João Victor de Mello Fagundes

Uso de Técnicas e Ferramentas de Embedding de Conhecimento para Desambiguação de Anotações Segundo Contextos Semânticos

Brasil

2019

João Victor de Mello Fagundes

Uso de Técnicas e Ferramentas de Embedding de Conhecimento para Desambiguação de Anotações Segundo Contextos Semânticos

Trabalho de Conclusão de Curso submetido ao Curso de Ciências da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.

Universidade Federal de Santa Catarina

Centro Tecnológico - CTC

Departamento de Informática e Estatística

Ciências da Computação

Orientador: Italo Lopes Oliveira

Brasil

2019

João Victor de Mello Fagundes

Uso de Técnicas e Ferramentas de Embedding de Conhecimento para Desambiguação de Anotações Segundo Contextos Semânticos

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de Bacharel em Ciências da Computação, e aprovado em sua forma final pelo Curso de Ciências da Computação da Universidade Federal de Santa Catarina.

Dr. Prof. **Italo Lopes Oliveira**
Orientador

Dr. Prof. **Mauro Roisenberg**
Convidado 1

Dr. Prof. **Alexandre Gonçalves Silva**
Convidado 2

Brasil
2019

Agradecimentos

Não seria justo iniciar esta seção sem agradecer a Deus logo na primeira linha. As oportunidades e condições que me levaram à conclusão deste curso foram, antes de tudo, permitidas por *Ele*. Agradeço também aos meus pais, José Carlos e Maritza, por me manterem no caminho correto e sempre me incentivarem em tudo. Os esforços que vocês fizeram para que eu chegasse até aqui é algo que nunca vou esquecer. Amo vocês.

A meus irmãos Leonardo e Leandro, agradeço pela parceria, ajuda e incentivos durante todo o meu trajeto até aqui. Vocês me ensinaram coisas que até mesmo as milhares de horas-aula deste curso jamais conseguiriam. Vocês, juntamente com minhas duas cunhadas e quatro sobrinhos também têm parte nessa minha conquista. Amo todos vocês.

Agradeço também à minha eterna caloura e namorada Fernanda. Meu amor, mesmo aparecendo na minha vida a apenas um ano, você me ajudou muito nessa conquista. Foram minhas escolhas durante o curso, com uma ajudinha do destino, que me levaram até você, e por isso tudo sou muito grato. Eu te amo muito, e espero conseguir retribuir tudo isso na sua formatura em breve, afinal de contas, veteranos também servem pra isso.

Ao meu orientador Italo e professor responsável Renato Fileto, fica o meu muito obrigado por toda a ajuda durante a escrita deste trabalho. Um TCC é assustador para qualquer universitário, e vocês me deram condições de finalizar esta etapa.

Aos meus supervisores de estágio, colegas de curso e também colegas de profissão agradeço pelos ensinamentos sobre os conhecimentos que o curso não cobre. Vocês me tornaram um profissional da área da computação ainda melhor e portanto também fazem parte dessa trajetória.

A todos que não se sentiram contemplados acima mas que de maneira direta ou indireta me ajudaram a ser quem sou hoje, também agradeço do fundo do coração.

Pesquisa desenvolvida com utilização dos recursos computacionais do Centro de Ciências Matemáticas Aplicadas à Indústria (CeMEAI) financiados pela FAPESP.

Mar calmo nunca fez bom marinheiro.

Resumo

Anotações semânticas permitem associar a dados ou porções de dados não-estuturados (e.g., menções relevantes em textos) recursos com semântica bem definida em bases de conhecimento (e.g., DBpedia, Wordnet, Babelnet) que ajudam a explicar a que os dados anotados se referem. Tais anotações permitem melhor explorar os dados anotados em uma miríade de domínios e aplicações, incluindo comércio, *marketing*, turismo, segurança pública, entre outras. Todavia, ao tentar capturar a semântica de dados, como, por exemplo, postagens em mídias sociais, aplicações de enriquecimento semântico esbarram em problemas como o uso de gírias e regionalismos, e principalmente, a ocorrência de ambiguidade. Várias técnicas ao longo dos anos tentam desambiguar menções a entidades do mundo real em textos para efetuar anotações semânticas corretas. Este trabalho consiste em realizar um estudo do estado da arte do problema da ambiguidade de palavras, tendo como objetivo o domínio das técnicas e ferramentas disponíveis atualmente e que vêm sendo utilizadas na solução deste problema, como por exemplo os *embeddings* de palavras e conhecimento. A partir disso, pretende-se desambiguar as menções existentes em postagens de mídias sociais com o auxílio dos *embeddings* em conjunto com redes neurais. Para tal, são identificadas e selecionadas implementações existentes de *embeddings*. Contextos semânticos são capturados, representados e explorados nesses *embeddings* para a desambiguação de anotações. A abordagem proposta é avaliada na melhoria da precisão de anotações de conjuntos de *tweets*, como por exemplo o *dataset Microposts*, que fornece conjuntos de *tweets* anuais.

Palavras-chaves: Anotações Semânticas, Desambiguação, Contextos Semânticos, *Embeddings* de Conhecimento, Grafos de Conhecimento, Redes neurais

Abstract

Semantic annotations allow to link unstructured data (e.g., relevant references in documents) to resources with well-founded semantics in knowledge information bases (e.g., DBPedia, Wordnet, Babelnet, Google Knowledge Graph) that help to explain what those annotated data are referring to. Such annotations enable better exploitation of the annotated data in a myriad of domains and applications, including marketing, tourism, and public security, among many others. However, by trying to capture the semantics in data, such as social media posts, the computer may face problems, such as the common use of slangs and regionalisms in informal conversation, and mainly the occurrence of ambiguity. Many techniques over the years try to disambiguate mentions to real world entities in documents to produce correct semantic annotations. This work consists in an study of the state of the art of the word ambiguity problem, having as objective the mastery of the techniques and tools available that are being used to solve this problem, such as knowledge and word embedding. From this, it is intended to disambiguate mentions present in social media contents by using embeddings together with neural networks. For such, implementations of embeddings are identified and selected. Semantic contexts are captured, represented and explored in these embeddings for annotation disambiguation. The proposed approach is evaluated in the improvement of the precision of the annotations from a set of tweets, such as the Microposts dataset, which provide annual sets of tweets.

Key-words: Semantic Annotation, Disambiguation, Semantic Contexts, Knowledge Embedding, Knowledge Graphs, Neural networks

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

KG	Knowledge Graph
KB	Knowledge Base
KE	Knowledge Embedding
EL	Entity Linking
IA	Inteligência Artificial
LD	Linked Data
NLP	Natural Language Processing
RNN	Recurrent Neural Network
LISA	Laboratório de Integração de Sistemas e Aplicações
LSTM	Long Short-Term Memory

Sumário

1 Introdução e objetivos

1.1 Introdução

Com o avançar da tecnologia, a quantidade de dados produzida a cada dia só aumenta. Mídias sociais (e.g., Facebook, Twitter, Instagram) e bibliotecas digitais estão entre as fontes de dados mais utilizadas atualmente. Os dados nelas disponíveis podem ser úteis para uma variedade de aplicações. Todavia, documentos texto e postagens em mídias sociais são dados não estruturados e costumam apresentar problemas como ambiguidades, tornando sua semântica difícil de capturar e tratar computacionalmente (LANDEGHEM, 2016).

Anotações semânticas são uma alternativa para contornar tais problemas. Uma anotação semântica associa uma porção de dado (e.g. uma menção a uma entidade em um texto) a uma descrição formal e processável por máquina daquilo a que se refere (e.g. recurso de grafo de conhecimento que descreve tal entidade). Por exemplo, uma menção à entidade nomeada **Michael Jordan** em um texto (e.g., artigo de revista, conteúdo textual de um *tweet*) pode ser ligada, por meio de uma anotação semântica, a um nodo de grafo de conhecimento (e.g. DBpedia[1]) (LEHMANN et al., 2015) representando o jogador de *basquete*, famoso por ter jogado no *Chicago Bulls* e ser 6 vezes campeão da *NBA*.

Entretanto, adicionar anotações semânticas a porções de textos, principalmente os provenientes de mídias sociais, não é uma tarefa trivial. Esses textos podem ter uma quantidade considerável de informalidade, com utilização de gírias ou regionalismos. Além disso, o significado de cada palavra em um texto depende do autor e de todo o contexto e da palavra dentro deste texto. Por exemplo, ao utilizar a palavra *apple* em um texto, o autor pode se referir à fruta ou à empresa de tecnologia. Esse é o problema conhecido como *ambiguidade* de palavras (LANDEGHEM, 2016), característica intrínseca em qualquer língua natural.

Uma popular rede social com uma variedade de assuntos é o *Twitter*. Essa rede social de *microblogging* tem uma média de 330 milhões de usuários ativos por mês e gera um total de 500 milhões de *tweets* por dia (OMNICORE, 2018). Ao obter um conjunto de *tweets* podemos nos deparar com dados escritos das mais diversas maneiras. Caso uma ferramenta computacional decida por sempre escolher, por exemplo, o significado estatisticamente mais provável para cada palavra, tal abordagem pode gerar anotações imprecisas.

Uma maneira intuitiva utilizada pelos seres humanos para resolver ambiguidades

[1] https://wiki.dbpedia.org/

durante uma conversa é analisar o contexto onde cada palavra está inserida, para então atribuir a ela o significado correto. Visando aumentar a precisão das anotações existentes e das novas que serão geradas, a ideia deste trabalho portanto é desenvolver uma ferramenta que durante o processo de anotações semânticas aumente o escopo de análise. Ou seja, utilizar as outras palavras presentes na mesma frase, ou em frases próximas, para desambiguar as menções à entidades do mundo real (entidades nomeadas), de modo que o significado delas auxilie a desambiguação, fornecendo um contexto semântico adequado.

Várias técnicas vêm sendo usadas ao longo dos anos para auxiliar o processo de desambiguação de porções de dados não-estruturados. Dentre tais técnicas, técnicas baseadas em *embeddings* têm se desenvolvido rapidamente e gerado bons resultados para uma variedade de problemas relacionados à semântica de dados. *Embeddings*, considerando o problema de anotação semântica de dados textuais, têm a função de mapear objetos discretos em vetores de números reais. A técnica se divide algumas vertentes, que se diferenciam no tipo de objeto mapeado. Este trabalho estuda duas dessas vertentes: os *embeddings* de palavra (MIKOLOV et al., 2013), que trabalham com palavras presentes em textos e documentos; e os de conhecimento (WANG et al., 2017), que mapeam triplas de um KG. Pelo fato de ambas as vertentes apresentarem um crescimento considerável em importância, este trabalho se aprofunda no conhecimento e domínio de ambas, mais precisamente no uso da biblioteca chamada *fastText*[2].

Abordagens atuais focam na utilização de *embeddings* de palavra e de entidades (uma variante de embeddings de palavras/documentos) com redes neurais profundas. No entanto, a utilização de *embeddings* de conhecimento em conjunto com os de palavras em redes neurais profundas ainda não foi devidamente analisado.

1.2 Justificativa

Anotação semântica de dados textuais oriundos de mídias sociais ainda sofre por enfrentar alguns problemas, sendo um dos principais deles, e foco deste trabalho, a ambiguidade de palavras. Informações relevantes e aproveitáveis podem ser extraídas de postagens de mídias sociais semanticamente enriquecidas. Por exemplo, inferir índices de aprovação e rejeição de políticas, analisar a popularidade de uma entidade (e.g. pessoa, instituição, produto, marca) em algum setor da sociedade, entre outros.

1.3 Objetivo geral

O objetivo geral deste trabalho é a aplicação de embeddings de conhecimento e palavras, gerados pela ferramenta *FastText*, em redes neurais profundas, visando obter re-

[2] https://fasttext.cc/

sultados melhores que as abordagens existentes de desambiguação de entidades nomeadas em postagens de mídias sociais.

1.4 Objetivos específicos

1. **Compreender o estado da arte:** estudar e conhecer as principais técnicas sendo utilizadas atualmente para desambiguar entidades nomeadas e que utilizem *embeddings* de conhecimento ou palavras. O intuito deste estudo é se habilitar para realizar experimentos no âmbito deste TCC e identificar oportunidades de contribuição científica para empreendimentos futuros, tais como um eventual mestrado;

2. **Estudar abordagens:** conhecer e compreender as abordagens existentes na literatura e identificar pontos fracos e fortes de cada uma;

3. **Escolher alternativas:** baseado no estudo anterior, selecionar qual abordagem adequa melhor ao objetivo geral desta proposta;

4. **Implementar**: usando uma ferramenta escolhida, propor e implementar uma alternativa que trata especificamente da resolução o problema de desambiguação de menções em texto;

5. **Realizar experimentos e documentar os resultados obtidos:** Após a implementação da ferramenta e a produção de resultados, realizar um estudo identificando se as alternativas de desambiguação de menções a entidades que foram estudadas, escolhidas e implementadas de fato melhoraram a precisão das anotações semânticas.

1.5 Metodologia de pesquisa

A metodologia adotada consiste primeiramente de uma revisão bibliográfica de trabalhos publicados, com o objetivo de conhecer as técnicas e ferramentas existentes na atualidade. A partir disso, é estabelecido domínio do problema em questão. Por fim, utilizar o que foi aprendido para se apoiar nas técnicas existentes estudadas em busca de uma solução eficiente e funcional para a desambiguação de menções nomeadas em postagens de mídias sociais.

A princípio, é necessário ter domínio das principais abordagens de desambiguação de menções a entidades nomeadas. Uma tendência observada em abordagens recentes é a utilização de redes neurais profundas com *embeddings* de palavras e *embeddings* de entidades. No entanto, há poucas abordagens que utilizem *embeddings* de conhecimento. Por isso, alguns artigos sobre o *FastText* (BOJANOWSKI et al., 2017; JOULIN et al., 2017)

foram estudados para melhorar o entendimento sobre a ferramenta e, como consequência, definir de maneira mais adequada os parâmetros utilizados para gerar os *embeddings*.

Uma rede neural profunda simples foi proposta e implementada a fim de averiguar a eficiência do uso de *embeddings* de palavras e conhecimento na desambiguação de entidades nomeadas em postagens de mídias sociais. Para verificar a eficiência da abordagem proposta em comparação a outras abordagens da literatura, foi utilizado o sistema de *benchmark* GERBIL[3] (RÖDER; USBECK; NGOMO, 2018). O sistema GERBIL mede a eficiência de uma solução para tarefa relacionada a *Entity Linking (EL)* através de medidas como precisão, cobertura e *F1 Score*. Neste trabalho, a principal medida de comparação é o *F1 Score*.

1.6 Estrutura do documento

O Capítulo 2 apresenta os conceitos relevantes ao trabalho para melhor compreensão do leitor. O Capítulo 3 discute trabalhos relacionados e o principal motivo que torna esta proposta interessante e inovadora. A seguir, o Capítulo 4 foca na proposta do trabalho, e discute mais a fundo os detalhes de implementação e experimentos a serem realizados. Após isso, o Capítulo 5 mostra os resultados dos experimentos realizados no trabalho, tendo como base o que foi dito no Capítulo 4. Por fim, o capítulo 6 apresenta as conclusões do trabalho, bem como ideias para trabalhos futuros.

[3] http://gerbil.aksw.org/gerbil/

2 Fundamentação teórica

Este capítulo apresenta os conceitos básicos para o entedimento deste trabalho. São apresentados os conceitos de grafos de conhecimento, *embeddings* e redes neurais profundas, procurando discutir como esses artefatos podem ser usados para desambiguação de entidades nomeadas.

2.1 Grafos de Conhecimento (KG)

Grafos de Conhecimento (KG - *Knowledge graph*) formalizam conhecimento na forma de recursos com conceitos ou entidades nomeadas (representados como nodos do grafo) e relações semânticas entre eles (representadas como arestas), sendo aderente ao conceito de dados ligados (*Linked Data*). Um KG pode ser armazenado como um conjunto de triplas $<h,\ r,\ t>$, onde h e t representam recursos (e.g., entidades, conceitos), origem e destino respectivamente, e r representa uma relação entre esses recursos (NGUYEN, 2017). Uma tripla é também chamada de fato, pois, usualmente, representa relações do mundo real. Uma outra notação que pode ser encontrada na literatura para descrever um fato é x_{ijk}, sendo i a origem, j o destino e k a relação. Entretando, esta notação não será utilizada neste trabalho.

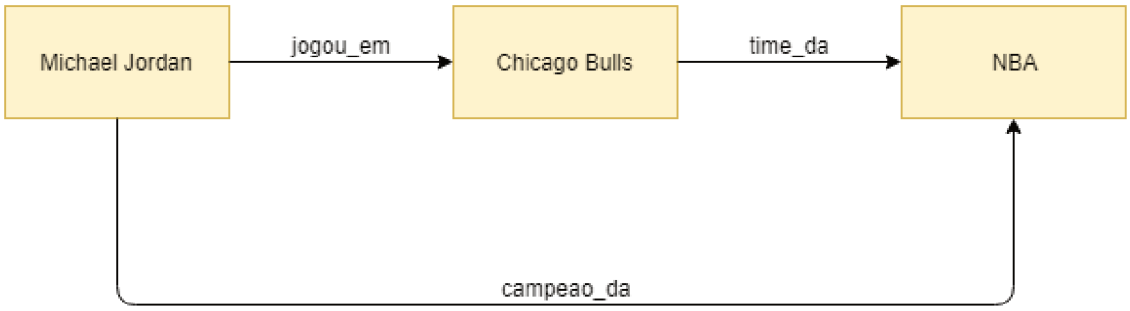

Figura 1 – Exemplo de nodos e relações de grafo de conhecimento.

A Figura 1 representa o exemplo de KG descrito na introdução, que trata do jogador de basquete Michael Jordan. Neste exemplo, a entidade nomeada *Michael Jordan* está relacionada com a entidade nomeada *Chicago Bulls*, já que *Jordan* jogou basquete no time de *Chicago*, que por sua vez está relacionado à *NBA*. Vale ressaltar que ao tratar de relações, cria-se a necessidade de um direcionamento no grafo, para determinar qual entidade faz o papel de domínio (origem) e qual faz o papel de imagem (destino).

Os KG vêm em uma crescente de popularidade recentemente, principalmente por serem utilizados e explorados em pesquisas relacionadas à Web Semântica (BERNERS-LEE; HENDLER; LASSILA, 2001). Apesar da tecnologia existir a mais tempo, foi após o *Google Knowledge Graph* ser divulgado, em 2012, que essa fama ganhou mais força. Ao apresentar o seu novo método de conduzir buscas, o conceito de Grafo de Conhecimento foi resumido em "*things, not strings*". Ou seja, "um grafo que compreende entidades do mundo real e suas relações umas com as outras" (SINGHAL, 2012).

Por estar em foco nas pesquisas na área de ciência de dados, diversas definições de grafo de conhecimento foram surgindo. Apesar do aprofundamento nos estudos sobre KG, não há uma convergência clara entre as inúmeras definições, com algumas delas sendo suficientemente genéricas, podendo ser utilizadas, inclusive, para representar ontologias (EHRLINGER; WÖSS, 2016).

Um KG amplamente utilizado e conhecido é a DBpedia. O grafo é composto principalmente por dados provenientes dos artigos da Wikipedia em mais de 111 idiomas. Apenas a versão inglês da Wikipedia é responsável por mais de 400 milhões de fatos que descrevem 3,7 milhões de entidades (LEHMANN et al., 2015).

2.2 Anotações Semânticas

Outro conceito importante para entendimento no desenvolvimento deste trabalho é o de anotações semânticas. Conforme mencionado, são elas que vão nos ajudar no processo de desambiguação de palavras e permitir representar os resultados de tal processo. O termo "anotação", de uma maneira geral, significa anexar dados a outros dados (OREN et al., 2006). Uma anotação semântica então liga o dado a ser anotado (e.g. menção em um texto) a uma definição daquilo a que tal dado se refere (e.g., conceito ou entidade em um grafo de conhecimento). Anotar semanticamente é o processo de anexar informações formalizadas em uma base de conhecimento tal como um KG. Quando um trecho de texto recebe uma anotação semântica, sua semântica se torna processável por computadores (ONTOTEXT, 2018).

Um paralelo do conceito de anotação no mundo real pode ser feito ao analisar o comportamento humano de rabiscar seus livros e documentos com palavras que facilitem o seu entendimento. Muitas vezes, os autores não conseguem passar a mensagem desejada ao leitor, já que por serem duas pessoas diferentes, eles pode ter maneiras distintas de pensar e interligar ideias. Por isso, muitas vezes o leitor pode escrever com suas próprias palavras o significado de uma frase ou palavra como forma de anotação no texto, para evitar uma pesquisa mais extensa e demorada toda vez que esbarrar naquele trecho de texto.

Realizar uma anotação em um texto ou documento significa demarcar um trecho

do texto (e.g., uma palavra, o nome de uma pessoa, um local) e ligá-lo a informação adicional. Anotar semanticamente é análogo, mas com uma diferença fundamental, o valor da anotação precisa ser tomado de uma base de conhecimento (e.g. grafo de conhecimento) onde as definições das coisas e as relações entre elas são formalmente descritas e processáveis por máquinas.

O uso de anotações semânticas no escopo desse trabalho é de suma importância no que diz respeito à desambiguação de palavras. É necessário ter uma base de conhecimento com informações consistentes, para que possam ser geradas anotações que apontam para as entidades corretas, a fim de desambiguar o texto a ser analisado. No exemplo apresentado anteriormente, que trata do ex-jogador de basquete Michael Jordan, uma anotação incorreta poderia apontar a palavra *Jordan* para o recurso que representa o país Jordânia (Jordan em inglês), desambiguando incorretamente esta palavra.

2.3 Embeddings de Palavra e Conhecimento

No que se refere à motivação deste trabalho, o problema de desambiguar entidades nomeadas em mídias sociais já foi explorado de diversas maneiras em pesquisas anteriores nos últimos anos, porém sem atingir resultados satisfatórios (GONGQING; YING; XUEGANG, 2018; DERCZYNSKI et al., 2014). Dentre as técnicas utilizadas, a *embedding* é uma das mais exploradas atualmente. Um *embedding* é o mapeamento de objetos discretos, como palavras, para vetores de números reais (TENSORFLOW, 2018), que condensam propriedades e relações relevantes desses objetos em espaços usualmente de dimensionalidade bem mais baixa que modelos vetoriais para vocabulários com milhares de palavras.

A técnica de *embeddings*, no entanto, possui uma classificação que a divide em duas outras técnicas mais específicas: os *embeddings* de palavra e os *embeddings* de conhecimento. A diferença entre as duas está no tipo de dado a ser mapeado, pois enquanto a primeira técnica mapeia palavras ou frases para vetores de números reais, a segunda trata não apenas disso, mas de triplas presentes em grafos de conhecimento.

Foi através do *word2vec* (MIKOLOV et al., 2013) que os *embeddings* de palavras começaram a receber uma atenção especial de pesquisadores, principalmente na área de Processamento de Linguagens Naturais (NLP). Isso se deve ao fato de *embeddings* de palavras terem mostrado uma melhora significativa ao realizar algumas tarefas de NLP, como análise de sentimentos (SOCHER et al., 2013).

Embeddings de palavra são importantes também para o aprendizado de máquina. Entradas para algoritmos de aprendizado de máquina podem utilizar palavras de um texto, que não possuem uma representação vetorial natural. A técnica de *embeddings* é uma maneira comum e efetiva de transformar objetos discretos em vetores de núme-

ros reais (TENSORFLOW, 2018). Juntando os vetores de todas as palavras analisadas, obtém-se uma matriz, que pode ter seus valores alterados baseado no conjunto de dados utilizados para treinar o modelo de *embedding*. Apesar dos avanços destacados, os *embeddings* de palavras não se mostraram tão satisfatórios ao enfrentar o problema de desambiguação de entidades nomeadas em postagens de mídias sociais (DERCZYNSKI et al., 2014).

O baixo desempenho dos *embeddings* de palavras ao tratar da desambiguação de entidades nomeadas se deve ao fato de que ao mapear as palavras para vetores, acaba-se misturando algumas características importantes da mensagem, como, por exemplo, os contextos em que a palavra aparece com significados distintos (LANDEGHEM, 2016). Portanto, a precisão do modelo depende muito do conjunto de dados utilizado para treiná-lo. Dados de experimentos de uma fonte diferente da utilizada para dados de treinamento podem acarretar em uma precisão baixa, como por exemplo tentar desambiguar artigos científicos usando um modelo treinado para desambiguar *tweets*, ou vice-versa. O baixo desempenho nesse caso seria atribuído ao fato dos conteúdos possuírem muitas diferenças quanto a maneira de escrita e tipo de linguagem utilizada.

Algumas técnicas que consistem em capturar outras informações adicionais, como, por exemplo, conhecimento prévio proveniente de bases de conhecimento, o que nos leva aos *embeddings* de conhecimento. Os *embeddings de conhecimento*, também chamados de *embeddings* de grafos de conhecimento, se aproveitam das características de um KG para realizar o mapeamento de recursos e suas relações em um KG para vetores, com mais precisão quanto a semântica do que *embeddings* de palavras por guardar informações de como a entidade se relaciona no mundo real. Isto é, não analisa uma palavra, mas sim a que (e.g. entidade, conceito) ela se refere, bem como suas relações com outros recursos descritos no KG. Apenas considerando isso, alguns trabalhos da literatura observaram uma melhora nos resultados ao inserir outro tipo de *embedding* nos seus experimentos (SEVGILI; PANCHENKO; BIEMANN, 2019).

2.4 Redes Neurais Long Short-Term Memory

As redes neurais também estão se tornando cada vez mais populares na ciência de dados. Sua motivação surgiu com avanços no campo de Inteligência Artificial (IA), suprindo a necessidade de um sistema computacional que simula os neurônios de um cérebro humano, sendo representada por diversos nós interconectados através de camadas.

Uma rede neural simples possui três camadas, sendo a primeira chamada de camada de entrada e a última a camada de saída. Entre elas, existe uma ou mais camadas que podem ser denominadas ocultas. São as camadas ocultas que têm seus nós ativados e estimulados de acordo com a entrada da rede neural, simulando o funcionamento dos

neurônios do cérebro humano e produzindo um resultado na saída (FUKUSHIMA, 1975).

Entretanto, tipicamente é difícil prever o quanto um raciocínio sobre eventos anteriores influencia os eventos futuros. Como este trabalho se propõe a desambiguar menções presentes em *tweets* a partir de seus contextos, uma rede neural simples não seria adequada por falhar em levar o contexto em consideração. Visando resolver este problema, as redes neurais recorrentes (RNN) implementam um laço em suas camadas ocultas onde aprendizados obtidos em camadas mais avançadas servem de estímulo para camadas anteriores (RUMELHART; HINTON; WILLIAMS, 1988).

Na teoria, uma RNN resolve o problema de utilizar o contexto de um *tweet* na sua desambiguação, mas na prática alguns problemas foram reportados ao utilizar RNNs quando a recorrência possuía um longo intervalo (BENGIO; SIMARD; FRASCONI, 1994).

As redes Long Short-Term Memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) são redes neurais especializadas em guardar informações no longo prazo, e desde sua concepção elas têm encontrado espaço nas mais variadas áreas de estudo da ciência de dados e inteligência artificial, como reconhecimento de escrita (SCHMIDHUBER, 2014; GRAVES; SCHMIDHUBER, 2009) e reconhecimento de fala (HANNUN et al., 2014; SAK et al., 2015).

3 Trabalhos Relacionados

Este capítulo apresenta e analisa os trabalhos que abordam o problema de desambiguação de entidades nomeadas, focando, principalmente, em trabalhos que utilizem *embeddings* ou que tratam de postagem em mídias sociais. A Tabela 1 apresenta os trabalhos levantados na revisão bibliográfica. As colunas representam, respectivamente, os trabalhos em questão (incluindo a proposta deste trabalho), as fontes de dados utilizadas, o tipo de dado anotado e o método utilizado na desambiguação.

Tabela 1 – Comparação entre os trabalhos relacionados

Trabalho	Entradas	Dado anotado	Método
(GUO et al., 2013)	Wikipedia	Texto informal (Microblog)	Graph-based Microblog Entity Linking (GMEL)
(FANG; CHANG, 2014)	Wikipedia	Texto informal (Microblog)	Spatiotemporal and End-to-end Entity Linking
(DERCZYNSKI et al., 2014)	DBpedia	Texto informal (Microblog)	Named Entity Recognition (NER)
(LI et al., 2016)	Linkless Wikipedia	Texto formal	Gibbs Sampling
(MORENO et al., 2017)	Wikipedia	Texto formal	Binary classifiers (Word Embedding, Entity Embedding)
(CHEN et al., 2017)	Wikipedia, Freebase	Texto formal	Pairwise boosting regression tree (Word Embedding, Entity Embedding)
(LE; TITOV, 2018)	Wikipedia, Yago	Texto formal	Conditional random field, loopy belief propagation (Word Embedding, Entity Embedding)
(KOLITSAS; GANEA; HOFMANN, 2018)	Wikipedia	Texto formal	Shallow FFNN and LSTM (Word Embedding, Entity Embedding)
(ZHU; IGLESIAS, 2018)	DBpedia	Texto formal	*Embedding* de Palavras
(MARTINS; MARINHO; MARTINS, 2019)	Wikipedia	Texto formal	*Embeddings* de Palavras e de Entidades
(SEVGILI; PANCHENKO; BIEMANN, 2019)	Wikipedia, DBpedia	Texto formal	*Embeddings* de Palavras e de Conhecimento e Redes Neurais
(WANG; IWAIHARA, 2019)	Wikipedia	Texto formal	*Embedding* de Palavras e Redes Neurais
(PARRAVICINI et al., 2019)	DBpedia	Texto formal	*Embedding* de Conhecimento
(HAN et al., 2019)	Wikipedia	Texto informal (Microblog)	Stanford CoreNLP
(HOSSEINI et al., 2019)	Wikipedia	Texto informal (Microblog)	*Embedding* de Entidades
Este Trabalho	Wikipedia, DBpedia	Texto informal (Microblog)	*Embeddings* de Palavras e de Conhecimento e Redes Neurais

Os trabalhos estão organizados por ordem cronológica e são analisados através das colunas das tabelas. Trabalhos recentes mais comumente utilizam a *Wikipedia* como fonte de dados, enquanto poucos deles utilizam a *DBpedia*. Dentre os benefícios da Wikipedia, podemos observar que descrições presentes em suas páginas servem para dar um contexto ao desambiguar palavras (WANG; IWAIHARA, 2019). A Wikipedia também é usada em (MARTINS; MARINHO; MARTINS, 2019) para calcular probabilidades a priori de correlações entre duas entidades.

É inegável que a Wikipedia é considerada uma boa fonte de dados para a tarefa de desambiguação de menções, principalmente por estar em constante atualização pela

própria comunidade. Todavia, espera-se que a DBpedia traga resultados mais satisfatórios, uma vez que traz os mesmos dados presentes na Wikipedia mas com a vantagem de reorganizá-los para o formato de triplas, permitindo a representação de entidades com semântica precisa e em formato padronizado (padrão RDF).

Além disso, como já apresentado no Capítulo 2, o formato de triplas permite que as entidades da DBpedia sejam representadas como *embeddings* de conhecimento. Dessa forma, este trabalho utiliza as duas fontes em momentos separados: a Wikipedia continua sendo utilizada, para geração de *embeddings* de palavras, enquanto a DBpedia gera os de conhecimento. Mais detalhes sobre este processo são descritos no Capítulo 4

Quanto ao tipo de dado anotado, embora a maioria dos trabalhos mencionados tenha utilizado textos formais, este trabalho optou por textos informais, mais precisamente *Tweets*. Essa escolha é justificada pelo fato de que abordagens que tratam de textos formais já são relativamente maduras, atingindo valores F1 acima de 90% (LIU et al., 2019; PARRAVICINI et al., 2019).

Alguns trabalhos serviram de inspiração para a implementação do método de desambiguação escolhido. A grande maioria se apoia no uso de redes neurais para produzir seus resultados, geralmente combinando com o uso de *embeddings*. Em (WANG; IWAIHARA, 2019) é possível perceber que tal combinação apresentou resultados superiores a de sistemas populares, como *DBpedia Spotlight*[1] e *Babelfy*[2].

Foi observado que uma pequena parcela de trabalhos existentes utilizam *embeddings* de conhecimento juntamente com o embedding de palavras, assim como esta proposta. Dentre os motivos para o uso desses *embeddings*, (SEVGILI; PANCHENKO; BIEMANN, 2019) apontam que houve uma melhora nos resultados após a inclusão de *embeddings* de conhecimento, sendo que seu modelo já utilizava *embeddings* de palavras e redes neurais.

No entanto, os *embeddings* eram gerados utilizando técnicas diferentes: os de palavra foram gerados utilizando *Word2Vec* e os de conhecimento foram gerados através do *DeepWalk* (PEROZZI; AL-RFOU; SKIENA, 2014). Este trabalho também gera os *embeddings* de palavra e conhecimento separadamente mas se difere por utilizar o *fastText* para ambos.

[1] https://www.dbpedia-spotlight.org/
[2] http://babelfy.org/

4 Proposta do Trabalho

Conforme discutido no Capítulo 3, o estado da arte da desambiguação têm obtido resultados mais satisfatórios utilizando *embeddings* de palavras em conjunto com os de conhecimento, sendo que o resultado deste processo de geração serve de alimento para uma rede neural. A Figura 2 traz uma visão geral da proposta deste trabalho e em seguida, a proposta é descrita em mais detalhes.

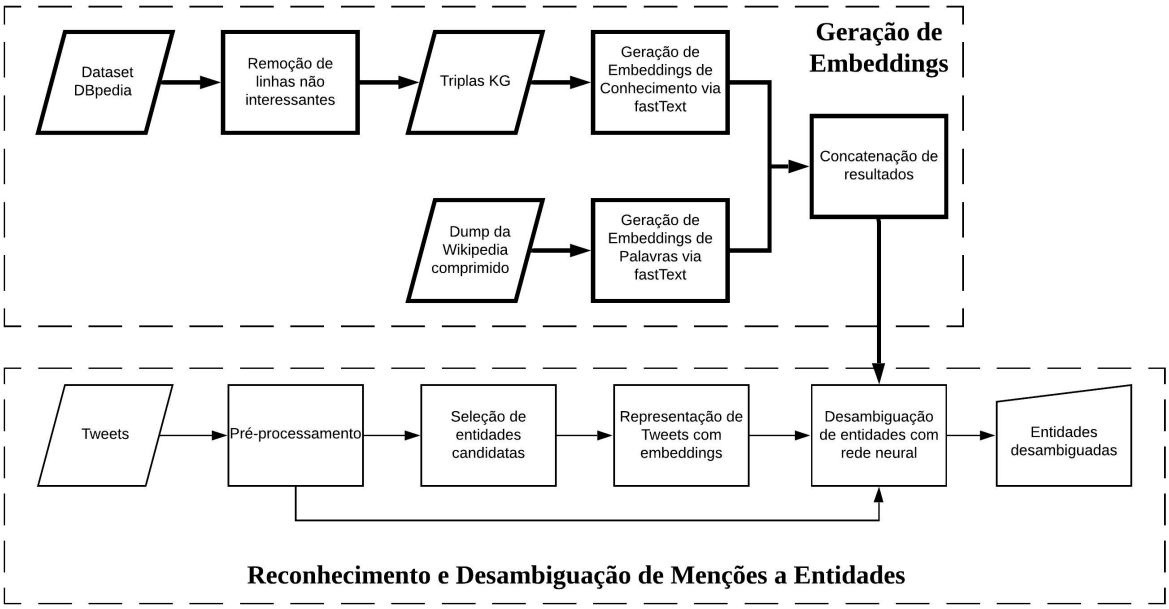

Figura 2 – Visão geral da proposta.

4.1 Geração dos *Embeddings*

Em (SEVGILI; PANCHENKO; BIEMANN, 2019), os autores treinam os *embeddings* de palavra e conhecimento usando duas técnicas diferentes, considerando que a rede neural será capaz de alinhar os dois tipos de *embedding*. Na proposta deste trabalho, os *embeddings* também são treinados de maneira separada, mas utilizando a mesma técnica, no caso *fastText*. Por conta disso, a mesma função será aplicada para geração dos *embeddings*, além de usar também os mesmos valores para dimensão, e com isso o resultado final de ambos tende a estar no mesmo espaço vetorial.

Apoiado no fato do resultado pertencer ao mesmo espaço vetorial, é esperado que a rede neural consiga comparar tipos de *embeddings* diferentes com mais facilidade e gere resultados mais satisfatórios. Isto é, em (SEVGILI; PANCHENKO; BIEMANN, 2019) os *embeddings* são gerados através de técnicas diferentes e consequentemente estão em espaços vetoriais diferentes. O referido trabalho confia na rede neural para que ela desambigue as entidades mesmo assim, correndo o risco de obter resultados inferiores por conta disso.

Para o treinamento de *embeddings* de palavras, foi utilizado o arquivo de entrada disponibilizado no próprio repositório da biblioteca *fastText*. Tal arquivo é um *dump* da Wikipedia em 9 idiomas diferentes (Árabe, Tcheco, Alemão, Inglês, Espanhol, Francês, Italiano, Romeno e Russo) que foi normalizado através de um *script* de compressão de dados (BOJANOWSKI et al., 2017). Além do arquivo de entrada, o repositório da biblioteca também contém um exemplo de linha de comando para utilizar a biblioteca para obter os *embeddings* de palavras. A linha de comando foi modificada para o propósito deste trabalho, utilizando parâmetros diferentes, como por exemplo a dimensão dos *embeddings* ou a curva de aprendizado, procurando sempre aperfeiçoar ao máximo os resultados com base nos recursos disponíveis.

Para o treinamento de *embeddings* de conhecimento também foi utilizado o exemplo presente no repositório do *fastText*. Visando obter resultados mais precisos e condizentes com uma aplicação real, foi escolhido utilizar o conjunto de dados da DBpedia para treinar os *embeddings* de conhecimento. Lembrando que a DBpedia é a versão em grafo de conhecimento da Wikipedia, representando os fatos que compõem as páginas da Wikipedia como triplas RDF.

Tendo escolhido a DBpedia como grafo de conhecimento, o passo inicial para gerar os *embeddings* de conhecimento foi extrair o conjunto de dados da DBpedia e transformá-los para o padrão do *fastText*. A DBpedia disponibiliza algumas opções de *datasets* para *download* em sua página. Para este trabalho, a opção escolhida é a que fornece apenas triplas de alta qualidade. Apesar de ser apenas um subconjunto das triplas presentes no *dataset* inteiro, a alta qualidade permite gerar *embeddings* mais precisos.

A transformação foi feita através de um script em *Python* em conjunto com a biblioteca RDFlib[1]. O script processa as triplas da DBpedia, retira o sujeito, predicado e objeto de cada tripla, e reconstrói as triplas no formato do fastText. Como o objetivo é treinar embeddings de conhecimentos, são utilizadas somente as triplas cujo sujeito e objeto são duas entidades. Triplas que apresentam atributos com valores literais (e.g., texto simples, datas, números) não são consideradas. Para identificar triplas que não contêm atributos primitivos, são consideradas somente as triplas que possuem o padrão *dbr:ENTIDADE dbo:RELACIONAMENTO dbr:ENTIDADE*. Um exemplo de uma linha presente no *da-*

[1] https://github.com/RDFLib/rdflib

taset da DBpedia que nos interessa, e já transformada no padrão do fastText, é

```
__label__dbr:What_Goes_On_(Velvet_Underground_song) 0_dbo:genre dbr:Art_rock
```

que conecta a música *What Goes On* da banda *Velvet Underground* com o gênero *Art rock*. O dígito presente no relacionamento indica a direção do fato, sendo que *0* indica que o relacionamento é dado da esquerda para a direita, ou seja, que a música *What Goes On* é do gênero *Art rock*. Devido à maneira que o *fastText* é implementado, é necessário especificar a tripla na sua relação reversa. O dígito presente no relacionamento também é alterado, para demonstrar que o relacionamento ocorre da direita para a esquerda, ou seja, que o gênero musical *Art rock* possui uma música chamada *What Goes On*. A mesma tripla acima na sua relação reversa é representada por:

```
dbr:What_Goes_On_(Velvet_Underground_song) 1_dbo:genre __label__dbr:Art_rock
```

Após essa transformação do *dataset* da DBpedia, foi possível utilizá-lo como entrada para a geração de *embeddings* de conhecimento. A geração foi feita novamente modificando os exemplos já presentes no repositório da biblioteca *fastText* para os propósitos deste trabalho.

O processo de geração dos *embeddings* resultou em dois arquivos (com embeddings de palavras e de conhecimento, respectivamente), sendo que a primeira linha de ambos os arquivos é composta da quantidade de elementos (respectivamente, número de palavras e, número de entidades e relacionamentos) e dimensão dos *embeddings* gerados, e as linhas seguintes compunham dos *embeddings* propriamente ditos. Entretanto, como este trabalho visa utilizar ambos embeddings de forma conjunta, os arquivos de embeddings foram concanetados. O único cuidado necessário na concatenação é garantir que a primeira linha contenha a soma das quantidades de ambos os *embeddings* de palavras como os de conhecimento.

4.2 Reconhecimento e Desambiguação de Menções

Após a geração dos *embeddings*, é realizado o processo de desambiguação de menções em tweets, usando tais *embeddings* e a rede neural. O primeiro passo no EL é reconhecer as menções à entidades nomeadas presentes no texto. No entanto, como o foco deste trabalho é a desambiguação, é considerado que as menções já foram reconhecidas, seja manualmente ou por alguma ferramenta. Portanto, este trabalho detalha somente os passos seleção de entidades candidatas para cada menção e desambiguação.

Dado um conjunto de menções M em um *tweet*, é necessário procurar por entidades

que possam descrever corretamenta cada menção $m_i \in M$. Cada conjunto de candidatas é denominado entidades candidatas C_i para a menção m_i. Este passo precisa garantir que o conjunto C_i não seja tão pequeno de forma que a entidade que corretamente descreve m_i não esteja nele, e nem muito grande que acabe gerando ruído e excesso de opções, comprometendo os resultados e o tempo de execução da tarefa de desambiguação.

O processo de seleção escolhido para este trabalho foi desenvolvido pelo Italo, orientador deste trabalho, mas que ainda não submeteu seu artigo. No referido trabalho, de início é necessário realizar um pré-processamento dos *tweets* que serão utilizados. Este pré-processamento é necessário porque menções em um *tweet* podem ocorrer tanto no conteúdo textual de um *tweet* como também na forma de *hashtags* – textos precedidos pelo caractere # – ou menções a usuários específicos (e.g. *@Estadao*, *@G1*).

O pré-processamento inicia removendo os caracteres @ e # dessas menções e após isso, através de expressões regulares, separa menções utilizando *camel case* (e.g. *MichaelJordan*) e *snake case* (e.g. *Michael_Jordan*) em *Michael Jordan*. Por fim, apenas a primeira letra de cada palavra de uma menção é mantida em letra maiúscula.

As menções pré-processadas são utilizadas para encontrar o conjunto de entidades candidatas C_i para cada menção m_i. Para encontrar as candidatas, é realizada uma busca em indíce previamente construído (MOUSSALLEM et al., 2017). Este indíce contém, para cada entidade na DBpedia, diversos nomes de superfície pelos quais essas entidades são denominadas e foi implementado na ferramenta ElasticSearch[2]. Para cada menção m_i, são realizadas duas consultas, de maneira simultânea, no ElasticSearch: Correspondência exata/contém e similaridade por *n-gram*. As duas consultas são realizadas para aumentar as chances de que a entidade que descreve corretamente m esteja entre as candidatas retornadas.

Caso nenhuma candidata seja retornada pela consulta, a menção m_i é quebrada em um conjunto de menções menores denominado M'_i, caso m_i seja composta por mais de uma palavra. Considerando que uma menção é um conjunto de palavras, nós retiramos uma palavra de m_i enquanto as demais são concatenatas, respeitando a ordem das mesmas. Esse passo é realizado em todas as palavras de m, gerando, portanto, o conjunto M'. Para cada menção $m_i j' \in M'_i$, são realizadas novamente as consultas no ElasticSearch. Caso nenhuma candidata seja retornada, é considerado que para esta menção m_i não existe uma entidade na DBpedia que a descreva corretamente. Portanto, m é ligada ao rótulo *NIL* (*Not Linkable*).

Por fim, caso a menção m_i possua candidatas, é necessário realizar o processo de desambiguação em si, utilizando os *embeddings* gerados e as entidades candidatas encontradas. A desambiguação, assim como a seleção de entidades candidatas, inicia com um

[2] https://www.elastic.co/pt/

passo de pré-processamento adicional dos *tweets* utilizados. Nesta etapa, este processo visa remover algumas informações que não auxiliam no processo de desambiguação, como *URLs*, *emoticons* (representações gráficas de expressões faciais) e alguns ruídos, como por exemplo palavras muito modificadas ou abreviações. Tais elementos podem ser considerados em outras tarefas, como análise de sentimentos, mas são pouco informativos no que diz respeito à desambiguação. O pré-processamento ainda limpa os símbolos # e @, de maneira análoga ao passo de seleção de entidades candidatas.

Após pré-processados, os *tweets* são utilizados em conjunto com as entidades candidatas selecionadas para descobrir quais entidades fazem mais sentido no contexto do *tweet*. De maneira geral, cada menção $m_i \in M$ em um *tweet* é substituída por cada uma das suas entidades candidatas $c_j \in C_i$, de forma a gerar um novo *tweet* para cada candidata. Com os novos tweets, a rede neural determina se a candidata faz sentido ou não no contexto em que ela está inserida. A implementação da rede neural utilizada é feita na forma de um classificador binário.

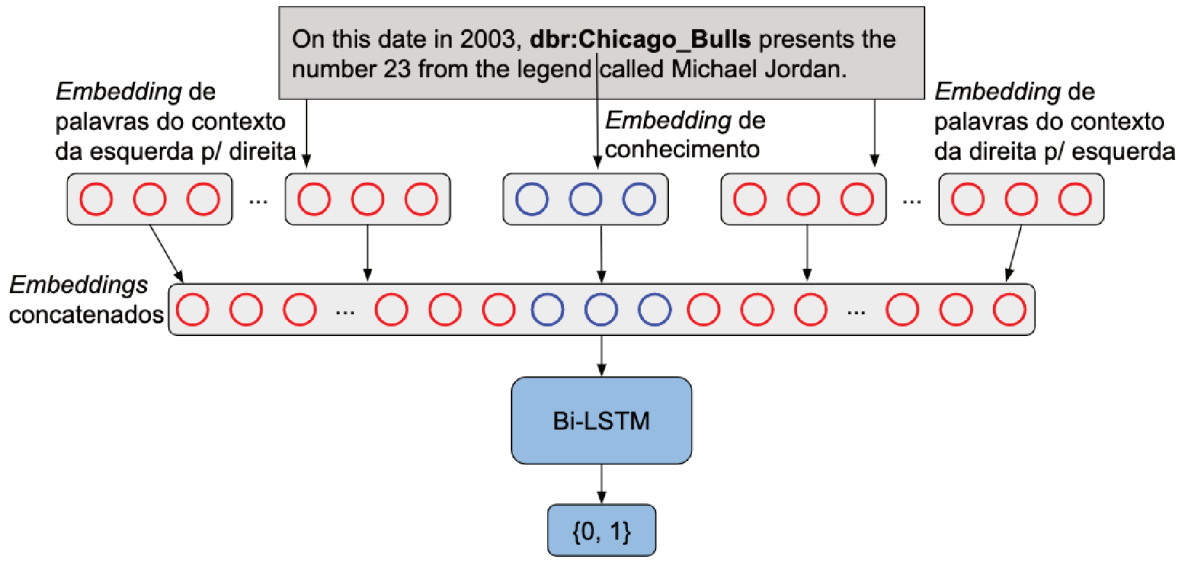

Figura 3 – Análise de um *tweet* após substituir uma menção por sua candidata.

Mais especificamente, a rede neural utilizada é uma *Bidirectional Long Short-Term Memory (Bi-LTSM)*. O principal motivo para a escolha da LTSM neste trabalho é por ser uma rede neural que leva em consideração a ordem que as palavras aparecem na entrada. Ou seja, a primeira palavra na entrada influencia as próximas, a segunda palavra influencia da terceira em diante, e assim sucessivamente. O *bidirecional* da rede neural também leva em consideração a ordem reversa. Devido a essa característica, redes neurais Bi-LTSM vem

sendo aplicadas com sucesso em outros trabalhos de EL, como mostrado no Capítulo 3. Levando em conta que queremos analisar o contexto de um *tweet* ao desambiguar, faz sentido essa abordagem.

Para cada tweet com uma candidata, ou seja, t_j, a rede neural atribui uma probabilidade desse tweet ser coerente com a entidade candidata presente nele. A candidata do tweet t_j que tiver a maior probabilidade de fazer sentido é a escolhida para descrever a menção m_i. A Figura 3 mostra o processo de análise, que tem como entrada um *tweet* com uma entidade no lugar da menção **Chicago Bulls** e uma saída de 0 ou 1, produzida pela rede LSTM que diz se a entidade candidata faz sentido.

4.2.1 Exemplo prático

Visando um melhor entendimento do reconhecimento e desambiguação de menções, um exemplo de *tweet* extraído do *dataset Microposts* é trazido, bem como sua análise até a desambiguação. O *tweet* extraído é o seguinte:

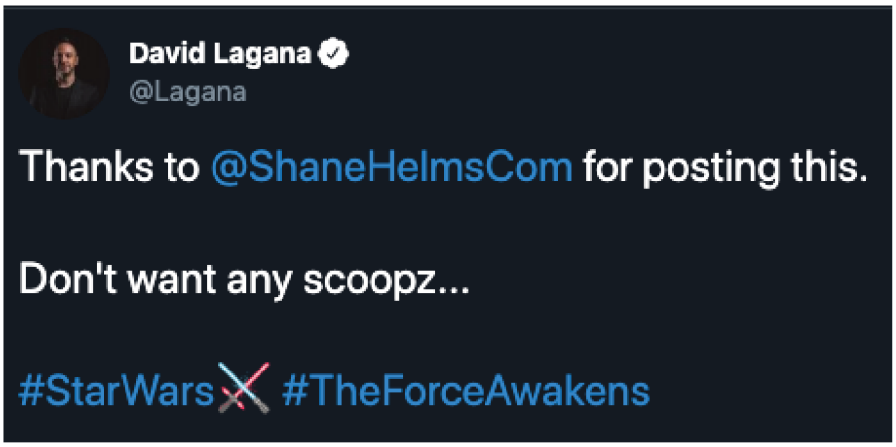

Figura 4 – *Tweet* presente no *dataset Microposts*.

O *tweet* ainda continha uma imagem que foi omitida pois é irrelevante para o processo, uma vez que o *dataset* guarda apenas a URL da imagem. Sendo assim, a maneira como o mesmo *tweet* é representado no *dataset*, e sobre o qual foi feita a análise, é a maneira textual simplificada:

> RT @Lagana: Thanks to @ShaneHelmsCom for posting this.
> Don't want any scoopz...
> #StarWars #TheForceAwakens https://t.co/oNpQgjPuOO

O primeiro passo consiste no pré-processamento, removendo símbolos como @ e #, separando expressões escritas em *camel case*[3] e *snake case*[4], removendo conteúdos de pouca semântica, como URLs e palavras muito modificadas, que atrapalham o processo de análise. Dessa forma, o resultado deste passo é:

> RT Lagana: Thanks to Shane Helms Com for posting this.
> Don't want any...
> Star Wars The Force Awakens

A seguir, obtemos as entidades candidatas para as menções encontradas no *tweet*. Por exemplo, um possível conjunto de entidades candidatas à menção *Star Wars The Force Awakens* seria:

```
dbr:Star_Wars:_The_Force_Awakens_(film);
dbr:Star_Wars:_The_Force_Awakens_(soundtrack)
```

o que geraria o mesmo *tweet* anterior, com a menção substituída pela suas possíveis entidades candidatas:

> RT Lagana: Thanks to Shane Helms Com for posting this.
> Don't want any...
> dbr:Star_Wars:_The_Force_Awakens_(film)

> RT Lagana: Thanks to Shane Helms Com for posting this.
> Don't want any...
> dbr:Star_Wars:_The_Force_Awakens_(soundtrack)

O processo de desambiguação finaliza alimentando a rede LSTM com as variações de *tweet*, que por sua vez informa quais entidades candidatas fazem sentido no contexto que foram inseridas. No caso, a entidade que representa a trilha sonora do filme, com o sufixo *(soundtrack)*, é descartada, enquanto que a entidade que representa o filme – sufixo *(film)* – é a escolhida.

[3] https://pt.wikipedia.org/wiki/CamelCase
[4] https://en.wikipedia.org/wiki/Snake_case

5 Experimentos e Discussão

Este capítulo relata os experimentos realizados para verificar a viabilidade do método proposto neste trabalho. As seções a seguir apresentam, respectivamente, o objetivo da realização desses experimentos, as configurações utilizadas, os resultados e uma discussão sobre os mesmos.

5.1 Objetivo dos Experimentos

Testar a viabilidade da proposta em *datasets* de *tweets* da literatura. Como o foco é a desambiguação, nossa avaliação é medida principalmente pela precisão. No entanto, somente a precisão não funciona adequadamente quando o *dataset* não está balanceado, isto é, as classes não estão distribuidas de maneira equilibrada. O *F1 Score* é uma medida que leva em consideração não só a precisão, mas também a cobertura (*recall*) do teste, realizando uma média harmônica entre a precisão e a cobertura. A fórmula utilizada para obter o *F1 Score* é a seguinte:

$$F1 = 2 \times \frac{precisão \times cobertura}{precisão + cobertura}$$

onde a *precisão* mede a quantidade de verdadeiros positivos dividido pelo total de assumidos positivos, certos ou errados. Já a *cobertura* mede a quantidade de verdadeiros positivos sobre o total acertos do modelo, chamados de *elementos relevantes*. O valor obtido varia de 0 a 1, sendo que o 1 é atingido quando tanto a precisão quanto a cobertura são perfeitas.

5.2 Configuração dos experimentos

Esta seção apresenta as configurações das máquinas utilizadas para a execução da proposta deste trabalho e todos os parâmetros que possam a vir a influenciar os resultados.

A rede neural proposta neste trabalho foi treinada no supercomputador Euler[1], em lâminas com as seguintes configurações:

- Hardware:
  - x2 Processador Intel Xeon E5-2650v4 de 2.2 GHz com doze núcleos
  - 128 GB DDR3 1866MHz de memória

[1] http://www.cemeai.icmc.usp.br/Euler/index.html

  - x1 GPU Nvidia Tesla P100 - 3584 Cuda cores - 16GB

- Software:

  - Red Hat Enterprise Linux, CentOS e Altair PBS Pro
  - Python 3.5.4
  - Cuda-toolkit 8.0.44
  - Cudnn 7.0
  - Torch 1.0.1.post2
  - Torchvision 0.4.0

A proposta em si foi executada em outra máquina, com as seguintes configurações:

- Hardware:

  - x2 Processador Intel Xeon E5-2620 v2 de 2.10GHz com 6 núcleos
  - 128 GB DDE3 1600 MHz de memória

- Software:

  - Debian GNU/Linux 8.10 (jessie)
  - Python 3.4.2
  - Torch 1.2.0
  - Torchvision 0.4.0
  - Elasticsearch engine 7.3.2

Por fim, a rede neural utilizada apresenta os seguintes parâmetros:

- 2 Camadas ocultas de leitura da esquerda para direita
- 2 Camadas ocultas de leitura da direita para esquerda
- 200 células em cada camada
- Dimensão dos *embeddings*: 200
- Otimizador: ADAM
- Função de perda: *Binary Cross-Entropy*

5.3 Resultados

A Tabela 2 apresenta os resultados desta proposta e de outras ferramentas na plataforma de *benchmarking* para tarefas relacionadas a EL GERBIL[2] (RÖDER; USBECK; NGOMO, 2018). Como o foco desta proposta está em postagens de mídias sociais, mais especificamente tweets, os conjuntos de dados utilizados para avaliar os resultados são os conjuntos de teste do Microposts2014, 2015 e 2016. As ferramentas presentes na Tabela 2 são disponibilizadas no serviço Web do GERBIL. Apesar de existirem outras abordagens que anotam semanticamente tweets, as mesmas não utilizam o GERBIL. Portanto, não foi possível e viável comparar esta proposta com elas. Além disso, outras abordagens de EL não presentes na tabela utilizam o GERBIL. No entanto, as mesmas não estão disponíveis no serviço Web do GERBIL, além de que em suas publicações, os conjuntos de dados utilizados não são os relacionados a tweets.

Tabela 2 – Comparação de resultados em termos de *Micro-F1 Score*

Trabalhos	Microposts2014-Test	Microposts2015-Test	Microposts2016-Test
AGDISTIS/MAG	**0,497**	**0,719**	**0,616**
AIDA	0,412	0,414	0,183
Babelfy	0,475	0,341	0,157
DBpedia Spotlight	0,452	ERR	ERR
FOX	0,252	0,311	0,068
FREME NER	0,419	0,313	0,162
OpenTapioca	0,215	0,259	0,053
Este trabalho	0,040	0,287	0,278

5.4 Discussão

O sistema de *benchmark GERBIL* fornece várias informações sobre os experimentos executados. No entanto, a mais relevante para o escopo desse trabalho é o *Micro-F1 Score*, que nos dá a pontuação atingida por cada ferramenta experimentada com base nos acertos e erros cometidos na desambiguação.

Utilizando esta pontuação como base, é possível concluir que o objetivo de obter resultados superiores aos das principais técnicas existentes atualmente foi atingido com êxito parcial. A Tabela 2 traz em sua primeira coluna os trabalhos utilizados para com-

[2] http://gerbil.aksw.org/gerbil/

paração e nas colunas seguintes traz o *Micro-F1 Score* das referidas técnicas para cada um dos *datasets* presentes no cabeçalho da tabela.

Sendo assim, observa-se que utilizando o conjunto de dados *Microposts2016*, este trabalho apresentou um *Micro-F1 Score* superior aos demais, alcançando uma pontuação de 0.2788 enquanto a maioria das técnicas falhou em chegar na pontuação de 0.200. Apenas uma técnica obteve resultados superiores, considerados significantemente altos, por superar a pontuação atingida por este trabalho em aproximadamente 0.300.

Esta pontuação alta foi obtida porque a técnica AGDISTIS utiliza medidas de popularidade para desambiguar menções em *tweets*. É possível obter 85% de precisão caso a técnica se baseie em ligar menções com entidades candidatas mais populares (GUO; CHANG; KICIMAN, 2013). Tal abordagem não foi utilizada neste trabalho por acreditar que desambiguar menções em *tweets* tentando entender o contexto em que ele está inserido é mais importante do que confiar na popularidade de entidades.

Analisando o *dataset Microposts2015*, apesar dos resultados obtidos nesse trabalho terem sido inferiores aos das técnicas utilizadas para comparação, observou-se que a pontuação foi parecida com a pontuação obtida com o conjunto de dados *Microposts2016*. Isso é importante devido ao fato de que o conjunto de 2016 foi utilizado para aprendizado da rede neural, então era esperado seu bom resultado com o conjunto de dados de 2016, mas manter uma pontuação semelhante com os dados de 2015 mostra a eficácia do processo, mesmo que outras técnicas tenham obtido resultados melhores com o *dataset* de 2015.

6 Conclusões e Trabalhos Futuros

Analisando o trabalho realizado e comparando com o esperado pelos objetivos descritos no Capítulo 1, foi possível concluir que os objetivos foram alcançados com êxito. A técnica proposta e implementada neste trabalho, utilizando os conjuntos de dados de *tweets* com regra ouro mais recentes que encontramos na literatura, conseguiu obter resultados melhores do que os resultados obtidos pela maioria das abordagens do estado da arte, atingindo assim seu objetivo geral.

Em relação a trabalhos futuros, este trabalho pode guiar pesquisadores que desejam se aprofundar na técnica aqui adotada e descrita. Quanto a isso, algumas linhas de pesquisa são sugeridas nas seções a seguir.

6.1 Geração dos *embeddings*

Este trabalho se diferenciou um pouco dos presentes no estado da arte da literatura ao gerar *embeddings* de conhecimento e de palavras de maneira separada, concatenando seus resultados antes de enviar como entrada para uma rede neural. Trabalhos futuros podem explorar mais maneiras de gerar os *embeddings* de forma que potencialize ainda melhores os resultados.

Ainda sobre os *embeddings*, este trabalho gerou *embeddings* com dimensão máxima de 200 por limitações de *hardware* nos recursos disponíveis para realização do trabalho. Trabalhos com recursos suficientes para gerar *embeddings* de dimensões maiores podem ter seus resultados melhorados.

6.2 Estudo das causas dos resultados obtidos

Apesar dos resultados documentados no capítulo 5 terem sido satisfatórios para a conclusão deste trabalho, gerando um *Micro-F1 Score* de 0.2871 e 0.2788 nos *datasets* de *tweets* de 2015 e 2016 respectivamente, sendo este último melhor do que a maioria das técnicas existentes, quando utilizado o *dataset Microposts* de 2014, o resultado foi muito abaixo do esperado, falhando em alcançar um *Micro-F1 score* de 0.05. Uma análise sobre os motivos que levaram a um resultado tão baixo e diferente dos demais também cabe em um estudo.

Referências

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, v. 5, p. 157–66, 02 1994. Citado na página 27.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific American*, v. 284, 2001. Disponível em: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>. Citado na página 24.

BOJANOWSKI, P. et al. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, v. 5, p. 135–146, 2017. ISSN 2307-387X. Citado 2 vezes nas páginas 21 e 32.

CHEN, H. et al. Bilinear joint learning of word and entity embeddings for entity linking. *Neurocomputing*, v. 294, 12 2017. Citado na página 29.

DERCZYNSKI, L. et al. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 2014. Citado 3 vezes nas páginas 25, 26 e 29.

EHRLINGER, L.; WÖSS, W. Towards a definition of knowledge graphs. In: *SEMANTiCS*. [S.l.: s.n.], 2016. Citado na página 24.

FANG, Y.; CHANG, M.-W. Entity linking on microblogs with spatial and temporal signals. *Transactions of ACL (TACL)*, p. 259–272, 2014. Citado na página 29.

FUKUSHIMA, K. Cognitron: A self-organizing multilayer neural network. *Biological Cybernetics*, v. 20, p. 121–136, 1975. Citado na página 27.

GONGQING, W.; YING, H.; XUEGANG, H. Entity linking: An issue to extract corresponding entity with knowledge base. v. 6, p. 6220–6231, 2018. Citado na página 25.

GRAVES, A.; SCHMIDHUBER, J. Offline handwriting recognition with multidimensional recurrent neural networks. In: KOLLER, D. et al. (Ed.). *Advances in Neural Information Processing Systems 21*. Curran Associates, Inc., 2009. p. 545–552. Disponível em: <http://papers.nips.cc/paper/3449-offline-handwriting-recognition-with-multidimensional-recurrent-neural-networks.pdf>. Citado na página 27.

GUO, S.; CHANG, M.-W.; KICIMAN, E. To link or not to link? a study on end-to-end tweet entity linking. In: . Atlanta, Georgia: Association for Computational Linguistics, 2013. p. 1020–1030. Disponível em: <https://www.aclweb.org/anthology/N13-1122>. Citado na página 42.

GUO, Y. et al. Microblog entity linking by leveraging extra posts. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, p. 863–868, 2013. Citado na página 29.

HAN, H. et al. Yet another framework for tweet entity linking (yaftel). In: . [S.l.: s.n.], 2019. p. 258–263. Citado na página 29.

HANNUN, A. et al. Deepspeech: Scaling up end-to-end speech recognition. 12 2014. Citado na página 27.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, v. 9, p. 1735–80, 12 1997. Citado na página 27.

HOSSEINI, H. et al. Implicit entity linking in tweets: An ad-hoc retrieval approach. *Applied Ontology*, p. 1–27, 2019. Citado na página 29.

JOULIN, A. et al. Bag of tricks for efficient text classification. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. [S.l.]: Association for Computational Linguistics, 2017. p. 427–431. Citado na página 21.

KOLITSAS, N.; GANEA, O.; HOFMANN, T. End-to-end neural entity linking. *CoRR*, abs/1808.07699, 2018. Disponível em: <http://arxiv.org/abs/1808.07699>. Citado na página 29.

LANDEGHEM, J. V. A survey of word embedding literature: Context representations and the challenge of ambiguity. Apr 2016. Citado 2 vezes nas páginas 19 e 26.

LE, P.; TITOV, I. Improving entity linking by modeling latent relations between mentions. 04 2018. Citado na página 29.

LEHMANN, J. et al. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, v. 6, n. 2, p. 167–195, 2015. Disponível em: <https://doi.org/10.3233/SW-140134>. Citado 2 vezes nas páginas 19 e 24.

LI, M. et al. Entity disambiguation by knowledge and text jointly embedding. In: . [S.l.: s.n.], 2016. p. 260–269. Citado na página 29.

LIU, C. et al. Attention-based joint entity linking with entity embedding. *Information*, Multidisciplinary Digital Publishing Institute, v. 10, n. 2, p. 46, 2019. Citado na página 30.

MARTINS, P. H.; MARINHO, Z.; MARTINS, A. F. T. Joint learning of named entity recognition and entity linking. 07 2019. Citado na página 29.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. Sep 2013. Citado 2 vezes nas páginas 20 e 25.

MORENO, J. et al. Combining word and entity embeddings for entity linking. In: . [S.l.: s.n.], 2017. p. 337–352. Citado na página 29.

MOUSSALLEM, D. et al. Mag: A multilingual, knowledge-base agnostic and deterministic entity linking approach. In: ACM. *Proceedings of the Knowledge Capture Conference*. [S.l.], 2017. p. 9. Citado na página 34.

NGUYEN, D. Q. An overview of embedding models of entities and relationships for knowledge base completion. Mar 2017. Citado na página 23.

OMNICORE. *Twitter by the Numbers*. 2018. Disponível em: <https://www.omnicoreagency.com/twitter-statistics/>. Acesso em: Maio 2018. Citado na página 19.

ONTOTEXT. *What is Semantic Annotation?* 2018. Disponível em: <https://www.ontotext.com/knowledgehub/fundamentals/semantic-annotation/>. Acesso em: Novembro 2018. Citado na página 24.

OREN, E. et al. *What are Semantic Annotations?* [S.l.], 2006. Disponível em: <http://www.siegfried-handschuh.net/pub/2006/whatissemannot2006.pdf>. Citado na página 24.

PARRAVICINI, A. et al. Fast and accurate entity linking via graph embedding. In: . [S.l.: s.n.], 2019. p. 1–9. Citado 2 vezes nas páginas 29 e 30.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. In: ACM. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2014. p. 701–710. Citado na página 30.

RÖDER, M.; USBECK, R.; NGOMO, A. N. GERBIL - benchmarking named entity recognition and linking consistently. *Semantic Web*, v. 9, 2018. Disponível em: <http://www.semantic-web-journal.net/system/files/swj1671.pdf>. Citado 2 vezes nas páginas 22 e 41.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Neurocomputing: Foundations of research. In: ANDERSON, J. A.; ROSENFELD, E. (Ed.). Cambridge, MA, USA: MIT Press, 1988. cap. Learning Representations by Back-propagating Errors, p. 696–699. ISBN 0-262-01097-6. Disponível em: <http://dl.acm.org/citation.cfm?id=65669.104451>. Citado na página 27.

SAK, H. et al. *Google voice search: faster and more accurate*. 2015. Disponível em: <https://ai.googleblog.com/2015/09/google-voice-search-faster-and-more.html>. Acesso em: Outubro 2019. Citado na página 27.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks*, v. 61, 04 2014. Citado na página 27.

SEVGILI, Ö.; PANCHENKO, A.; BIEMANN, C. Improving neural entity disambiguation with graph embeddings. In: *ACL*. [S.l.: s.n.], 2019. Citado 5 vezes nas páginas 26, 29, 30, 31 e 32.

SINGHAL, A. *Introducing the Knowledge Graph: things, not strings*. 2012. Disponível em: <https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html>. Acesso em: Outubro 2018. Citado na página 24.

SOCHER, R. et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Stroudsburg, PA: Association for Computational Linguistics, 2013. p. 1631–1642. Disponível em: <http://www.aclweb.org/anthology/D13-1170>. Citado na página 25.

TENSORFLOW. *Embeddings*. 2018. Disponível em: <https://www.tensorflow.org/guide/embedding>. Acesso em: Novembro 2018. Citado 2 vezes nas páginas 25 e 26.

WANG, Q.; IWAIHARA, M. Deep neural architectures for joint named entity recognition and disambiguation. In: . [S.l.: s.n.], 2019. Citado 2 vezes nas páginas 29 e 30.

WANG, Q. et al. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge & Data Engineering*, v. 29, n. 12, p. 2724–2743, Dec. 2017. ISSN 1041-4347. Disponível em: <doi.ieeecomputersociety.org/10.1109/TKDE.2017.2754499>. Citado na página 20.

ZHU, G.; IGLESIAS, C. Exploiting semantic similarity for named entity disambiguation in knowledge graphs. *Expert Systems with Applications*, v. 101, 02 2018. Citado na página 29.

Apêndices

APÊNDICE A – Artigo

Uso de Técnicas e Ferramentas de Embedding de Conhecimento para Desambiguação de Anotações Segundo Contextos Semânticos

João V. Fagundes[1], **Italo L. Oliveira** [1], **Renato Fileto**[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
88.040-900 – Florianópolis – SC – Brazil

joaovictordmf@gmail.com, italo.oliveira@posgrad.ufsc.br, r.fileto@ufsc.br

Abstract. *Semantic annotations allow to link unstructured data (e.g., relevant references in documents) to resources with well-founded semantics in knowledge information bases (e.g., DBPedia, Wordnet, Babelnet, Google Knowledge Graph) that help to explain what those annotated data are referring to. However, by trying to capture the semantics in data, such as social media posts, the computer may face problems, such as the common use of slangs and regionalisms in informal conversation, and mainly the occurrence of ambiguity. This work consists in an study of the state of the art of the word ambiguity problem, having as objective the mastery of the techniques and tools available that are being used to solve this problem, such as knowledge and word embedding. From this, it is intended to disambiguate mentions present in social media contents by using embeddings together with neural networks. Semantic contexts are captured, represented and explored in these embeddings for annotation disambiguation. The proposed approach is evaluated in the improvement of the precision of the annotations from a set of tweets, such as the Microposts dataset, which provide annual sets of tweets.*

Resumo. *Anotações semânticas permitem associar a dados ou porções de dados não-estuturados (e.g., menções relevantes em textos) recursos com semântica bem definida em bases de conhecimento (e.g., DBpedia, Wordnet, Babelnet) que ajudam a explicar a que os dados anotados se referem. Todavia, ao tentar capturar a semântica de dados, como, por exemplo, postagens em mídias sociais, aplicações de enriquecimento semântico esbarram em problemas como o uso de gírias e regionalismos, e principalmente, a ocorrência de ambiguidade. Este trabalho consiste em realizar um estudo do estado da arte do problema da ambiguidade de palavras, tendo como objetivo o domínio das técnicas e ferramentas disponíveis atualmente, como por exemplo os embeddings de palavras e conhecimento. A partir disso, pretende-se desambiguar as menções existentes em postagens de mídias sociais com o auxílio dos embeddings em conjunto com redes neurais. Contextos semânticos são capturados, representados e explorados nesses embeddings para a desambiguação de anotações. A abordagem proposta é avaliada na melhoria da precisão de anotações de conjuntos de tweets, como por exemplo o dataset Microposts, que fornece conjuntos de tweets anuais.*

1. Introdução

Com o avançar da tecnologia, a quantidade de dados produzida a cada dia só aumenta. Mídias sociais (e.g., Facebook, Twitter, Instagram) e bibliotecas digitais estão entre as fontes de dados mais utilizadas atualmente. Os dados nelas disponíveis podem ser úteis para uma variedade de aplicações. Todavia, documentos texto e postagens em mídias sociais são dados não estruturados e costumam apresentar problemas como ambiguidades, tornando sua semântica difícil de capturar e tratar computacionalmente [Landeghem 2016].

Anotações semânticas são uma alternativa para contornar tais problemas. Uma anotação semântica associa uma porção de dado (e.g. uma menção a uma entidade em um texto) a uma descrição formal e processável por máquina daquilo a que se refere (e.g. recurso de grafo de conhecimento que descreve tal entidade). Por exemplo, uma menção à entidade nomeada **Michael Jordan** em um texto (e.g., artigo de revista, conteúdo textual de um *tweet*) pode ser ligada, por meio de uma anotação semântica, a um nodo de grafo de conhecimento (e.g. DBpedia[1]) [Lehmann et al. 2015] representando o jogador de *basquete*, famoso por ter jogado no *Chicago Bulls* e ser 6 vezes campeão da *NBA*.

Entretanto, adicionar anotações semânticas a porções de textos, principalmente os provenientes de mídias sociais, não é uma tarefa trivial. Esses textos podem ter uma quantidade considerável de informalidade, com utilização de gírias ou regionalismos. Além disso, o significado de cada palavra em um texto depende do autor e de todo o contexto e da palavra dentro deste texto. Por exemplo, ao utilizar a palavra *apple* em um texto, o autor pode se referir à fruta ou à empresa de tecnologia. Esse é o problema conhecido como *ambiguidade* de palavras [Landeghem 2016], característica intrínseca em qualquer língua natural.

Uma popular rede social com uma variedade de assuntos é o *Twitter*. Essa rede social de *microblogging* tem uma média de 330 milhões de usuários ativos por mês e gera um total de 500 milhões de *tweets* por dia [Omnicore 2018]. Ao obter um conjunto de *tweets* podemos nos deparar com dados escritos das mais diversas maneiras. Caso uma ferramenta computacional decida por sempre escolher, por exemplo, o significado estatisticamente mais provável para cada palavra, tal abordagem pode gerar anotações imprecisas.

Uma maneira intuitiva utilizada pelos seres humanos para resolver ambiguidades durante uma conversa é analisar o contexto onde cada palavra está inserida, para então atribuir a ela o significado correto. Visando aumentar a precisão das anotações existentes e das novas que serão geradas, a ideia deste trabalho portanto é desenvolver uma ferramenta que durante o processo de anotações semânticas aumente o escopo de análise. Ou seja, utilizar as outras palavras presentes na mesma frase, ou em frases próximas, para desambiguar as menções à entidades do mundo real (entidades nomeadas), de modo que o significado delas auxilie a desambiguação, fornecendo um contexto semântico adequado.

Várias técnicas vêm sendo usadas ao longo dos anos para auxiliar o processo de desambiguação de porções de dados não-estruturados. Dentre tais técnicas, técnicas baseadas em *embeddings* têm se desenvolvido rapidamente e gerado bons resultados para uma variedade de problemas relacionados à semântica de dados. *Embeddings*, considerando o problema de anotação semântica de dados textuais, têm a função de mapear

[1] https://wiki.dbpedia.org/

objetos discretos em vetores de números reais. A técnica se divide algumas vertentes, que se diferenciam no tipo de objeto mapeado. Este trabalho estuda duas dessas vertentes: os *embeddings* de palavra [Mikolov et al. 2013], que trabalham com palavras presentes em textos e documentos; e os de conhecimento [Wang et al. 2017], que mapeam triplas de um KG. Pelo fato de ambas as vertentes apresentarem um crescimento considerável em importância, este trabalho se aprofunda no conhecimento e domínio de ambas, mais precisamente no uso da biblioteca chamada *fastText*[2].

Abordagens atuais focam na utilização de *embeddings* de palavra e de entidades (uma variante de embeddings de palavras/documentos) com redes neurais profundas. No entanto, a utilização de *embeddings* de conhecimento em conjunto com os de palavras em redes neurais profundas ainda não foi devidamente analisado.

2. Fundamentação Teórica

2.1. Grafos de Conhecimento (KG)

Grafos de Conhecimento (KG - *Knowledge graph*) formalizam conhecimento na forma de recursos com conceitos ou entidades nomeadas (representados como nodos do grafo) e relações semânticas entre eles (representadas como arestas), sendo aderente ao conceito de dados ligados (*Linked Data*). Um KG pode ser armazenado como um conjunto de triplas ¡*h, r, t*¿, onde *h* e *t* representam recursos (e.g., entidades, conceitos), origem e destino respectivamente, e *r* representa uma relação entre esses recursos [Nguyen 2017]. Uma tripla é também chamada de fato, pois, usualmente, representa relações do mundo real. Uma outra notação que pode ser encontrada na literatura para descrever um fato é x_{ijk}, sendo i a origem, j o destino e k a relação. Entretando, esta notação não será utilizada neste trabalho.

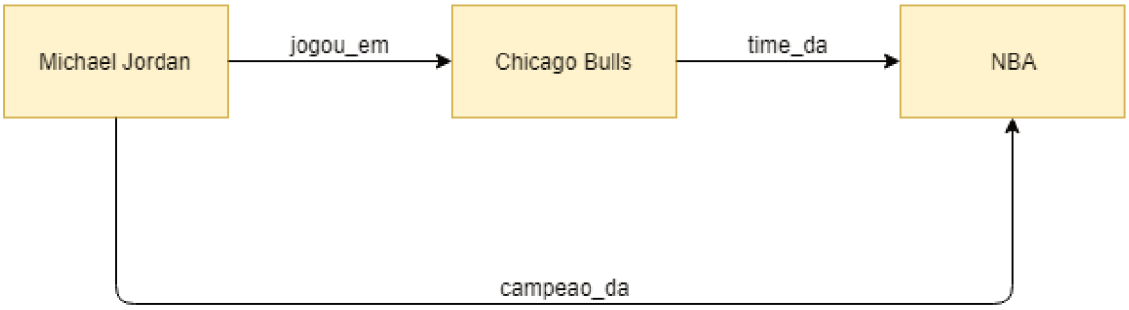

Figura 1. Exemplo de nodos e relações de grafo de conhecimento.

A Figura 1 representa o exemplo de KG descrito na introdução, que trata do jogador de basquete Michael Jordan. Neste exemplo, a entidade nomeada *Michael Jordan* está relacionada com a entidade nomeada *Chicago Bulls*, já que *Jordan* jogou basquete no time de *Chicago*, que por sua vez está relacionado à *NBA*. Vale ressaltar que ao tratar de relações, cria-se a necessidade de um direcionamento no grafo, para determinar qual entidade faz o papel de domínio (origem) e qual faz o papel de imagem (destino).

[2]https://fasttext.cc/

Os KG vêm em uma crescente de popularidade recentemente, principalmente por serem utilizados e explorados em pesquisas relacionadas à Web Semântica [Berners-Lee et al. 2001]. Apesar da tecnologia existir a mais tempo, foi após o *Google Knowledge Graph* ser divulgado, em 2012, que essa fama ganhou mais força. Ao apresentar o seu novo método de conduzir buscas, o conceito de Grafo de Conhecimento foi resumido em "*things, not strings*". Ou seja, "um grafo que compreende entidades do mundo real e suas relações umas com as outras" [Singhal 2012].

Por estar em foco nas pesquisas na área de ciência de dados, diversas definições de grafo de conhecimento foram surgindo. Apesar do aprofundamento nos estudos sobre KG, não há uma convergência clara entre as inúmeras definições, com algumas delas sendo suficientemente genéricas, podendo ser utilizadas, inclusive, para representar ontologias [Ehrlinger and Wöß 2016].

Um KG amplamente utilizado e conhecido é a DBpedia. O grafo é composto principalmente por dados provenientes dos artigos da Wikipedia em mais de 111 idiomas. Apenas a versão inglês da Wikipedia é responsável por mais de 400 milhões de fatos que descrevem 3,7 milhões de entidades

2.2. *Embeddings* de Palavra e Conhecimento

No que se refere à motivação deste trabalho, o problema de desambiguar entidades nomeadas em mídias sociais já foi explorado de diversas maneiras em pesquisas anteriores nos últimos anos, porém sem atingir resultados satisfatórios [Gongqing et al. 2018, Derczynski et al. 2014]. Dentre as técnicas utilizadas, a *embedding* é uma das mais exploradas atualmente. Um *embedding* é o mapeamento de objetos discretos, como palavras, para vetores de números reais [Tensorflow 2018], que condensam propriedades e relações relevantes desses objetos em espaços usualmente de dimensionalidade bem mais baixa que modelos vetoriais para vocabulários com milhares de palavras.

A técnica de *embeddings*, no entanto, possui uma classificação que a divide em duas outras técnicas mais específicas: os *embeddings* de palavra e os *embeddings* de conhecimento. A diferença entre as duas está no tipo de dado a ser mapeado, pois enquanto a primeira técnica mapeia palavras ou frases para vetores de números reais, a segunda trata não apenas disso, mas de triplas presentes em grafos de conhecimento.

Foi através do *word2vec* [Mikolov et al. 2013] que os *embeddings* de palavras começaram a receber uma atenção especial de pesquisadores, principalmente na área de Processamento de Linguagens Naturais (NLP). Isso se deve ao fato de *embeddings* de palavras terem mostrado uma melhora significativa ao realizar algumas tarefas de NLP, como análise de sentimentos [Socher et al. 2013].

Embeddings de palavra são importantes também para o aprendizado de máquina. Entradas para algoritmos de aprendizado de máquina podem utilizar palavras de um texto, que não possuem uma representação vetorial natural. A técnica de *embeddings* é uma maneira comum e efetiva de transformar objetos discretos em vetores de números reais [Tensorflow 2018]. Juntando os vetores de todas as palavras analisadas, obtém-se uma matriz, que pode ter seus valores alterados baseado no conjunto de dados utilizados para treinar o modelo de *embedding*. Apesar dos avanços destacados, os *embeddings* de palavras não se mostraram tão satisfatórios ao enfrentar o problema de desambiguação de entidades nomeadas em postagens de mídias sociais [Derczynski et al. 2014].

O baixo desempenho dos *embeddings* de palavras ao tratar da desambiguação de entidades nomeadas se deve ao fato de que ao mapear as palavras para vetores, acaba-se misturando algumas características importantes da mensagem, como, por exemplo, os contextos em que a palavra aparece com significados distintos [Landeghem 2016]. Portanto, a precisão do modelo depende muito do conjunto de dados utilizado para treiná-lo. Dados de experimentos de uma fonte diferente da utilizada para dados de treinamento podem acarretar em uma precisão baixa, como por exemplo tentar desambiguar artigos científicos usando um modelo treinado para desambiguar *tweets*, ou vice-versa. O baixo desempenho nesse caso seria atribuído ao fato dos conteúdos possuírem muitas diferenças quanto a maneira de escrita e tipo de linguagem utilizada.

Algumas técnicas que consistem em capturar outras informações adicionais, como, por exemplo, conhecimento prévio proveniente de bases de conhecimento, o que nos leva aos *embeddings* de conhecimento. Os *embeddings de conhecimento*, também chamados de *embeddings* de grafos de conhecimento, se aproveitam das características de um KG para realizar o mapeamento de recursos e suas relações em um KG para vetores, com mais precisão quanto a semântica do que *embeddings* de palavras por guardar informações de como a entidade se relaciona no mundo real. Isto é, não analisa uma palavra, mas sim a que (e.g. entidade, conceito) ela se refere, bem como suas relações com outros recursos descritos no KG. Apenas considerando isso, alguns trabalhos da literatura observaram uma melhora nos resultados ao inserir outro tipo de *embedding* nos seus experimentos [Sevgili et al. 2019].

2.3. Redes Neurais Long Short-Term Memory

As redes neurais também estão se tornando cada vez mais populares na ciência de dados. Sua motivação surgiu com avanços no campo de Inteligência Artificial (IA), suprindo a necessidade de um sistema computacional que simula os neurônios de um cérebro humano, sendo representada por diversos nós interconectados através de camadas.

Uma rede neural simples possui três camadas, sendo a primeira chamada de camada de entrada e a última a camada de saída. Entre elas, existe uma ou mais camadas que podem ser denominadas ocultas. São as camadas ocultas que têm seus nós ativados e estimulados de acordo com a entrada da rede neural, simulando o funcionamento dos neurônios do cérebro humano e produzindo um resultado na saída [Fukushima 1975].

Entretanto, tipicamente é difícil prever o quanto um raciocínio sobre eventos anteriores influencia os eventos futuros. Como este trabalho se propõe a desambiguar menções presentes em *tweets* a partir de seus contextos, uma rede neural simples não seria adequada por falhar em levar o contexto em consideração. Visando resolver este problema, as redes neurais recorrentes (RNN) implementam um laço em suas camadas ocultas onde aprendizados obtidos em camadas mais avançadas servem de estímulo para camadas anteriores [Rumelhart et al. 1988].

Na teoria, uma RNN resolve o problema de utilizar o contexto de um *tweet* na sua desambiguação, mas na prática alguns problemas foram reportados ao utilizar RNNs quando a recorrência possuía um longo intervalo [Bengio et al. 1994].

As redes Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] são redes neurais especializadas em guardar informações no longo prazo, e desde sua concepção elas têm encontrado espaço nas mais variadas áreas de es-

tudo da ciência de dados e inteligência artificial, como reconhecimento de escrita [Schmidhuber 2014, Graves and Schmidhuber 2009] e reconhecimento de fala [Hannun et al. 2014, Sak et al. 2015].

3. Desenvolvimento

O estado da arte da desambiguação têm obtido resultados mais satisfatórios utilizando *embeddings* de palavras em conjunto com os de conhecimento, sendo que o resultado deste processo de geração serve de alimento para uma rede neural.

3.1. Geração dos *Embeddings*

Na proposta deste trabalho, os *embeddings* são treinados de maneira separada, mas utilizando a mesma técnica, no caso *fastText*. Por conta disso, a mesma função será aplicada para geração dos *embeddings*, além de usar também os mesmos valores para dimensão, e com isso o resultado final de ambos tende a estar no mesmo espaço vetorial. Apoiado no fato do resultado pertencer ao mesmo espaço vetorial, é esperado que a rede neural consiga comparar tipos de *embeddings* diferentes com mais facilidade e gere resultados mais satisfatórios.

Para o treinamento de *embeddings* de palavras, foi utilizado o arquivo de entrada disponibilizado no próprio repositório da biblioteca *fastText*. Além do arquivo de entrada, o repositório da biblioteca também contém um exemplo de linha de comando para utilizar a biblioteca para obter os *embeddings* de palavras. A linha de comando foi modificada para o propósito deste trabalho, utilizando parâmetros diferentes, como por exemplo a dimensão dos *embeddings* ou a curva de aprendizado, procurando sempre aperfeiçoar ao máximo os resultados com base nos recursos disponíveis.

Para o treinamento de *embeddings* de conhecimento também foi utilizado o exemplo presente no repositório do *fastText*. Visando obter resultados mais precisos e condizentes com uma aplicação real, foi escolhido utilizar o conjunto de dados da DBpedia para treinar os *embeddings* de conhecimento. Lembrando que a DBpedia é a versão em grafo de conhecimento da Wikipedia, representando os fatos que compõem as páginas da Wikipedia como triplas RDF.

Tendo escolhido a DBpedia como grafo de conhecimento, o passo inicial para gerar os *embeddings* de conhecimento foi extrair o conjunto de dados da DBpedia e transformá-los para o padrão do *fastText*. A transformação foi feita através de um script em *Python* em conjunto com a biblioteca RDFlib[3]. O script processa as triplas da DBpedia, retira o sujeito, predicado e objeto de cada tripla, e reconstrói as triplas no formato do fastText. Como o objetivo é treinar embeddings de conhecimentos, são utilizadas somente as triplas cujo sujeito e objeto são duas entidades.

Triplas que apresentam atributos com valores literais (e.g., texto simples, datas, números) não são consideradas. Para identificar triplas que não contêm atributos primitivos, são consideradas somente as triplas que possuem o padrão *dbr:ENTIDADE dbo:RELACIONAMENTO dbr:ENTIDADE*.

Após essa transformação do *dataset* da DBpedia, foi possível utilizá-lo como entrada para a geração de *embeddings* de conhecimento. A geração foi feita novamente mo-

[3]https://github.com/RDFLib/rdflib

dificando os exemplos já presentes no repositório da biblioteca *fastText* para os propósitos deste trabalho.

O processo de geração dos *embeddings* resultou em dois arquivos (com embeddings de palavras e de conhecimento, respectivamente), sendo que a primeira linha de ambos os arquivos é composta da quantidade de elementos (respectivamente, número de palavras e, número de entidades e relacionamentos) e dimensão dos *embeddings* gerados, e as linhas seguintes compunham dos *embeddings* propriamente ditos. Entretanto, como este trabalho visa utilizar ambos embeddings de forma conjunta, os arquivos de embeddings foram concanetados. O único cuidado necessário na concatenação é garantir que a primeira linha contenha a soma das quantidades de ambos os *embeddings* de palavras como os de conhecimento.

3.2. Reconhecimento e Desambiguação de Menções

Após a geração dos *embeddings*, é realizado o processo de desambiguação de menções em tweets, usando tais *embeddings* e a rede neural. O primeiro passo no EL é reconhecer as menções à entidades nomeadas presentes no texto. No entanto, como o foco deste trabalho é a desambiguação, é considerado que as menções já foram reconhecidas, seja manualmente ou por alguma ferramenta. Portanto, este trabalho detalha somente os passos seleção de entidades candidatas para cada menção e desambiguação.

Dado um conjunto de menções M em um *tweet*, é necessário procurar por entidades que possam descrever corretamenta cada menção $m_i \in M$. Cada conjunto de candidatas é denominado entidades candidatas C_i para a menção m_i. Este passo precisa garantir que o conjunto C_i não seja tão pequeno de forma que a entidade que corretamente descreve m_i não esteja nele, e nem muito grande que acabe gerando ruído e excesso de opções, comprometendo os resultados e o tempo de execução da tarefa de desambiguação.

O processo de seleção escolhido para este trabalho foi desenvolvido pelo Italo, orientador deste trabalho, mas que ainda não submeteu seu artigo. No referido trabalho, de início é necessário realizar um pré-processamento dos *tweets* que serão utilizados. Este pré-processamento é necessário porque menções em um *tweet* podem ocorrer tanto no conteúdo textual de um *tweet* como também na forma de *hashtags* – textos precedidos pelo caractere # – ou menções a usuários específicos (e.g. *@Estadao*, *@G1*).

O pré-processamento inicia removendo os caracteres @ e # dessas menções e após isso, através de expressões regulares, separa menções utilizando *camel case* (e.g. *MichaelJordan*) e *snake case* (e.g. *Michael_Jordan*) em *Michael Jordan*. Por fim, apenas a primeira letra de cada palavra de uma menção é mantida em letra maiúscula.

As menções pré-processadas são utilizadas para encontrar o conjunto de entidades candidatas C_i para cada menção m_i. Para encontrar as candidatas, é realizada uma busca em indíce previamente construído [Moussallem et al. 2017]. Este indíce contém, para cada entidade na DBpedia, diversos nomes de superfície pelos quais essas entidades são denominadas e foi implementado na ferramenta ElasticSearch[4]. Para cada menção m_i, são realizadas duas consultas, de maneira simultânea, no ElasticSearch: Correspondência exata/contém e similaridade por *n-gram*. As duas consultas são realizadas para aumentar as chances de que a entidade que descreve corretamente m esteja entre as candidatas

[4]https://www.elastic.co/pt/

retornadas.

Caso nenhuma candidata seja retornada pela consulta, a menção m_i é quebrada em um conjunto de menções menores denominado M_i', caso m_i seja composta por mais de uma palavra. Considerando que uma menção é um conjunto de palavras, nós retiramos uma palavra de m_i enquanto as demais são concatenatas, respeitando a ordem das mesmas. Esse passo é realizado em todas as palavras de m, gerando, portanto, o conjunto M'. Para cada menção $m_ij' \in M_i'$, são realizadas novamente as consultas no ElasticSearch. Caso nenhuma candidata seja retornada, é considerado que para esta menção m_i não existe uma entidade na DBpedia que a descreva corretamente. Portanto, m é ligada ao rótulo *NIL* (*Not Linkable*).

Por fim, caso a menção m_i possua candidatas, é necessário realizar o processo de desambiguação em si, utilizando os *embeddings* gerados e as entidades candidatas encontradas. A desambiguação, assim como a seleção de entidades candidatas, inicia com um passo de pré-processamento adicional dos *tweets* utilizados. Nesta etapa, este processo visa remover algumas informações que não auxiliam no processo de desambiguação, como *URLs*, *emoticons* (representações gráficas de expressões faciais) e alguns ruídos, como por exemplo palavras muito modificadas ou abreviações. Tais elementos podem ser considerados em outras tarefas, como análise de sentimentos, mas são pouco informativos no que diz respeito à desambiguação. O pré-processamento ainda limpa os símbolos # e @, de maneira análoga ao passo de seleção de entidades candidatas.

Após pré-processados, os *tweets* são utilizados em conjunto com as entidades candidatas selecionadas para descobrir quais entidades fazem mais sentido no contexto do *tweet*. De maneira geral, cada menção $m_i \in M$ em um *tweet* é substituída por cada uma das suas entidades candidatas $c_j \in C_i$, de forma a gerar um novo *tweet* para cada candidata. Com os novos tweets, a rede neural determina se a candidata faz sentido ou não no contexto em que ela está inserida. A implementação da rede neural utilizada é feita na forma de um classificador binário.

Mais especificamente, a rede neural utilizada é uma *Bidirectional Long Short-Term Memory (Bi-LTSM)*. O principal motivo para a escolha da LTSM neste trabalho é por ser uma rede neural que leva em consideração a ordem que as palavras aparecem na entrada. Ou seja, a primeira palavra na entrada influencia as próximas, a segunda palavra influencia da terceira em diante, e assim sucessivamente. O *bidirecional* da rede neural também leva em consideração a ordem reversa. Levando em conta que queremos analisar o contexto de um *tweet* ao desambiguar, faz sentido essa abordagem.

Para cada tweet com uma candidata, ou seja, t_j, a rede neural atribui uma probabilidade desse tweet ser coerente com a entidade candidata presente nele. A candidata do tweet t_j que tiver a maior probabilidade de fazer sentido é a escolhida para descrever a menção m_i.

4. Resultados

Como o foco é a desambiguação, nossa avaliação é medida principalmente pela precisão. No entanto, somente a precisão não funciona adequadamente quando o *dataset* não está balanceado, isto é, as classes não estão distribuidas de maneira equilibrada. O *F1 Score* é uma medida que leva em consideração não só a precisão, mas também a cobertura

Tabela 1. Comparação de resultados em termos de *Micro-F1 Score*

Trabalhos	Microposts2014	Microposts2015	Microposts2015
AGDISTIS/MAG	**0,497**	**0,719**	**0,616**
AIDA	0,412	0,414	0,183
Babelfy	0,475	0,341	0,157
DBpedia Spotlight	0,452	ERR	ERR
FOX	0,252	0,311	0,068
FREME NER	0,419	0,313	0,162
OpenTapioca	0,215	0,259	0,053
Este Trabalho	0,040	0,287	0,278

(*recall*) do teste, realizando uma média harmônica entre a precisão e a cobertura. A fórmula utilizada para obter o *F1 Score* é a seguinte:

$$F1 = 2 \times \frac{precisão \times cobertura}{precisão + cobertura}$$

onde a *precisão* mede a quantidade de verdadeiros positivos dividido pelo total de assumidos positivos, certos ou errados. Já a *cobertura* mede a quantidade de verdadeiros positivos sobre o total acertos do modelo, chamados de *elementos relevantes*. O valor obtido varia de 0 a 1, sendo que o 1 é atingido quando tanto a precisão quanto a cobertura são perfeitas.

A Tabela 1 apresenta os resultados desta proposta e de outras ferramentas na plataforma de *benchmarking* para tarefas relacionadas a EL GERBIL[5] [Röder et al. 2018]. O sistema de *benchmark GERBIL* fornece várias informações sobre os experimentos executados. No entanto, a mais relevante para o escopo desse trabalho é o *Micro-F1 Score*, que nos dá a pontuação atingida por cada ferramenta experimentada com base nos acertos e erros cometidos na desambiguação.

Utilizando esta pontuação como base, é possível concluir que o objetivo de obter resultados superiores aos das principais técnicas existentes atualmente foi atingido com êxito parcial. A Tabela 1 traz em sua primeira coluna os trabalhos utilizados para comparação e nas colunas seguintes traz o *Micro-F1 Score* das referidas técnicas para cada um dos *datasets* presentes no cabeçalho da tabela.

Sendo assim, observa-se que utilizando o conjunto de dados *Microposts2016*, este trabalho apresentou um *Micro-F1 Score* superior aos demais, alcançando uma pontuação de 0.2788 enquanto a maioria das técnicas falhou em chegar na pontuação de 0.200. Apenas uma técnica obteve resultados superiores, considerados significantemente altos, por superar a pontuação atingida por este trabalho em aproximadamente 0.300.

Analisando o *dataset Microposts2015*, apesar dos resultados obtidos nesse trabalho terem sido inferiores aos das técnicas utilizadas para comparação, observou-se que a pontuação foi parecida com a pontuação obtida com o conjunto de dados *Microposts2016*. Isso é importante devido ao fato de que o conjunto de 2016 foi utilizado para aprendizado da rede neural, então era esperado seu bom resultado com o conjunto de dados de 2016,

[5]http://gerbil.aksw.org/gerbil/

mas manter uma pontuação semelhante com os dados de 2015 mostra a eficácia do processo, mesmo que outras técnicas tenham obtido resultados melhores com o *dataset* de 2015.

5. Conclusão

Analisando o trabalho realizado e comparando com o esperado pelos objetivos, foi possível concluir que os objetivos foram alcançados com êxito. A técnica proposta e implementada neste trabalho, utilizando os conjuntos de dados de *tweets* com regra ouro mais recentes que encontramos na literatura, conseguiu obter resultados melhores do que os resultados obtidos pela maioria das abordagens do estado da arte, atingindo assim seu objetivo geral.

Referências

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284.

Derczynski, L., Maynard, D., Rizzo, G., Erp, M., Gorrell, G., Troncy, R., Petrak, J., and Bontcheva, K. (2014). Analysis of named entity recognition and linking for tweets. *Information Processing & Management*.

Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. In *SEMANTiCS*.

Fukushima, K. (1975). Cognitron: A self-organizing multilayer neural network. *Biological Cybernetics*, 20:121–136.

Gongqing, W., Ying, H., and Xuegang, H. (2018). Entity linking: An issue to extract corresponding entity with knowledge base. 6:6220–6231.

Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 545–552. Curran Associates, Inc.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. (2014). Deepspeech: Scaling up end-to-end speech recognition.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Landeghem, J. V. (2016). A survey of word embedding literature: Context representations and the challenge of ambiguity.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Moussallem, D., Usbeck, R., Röeder, M., and Ngomo, A.-C. N. (2017). Mag: A multilingual, knowledge-base agnostic and deterministic entity linking approach. In *Proceedings of the Knowledge Capture Conference*, page 9. ACM.

Nguyen, D. Q. (2017). An overview of embedding models of entities and relationships for knowledge base completion.

Omnicore (2018). Twitter by the numbers. `https://www.omnicoreagency.com/twitter-statistics/`. Acesso em Maio de 2018.

Röder, M., Usbeck, R., and Ngomo, A. N. (2018). GERBIL - benchmarking named entity recognition and linking consistently. *Semantic Web*, 9.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

Sak, H., Senior, A., Rao, K., Beaufays, F., and Schalkwyk, J. (2015). Google voice search: faster and more accurate. `https://ai.googleblog.com/2015/09/google-voice-search-faster-and-more.html`. Acesso em Outubro de 2019.

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *Neural Networks*, 61.

Sevgili, Ö., Panchenko, A., and Biemann, C. (2019). Improving neural entity disambiguation with graph embeddings. In *ACL*.

Singhal, A. (2012). Introducing the knowledge graph: things, not strings. `https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html`. Acesso em Outubro de 2018.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA. Association for Computational Linguistics.

Tensorflow (2018). Embeddings. `https://www.tensorflow.org/guide/embedding`. Acesso em Novembro de 2018.

Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge & Data Engineering*, 29(12):2724–2743.

APÊNDICE B – Código Fonte

```
import os

class WordLearner:
  def __init__(self, ft_dir, result_dir, args):
    self._ft_dir = ft_dir
    self._result_dir = result_dir
    self._args = args

  def learn(self):
    print("==== START WORD LEARNING PROCESS ====\n\n")
    os.system(self.get_command())
    print("==== FINISH WORD LEARNING PROCESS ====\n\n")

    return self.result_file()

  def get_command(self):
    command  = "{0}/fasttext skipgram ".format(self._ft_dir)
    command += "-input {0} ".format(self.input_param())
    command += "-output {0} -lr 0.025
    ↪  ".format(self.result_file())
    command += "-dim {0} -ws 5 ".format(self.dim_param())
    command += "-epoch {0} -minCount 5
    ↪  ".format(self.epoch_param())
    command += "-neg 5 -loss ns -bucket 2000000 "
    command += "-minn 3 -maxn 6 -thread 4 -t 1e-4 "
    command += "-lrUpdateRate 100"

    return command

  def input_param(self):
    return self._args["-winput"] if "-winput" in
    ↪  self._args.keys() else Exception("missing input file")

  def dim_param(self):
    return self._args["-wdim"] if "-wdim" in self._args.keys()
    ↪  else "200"

  def epoch_param(self):
    return self._args["-wepoch"] if "-wepoch" in
    ↪  self._args.keys() else "100"

  def result_file(self):
    return "{0}/fil9".format(self._result_dir)
```

```
import os

class KnowledgeLearner:
  def __init__(self, ft_dir, output_dir, args):
    self._ft_dir = ft_dir
    self._output_dir = output_dir
    self._args = args

  def learn(self):
    print("==== START KNOWLEDGE LEARNING PROCESS ====\n\n")
    os.system(self.get_command())
    print("==== FINISH KNOWLEDGE LEARNING PROCESS ====\n\n")

    return self.output_file()

  def get_command(self):
    command  = "{0}/fasttext supervised ".format(self._ft_dir)
    command += "-input {0} ".format(self.input_param())
    command += "-dim {0} ".format(self.dim_param())
    command += "-epoch {0} ".format(self.epoch_param())
    command += "-output {0} ".format(self.output_file())
    command += "-lr .2 -thread 20 -loss ns -neg 100 -minCount 0"

    return command

  def input_param(self):
    return self._args["-kinput"] if "-kinput" in
    ↪  self._args.keys() else Exception("missing input file")

  def dim_param(self):
    return self._args["-kdim"] if "-kdim" in self._args.keys()
    ↪  else "200"

  def epoch_param(self):
    return self._args["-kepoch"] if "-kepoch" in
    ↪  self._args.keys() else "100"

  def output_file(self):
    return "{0}/fb15".format(self._output_dir)
```

```
import sys
import configparser
from learner.word_learner import WordLearner
from learner.knowledge_learner import KnowledgeLearner

class Learner:
  _word_learning_args = dict()
  _knowledge_learning_args = dict()
  _ft_dir = str()
  _output_dir = str()

  def __init__(self, args):
    self._args = dict(zip(*[iter(args)]*2))
    self._parser = configparser.SafeConfigParser()

  def learn(self):
    print("==== START LEARNING PROCESS ====\n\n")
    self.split_args()
    self._parser.read("../.config")
    self._ft_dir =  self._parser.get('directories', 'fast_text')
    self._output_dir = self._parser.get('directories', 'output')

    word_learner = WordLearner(self._ft_dir, self._output_dir,
    ↪  self.get_word_learning_args())
    word_output = word_learner.learn()

    knowledge_learner = KnowledgeLearner(self._ft_dir,
    ↪  self._output_dir, self.get_knowledge_learning_args())
    knowledge_output = knowledge_learner.learn()

    print("==== FINISH LEARNING PROCESS ====")

    print("Preparing training file...")
    self.concat_results([word_output + ".vec", knowledge_output +
    ↪  ".vec"])
    print("Finished! File present in
    ↪  {0}/training_result.txt".format(self._output_dir))



  def concat_results(self, outputs):
    with open(self._output_dir + '/training_result.txt', 'w') as
    ↪  training_file:
      training_file.write(self.parse_first_line(outputs))

      for output in outputs:
```

```
      with open(output, 'r') as f:
        # skip first line
        f.readline()

        while True:
          line = f.readline()
          if not line: break
          training_file.write('{0}'.format(line))

def parse_first_line(self, outputs):
  count = 0
  dim = 0

  for output in outputs:
      with open(output, 'r') as f:
        line = f.readline().split()
        count += int(line[0])
        dim = int(line[1])

  return "{0} {1}\n".format(count, dim)

def split_args(self):
  for k,v in self._args.items():
    if k[1] in "w":
      self._word_learning_args[k] = v
    elif k[1] in "k":
      self._knowledge_learning_args[k] = v
    else:
      print("'{0}' argument unknown. Being
       ↪  discarted...".format(k))

def get_word_learning_args(self):
  return self._word_learning_args

def get_knowledge_learning_args(self):
  return self._knowledge_learning_args
```

```
import sys
from learner.learner import Learner

def run_learner():
  args = sys.argv[1:]

  learner = Learner(args)
  learner.learn()

run_learner()

import re

def run_parser():
  regex = re.compile('^.*\d_dbo\S*.*$')
  results = list()

  with open('results_file.txt', 'w') as output_file:
    with open('full_training.txt', 'r') as input_file:
      while True:
        line = input_file.readline()
        if not line: break
        result = re.finditer(regex, line)
        for match in result:
          print(match)
          output_file.write('{0}\n'.format(match.group(0).strip()))

run_parser()
```

```
#### Configuration file
#
# This file is used to configure the sorting order. It must
# contain a path to a fastText directory and to a result directory
#
#
#       Example:
#       fast_text = ../fastText
####

[directories]
fast_text = ../../fastText
output = ../outputs
```