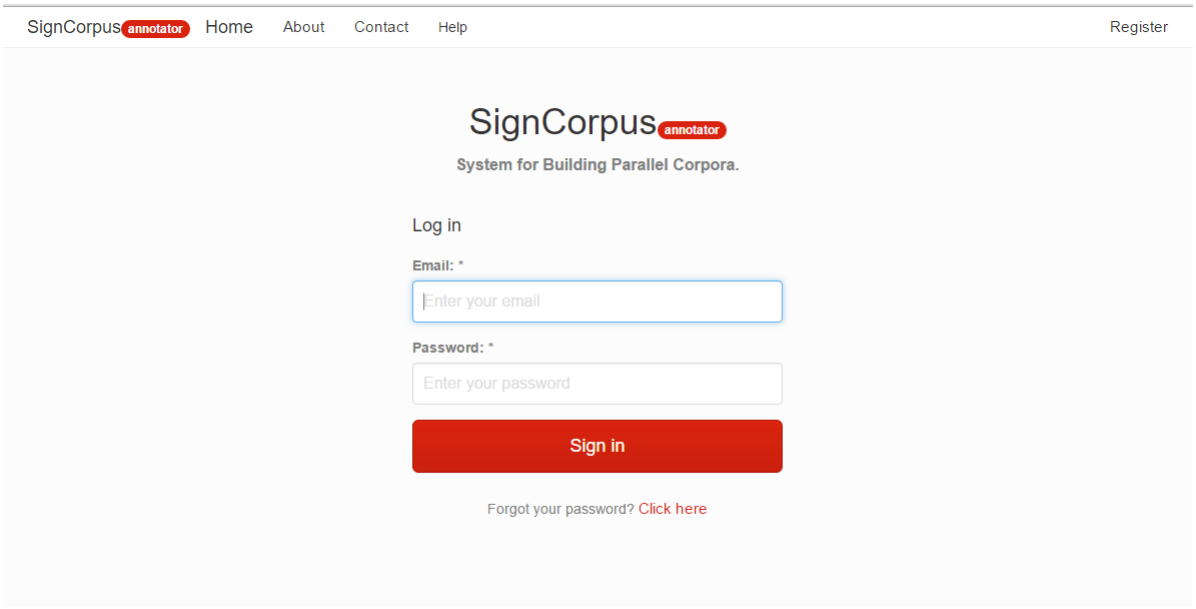
Com essa configuração feita e os arquivos traduzidos para cada idioma, é necessário vincular cada chave das mensagens na *view* e no *managed bean*. Para os arquivos da view é necessário apenas usar o seguinte código language['login.email'], onde language é o nome da variável configurado no arquivo faces-config e login.email é o nome da chave que possui a tradução nos arquivos properties. E para traduzir as mensagens do managed bean, basta só usar o método getMsg, conforme mostra a Figura 35, passando a chave da mensagem como parâmetro do método.

A Figura 36 apresenta a tela inicial da ferramenta no idioma Inglês. O sistema fará a leitura do arquivo que contêm as traduções do idioma correto a ser mostrado conforme a configuração do idioma no navegador.

Figura 35 – Código para leitura das mensagens no managed bean.

```
public class InternationalizationMessage {

    public static String getMsg(String messageId) {
        FacesContext facesContext = FacesContext.getCurrentInstance();
        String msg = "";
        Locale locale = facesContext.getViewRoot().getLocale();
        ResourceBundle bundle = ResourceBundle.getBundle("language.language", locale);
        try {
            msg = bundle.getString(messageId);
        } catch (Exception e) {
        }
        return msg;
    }

}
```

Figura 36 – Tela inicial da ferramenta com o idioma em Inglês.



4.4.9 Incremento 9

O processo de validação da aplicação foi realizado mediante uma avaliação cooperativa executada com duas pessoas especialistas em SW. Na Figura 37 apresentamos as tarefas executada pelos especialistas, ficando o avaliador a tarefa de observar a execução da mesma e realizar a anotação da avaliação. Por fim, obtemos uma melhor visão de como o sistema está quanto a usabilidade e suas funcionalidade, ficando sugerido as seguintes melhorias no sistema: poder realizar a leitura da sentença anotada somente em SW; poder editar um sinal criado; adicionar um dicionário em português para mostrar sinônimos de cada palavra e assim auxiliar o anotador surdo que muitas vezes desconhece o significado de alguma palavra em português; e por último, acrescentar o alfabeto manual e de

números, para facilitar na hora de escrever o sinal em datilologia. Com isso, a avaliação final nos mostrou que o sistema atende seu propósito, pode ser levado à produção, ficando essas sugestões de melhorias para a versão 2.0 do sistema.

Figura 37 – Tarefas realizadas na avaliação cooperativa.

Sistema
Nome: SignCorpus Annotator
URL: http://www.sistemas.porthal.com.br/SignCorpus
Tarefas
Tarefa 1: Como faço para me registrar no sistema?
Tarefa 2: Como faço para entrar no sistema?
Tarefa 3: Como faço para acessar as palavras de um dicionário?
Tarefa 4: Como faço para importar um documento para anotação?
Tarefa 5: Como faço para preparar um documento para anotação?
Tarefa 6: Como faço para anotar uma sentença do documento?
Tarefa 7: Como faço para criar um sinal e vincular na palavra?
Tarefa 8: Como faço para selecionar um sinal para palavra?
Tarefa 9: Como faço para unir duas palavras?
Tarefa 10: Como faço para separar uma palavra composta?
Tarefa 11: Como faço para desfazer anotação?
Tarefa 12: Como faço para buscar mais sinais para a palavra?
Tarefa 13: Como faço para incluir um FSW direto na anotação?
Tarefa 14: Como faço para anotar uma palavra sem sinal?

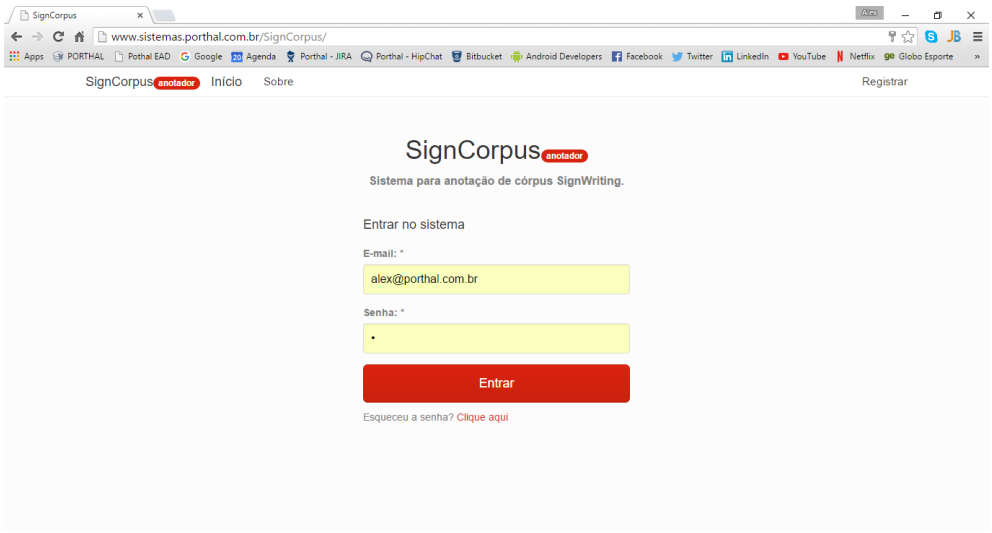
4.4.10 Incremento 10

O último incremento, após o processo de validação, foi disponibilizar o sistema online em modo de produção. Para poder realizar essa tarefa foi necessário a configuração de alguns programas para que o sistema funcionasse perfeitamente. O primeiro programa necessário no servidor é um SGBD, e nesse nosso contexto, o MySql. Com a instalação do MySql pronta, precisamos apenas criar o banco de dados, que após a execução do sistema, o mesmo irá criar automaticamente todas as tabelas e demais artefatos do banco de dados. O segundo passo foi certificar que possui um servidor web instalado, e para a nossa aplicação, precisamos do servidor de aplicação Glassfish 4. Feito isso, bastou fazer o *deploy* do sistema no servidor, e abrir o sistema em um navegador web. A instância online pode ser acessada em <http://www.sistemas.porthal.com.br/SignCorpus/> como mostra na Figura 38.

4.5 Testes

Com o intuito de testar a ferramenta, foram criados 26 casos de testes para todas as funcionalidades do sistema e os mesmos foram instanciados utilizando o navegador Chrome, e a validação da execução desses testes encontram-se disponível no repositório do projeto para comprovação da execução e assim a garantia da qualidade do sistema. Dos 26 casos de testes apenas um não foi executado conforme o previsto, pois apesar de

Figura 38 – Sistema online em modo de produção.



retornar o feedback ao usuário, ele retornava com a mensagem em inglês mesmo quando a aplicação estava executada no idioma Português. Porém, foi considerado aceito pois a mensagem é própria do componente utilizado para leitura de arquivos, e caso selecione um arquivo no formato errado, o componente mesmo gera o *feedback*, o qual não encontramos o lugar para configurar a mensagem em português.

4.6 Contribuições do trabalho

O desenvolvimento dessa ferramenta vem a contribuir de forma significativa com a comunidade surda, uma vez que abre portas para que outras pesquisas possam ser desenvolvidas com a utilização do córpus paralelo. A principal contribuição é a implantação da ferramenta online, permitindo assim que várias pessoas do mundo inteiro possam contribuir com a anotação de um córpus e uso dos mesmos. A segunda contribuição é disponibilizar gratuitamente o código fonte para que outras pessoas possam contribuir com a evolução da ferramenta, estando disponível em <https://bitbucket.org/unipampa/signcorpus>.

Também, vale destacar que submetemos um artigo na conferência LREC 2016 (Language Resources and Evaluation Conference) <http://lrec2016.lrec-conf.org/en/>, sendo esta uma das melhores conferência no segmento que este trabalho está inserido, onde obtêm uma classificação A2 no Qualis, e por fim o artigo encontra-se em processo de avaliação.

5 Discussão final e trabalhos futuros

Como mencionado nos Capítulos anteriores, este trabalho teve como principal objetivo construir uma ferramenta online para anotar manualmente textos em qualquer língua falada com SignWriting (SW) em qualquer língua de sinais, e com este córpus anotado permitir o uso para a criação de ferramentas eficientes para a comunidade surda.

Com o desenvolvimento dessa ferramenta, a qual é considerada até então a primeira ferramenta para anotação de córpus paralelo em SW, estamos contribuindo com toda comunidade surda. Uma vez que um córpus esteja anotado, ele permitirá que novas tecnologias sejam desenvolvidas nesse contexto. Além disso, este trabalho deixa várias contribuições, não somente com a ferramenta em produção e disponibilização dos córpus anotados, mas também a liberação de seu código fonte para que outras pessoas possam contribuir com o mesmo. Inclusive, com intuito de divulgar o trabalho, submetemos um artigo na conferência LREC, considerada umas das melhores conferência no segmento que este trabalho está inserido, que está em processo de avaliação.

Como trabalhos futuros que podem ser desenvolvidos utilizando um córpus paralelo, podemos citar ferramentas para tradução automática da escrita de sinais, jogos para ensino do português e de SW para crianças surdas e até mesmo ferramentas para reconhecimento de sinais a partir de vídeos, o que para esse último seria necessário incluir no córpus mais uma anotação, a do sinal visual.

Referências

ALMASOUD, A. M.; AL-KHALIFA, H. S. Semsignwriting: A proposed semantic system for arabic text-to-signwriting translation. Scientific Research Publishing, 2012. Disponível em: <http://file.scirp.org/Html/9-9301449_21956.htm>. Citado na página 34.

ALMEIDA, S. G. M.; GUIMARÃES, F. G.; RAMÍREZ, J. A. Feature extraction in brazilian sign language recognition based on phonological structure and using rgb-d sensors. *Expert Systems with Applications*, Elsevier, v. 41, n. 16, p. 7259–7271, 2014. Disponível em: <http://ac.els-cdn.com/S0957417414003042/1-s2.0-S0957417414003042-main.pdf?_tid=b1221cb4-f101-11e4-a16d-00000aacb35e&acdnat=1430595209_bf193364ddfc6cb1161c80fbe253fa77>. Citado na página 35.

BARRETO, M.; BARRETO, R. Escrita de sinais sem mistérios. *Ed. do Autor, BH*, 2012. Citado 4 vezes nas páginas 21, 25, 27 e 31.

BISWAS, K.; BASU, S. K. Gesture recognition using microsoft kinect®. In: IEEE. *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*. 2011. p. 100–103. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6144864>. Citado na página 33.

CASELI, H. de M. Indução de léxicos bilíngües e regras para a tradução automática. 2007. Disponível em: <http://www2.dc.ufscar.br/~helenacaseli/pdf/2007/TeseDoutorado.pdf>. Citado na página 32.

CIVIL, P. da R. C. *LEI N.10.436, DE 24 DE ABRIL DE 2002*. 2015. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/2002/l10436.htm>. Citado 2 vezes nas páginas 21 e 23.

ESPINDOLA, L. da S. *Um Estudo sobre Modelos Ocultos de Markov HMM-Hidden Markov Model*. June, 2010. Disponível em: <https://www.inf.pucrs.br/peg/pub/tr/TI1_Luciana.pdf>. Nenhuma citação no texto.

ESTATíSTICA, I. I. B. de Geografia e. *Censo 2010*. 2015. Disponível em: <http://www.censo2010.ibge.gov.br/apps/mapa/>. Citado 2 vezes nas páginas 21 e 23.

GESSER, A. *Libras? que língua é essa?: crenças e preconceitos em torno da língua de sinais e da realidade surda*. [S.l.]: Parábola Ed., 2009. Citado na página 23.

KAR, A.; CHATTERJEE, P. S. A light-weight mobile application of sign writing translator for translating sign writing symbols to simple sentence in english. *Int. J. of Recent Trends in Engineering & Technology*, v. 11, 2014. Disponível em: <http://searchdl.org/public/journals/2014/IJRTET/11/2/1006.pdf>. Citado na página 35.

KISHORE, P.; KUMAR, P. R. Segment, track, extract, recognize and convert sign language videos to voice/text. *International Journal of Advanced Computer Science and Applications (IJACSA) ISSN (Print)-2156*, Citeseer, v. 5570, 2012. Disponível em:

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.261.781&rep=rep1&type=pdf#page=45>. Citado na página 34.

LI, K. F. et al. A web-based sign language translator using 3d video processing. In: IEEE. *Network-Based Information Systems (NBiS), 2011 14th International Conference on*. 2011. p. 356–361. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6041939>. Citado na página 33.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007. Disponível em: <http://www.seer.ufrgs.br/index.php/rita/article/view/rita_v14_n2_p43-67/3543>. Nenhuma citação no texto.

MARSLAND, S. *Machine learning: an algorithmic perspective*. [S.l.]: CRC press, 2014. Nenhuma citação no texto.

MELO, F. R. de; MATOS, H. C. de O.; DIAS, E. R. B. Aplicação da métrica bleu para avaliação comparativa dos tradutores automáticos bing tradutor e google tradutor. *Revista e-scrita: Revista do Curso de Letras da UNIABEU*, v. 5, n. 3, p. 33–45, 2015. Disponível em: <http://www.uniabeu.edu.br/publica/index.php/RE/article/view/1719/0>. Nenhuma citação no texto.

NGUYEN, T. D.; RANGANATH, S. Facial expressions in american sign language: Tracking and recognition. *Pattern Recognition*, Elsevier, v. 45, n. 5, p. 1877–1891, 2012. Disponível em: <http://ac.els-cdn.com/S0031320311004559/1-s2.0-S0031320311004559-main.pdf?_tid=25b7d3a2-f134-11e4-b6e2-00000aab0f26&acdnat=1430616880_5509dbb101726bc534b2dcf456b521aa>. Citado na página 34.

OZ, C.; LEU, M. C. American sign language word recognition with a sensory glove using artificial neural networks. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 24, n. 7, p. 1204–1213, 2011. Disponível em: <http://ac.els-cdn.com/S0952197611001230/1-s2.0-S0952197611001230-main.pdf?_tid=5d850db4-f368-11e4-a8a9-00000aab0f26&acdnat=1430859210_c68be0881fdd7ab911aa1547be4cac2e>. Citado na página 33.

PAULRAJ, M. P. et al. A phoneme based sign language recognition system using skin color segmentation. In: IEEE. *Signal Processing and Its Applications (CSPA), 2010 6th International Colloquium on*. 2010. p. 1–5. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5545253>. Citado 2 vezes nas páginas 21 e 23.

PORFIRIO, A. J. et al. Libras sign language hand configuration recognition based on 3d meshes. In: IEEE. *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. 2013. p. 1588–1593. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6722027>. Citado 2 vezes nas páginas 25 e 34.

QUADROS, R. M. de; KARNOPP, L. B. *Língua de sinais brasileira: estudos lingüísticos*. [S.l.]: Artmed, 2007. Citado na página 24.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A. *Redes Neurais artificiais para engenharia e ciências aplicadas curso prático*. [S.l.]: SciELO Brasil, 2010. Nenhuma citação no texto.

SLEVINSKI, S. *The SignPuddle Standard for SignWriting Text*. 2015. Disponível em: <http://signpuddle.net/download/draft-slevinski-signwriting-text-03.html>. Citado na página 29.

SOMMERVILLE, I. Engenharia de software-8ª edição. *Ed Person Education*, 2007. Citado na página 39.

STUMPF, M. R. Aprendizagem de escrita de língua de sinais pelo sistema signwriting: língua de sinais no papel e no computador. 2005. Disponível em: <http://www.lume.ufrgs.br/handle/10183/5429>. Citado 3 vezes nas páginas 26, 27 e 31.

SUN, C.; ZHANG, T.; XU, C. Latent support vector machine modeling for sign language recognition with kinect. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 6, n. 2, p. 20, 2015. Disponível em: <http://dl.acm.org/citation.cfm?id=2753829.2629481&coll=DL&dl=ACM&CFID=510167635&CFTOKEN=70823312>. Citado na página 36.

WANG, H. et al. Depth sensor assisted real-time gesture recognition for interactive presentation. *Journal of Visual Communication and Image Representation*, Elsevier, v. 24, n. 8, p. 1458–1468, 2013. Disponível em: <http://ac.els-cdn.com/S1047320313001843/1-s2.0-S1047320313001843-main.pdf?_tid=2c69ceae-f101-11e4-8dc2-00000aacb361&acdnat=1430594987_f9a571397e324665825a96ecd06b0858>. Citado na página 35.

WEB, R. B. de. *Aprendizado de Máquina: Aprendizado Supervisionado*. 2015. Disponível em: <http://www.revistabw.com.br/revistabw/aprendizagem-de-maquina-aprendizado-supervisionado/>. Nenhuma citação no texto.

YANG, H.-D.; LEE, S.-W. Robust sign language recognition by combining manual and non-manual features based on conditional random field and support vector machine. *Pattern Recognition Letters*, Elsevier, v. 34, n. 16, p. 2051–2056, 2013. Disponível em: <http://ac.els-cdn.com/S0167865513002559/1-s2.0-S0167865513002559-main.pdf?_tid=da8f6a3e-f101-11e4-bcde-00000aacb362&acdnat=1430595279_c5ca05526fedeb8c1925bea47f87319d>. Citado na página 35.

ZAFRULLA, Z. et al. American sign language recognition with the kinect. In: ACM. *Proceedings of the 13th international conference on multimodal interfaces*. 2011. p. 279–286. Disponível em: <http://delivery.acm.org/10.1145/2080000/2070532/p279-zafrulla.pdf?ip=200.132.146.39&id=2070532&acc=ACTIVE%20SERVICE&key=344E943C9DC262BB%2E63D8E28469F48F94%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=670861893&CFTOKEN=23428246&__acm__=1430595183_41f43ae7f346a1d711e9fbdcd7e664ff>. Citado na página 33.

ZAKI, M. M.; SHAHEEN, S. I. Sign language recognition using a combination of new vision based features. *Pattern Recognition Letters*, Elsevier, v. 32, n. 4, p. 572–577, 2011. Disponível em: <http://ac.els-cdn.com/S016786551000379X/1-s2.0-S016786551000379X-main.pdf?_tid=fd153546-f36f-11e4-aeee-00000aacb360&acdnat=1430862484_18a2246d16ba05b5994b6f4c2f2464d0>. Citado na página 34.