UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Arthur Silva Sens, Isabelle Pinheiro

USO DE CLASSIFICADORES BINÁRIOS EM EMBEDDINGS DE PALAVRAS E CONHECIMENTO NA TAREFA DE LIGAÇÃO DE ENTIDADES

Florianópolis

2019

Arthur Silva Sens, Isabelle Pinheiro

USO DE CLASSIFICADORES BINÁRIOS EM EMBEDDINGS DE PALAVRAS E CONHECIMENTO NA TAREFA DE LIGAÇÃO DE ENTIDADES

Trabalho de Conclusão de Curso submetido ao curso de Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação.
Orientador: Me. Italo Lopes Oliveira
Coorientador: Prof. Dr. Renato Fileto

Florianópolis

2019

RESUMO

Anotar semanticamente dados cuja semântica não é bem definida ou processável por máquinas, tais como grandes quantidades de dados semi ou não-estruturados atualmente disponíveis na Web, tem o potencial de alavancar aplicações que podem tirar proveito de interpretações automáticas de tais dados. Entretanto, os atuais processos automáticos de anotação semântica falham em entregar resultados com boa qualidade. Uma maneira de melhorar a qualidade dos resultados gerados pelos processos de anotação atuais é considerar um contexto mais amplo no qual os dados se encontram. Contextos semânticos construídos a partir de algumas anotações confiáveis podem auxiliar no processo de desambiguação de novas anotações. Eles podem ser representados como *embeddings*, que são uma categoria de modelos de processamento natural da linguagem que mapeiam matematicamente as palavras para vetores numéricos. Esse recurso facilita e agiliza a determinação de palavras (vetores) semelhantes ou clusters vetoriais. Este trabalho propõe algoritmos para a criação e utilização de contextos semânticos usando técnicas de aprendizado de máquina em grafos, para melhorar o processo de desambiguação de novas anotações. Os resultados das anotações produzidas pelo modelo proposto são comparadas com aqueles de ferramentas do estado da arte em anotação semântica de dados textuais.

Palavras-chave: Contexto, anotação semântica, desambiguação, aprendizado de máquina em grafos, dados textuais, postagens de mídias sociais.

ABSTRACT

Semantically annotating data whose semantics are not well-defined by machines, such as large amounts of semi-structured unstructured data currently available on the Web, has the potential to leverage applications that can take advantage of automatic interpretations of such data. However, the current automatic processes of semantic annotation fail to deliver results with good quality. One way to improve the quality of the results generated by the current annotation processes is to consider a broader context in which the data are found. Semantic contexts constructed from reliable annotation can help in the disambiguation process of new annotations. They can be represented as embedding, which is a category of natural language processing models that mathematically map the words to numerical vectors. This feature makes it much easier and streamlines the determination of similar words (vectors) or vector clusters This work proposes algorithms for the creation and use of semantic contexts using machine learning techniques in graphs to improve the disambiguation process of new annotations. The results of the annotations produced by the proposed model are compared with those of state of the art tools in semantic annotation of textual data.

Keywords: Context, semantic annotation, disambiguation, machine learning in graphs, textual data, postings of social media.

LISTA DE FIGURAS

LISTA DE TABELAS

LIST OF ALGORITHMS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Muitos dados não estruturados e semi-estruturados estão atualmente disponíveis na Web (e.g., documentos de texto, documentos HTML (*Hypertext Markup Language*), postagens em mídias sociais). Entretanto, a aplicação destes dados é prejudicada por dificuldades em processar computacionalmente sua semântica correta (LAENDER, 2002). O não entendimento do significado de termos e menções à entidades do mundo real presentes em dados textuais impede o uso dos mesmos em todo o seu potencial. Uma maneira de atenuar tal problema é anotá-los semanticamente (BONTCHEVA; ROUT,).

Uma anotação semântica descreve o sentido de dados (ou partes deles) mediante a associação de tais dados a recursos de informação com semântica bem definida (e.g., conceitos, instâncias de conceito) (KOIVUNEN et al., 2001) presentes em, por exemplo, grafos de conhecimento (*Knowledge Graph* - KG) (e.g., DBpedia (AUER et al., 2007) (LEHMANN et al., 2009), Yago (FABIAN; GJERGJI; GERHARD, 2007), Freebase (BOLLACKER et al., 2008)). Entretanto, anotar manualmente dados disponíveis na Web é um processo inviável, seja pela quantidade proibitiva de dados disponíveis, seja pela dificuldade em produzir resultados padronizados e com qualidade. Assim, diversos processos automáticos de anotação semântica têm sido propostos.

Entre os diversos processos automáticos de anotação existentes, destaca-se a ligação de entidades (*Entity Linking* - EL). EL é uma tarefa de anotação semântica responsável por identificar menções a entidades do mundo real em um documento de texto (este passo é chamado de reconhecimento de entidades nomeadas, *Named Entity Recognition* - NER) e ligar cada menção encontrada a uma definição semanticamente bem descrita daquilo a que cada menção se refere (este passo é chamado de desambiguação de entidades nomeadas, *Named Entity Disambiguation* - NED) (RATINOV L., 2011; SHEN; WANG; HAN, 2015). A etapa de desambiguação da EL é o foco deste trabalho.

No entanto, abordagens e ferramentas automáticas atuais para anotação semântica (e.g., DBpedia-Spotlight (MENDES, 2011), FOX (SPECKAXEL; NGOMO, 2014), Babelfy (MORO; RAGANATO; NAVIGLI, 2014)) falham em entregar anotações com boa precisão e cobertura, principalmente quando aplicadas a documentos de texto com muitos ruídos (e.g., erros ortográficos e gramaticais) e linguagem coloquial (e.g. uso de gírias), tais como postagens em mídias sociais (BONTCHEVA; ROUT,). Uma das razões para a baixa precisão e cobertura é que tais

ferramentas não consideram todo o contexto (e.g., textual, semântico) que envolvem as anotações e os dados a serem anotados.

Uma forma de contornar tal limitação é considerar o contexto existente em dados textuais já anotados, como, por exemplo, considerar as relações entre as palavras e os recursos presentes em grafos de conhecimento. A exploração de tais relações, que podem ser expressas através de embeddings de palavras e conhecimento por técnicas de *Machine Learning* - ML), podem ajudar a desambiguar com maior precisão novas anotações.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

Mídias sociais são uma grande fonte de informações. A cada minuto milhares de pessoas ao redor do globo publicam conteúdo como fotos, textos e/ou vídeos, onde expressam seus interesses, experiências e opiniões[1].

No entanto, postagens em mídias sociais apresentam um novo e desafiador estilo de texto para as tarefas de anotação semântica. Postagens de mídias sociais podem apresentar padrões linguísticos dinâmicos, como natureza informal; ruídos, tais como erros ortográficos e gramaticais; gírias; etc. Além disso, também podem apresentar contexto limitado, devido a restrição de caracteres. Tais fatores tornam o processo de anotação semântica particularmente complexo, sendo difícil obter bons resultados por meio dos métodos e ferramentas atuais.

A Figura 1 mostra algumas das dificuldades de anotar semanticamente dados textuais de modo automático devido a contexto limitado. Considerando, respectivamente, as postagens A e B, ambas extraídas da plataforma Twitter[2], não há garantias que a menção *George Lucas* em ambas postagens se referem à mesma entidade no mundo real. Isso ocorre devido ao pouco contexto presente nas postagens, devido à limitação de caracteres imposta pelo Twitter. Além disso, diferentes entidades podem ter o mesmo nome de superfície (termo utilizado para se referir a alguma entidade), como exemplificado na Figura 1.

Uma maneira de amenizar o problema de contexto limitado é considerar as relações entre as palavras e entidades candidatas, além das relações entre as entidades candidatas. Uma maneira de explorar tais relações é através do uso de vetores n-dimensionais (*embeddings*) que representem as palavras e entidades de um KG em técnicas de

[1] https://www.visualcapitalist.com/internet-minute-2018/
[2] https://twitter.com/

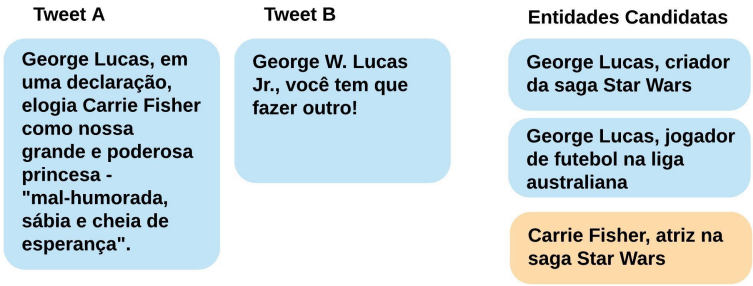

Figura 1 – Exemplo de ambiguidade presente em postagens de mídias sociais devido ao contexto limitado.

machine learning.

Trabalhos recentes na área mostram que o uso de *embeddings* de palavras em conjunto com o uso de *embeddings* de entidades pode ajudar a desambiguar melhor as entidades candidatas encontradas. No entanto, tais trabalhos treinam ambos os *embeddings* de maneira separada. Além disso, os *embeddings* de entidades são gerados a partir de textos de páginas da Wikipedia e não através da estrutura de grafos presente em grafos de conhecimento (*embeddings* gerados dessa forma recebem o nome de *embeddings* de conhecimento). O uso de *embeddings* de conhecimento combinados a *embeddings* de texto para auxiliar a desambiguação de menções ainda não foi suficientemente explorado na literatura.

Trabalhos atuais focam em redes neurais de aprendizado profundo (*deep learning*) como técnica de *machine learning* para explorar o uso de *embeddings* de palavras e entidades na tarefa de EL. Todavia, redes de *deep learning* são computacionalmente pesadas, exigindo máquinas relativamente caras e poderosas para seu treinamento. Devido a isso, organizações que possuem recursos financeiros ou processamento computacional limitado não podem utilizar tais soluções por questões de eficiência.

Portanto, este trabalho parte da hipótese de que a utilização de *embeddings* de palavras e conhecimento, gerados em conjunto, em técnicas de *machine learning* tradicionais, e computacionalmente mais leves que *deep learning*, têm o potencial de produzir resultados iguais ou melhores na etapa de desambiguação da tarefa de EL para postagens de mídias sociais.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é propor e implementar um método que utilize *embeddings* de palavras e conhecimento de forma conjunta no processo de desambiguação de novas anotações semânticas em postagens de mídias sociais utilizando técnicas de *machine learning* computacionalmente menos exigentes do que redes neurais profundas.

1.2.2 Objetivos Específicos

Os objetivos específicos deste trabalho são:

1. Melhorar o entendimento do estado-da-arte em anotação semântica, mais especificamente *Entity Linking*;
2. Treinar modelos com diferentes técnicas de *machine learning* utilizando *embeddings* de palavra e conhecimento como entrada;
3. Propor e implementar um método de desambiguação com base nos modelos gerados;
4. Validar os métodos propostos através de experimentos.

1.3 METODOLOGIA

Esta seção apresenta a metodologia a ser seguida para a execução deste trabalho, a qual está resumida no fluxograma apresentado na Figura 2. A metodologia do trabalho é dividida em 8 partes. Primeiramente é realizado um estudo bibliográfico para entender o estado-da-arte em processos de anotação semântica e técnicas *Statistical Relational Learning* - SRL). São utilizados como fonte de pesquisa as bases *ACM Digital Library* e *Springer*, além do indexador *Google Scholar*. Tais bases e indexador foram escolhidos porque apresentaram melhores resultados em uma pesquisa preliminar.

A segunda parte da metodologia envolve selecionar algoritmos de *machine learning* para classificação binária. Esta proposta utiliza classificadores binários devido que o problema de desambiguação é modelado de modo a classificar se uma entidade candidata faz sentido ou

não no contexto em que ela se encontra. Mais detalhes são explicados no Capítulo 4. Em seguida, o conjunto de treinamento e testes são preparados com base em conjuntos de dados disponíveis na literatura. São gerados modelos para diferentes classificadores binários. Esses modelos são avaliados através de uma métrica (detalhada no Capítulo 2), sendo que o melhor deles, baseado nessa métrica, é utilizado na desambiguação de anotações. A desambiguação das anotações é realizada com a proposta detalhada no Capítulo 4. Os resultados são analisados e é feita uma comparação com os trabalhos relacionados na área de interesse.

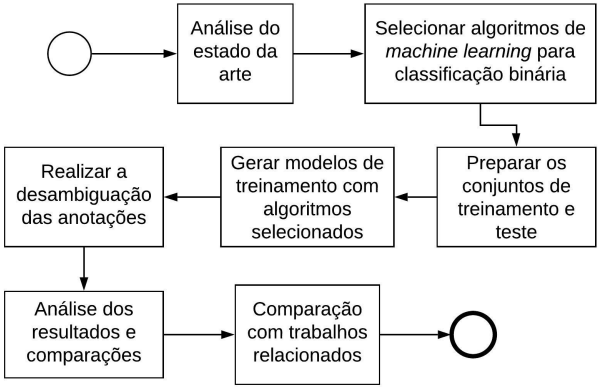

Figura 2 – Fluxograma da metodologia a ser seguida nesta proposta.

1.4 ORGANIZAÇÃO DO TEXTO

O trabalho está estruturado como segue. O Capítulo 2 apresenta a fundamentação teórica, descrevendo os conceitos básicos para o entendimento deste trabalho e justificando o uso das tecnologias escolhidas para o desenvolvimento. O Capítulo 3 apresenta os trabalhos relacionados, suas características e abordagens escolhidas. O Capítulo 4 apresenta o algoritmo proposto para a desambiguação de anotações semânticas com base em técnicas de embeddings de palavras e conhecimento. O Capítulo 5 relata experimentos realizados para avaliar a proposta, apresenta os resultados obtidos e os discute. Por fim, o Capítulo 7 apresenta as considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os fundamentos necessários para compreensão do trabalho desenvolvido. São apresentados os conceitos de EL, grafo de conhecimento, embeddings de forma geral, embeddings de conhecimento, embeddings de palavras, classificadores binários e as medidas de desempenho que foram utilizadas.

Como já apresentado no Capítulo 1, EL é uma tarefa de anotação semântica que visa identificar menções à entidades nomeadas em um texto e ligar cada menção à um recurso, semanticamente bem descritos, que melhor descreva a menção no contexto em que ela se encontra. A Definição 2.1.1 apresenta uma definição formal do EL baseada nas definições propostas por (NADEAU; SEKINE, 2007; SHEN; WANG; HAN, 2015).

2.1 ENTITY LINKING

Definition 2.1.1 (Entity Linking) *Dado um conjunto de entidades E em um grafo de conhecimento K (i.e., $E \subset K$) e um conjunto de entidades nomeadas M em um texto T (i.e., $M \subset T$), a tarefa EL visa reconhecer o conjunto de menções a entidades $\boldsymbol{M}$ no texto $\boldsymbol{T}$ e ligar cada menção $m \in \boldsymbol{M}$ à entidade $e \in \boldsymbol{E}$ que melhor descreve m no contexto onde ela aparece em T.*

Além de fornecer as entidades que serão apontadas pelas anotações semânticas, os recursos (entidades e relacionamentos) presentes em grafos de conhecimento, tais como DBpedia, Yago e outros, têm sido utilizados por diversas abordagens (SINGLA; DOMINGOS, 2006; BHATTACHARYA; GETOOR, 2007; HAN; SUN; ZHAO, 2011; WHANG; GARCIA-MOLINA, 2012; HUANG et al., 2014; KALLOUBI; NFAOUI et al., 2016; LI et al., 2016; GANEA et al., 2016; CHONG; LIM; COHEN, 2017) na tarefa EL para enriquecer textos semanticamente. Devido à sua importância na tarefa EL, este trabalho apresenta uma definição formal de grafos de conhecimento na Definição 2.1.2 baseada na definição dada por (NICKEL et al., 2016).

Definition 2.1.2 (Grafo de Conhecimento) *Dado um conjunto de entidades E e um conjunto de relações L, um grafo de conhecimento K é composto por triplas $< e_i, l_k, e_j >$, onde $e_i, e_j \in E$ e $l_k \in L$. No entanto, tal definição engloba todas as possiveis triplas entre $E \times$*

$L \times E$. Devido a isso, este trabalho utiliza os termos K^+ e K^- para definir, respectivamente, o subconjunto de triplas ($K^+ \subset K$) que são verdadeiras (i.e., efetivamente existem em K) e o subconjunto de triplas $K^- \subset K$ que são falsas (i.e., não existem em K).

A Definição 2.1.2 permite que grafos de conhecimento sejam expressos como triplas *Resource Description Framework (RDF)* (GROUP, 2018), as quais podem ser utilizadas para expressar um grafo. RDF é uma família de especificações da World Wide Web Consortium (W3C) planejada como um modelo para representar conhecimento e dados em acordo com um conhecimento, os quais podem ser usado como metadados. Esse tipo de especificação vem sendo utilizada para a descrição conceitual ou de modelagem de informação na Web Semântica e está implementada em diversas ferramentas e sistemas na Web, utilizando uma variedade de sintaxes e formatos de serialização.

Por exemplo, a informação "O ator Leonardo DiCaprio interpretou o personagem Jack Dawson no clássico filme de drama Titanic" pode ser expressa através do seguinte conjunto de triplas Sujeito (e_i) Predicado (l_k) Objeto (e_j) (SPO) que consta na Tabela 1:

Conjunto de triplas SPO		
Sujeito	**Predicado**	**Objeto**
Leonardo DiCaprio	profissão	Ator
Leonardo DiCaprio	estrelou em	Titanic
Leonardo DiCaprio	interpretou	Jack Dawson
Jack Dawson	personagem em	Titanic
Titanic	gênero	Drama

Tabela 1 – Conjunto de triplas SPO.

2.2 EMBEDDINGS

Embeddings são vetores n-dimensionais que representam dados de dimensionalidade maior mantendo as suas propriedades. O uso de *embeddings* são frequentemente utilizados em técnicas de *machine learning* que recebem entradas maiores, como vetores esparsos que representam palavras. Idealmente, um *embedding* captura a semântica do dado que ele representa, colocando dados semelhantes próximos um dos outros em um mesmo espaço vetorial.

2.2.1 Embedding de palavras

Nos últimos anos, o uso de *embeddings* de palavras tornou-se um dos métodos mais populares entre a comunidade de processamento de linguagem natural (NPL) (RUDER; VULIĆ; SØGAARD, 2017). *Embeddings* de palavras é um tipo de representação que permite que palavras com significado similar possuam representação similar.

Técnicas de *embeddings* de palavras transformam palavras em vetores n-dimensionais. Palavras relacionadas, como, por exemplo, *casa* e *lar* são mapeadas para vetores n-dimensionais semelhantes, enquanto palavras diferentes, como, por exemplo, *casa* e *carro*, possuem vetores diferentes. Devido a esta característica, o significado de uma palavra pode ser refletido em seu *embedding*. Um modelo pode usar essas informações para aprender o relacionamento entre as palavras. O principal ganho em utilizar *embeddings* de palavras é que um modelo treinado na palavra *casa* será capaz de reagir à palavra *lar*, mesmo que nunca tenha visto essa palavra em treinamento.

Observa-se na Figura 3 que *embeddings* de palavras similares possuem representações similares e vão ficar próximas entre si, mesmo que estejam escritas em idiomas diferentes.

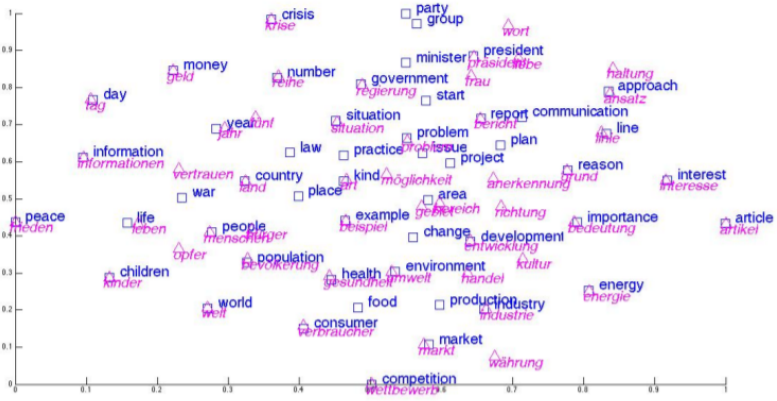

Figura 3 – Um espaço de embedding compartilhado entre dois idiomas (Luong et al., 2015)

Utiliza-se, portanto, do recurso de *embeddings* de palavras nesse trabalho, pois existe a necessidade da representação de palavras vindas de microtextos e que consiga-se entender de que forma diferentes

palavras estão relacionadas entre si.

2.2.2 Embeddings de Conhecimento

Embeddings de conhecimento são representações de recursos de um KG em vetores densos n-dimensionais, preservando a estrutura inerente do KG e simplificando sua manipulação. A maioria das técnicas realiza a geração de embeddings com base nas triplas presentes em K^+. Isso ocorre devido a dificuldade de se prever corretamenta triplas em K^-, pois não se sabe se a tripla, quando existir, será correta ou não (NICKEL et al., 2016; WANG et al., 2017).

Técnicas de *embeddings* de conhecimento geralmente consistem em três etapas: (i) representar entidades e relações, (ii) definir uma função de pontuação e (iii) aprendizado de entidade e relação de representações. A etapa (i) especifica a forma na qual entidades e relações são representadas em um vetor de espaço contínuo. As entidades são geralmente representadas como vetores, enquanto relações são normalmente tomadas como operações no espaço vetorial. Então, na segunda etapa, uma função de pontuação é definida para cada tripla para medir sua coerência de acordo com K^+. As triplas observadas no KG tendem a ter pontuações mais altas do que aquelas que não foram observadas (i.e., triplas presentes em K^-). Finalmente, para aprender as representações de entidade e relação (ou seja, *embeddings*), a terceira etapa resolve um problema de otimização que maximiza a coerência total das triplas observadas (isto é, fatos contidos em K^+) (WANG et al., 2017).

A representação de recursos de um KG como vetores permite a manipulação para diversos fins, como: verificar se entidades são semelhantes; analizar se o relacionamento entre duas entidades é coerente (Figura 4), por exemplo. Para a tarefa de EL, pode-se inferir que, dado uma menção $m \in M$ e uma entidade $e \in E$, a entidade e será considerada a desambiguação correta de m caso o *embedding* de e seja coerente com os *embeddings* de palavras ao seu redor.

2.3 CLASSIFICAÇÃO BINÁRIA

Uma vasta diversidade de problemas envolvem a classificação de dados em categorias, também conhecidas como classes. A partir de um conjunto de dados cujas classes são conhecidas, algoritmos de *machine learning* supervisionados podem ser utilizados para treinar

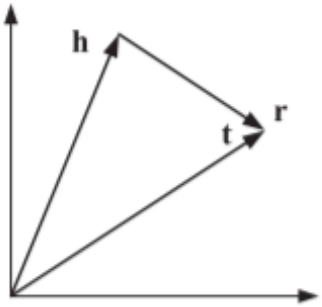

Figura 4 – Entidade e espaço de relação

um classificador capaz de predizer a classe de novos dados do mesmo domínio (LORENA; CARVALHO, 2008). Alguns exemplos de situações em que os classificadores podem ser aplicados são: verificar se um e-mail recebido é *spam* ou não, avaliar o resultado final de um procedimento médico, identificar se o perfil de um cliente é adequado para a empresa ou não, identificar se uma pessoa é homem ou mulher, etc. (ALMEIDA, 2010) (MIN; JEONG, 2009).

Existem dois tipos de classificadores: binários e multiclasses. Classificadores binários são ferramentas populares em *machine learning*. Diversas técnicas de ML são originalmente formuladas para a solução de problemas de classificação binários. Entre elas pode-se citar o Suport Vector Machine (SVM), que é um dos classificadores mais comuns para se trabalhar com EL (RENNIE; RIFKIN, 2001). Esse trabalho utilizará de classificadores binários afim de verificar se um determinado *tweet* tem relação com uma certa entidade candidata e avaliar entre as entidades candidatas obtidas, qual delas possuem mais similaridade com nosso dado de entrada.

2.4 MEDIDAS DE DESEMPENHO

Nesse trabalho são utilizadas como medidas de desempenho a curva ROC como métrica de classificação para avaliar os classificadores, a qual será melhor explicada no capítulo 4. Também escolhemos utilizar as medidas F1 macro e micro para desenvolvimento dos experimentos. A seguir, faremos uma breve explicação sobre ambas as medidas e suas respectivas utilizações em nosso modelo.

2.4.1 Curva ROC

Dentre diferentes medidas existentes para alivar o desempenho de classificadores binários, este trabalho utiliza a área sob a curva ROC (*Area Under The Curve ROC - AUC-ROC*). A curva *Receiver Operator Characteristic*(ROC) é amplamente utilizada para verificar ou visualizar a performance de um problema de classificação (DAVIS; GOADRICH, 2006).

A curva ROC é usada para comparação visual de modelos de classificação que mostram o *trade-off*, ou limite, entre a taxa positiva verdadeira e a taxa positiva falsa (EKELUND, 2012). A área sob a curva ROC é uma medida da precisão do modelo. Quando a curva que representa um modelo está mais próximo da diagonal, é menos preciso e o modelo com perfeita precisão terá uma área de 1,0 (FAN; UPADHYE; WORSTER, 2006). Uma vantagem na utilização da medida AUC-ROC em detrimento de medidas convencionais de precisão e cobertura é que a medida AUC-ROC é pouco influenciada pelo desbalanceamento de classes positivas e negativas no conjunto de teste.

2.4.2 Medida F1

A métrica de pontuação F1 é uma métrica de classificação que calcula a média harmônica da precisão e *recall*, ou cobertura, onde uma pontuação F1 atinge seu melhor valor em 1 (precisão e cobertura ideais) e o pior em 0.

Neste trabalho, são utilizados duas variações da métrica F1: F1 micro e F1 macro. O F1 micro é obtido da soma dos verdadeiros positivos e verdadeiros negativos de todas as instâncias do conjunto de teste. Deste modo, o F1 micro é uma medida que mostra o desempenho do classificador no conjunto de dados como um todo. Já o F1 macro é a média da métrica F1 obtida para cada instância do conjunto de teste. Portanto, o F1 macro mostra o desempenho, na média, do classificador para cada instância (USBECK et al., 2015).

3 TRABALHOS RELACIONADOS

Este capítulo apresenta e discute os principais trabalhos relacionados à proposta deste trabalho. A Tabela 2 lista abordagens para EL que usam *embeddings* ou cujo foco seja anotar semanticamente postagens de mídias sociais, particularmente *tweets*.

Os trabalhos na Tabela 2 estão ordenados por sua data de publicação e são comparados segundo três critérios especificados nas demais colunas: Textos anotados; Entradas externas; Métodos. A coluna Textos anotados informa o tipo de documento de texto a ser anotado, podendo ser classificado como notícias, textos literários etc, afim de explicitar o tipo de texto formal utilizado no trabalho, ou como *tweets* (postagens de mídias sociais). Entradas externas são as entradas utilizadas para anotar semanticamente os documentos de texto. Por fim, a coluna Métodos apresenta a principal técnica/algoritmo/método a ser utilizada pelo trabalho para tratar a tarefa de EL.

Na coluna Tipos de dados, percebe-se que poucos são os trabalhos atuais que trabalham com *tweets*. Isso se deve a diversos motivos, entre eles, dificuldade de obtenção de *tweets*, poucos conjuntos de dados públicos disponíveis e questões de privacidade. Na coluna Entradas externas, a maioria dos trabalhos utilizam como fonte de dados utilizados para anotação semântica a Wikipedia. Alguns trabalhos, como (LUO et al., 2015; FANG et al., 2016; CHEN et al., 2018; LE; TITOV, 2018), utilizam outros KG (Freebase, DBpedia) para complementar os dados provenientes da Wikipedia. Somente (KALLOUBI; NFAOUI et al., 2016; ZHU; IGLESIAS, 2018; SEVGILI; PANCHENKO; BIEMANN, 2019; PARRAVICINI et al., 2019) utilizam a DBpedia como fonte para as anotações semânticas. Essa diferença entre Wikipedia e outros KGs ocorre por que KGs, apesar de serem estruturados em triplas, possuem consideravelmente menos dados textuais que as páginas da Wikipedia. Além disso, algumas estatísticas das páginas da Wikipedia, como número de visitas, não estão disponíveis nos KGs. Por fim, conclui-se pela coluna Métodos, que grande parte das abordagens que utilizam *embeddings* são voltadas para textos formais e baseadas em redes neurais. Tais redes neurais são computacionalmente pesadas, exigindo máquinas com poder de processamento considerável. Por exemplo, no artigo (CHEN et al., 2018) é utilizado um computador com um processador Intel (R)CPU Xeon (R) E5-2620 que possui 8 núcleos, podendo utilizar até 16 threads. Já no artigo (PARRAVICINI et al., 2019), o desenvolvimento do trabalho utiliza uma máquina com um processador Intel Xeon CPU E5-

2680 v2 com 20 núcleos, 40 threads em 2.80GHz e **378GB** de memória RAM. Trabalhos com redes neurais produzem resultados satisfatórios para textos formais, porém necessitam de máquinas poderosas e custosas de manter financeiramente, o que é inviável para organizações com recursos computacionais ou financeiros limitados.

Tabela 2 – Comparação de abordagens holísticas para Entity Linking

Artigo	Textos anotados	Entradas externas	Métodos
(GUO et al., 2013)	Tweets	Freebase, Wikipedia	Modelo de espaço vetorial (VSM) e Aprendizado de Classificação (LTR)
(FANG; CHANG, 2014)	Tweets	Wikipedia	Extração de informação e Recuperação de informação
(DERCZYNSKI et al., 2015)	Tweets	Lupedia, DBpedia, TextRazor, Zemanta	FFNN (Embedding de palavras)
(FANG et al., 2016)	Noticiários	Wikipedia, Freebase	Regressão logística para o modelo de duas camadas (Embedding de palavras, Embedding de conhecimento)
(YAMADA et al., 2016)	Documento de texto	Wikipedia	Regressão aumentada por gradiente Árvores (Embedding de palavras, Embedding de entidades)
(MORENO et al., 2017)	Noticiários	Wikipedia	Classificadores binários (Embedding de palavras, Embedding de entidades)
(CHEN et al., 2018)	Noticiários	Wikipedia, Freebase	Árvore de regressão aumentando em pares (Embedding de palavras, Embedding de entidades)
(LE; TITOV, 2018)	Documento de texto	Wikipedia, Yago	Campo aleatório condicional, loopy belief propagation (Embedding de palavras, Embedding de entidades)
(KOLITSAS; GANEA; HOFMANN, 2018)	Documento de texto	Wikipedia	Shallow FFNN and LSTM (Embedding de palavras, Embedding de entidades)
(ZHU; IGLESIAS, 2018)	Document de texto	DBpedia	Algoritmo de similaridade contextual semântica (Embedding de palavras, Embedding de categorias)
(MARTINS; MARINHO; MARTINS, 2019)	Noticiários	Wikipedia	Stack-LSTM (Embedding de palavras, Embedding de entidades)
(SEVGILI; PANCHENKO; BIEMANN, 2019)	Noticiários	Wikipedia, DBpedia	FFNN (Embedding de palavras, Embedding de grafos)
(WANG; IWAIHARA, 2019)	Noticiários	Wikipedia	TNN and CNN (Embedding de palavras)
(PARRAVICINI et al., 2019)	Noticiários	DBpedia	Similaridade semântica(Embedding de grafos) e heurística de pesquisa no espaço de estados
(LIU et al., 2019)	Noticiários	Wikipedia	Algoritmo Forward-Backward (Embedding de entidades)
(HOSSEINI et al., 2019)	Tweets	DBpedia	Redes neurais e embedding à estrutura de recuperação baseada em MRF
(HAN et al., 2019)	Tweets	DBpedia	Embedding de entidades
(SENS; PINHEIRO, 2019)	Tweets	DBpedia	Embedding de palavras, Embeddings de conhecimento

4 PROPOSTA

Este capítulo apresenta, respectivamente, o método para o treinamento dos modelos de classificação a serem utilizados para desambiguação de anotações da tarefa EL e o método para a etapa de desambiguação da tarefa EL em si.

A Figura 5 exemplifica os passos realizados pela proposta deste trabalho para a anotação semântica de um *tweet*. A proposta deste trabalho considera que a etapa de desambiguação do EL pode ser resumida como um problema de classificação binária. No caso, as entidades que descrevem as menções são classificadas como *positivas* caso façam sentido na postagem ou como *negativas* caso não façam sentido.

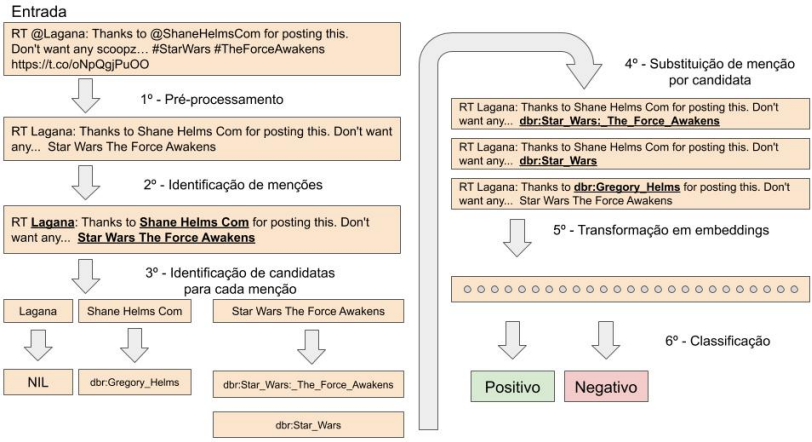

Figura 5 – Fluxograma geral da proposta

4.1 PRÉ-PROCESSAMENTO

O método proposto recebe como entrada uma postagem de mídia social, no caso um *tweet*. O primeiro passo do modo proposto é o pré-processamento, que consiste em: (i) retirar links; (ii) palavras consideradas sem relevância; (ii) remoção de alguns caracteres especiais, como o # e @; (iv) a quebra de palavras, como apresentado no exemplo onde "StarWars"foi transformado em "Star Wars".

4.2 RECONHECIMENTO DE MENÇÕES

Após o pré-processamento, usalmente o próximo passo em uma abordagem EL é o reconhecimento de entidades nomeadas no texto. Este passo, no entanto, não é abordado nesta proposta, pois o foco da mesma é tratar a etapa de desambiguação. Devido a isso, é considerado que as menções já foram identificadas, seja manualmente ou por alguma ferramenta do estado-da-arte.

4.3 BUSCA DE ENTIDADES CANDIDATAS PARA MENÇÕES

Com as menções identificadas, a próxima etapa desta proposta visa encontrar possíveis entidades que descrevam corretamente cada menção, ou seja, suas entidades candidatas. Para encontrar as entidades candidatas, os recursos da DBpedia que descrevem entidades foram transformados em documentos a serem pesquisados na ferramenta ElasticSearch[1]. Elasticsearch é um indexador de documentos especializado em buscas textuais com alta performance e que fornece inúmeras estratégias para realizar as buscas.

Cada documento indexado possui os seguintes atributos: o endereço da entidade na DBpedia (URI), possíveis nomes relacionados à entidade (nome de superfície) e o texto introdutório da entidade em questão (Contexto). Os documentos inseridos no ElasticSearch foram obtidos através de (MOUSSALLEM et al., 2017).

A estratégia de busca utilizada no ElasticSearch é a pesquisa por N-Grams, onde a menção é quebrada em caracteres de tamanho n e a busca é realizada por similaridade com os nomes de superfície dos documentos armazenados, que também são quebrados em partes de tamanho n. Caso nenhuma entidade candidata seja encontrada para a menção, como é no caso para a menção Lagana, é atribuída a menção o valor *NIL*, indicando que não foi encontrada no KG uma entidade que a descreva corretamente.

A Figura 6 exemplifica a busca por N-Gram, onde é dada uma menção de entrada, e.g. "Star Wars The Force Awakens", a mesma passa pelo processo de quebra do N-Gram e a busca é realizada. Então é retornado os endereços das entidades na DBPedia(URI) de candidatas que foram consideradas similares pelo ElasticSearch.

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

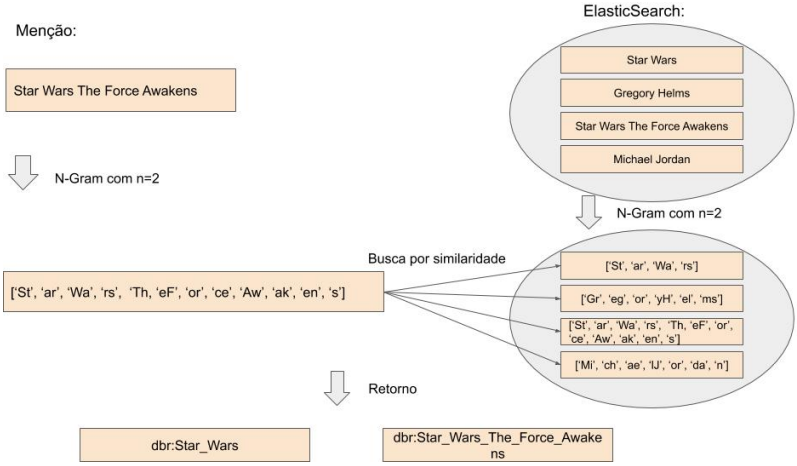

Figura 6 – Exemplo de busca por N-Gram com N=2

4.4 DESAMBIGUAÇÃO DE ENTIDADES CANDIDATAS

A quarta etapa do modo proposto é a desambiguação das entidades candidatas. Para um conjunto de menções M, cada menção $m_i \in M$ possui um conjunto de candidatas C_i. Para cada candidata $c_j \in C_i$, é gerado uma sentença onde é utilizado o resultado do primeiro passo, mas cada menção $m_i \in M$ é substituída por cada uma de suas candidatas $c_j \in C_i$.

4.5 SUBSTITUIÇÃO DE PALAVRAS E ENTIDADES CANDIDATAS POR EMBEDDINGS

O quinto passo é a substituição das palavras e entidades por seus respectivos embeddings, ou seja, pelas suas representações vetoriais. Os embeddings que foram armazenados no MongoDB foram gerados seguindo o seguinte processo: (i) adquirir as triplas da DBpedia, através do dataset Mappingbased Objects[2]; (ii) adquirir os resumos de cada entidade da DBpedia, através do dataset Long Abstracts[3]. Os dois datasets foram alinhados pelas entidades descritas e então transformados em *embeddings*(vetores) para cada entidade através da biblioteca

[2]http://wiki.dbpedia.org/services-resources/documentation/datasets#MappingbasedObject
[3]http://wiki.dbpedia.org/services-resources/documentation/datasets#LongAbstracts

FastText(JOULIN et al., 2016), que foi escolhida por ser capaz de criar *embeddings* de palavras e conhecimento.

4.6 DESAMBIGUAÇÃO DE ENTIDADES

O sexto e último passo é onde ocorre a desambiguação de entidades, isto é, a escolha de uma entidade candidata para ligar à sua respectiva menção, sendo definida a partir de classificação binária que irá julgar se uma menção foi corretamente anotada ou não.

Cada instância gerada no quinto passo será classificada dado um modelo de classificação previamente calculado (mais informações no Capítulo 5). Tal modelo fornece uma probabilidade de que a candidata faz sentido no contexto em que ela se encontra. Porém, apenas a porcentagem sozinha não é suficiente para garantir que a desambiguação é feita de maneira correta. Uma série de regras são utilizadas para auxiliar no processo de EL, entre elas:

1. Caso uma menção tenha sido relacionada apenas com a entidade candidata NIL, ou seja, não foi possível encontrar possíveis candidatas na etapa de Identificação de Candidatas, pode-se pular a etapa de classificação e concluir que não foi possível desambiguar tal menção.

2. É definido um *threshold* que define uma porcentagem mínima aceitável para que uma candidata possa realmente ser considerada uma boa candidata. Se nenhuma candidata exceder esse limite, o resultado da desambiguação não recebe candidatas retornadas.

3. Se as porcentagens de todas as candidatas para uma dada menção forem iguais, é considerado que não foi possível realizar a desambiguação e nenhuma candidata é retornada.

4. Se uma ou mais candidatas excedem o *threshold*, é retornado a candidata com maior chance de pertencer à classe positiva.

A figura 7 demonstra 3 menções, com suas respectivas candidatas, que passam pela etapa de classificação. É demonstrado que cada candidata recebe sua própria porcentagem. Com um threshold igual à 0.9, foi possível realizar a desambiguação das menções "Shane Helms Com"e "Star Wars The Force Awakens", que foram ligadas às entidades "dbr:Gregory_Helms"e "dbr:Star_Wars_The_Force_Awakens"respectivamente. (A etapa de formação do vetor de embeddings foi ocultada no exemplo)

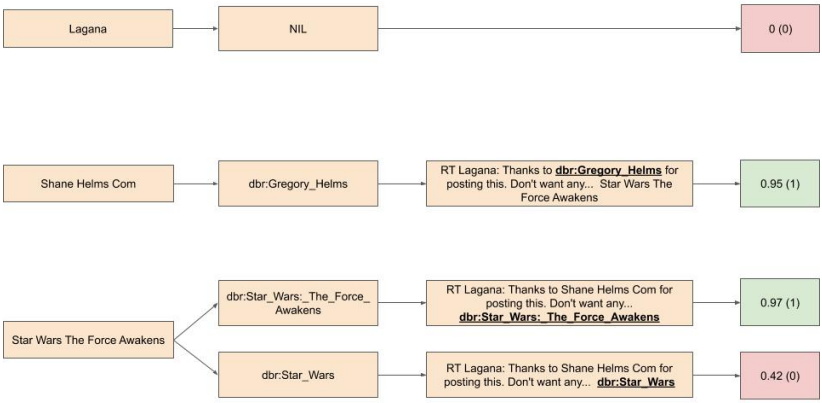

Figura 7 – Exemplo de classificação

5 EXPERIMENTOS

Este capítulo apresenta as ferramentas e frameworks utilizados durante a realização deste trabalho, além dos resultados de testes e experimentos realizados.

Inicialmente, vários classificadores foram testados com a finalidade de escolher o mais apropriado, utilizando a curva ROC como medida de desempenho. Com o classificador escolhido, foi calculada a medida F a fim de comparar o anotador semântico desenvolvido neste trabalho com outas soluções do estado da arte, em termos de macro e micro F1.

5.1 MATERIAIS E MÉTODOS

Para a realização dos experimentos em que os classificadores foram comparados, foi utilizado um notebook com as seguintes configurações:

Memória RAM	16GB
CPU Cores	4 Intel(R) Core(TM) i7-4510U CPU @ 2.00G
Sistema Operacional	Sistema operacional Ubuntu LTS 18.04
Linguagem de desenvolvimento	Python versão 3.6.5

Foi desenvolvido na linguagem Python, o script para cálculo das curvas ROC de um conjunto de algoritmos de classificação binária, disponibilizados pela biblioteca Scikit-Learn((PEDREGOSA et al., 2011)). Como técnica de treinamento dos classificadores, foi utilizada Cross-Validation.

5.2 OBJETIVOS DOS EXPERIMENTOS

O objetivo dos experimentos é descobrir a viabilidade deste trabalho através do cálculo do F1, utilizando a ferramenta GERBIL.

Em análises estatíticas de classificação binária, o F1 é utilizado para mensurar a acurária de um teste. Ele considera tanto a precisão p quanto a cobertura c de um teste para calcular sua pontuação, onde p é o número de resultados positivos verdadeiros dividido pelo total de resultados positivos retornados pelo classificador, e c é o número de resultados positivos verdadeiros divido pelo número total de ocorrências

que deveriam ser identificadas como positivas.

O F1 é a média harmônica entre precisão e cobertura, onde 1 seria o valor ideal para o F1 e 0 o pior.

5.2.1 Algoritmo de Classificação binária

Com o objetivo de diminuir a quantidade de testes realizados nos experimentos apresentados na seção 5.2, um conjunto de algoritmos de classificação binária disponibilizados pela biblioteca Scikit-Learn((PEDREGOSA et al., 2011)), foi testado com a finalidade de decidir qual seria utilizado nas demais etapas deste trabalho.

Foi utilizada a curva ROC (DAVIS; GOADRICH, 2006) para julgar se um algoritmo de classificação é melhor ou pior que o outro. A curva ROC, é uma representação gráfica que ilustra o desempenho (ou performance) de um sistema classificador binário e como o seu limiar de discriminação é variado.

É obtido pela representação da fração de Positivos Verdadeiros dos Positivos Totais (RPV=PV/P) versus a fração de Positivos Falsos dos Negativos Totais (RPF=PF/N), em várias configurações do limite.

5.3 CONFIGURAÇÃO DOS EXPERIMENTOS

Para testar a viabilidade deste trabalho foi utilizada a ferramenta GERBIL. Ela nos fornece uma referência confiável por utilizar métodos para EL/ED que seguem o estado-da-arte, além de utilizar diversos datasets públicos voltados para esse tipo de tarefa.

No período de elaboração deste trabalho, foi utilizada a ferramenta GERBIL na versão 1.2.7, pois era a versão disponível no momento em seu website[1].

5.3.1 Algoritmo de Classificação binária

Para o cálculo da curva ROC foi desenvolvido um script em Python. Tal algoritmo utiliza a técnica de treinamento Cross-Validation, realizando o cálculo da curva ROC em todos os batches. Na etapa de Cross-Validation o treinamento é realizado em 10 batches. Ao final do treinamento é calculado a média das curvas ROC e o seu

[1] http://gerbil.aksw.org/gerbil/config

desvio padrão.

O processo descrito no parágrafo acima é repetido para cada algoritmo selecionado afim de testes neste trabalho. O processo também é repetido utilizando embeddings de 200 e 300 dimensões.

5.4 RESULTADOS DOS EXPERIMENTOS

O resultado dos experimentos é apresentado comparando com os experimentos de outras ferramentas de anotação, na tabela abaixo:

A tabela 3 demonstra que a abordagem adotada por este trabalho ultrapassa a performance da maioria das ferramentas atuais no dataset "Microposts2016-Test"e também possui bons resultados no dataset "Microposts2015-Test".

Fica claro na tabela que a ferramenta AGDISTIS/MAG apresenta os melhores resultados nos 3 datasets e isso se deve à utilização de atributos adicionais em seus classificadores, como popularidade da entidade candidata, e uma etapa de pré-processamento mais elaborada.

5.4.1 Algoritmo de Classificação

O resultado dos experimentos é apresentado por ordem decrescente em relação à media das curvas ROC calculadas durante a Cross-Validation.

O algoritmo que apresentou a melhor curva ROC foi o Random Forest, apresentado na tabela 4, portanto esse foi o algoritmo utilizado para o desenvolvimento da etapa de classificação deste trabalho. É notável que a utilização de 200 ou 300 dimensões para os embeddings não alterou os resultados de maneira significativa.

F1@Micro F1@Macro	Microposts2014-Test	Microposts2015-Test	Microposts2016-Test
AGDISTIS/MAG	0.497 0.701	0.719 0.768	0.616 0.964
AIDA	0.412 0.588	0.414 0.439	0.183 0.919
Babelfy	0.475 0.623	0.341 0.384	0.157 0.917
DBpedia Spotlight	0.452 0.634	ERR	ERR
FOX	0.252 0.508	0.311 0.355	0.068 0.910
FREME NER	0.419 0.597	0.313 0.353	0.162 0.916
OpenTapioca	0.215 0.484	0.259 0.310	0.053 0.909
Nossa abordagem	0.059 0.445	0.319 0.449	0.253 0.928

Tabela 3 – Macro e Micro F1 de abordagens executadas no sistema de benchmark GERBIL. ERR indica que o anotador causou muitos erros no GERBIL.

200 dimensões 300 dimensões	Memória RAM	Tempo de treinamento	Curva ROC
Naive Bayes	4.2G 5.3G	26s 55s	0.5 0.5
Decision Tree	4.2G 5.3G	26s 56s	0.5 0.5
K-Nearest Neighbors	4.2G 5.3G	25m 47m	0.75 0.75
Sigmoid SVM	4.4G 5.8G	8h 20m 12h 12m	0.78 0.79
Linear SVM	4.4G 5.8G	6h 22m 9h 7m	0.79 0.79
RBF SVM	4.4G 5.8G	8h 32m 12h 46m	0.87 0.86
Random Forest	4.2G 5.3G	50s 1m 27s	0.89 0.89

Tabela 4 – Demonstração dos resultados dos experimentos com classificadores binários. A tabela mostra a quantidade de memória utilizada, tempo para o treinamento e a pontuação da curva ROC para cada um dos algoritmos, usando embeddings de 200 dimensões e 300 dimensões.

6 TRABALHOS FUTUROS

Apesar dos resultados satisfatórios, com certeza este trabalho pode ser aprimorado. A etapa de pré-processamento ainda se mostra um tanto ineficaz e outras ferramentas já demonstraram boas alternativas, além de novos atributos que poderiam ser levados em consideração na etapa de classificação.

6.1 PRÉ-PROCESSAMENTO

O uso de gírias e abreviações é uma prática comum no dia-a-dia dos seres humanos, ainda mais comum em postagens em mídias sociais, e não há dúvidas de que as mesmas possuem forte valor semântico. Infelizmente a técnica de pré-processamento utilizada neste trabalho frequentemente não reconhece esse tipo de linguagem e remove tais palavras da sentença.

6.2 CLASSIFICAÇÃO

Diversas plataformas de mídias sociais implementam algoritmos para divulgar assuntos mais populares, conhecidos como "Thrends", consequentemente as Thrends ficam ainda mais populares em um certo período de tempo e as chances de postagens se referirem à este determinado assunto são grandes. Estudos apontam apenas por selecionar a candidata mais popular ao tentar desambiguar uma menção, as chances de a tarefa de EL ter sido realizada corretamente aumentam em 85% (MOUSSALLEM et al., 2017).

Assuntos populares também variam de acordo com a localidade. A abertura do primeiro shopping de uma pequena cidade no interior pode ser uma thrend para as pessoas que residem esse local, porém é pouco provável que esse seja um assunto popular no restante do planeta. O mesmo acontece com o período no tempo em que uma postagem é feita, o que pode ser popular hoje provavelmente não foi popular, dias, meses ou anos atrás e vice-versa. Levar em consideração metadados de uma postagem, como sua posição GPS e data de publicação são bons atributos à serem usados.

Um último atributo à ser levado em consideração é o autor da publicação. Cada pessoa possui sentimentos, preferências, costumes e

interesses que são únicos. Ao guardar um histórico de postagens de mídias sociais, é provavel que uma nova postagem repita candidatas já classificadas corretamente em postagens anteriores.

7 CONCLUSÃO

Muitos dados não estruturados e semi-estruturados estão atualmente disponíveis na Web, porém é difícil de determinar sua semântica correta. Uma anotação semântica descreve o sentido de dados (ou partes deles) mediante a associação de tais dados a recursos de informação com semântica bem definida (KOIVUNEN et al., 2001) presentes em, por exemplo, grafos de conhecimento.

Entre os diversos processos automáticos de anotação existentes, destaca-se o EL. No entanto, abordagens e ferramentas automáticas atuais para anotação semântica falham em entregar anotações com boa precisão e cobertura, principalmente em dados provenientes de mídias sociais. Uma das razões para a baixa precisão e cobertura é que tais ferramentas não consideram todo o contexto que envolvem as anotações. Uma forma de contornar tal limitação é considerar o contexto existente em dados textuais já anotados, como, por exemplo, considerar as relações entre as palavras e os recursos presentes em grafos de conhecimento. A exploração de tais relações, podem ajudar a desambiguar com maior precisão novas anotações.

Este trabalho apresentou a proposta de utilizar de maneira conjunta embeddings de palavras e de conhecimento em classificadores binários. O classificador binário utilizado nesta proposta, Random Forest, foi selecionado dentre diversos outros classificadores binários devido que o mesmo obteve a melhor pontuação com a métrica AUC-ROC. Com o classificador escolhido e o modelo do mesmo gerado, podemos classificar candidatas para menções de novas publicações e realizar a tarefa de desambiguação para EL.

Comparando os resultados obtidos com outras ferramentas do estado-da-arte, é possível perceber que a abordagem deste trabalho obteve bons resultados com os datasets mais atuais "Microposts2016-Test"e "Microposts2015-Test"e uma pior performance no dataset "Micropost2014-Test".

Apesar dos resultados satisfatórios, a proposta pode ser melhorada em algumas partes. O uso de gírias e abreviações não é tratado de maneira adequada neste trabalho, além de diversos atributos (e.g. localização e data/hora da publicação, popularidade da candidata) que poderiam ser levados em consideração na etapa de classificação são completamente ignorados.

REFERÊNCIAS

ALMEIDA, E. Classificação ordinal com opção de rejeição. In: . [S.l.: s.n.], 2010.

AUER, S. et al. Dbpedia: A nucleus for a web of open data. In: *The semantic web*. [S.l.]: Springer, 2007. p. 722–735.

BHATTACHARYA, I.; GETOOR, L. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, ACM, v. 1, n. 1, p. 5, 2007.

BOLLACKER, K. et al. Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM. *Proc. of the 2008 ACM SIGMOD international conference on Management of data*. [S.l.], 2008. p. 1247–1250.

BONTCHEVA, K.; ROUT, D. Making sense of social media streams through semantics: A survey. *Semantic Web*.

CHEN, H. et al. Bilinear joint learning of word and entity embeddings for entity linking. *Neurocomputing*, Elsevier, v. 294, p. 12–18, 2018.

CHONG, W.-H.; LIM, E.-P.; COHEN, W. Collective entity linking in tweets over space and time. In: SPRINGER. *European Conf. on Information Retrieval*. [S.l.], 2017. p. 82–94.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. In: *ICML*. [S.l.: s.n.], 2006.

DERCZYNSKI, L. et al. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, Elsevier, v. 51, p. 32–49, 2015.

EKELUND, S. Roc curves—what are they and how are they used? In: . [S.l.: s.n.], 2012.

FABIAN, M.; GJERGJI, K.; GERHARD, W. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In: *16th Intl. World Wide Web Conf., WWW*. [S.l.: s.n.], 2007. p. 697–706.

FAN, J.; UPADHYE, S.; WORSTER, A. Understanding receiver operating characteristic (roc) curves. *CJEM*, v. 8 1, p. 19–20, 2006.

FANG, W. et al. Entity disambiguation by knowledge and text jointly embedding. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. [S.l.: s.n.], 2016. p. 260–269.

FANG, Y.; CHANG, M.-W. Entity linking on microblogs with spatial and temporal signals. *Association for Computational Linguistics*, Association for Computational Linguistics, v. 2, p. 259–272, 2014.

GANEA, O.-E. et al. Probabilistic bag-of-hyperlinks model for entity linking. In: INTL. WORLD WIDE WEB CONF. STEERING COMMITTEE. *Proc. of the 25th Intl. Conf. on World Wide Web*. [S.l.], 2016. p. 927–938.

GROUP, R. W. *RDF 1.1 Concepts and Abstract Syntax*. Nov. 2018. https://www.w3.org/TR/rdf11-concepts/. <https://www.w3.org/TR/rdf11-concepts/>.

GUO, Y. et al. Microblog entity linking by leveraging extra posts. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, p. 863–868, 2013.

HAN, H. et al. Yet another framework for tweet entity linking (yaftel). *IEEE Xplore Digital Library*, 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2019.

HAN, X.; SUN, L.; ZHAO, J. Collective entity linking in web text: a graph-based method. In: ACM. *Proc. of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. [S.l.], 2011. p. 765–774.

HOSSEINI, H. . et al. Implicit entity linking in tweets: an ad-hoc retrieval approach. *Applied Ontology*, IOS Press, Pre-press, p. 1–27, 2019.

HUANG, H. et al. Collective tweet wikification based on semi-supervised graph regularization. In: *ACL (1)*. [S.l.: s.n.], 2014. p. 380–390.

JOULIN, A. et al. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

KALLOUBI, F.; NFAOUI, E. H. et al. Microblog semantic context retrieval system based on linked open data and graph-based theory. *Expert Systems with Applications*, Elsevier, v. 53, p. 138–148, 2016.

KOIVUNEN, M.-R. et al. Annotea: An open rdf infrastructure for shared web annotations. 2001.

KOLITSAS, N.; GANEA, O.-E.; HOFMANN, T. End-to-end neural entity linking. In: *CoNLL*. [S.l.: s.n.], 2018.

LAENDER, A. H. e. a. A brief survey of web data extraction tools. ACM Sigmod Record, ACM, v. 31, n. 2, p. 84–93, 2002.

LE, P.; TITOV, I. Improving entity linking by modeling latent relations between mentions. Association for Computational Linguistics, v. 1, p. 1595–1604, 2018.

LEHMANN, J. et al. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, Elsevier Science Publishers B. V., v. 7, n. 3, p. 154–165, 2009.

LI, Y. et al. Entity disambiguation with linkless knowledge bases. In: INTL. WORLD WIDE WEB CONF. STEERING COMMITTEE. *Proc. of the 25th Intl. Conf. on World Wide Web*. [S.l.], 2016. p. 1261–1270.

LIU, C. et al. Attention-based joint entity linking with entity embedding. *Information*, v. 10, p. 46, 2019.

LORENA, A. C.; CARVALHO, A. C. P. de Leon Ferreira de. Estratégias para a combinação de classificadores binários em soluções multiclasses. *RITA*, v. 15, p. 65–86, 2008.

LUO, G. et al. Joint named entity recognition and disambiguation. In: *Proc. of the Conf. on Empirical Methods in Natural Language Processing*. [S.l.: s.n.], 2015. p. 879–888.

MARTINS, P. H.; MARINHO, Z.; MARTINS, A. F. T. Joint learning of named entity recognition and entity linking. In: *ACL*. [S.l.: s.n.], 2019.

MENDES, P. N. e. a. Dbpedia spotlight: shedding light on the web of documents. *In: ACM. Proceedings of the 7th international conference on semantic systems. [S.l.]*, p. 1–8, 2011.

MIN, J. H.; JEONG, C. A binary classification method for bankruptcy prediction. *Expert Syst. Appl.*, v. 36, p. 5256–5263, 2009.

MORENO, J. G. et al. Combining word and entity embeddings for entity linking. In: SPRINGER. *European Semantic Web Conference*. [S.l.], 2017. p. 337–352.

MORO, A.; RAGANATO, A.; NAVIGLI, R. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, v. 2, 2014.

MOUSSALLEM, D. et al. Mag: A multilingual, knowledge-base agnostic and deterministic entity linking approach. In: *K-CAP*. [S.l.: s.n.], 2017.

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, John Benjamins publishing company, v. 30, n. 1, p. 3–26, 2007.

NICKEL, M. et al. A review of relational machine learning for knowledge graphs. *Proc. of the IEEE*, v. 104, n. 1, p. 11–33, 2016.

PARRAVICINI, A. et al. Fast and accurate entity linking via graph embedding. In: *GRADES/NDA@SIGMOD/PODS*. [S.l.: s.n.], 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, v. 12, p. 2825–2830, 2011.

RATINOV L., R. D. D. D. . A. M. Local and global algorithms for disambiguation to wikipedia. *In 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT '11)*, p. 1375–1384, 2011.

RENNIE, J. D. M.; RIFKIN, R. Improving multiclass text classification with the support vector machine. In: . [S.l.: s.n.], 2001.

RUDER, S.; VULIĆ, I.; SØGAARD, A. A survey of cross-lingual word embedding models. In: . [S.l.: s.n.], 2017.

SENS, A.; PINHEIRO, I. Construção de contextos semânticos a partir dos resultados de ferramentas de anotação. In: . [S.l.: s.n.], 2019.

SEVGILI, Ö.; PANCHENKO, A.; BIEMANN, C. Improving neural entity disambiguation with graph embeddings. In: *ACL*. [S.l.: s.n.], 2019.

SHEN, W.; WANG, J.; HAN, J. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 27, n. 2, p. 443–460, 2015.

SINGLA, P.; DOMINGOS, P. Entity resolution with markov logic. In: IEEE. *Data Mining, 2006. ICDM'06. Sixth International Conference on*. [S.l.], 2006. p. 572–582.

SPECKAXEL, R.; NGOMO, C. N. Ensemble learning for named entity recognition. *International Semantic Web Conference*, 2014.

USBECK, R. et al. Gerbil: general entity annotator benchmarking framework. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 24th international conference on World Wide Web*. [S.l.], 2015. p. 1133–1143.

WANG, Q.; IWAIHARA, M. Deep neural architectures for joint named entity recognition and disambiguation. *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, p. 1–4, 2019.

WANG, Q. et al. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 29, n. 12, p. 2724–2743, 2017.

WHANG, S. E.; GARCIA-MOLINA, H. Joint entity resolution. In: IEEE. *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. [S.l.], 2012. p. 294–305.

YAMADA, I. et al. Joint learning of the embedding of words and entities for named entity disambiguation. In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. [S.l.: s.n.], 2016. p. 250–259.

ZHU, G.; IGLESIAS, C. A. Exploiting semantic similarity for named entity disambiguation in knowledge graphs. *Expert Systems With Applications*, Elsevier, v. 101, p. 8–24, 2018.

APÊNDICE A – Comparadores Curva ROC

As figuras abaixo apresentam gráficos com resultados dos experimentos com classificadores binários. Foram calculados as curvas ROC para todos os classificadores testados, com um teste utilizando embeddings de 200 dimensões e outro com 300 dimensões.

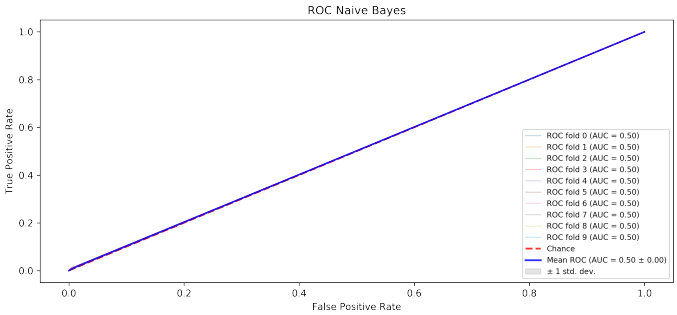

Figura 8 – Curva ROC -> Naive Bayes com embeddings de 200 dimensões

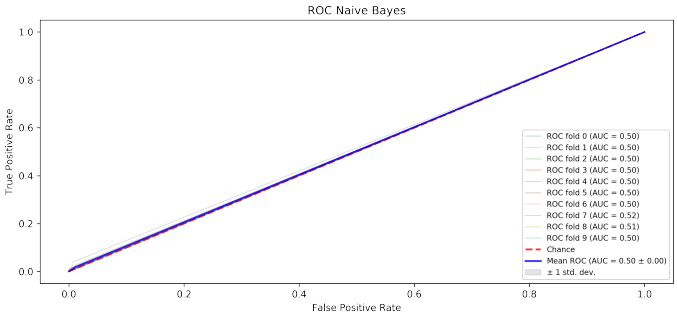

Figura 9 – Curva ROC -> Naive Bayes com embeddings de 300 dimensões

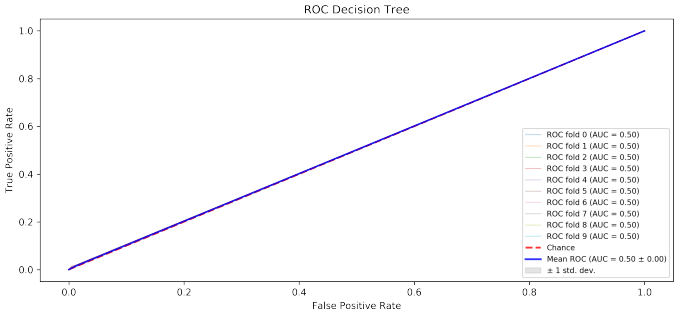

Figura 10 – Curva ROC -> Decision Tree com embeddings de 200 dimensões

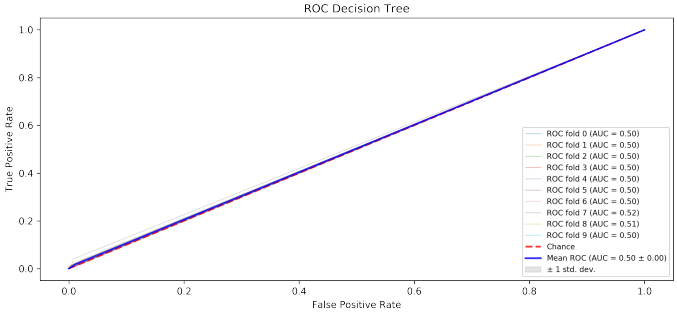

Figura 11 – Curva ROC -> Decision Tree com embeddings de 300 dimensões

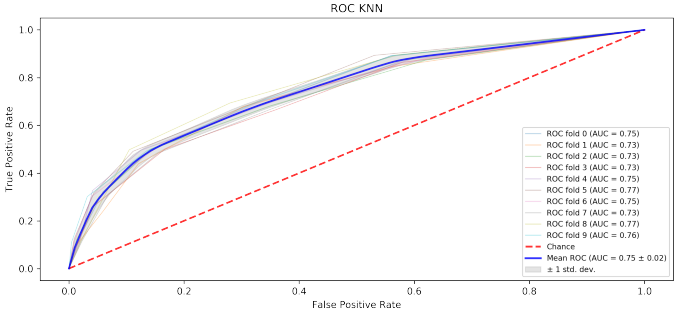

Figura 12 – Curva ROC -> KNN com embeddings de 200 dimensões

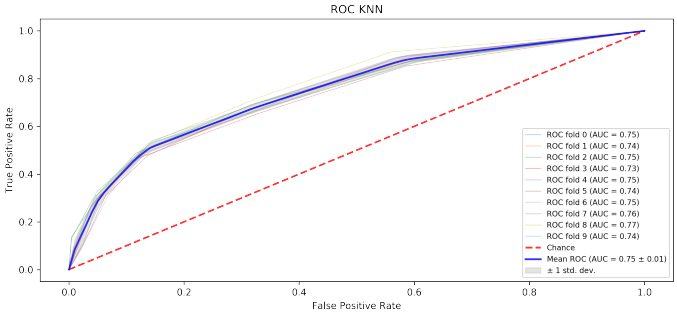

Figura 13 – Curva ROC -> KNN com embeddings de 300 dimensões

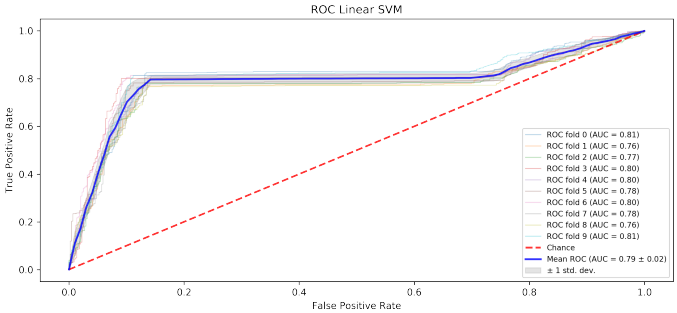

Figura 14 – Curva ROC -> Linear SVM com embeddings de 200 dimensões

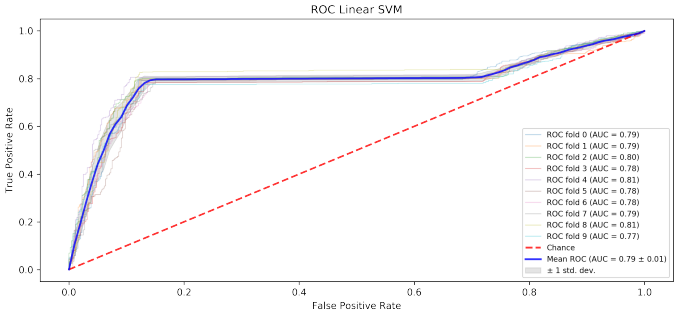

Figura 15 – Curva ROC -> Linear SVM com embeddings de 300 dimensões

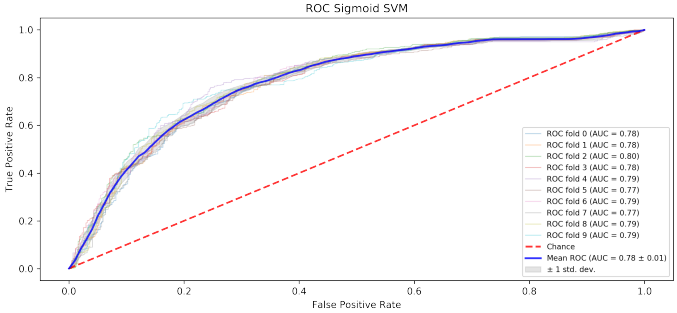

Figura 16 – Curva ROC -> Sigmoid SVM com embeddings de 200 dimensões

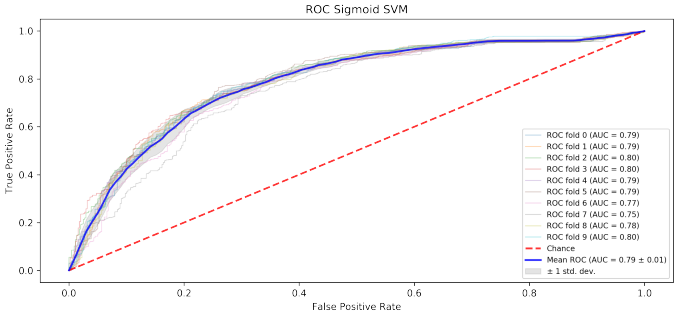

Figura 17 – Curva ROC -> Sigmoid SVM com embeddings de 300 dimensões

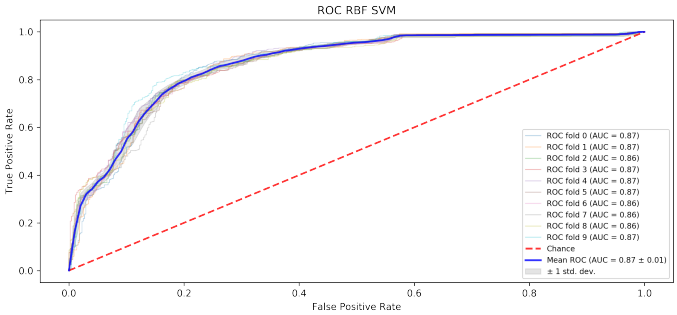

Figura 18 – Curva ROC -> Gaussian SVM com embeddings de 200 dimensões

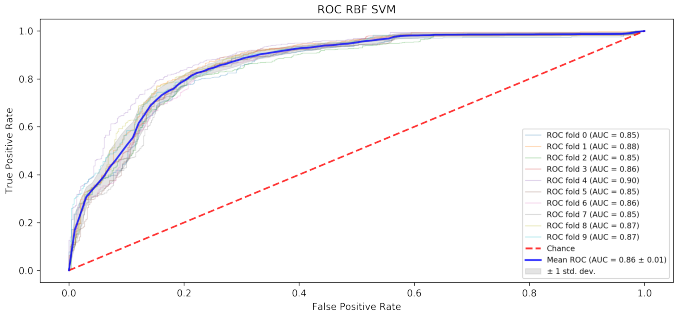

Figura 19 – Curva ROC -> Gaussian SVM com embeddings de 300 dimensões

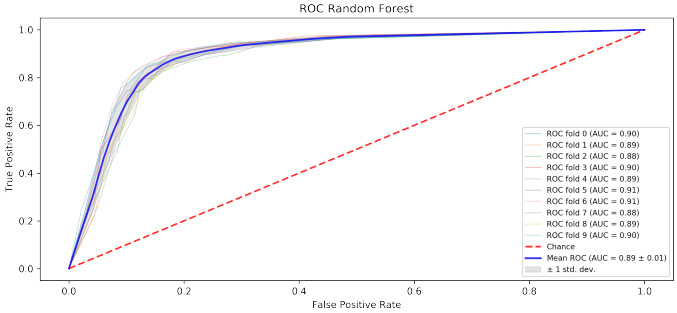

Figura 20 – Curva ROC -> Random Forest com embeddings de 200 dimensões

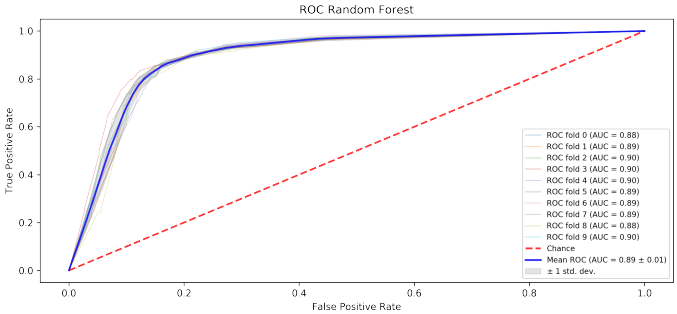

Figura 21 – Curva ROC -> Random Forest com embeddings de 300 dimensões

ANEXO A – Código para treinamento de classificadores

```
Training.py
import pymongo
import time
import pickle
import numpy
import pandas
import matplotlib.pyplot as plt
from scipy import interp
from sklearn.model_selection import train_test_split,
    cross_val_score, cross_val_predict, StratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, roc_auc_score,
    roc_curve, auc
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from joblib import dump, load
from time import gmtime, strftime


start = time.time()

df = pandas.read_csv('./storage/embeddings.csv')
df = df.sample(frac=1).reset_index(drop=True)

# # ----------------- > KNN <----------------------
startClassifier = time.time()
print('\nIncio KNN: ' + strftime("%H:%M:%S", gmtime()))

cv = StratifiedKFold(n_splits=10)
classifier = KNeighborsClassifier()
pipeline = Pipeline([('transformer', sc), ('estimator',
    classifier)])

tprs = []
aucs = []
mean_fpr = numpy.linspace(0, 1, 100)

i = 0
plt.figure(1, (12, 5))
for train, test in cv.split(X, y):
```

```
    pipeline = pipeline.fit(X[train], numpy.ravel(y[train]))
    probas_ = pipeline.predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1


plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
    label='Chance', alpha=.8)

mean_tpr = numpy.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = numpy.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
     label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc,
         std_auc),
     lw=2, alpha=.8)

std_tpr = numpy.std(tprs, axis=0)
tprs_upper = numpy.minimum(mean_tpr + std_tpr, 1)
tprs_lower = numpy.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey',
    alpha=.2,
     label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC KNN')
plt.legend(loc="lower right", prop={'size': 8})
plt.savefig('plots/embeddings300/KNN.png', format='png',
    dpi=300, transparent=True)
# plt.savefig('plots/embeddings200/KNN.png', format='png',
    dpi=300, transparent=True)
plt.close()
```

```
endClassifier = time.time()
print(f'Running time creating KNN classifier: {endClassifier -
    startClassifier}')

# # ----------------- > Linear SVM <----------------------
startClassifier = time.time()
print('\nIncio Linear SVM: ' + strftime("%H:%M:%S", gmtime()))

cv = StratifiedKFold(n_splits=10)
classifier = SVC(kernel='linear', probability=True, gamma='auto')
pipeline = Pipeline([('transformer', sc), ('estimator',
    classifier)])

tprs = []
aucs = []
mean_fpr = numpy.linspace(0, 1, 100)
i = 0
plt.figure(1, (12, 5))
for train, test in cv.split(X, y):
    probas_ = pipeline.fit(X[train],
        numpy.ravel(y[train])).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
        label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1


plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
     label='Chance', alpha=.8)

mean_tpr = numpy.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = numpy.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
     label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc,
         std_auc),
     lw=2, alpha=.8)
```

```
std_tpr = numpy.std(tprs, axis=0)
tprs_upper = numpy.minimum(mean_tpr + std_tpr, 1)
tprs_lower = numpy.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey',
    alpha=.2,
             label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Linear SVM')
plt.legend(loc="lower right", prop={'size': 8})
plt.savefig('plots/embeddings300/LinearSVM.png', format='png',
    dpi=300, transparent=True)
# plt.savefig('plots/embeddings200/LinearSVM.png', format='png',
    dpi=300, transparent=True)
plt.close()

endClassifier = time.time()
print(f'Running time creating Linear SVM classifier:
    {endClassifier - startClassifier}')

# # ----------------- > RBF SVM <----------------------
startClassifier = time.time()
print('\nIncio RBF SVM: ' + strftime("%H:%M:%S", gmtime()))

cv = StratifiedKFold(n_splits=10)
classifier = SVC(kernel='rbf', probability=True, gamma='auto')
pipeline = Pipeline([('transformer', sc), ('estimator',
    classifier)])

tprs = []
aucs = []
mean_fpr = numpy.linspace(0, 1, 100)

i = 0
plt.figure(1, (12, 5))
for train, test in cv.split(X, y):
    probas_ = pipeline.fit(X[train],
        numpy.ravel(y[train])).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
```

```
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
          label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1


plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
      label='Chance', alpha=.8)

mean_tpr = numpy.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = numpy.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
      label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc,
          std_auc),
      lw=2, alpha=.8)

std_tpr = numpy.std(tprs, axis=0)
tprs_upper = numpy.minimum(mean_tpr + std_tpr, 1)
tprs_lower = numpy.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey',
     alpha=.2,
             label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC RBF SVM')
plt.legend(loc="lower right", prop={'size': 8})
plt.savefig('plots/embeddings300/RbfSVM.png', format='png',
     dpi=300, transparent=True)
# plt.savefig('plots/embeddings200/RbfSVM.png', format='png',
     dpi=300, transparent=True)
plt.close()

endClassifier = time.time()
print(f'Running time creating RBF SVM classifier: {endClassifier
     - startClassifier}')
```

```
# ----------------- > Sigmoid SVM <----------------------
startClassifier = time.time()
print('\nIncio Sigmoid SVM: ' + strftime("%H:%M:%S", gmtime()))

cv = StratifiedKFold(n_splits=10)
classifier = SVC(kernel='sigmoid', probability=True,
    gamma='auto')
pipeline = Pipeline([('transformer', sc), ('estimator',
    classifier)])

tprs = []
aucs = []
mean_fpr = numpy.linspace(0, 1, 100)

i = 0
plt.figure(1, (12, 5))
for train, test in cv.split(X, y):
    probas_ = pipeline.fit(X[train],
        numpy.ravel(y[train])).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
            label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1


plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
        label='Chance', alpha=.8)

mean_tpr = numpy.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = numpy.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
        label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' %
            (mean_auc, std_auc),
        lw=2, alpha=.8)

std_tpr = numpy.std(tprs, axis=0)
tprs_upper = numpy.minimum(mean_tpr + std_tpr, 1)
```

```
tprs_lower = numpy.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey',
    alpha=.2,
                label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Sigmoid SVM')
plt.legend(loc="lower right", prop={'size': 8})
plt.savefig('plots/embeddings300/SigmoidSVM.png', format='png',
    dpi=300, transparent=True)
# plt.savefig('plots/embeddings200/SigmoidSVM.png',
    format='png', dpi=300, transparent=True)
plt.close()

endClassifier = time.time()
print(f'Running time creating Sigmoid SVM classifier:
    {endClassifier - startClassifier}')

# ----------------- > Naive Bayes <----------------------
startClassifier = time.time()
print('\nIncio Naive Bayes: ' + strftime("%H:%M:%S", gmtime()))

cv = StratifiedKFold(n_splits=10)
classifier = GaussianNB()
pipeline = Pipeline([('transformer', sc), ('estimator',
    classifier)])

tprs = []
aucs = []
mean_fpr = numpy.linspace(0, 1, 100)

i = 0
plt.figure(1, (12, 5))
for train, test in cv.split(X, y):
    probas_ = pipeline.fit(X[train],
        numpy.ravel(y[train])).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
```

```
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1


plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)

mean_tpr = numpy.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = numpy.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' %
             (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = numpy.std(tprs, axis=0)
tprs_upper = numpy.minimum(mean_tpr + std_tpr, 1)
tprs_lower = numpy.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey',
    alpha=.2,
                 label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Naive Bayes')
plt.legend(loc="lower right", prop={'size': 8})
plt.savefig('plots/embeddings300/NaiveBayes.png', format='png',
    dpi=300, transparent=True)
# plt.savefig('plots/embeddings200/NaiveBayes.png',
    format='png', dpi=300, transparent=True)
plt.close()

endClassifier = time.time()
print(f'Running time creating Naive Bayes classifier:
    {endClassifier - startClassifier}')

# ----------------- > Decision Tree <----------------------
startClassifier = time.time()
print('\nIncio Decision Tree: ' + strftime("%H:%M:%S",
    gmtime()))
```

```
cv = StratifiedKFold(n_splits=10)
classifier = GaussianNB()
pipeline = Pipeline([('transformer', sc), ('estimator',
    classifier)])

tprs = []
aucs = []
mean_fpr = numpy.linspace(0, 1, 100)

i = 0
plt.figure(1, (12, 5))
for train, test in cv.split(X, y):
    probas_ = pipeline.fit(X[train],
        numpy.ravel(y[train])).predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
            label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1


plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
        label='Chance', alpha=.8)

mean_tpr = numpy.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = numpy.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
        label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' %
            (mean_auc, std_auc),
        lw=2, alpha=.8)

std_tpr = numpy.std(tprs, axis=0)
tprs_upper = numpy.minimum(mean_tpr + std_tpr, 1)
tprs_lower = numpy.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey',
    alpha=.2,
                label=r'$\pm$ 1 std. dev.')
```

```
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Decision Tree')
plt.legend(loc="lower right", prop={'size': 8})
plt.savefig('plots/embeddings300/DecisionTree.png',
    format='png', dpi=300, transparent=True)
# plt.savefig('plots/embeddings200/DecisionTree.png',
    format='png', dpi=300, transparent=True)
plt.close()

endClassifier = time.time()
print(f'Running time creating Decision Tree classifier:
    {endClassifier - startClassifier}')

# ----------------- > Random Forest <----------------------

startClassifier = time.time()
print('\nIncio Random Forest: ' + strftime("%H:%M:%S",
    gmtime()))


cv = StratifiedKFold(n_splits=10)
classifier = RandomForestClassifier()
pipeline = Pipeline([('transformer', sc), ('estimator',
    classifier)])

tprs = []
aucs = []
mean_fpr = numpy.linspace(0, 1, 100)

i = 0
plt.figure(1, (12, 5))
for train, test in cv.split(X, y):
    pipeline = pipeline.fit(X[train], numpy.ravel(y[train]))
    dump(pipeline, 'randomForest.model')
    probas_ = pipeline.predict_proba(X[test])
    # Compute ROC curve and area the curve
    fpr, tpr, thresholds = roc_curve(y[test], probas_[:, 1])
    tprs.append(interp(mean_fpr, fpr, tpr))
    tprs[-1][0] = 0.0
    roc_auc = auc(fpr, tpr)
    aucs.append(roc_auc)
```

```
    plt.plot(fpr, tpr, lw=1, alpha=0.3,
             label='ROC fold %d (AUC = %0.2f)' % (i, roc_auc))

    i += 1

plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
         label='Chance', alpha=.8)

mean_tpr = numpy.mean(tprs, axis=0)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
std_auc = numpy.std(aucs)
plt.plot(mean_fpr, mean_tpr, color='b',
         label=r'Mean ROC (AUC = %0.2f $\pm$ %0.2f)' %
             (mean_auc, std_auc),
         lw=2, alpha=.8)

std_tpr = numpy.std(tprs, axis=0)
tprs_upper = numpy.minimum(mean_tpr + std_tpr, 1)
tprs_lower = numpy.maximum(mean_tpr - std_tpr, 0)
plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey',
    alpha=.2,
                label=r'$\pm$ 1 std. dev.')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Random Forest')
plt.legend(loc="lower right", prop={'size': 8})
plt.savefig('plots/embeddings300/RandomForest.png',
    format='png', dpi=300, transparent=True)
# plt.savefig('plots/embeddings200/RandomForest.png',
    format='png', dpi=300, transparent=True)
plt.close()

endClassifier = time.time()
print(f'Running time creating Random Forest classifier:
    {endClassifier - startClassifier}')
```

```
disambiguator.py
from util import *

import torch
```

```
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from torch.autograd import Variable
import model
import pymongo
import numpy
myclient =
    pymongo.MongoClient("mongodb://localhost:27017/",username='lisa',password='l
mydb = myclient["oya"]
mycol = mydb["words"]


class DatasetOya(Dataset):
    def __init__(self, file):
        self.data = file

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        text = torch.tensor(self.data[idx], dtype=torch.long)
        return text

def get_word2idx_vocab(vocab):
    vocab2idx = dict()
    idx = 0
    for word in vocab:
        vocab2idx[word] = idx
        idx += 1
    return vocab2idx

def prepare_sequence(instance):
    X = []
    for i in range(0, 41):
        try:
            X.append(instance.split()[i])
        except:
            X.append('<PAD>')
    return X

def build_weight_matrix(fast_text, target_vocab, embed_dim):
    matrix_len = len(target_vocab)
    weight_matrix = np.zeros((matrix_len, embed_dim))
    words_found = 0
    for i, word in enumerate(target_vocab):
```

```
            try:
                weight_matrix[i] = fast_text[word]
                words_found += 1
            except KeyError:
                weight_matrix[i] = np.zeros(embed_dim)
        weight_tensor = torch.from_numpy(weight_matrix)
        return weight_tensor


def disambiguate_mentions(tweet, threshold, model):
    entities = []

    instances_mentions = []
    print("Printando candidatas: " + str(tweet.candidates))
    for i, mention in enumerate(tweet.mentions):
        print(" Meno  " +str(i) + ": " + str(mention))
        instances = []
        if tweet.candidates[i] is None:
            instance = tweet.procText.replace(mention[1], 'NIL')
            instance = prepare_sequence(instance)
            instances.append(instance)
            instances_mentions.append(instances)
        else:
            for candidate in tweet.candidates[i]:
                instance = tweet.procText.replace(mention[1], "
                    {} ".format(candidate))
                instance = prepare_sequence(instance)
                instances.append(instance)
            instances_mentions.append(instances)



    print("\n\nComeando  clculo  de scores --")
    for i, instances in enumerate(instances_mentions):
        print("Printando instances: " + str(instances))

        if instances == None:
            entities.append('NIL')
        else:
            scores = []

            for instance in instances:
                X_test = []
                embeddings = []
                for word in instance:
```

```
            if word != '<PAD>':
                try:
                    embeddings += list(map(float,
                        mycol.find_one({"_id":
                        word})['embedding'].split(' ')))
                except:
                    if 'dbr:' in word:
                        print("Palavra no encontrada no
                            Mongo: " + str(word))

                    e = []
                    for aux in range(0, 200):
                        e += [0]
                    embeddings += e

            else:
                e = []
                for aux in range(0, 200):
                    e += [0]
                embeddings += e

        X_test.append(embeddings)
        scores.append(model.predict_proba(X_test)[0][1])
    print("\nScores: " + str(scores))
    print("-------")


    print("\n\nComeando  seleo  da candidata")
    if(tweet.candidates[i] == None):
        entities.append('NIL')
    # Uma candidata s e ela est acima do threshold
    elif(len(scores) == 1 and scores[0] > threshold):
        entities.append(tweet.candidates[i][0])
    elif(len(scores) == 1 and scores[0] < threshold):
        entities.append('NIL')
    # Se todos os scores foram iguais, no conseguimos
        desambiguar, retorna nulo
    elif(scores.count(scores[0]) == len(scores)):
        entities.append('NIL')
    # Ordena lista e pega maior score
    else:
        index_max = max(range(len(scores)),
            key=scores.__getitem__)
        # Retorna nulo se melhor score estiver abaixo do
            threshold
```

```
            if max(scores) < threshold:
                entities.append('NIL')
            else:
                entities.append(tweet.candidates[i][index_max])

    return entities
```

ANEXO A – Artigo

Uso de classificadores binários em embeddings de palavras e conhecimento na tarefa de ligação de entidades

Arthur Silva Sens[1], Isabelle Pinheiro[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.010-970 – Florianópolis – SC – Brasil

arthursens2005@gmail.com, isabellepinheiro00@gmail.com

Abstract. *Semantically annotating data whose semantics are not well-defined by machines, such as large amounts of semi-structured unstructured data currently available on the Web, has the potential to leverage applications that can take advantage of automatic interpretations of such data. However, the current automatic processes of semantic annotation fail to deliver results with good quality. One way to improve the quality of the results generated by the current annotation processes is to consider a broader context in which the data are found.Semantic contexts constructed from reliable annotation can help in the disambiguation process of new annotations. They can be represented as embedding, which is a category of natural language processing models that mathematically map the words to numerical vectors. This feature makes it much easier and streamlines the determination of similar words (vectors) or vector clusters This work proposes algorithms for the creation and use of semantic contexts using machine learning techniques in graphs to improve the disambiguation process of new annotations. The results of the annotations produced by the proposed model are compared with those of state of the art tools in semantic annotation of textual data.*

Resumo. *Anotar semanticamente dados cuja semântica não é bem definida ou processável por máquinas, tais como grandes quantidades de dados semi ou não-estruturados atualmente disponíveis na Web,tem o potencial de alavancar aplicações que podem tirar proveito de interpretações automáticas de tais dados. Entretanto, os atuais processos automáticos de anotação semântica falham em entregar resultados com boa qualidade. Uma maneira de melhorar a qualidade dos resultados gerados pelos processos de anotação atuais é considerar um contexto mais amplo no qual os dados se encontram. Contextos semânticos construídos a partir de algumas anotações confiáveis podem auxiliar no processo de desambiguação de novas anotações.Eles podem ser representados como embeddings, que são uma categoria de modelos de processamento natural da linguagem que mapeiam matematicamente as palavras para vetores numéricos.*

Esse recurso facilita e agiliza a determinação de palavras (vetores) semelhantes ou clusters vetoriais. Este trabalho propõe algoritmos para a criação e utilização de contextos semânticos usando técnicas de aprendizado de máquina em grafos, para melhorar o processo de desambiguação de novas anotações. Os resultados das anotações produzidas pelo modelo proposto são comparadas com aqueles de ferramentas do estado da arte em anotação semântica de dados textuais.

1. Introdução

Muitos dados não estruturados e semi-estruturados estão atualmente disponíveis na Web (e.g., documentos de texto, documentos HTML (Hypertext Markup Language), postagens em mídias sociais). Entretanto, a aplicação destes dados é prejudicada por dificuldades em processar computacionalmente sua semântica correta (LAENDER,

2002). O não entendimento do significado de termos e menções à entidades do mundo real presentes em dados textuais impede o uso dos mesmos em todo o seu potencial. Uma maneira de atenuar tal problema é anotá-los semanticamente (BONTCHEVA; ROUT,).

Uma anotação semântica descreve o sentido de dados (ou partes deles) mediante a associação de tais dados a recursos de informação com semântica bem definida (e.g., conceitos, instâncias de conceito) (KOIVUNEN et al., 2001) presentes em, por exemplo, grafos de conhecimento (Knowledge Graph - KG) (e.g., DBpedia (AUER et al., 2007) (LEHMANN et al., 2009), Yago (FABIAN; GJERGJI; GERHARD, 2007), Freebase (BOLLACKER et al., 2008)). Entretanto, anotar manualmente dados disponíveis na Web é um processo inviável, seja pela quantidade proibitiva de dados disponíveis, seja pela dificuldade em produzir resultados padronizados e com qualidade. Assim, diversos processos automáticos de anotação semântica têm sido propostos.

Entre os diversos processos automáticos de anotação existentes, destaca-se a ligação de entidades (Entity Linking - EL). EL é uma tarefa de anotação semântica responsável por identificar menções a entidades do mundo real em um documento de texto (este passo é chamado de reconhecimento de entidades nomeadas, Named Entity Recognition -NER) e ligar cada menção encontrada a uma definição semanticamente bem descrita daquilo a que cada menção se refere (este passo é chamado de desambiguação de entidades nomeadas, Named Entity Disambiguation - NED) (RATINOV L., 2011; SHEN; WANG; HAN, 2015). A etapa de desambiguação da EL é o foco deste trabalho.

No entanto, abordagens e ferramentas automáticas atuais para anotação semântica (e.g., DBpedia-Spotlight (MENDES, 2011), FOX (SPECKAXEL; NGOMO, 2014), Babelfy (MORO; RAGANATO; NAVIGLI, 2014)) falham em entregar anotações com boa precisão e cobertura, principalmente quando aplicadas a documentos de texto com muitos ruídos (e.g., erros ortográficos e gramaticais) e linguagem coloquial (e.g. uso de gírias), tais como postagens em mídias sociais (BONTCHEVA;ROUT,). Uma das razões para a baixa precisão e cobertura é que tais ferramentas não consideram todo o contexto (e.g., textual, semântico) que envolvem as anotações e os dados a serem anotados.

Uma forma de contornar tal limitação é considerar o contexto existente em dados textuais já anotados, como, por exemplo, considerar as relações entre as palavras e os recursos presentes em grafos de conhecimento. A exploração de tais relações, que podem ser expressas através de embeddings de palavras e conhecimento por técnicas de Machine Learning - ML), podem ajudar a desambiguar com maior precisão novas anotações.

1.1. Objetivos

Esta seção tratará dos objetivos e metas a serem alcançados através da pesquisa e execução do trabalho de conclusão de curso.

1.1.1. Objetivos gerais

O objetivo geral deste trabalho é propor e implementar um método que utilize embeddings de palavras e conhecimento de forma conjunta no processo de desambiguação de novas anotações semânticas em postagens de mídias sociais utilizando técnicas de machine learning computacionalmente menos exigentes do que redes neurais profundas.

1.1.2. Objetivos específicos

Os objetivos específicos deste trabalho são:

1. Melhorar o entendimento do estado-da-arte em anotação semântica, mais especificamente *Entity Linking* (EL);

2. Treinar modelos com diferentes técnicas de machine learning utilizando embeddings de palavra e conhecimento como entrada;

3. Propor e implementar um método de desambiguação com base nos modelos gerados;

4. Validar os métodos propostos através de experimentos.

2. Trabalhos Relacionados

Este capítulo apresenta e discute os principais trabalhos relacionados à proposta deste trabalho. A Tabela abaixo lista abordagens para EL que usam embeddings ou cujo foco seja anotar semanticamente postagens de mídias sociais, particularmente tweets.

Os trabalhos na Tabela estão ordenados por sua data de publicação e são comparados segundo 3 critérios especificados nas demais colunas: Textos anotados; Entradas externas; Métodos. A coluna Textos anotados informa o tipo de documento de texto a ser anotado,podendo ser classificado como notícias, textos literários etc, afim de explicitar o tipo de texto formal utilizado no trabalho, ou como tweets(postagens de mídias sociais). Entradas externas são as entradas utilizadas para anotar semanticamente os documentos de texto. Por fim, a coluna Métodos apresenta a principal técnica/algoritmo/método a ser utilizada pelo trabalho para tratar a tarefa de EL.

Na coluna Tipos de dados, percebe-se que poucos são os trabalhos atuais que trabalham com tweets. Isso se deve a diversos motivos,entre eles, dificuldade de obtenção de tweets, poucos conjuntos de dados públicos disponíveis e questões de privacidade. Na coluna Entradas Externas, a maioria dos trabalhos utilizam como fonte de dados utilizados para anotação semântica a Wikipedia. Alguns trabalhos, como (LUO et al., 2015;FANG et al., 2016;CHEN et al., 2018;LE; TITOV, 2018), utilizam outros KG (Freebase, DBpedia) para complementar os dados provenientes da Wikipedia. Somente (KALLOUBI; NFAOUI et al., 2016;ZHU; IGLESIAS, 2018;SEVGILI; PANCHENKO; BIEMANN, 2019;PARRAVICINI et al., 2019) utilizam a DBpedia como fonte para as anotações semânticas. Essa diferença entre Wikipedia e outros KGs ocorre porque KGs, apesar de serem estruturados em triplas, possuem consideravelmente menos dados textuais que as páginas da Wikipédia. Além Disso, algumas estatísticas das páginas da Wikipédia, como número de visitas, não estão disponíveis nos KGs. Por fim, conclui-se pela coluna Métodos, que grande parte das abordagens que utilizam embedding são voltadas para textos formais e baseadas em redes neurais. Tais redes neurais são computacionalmente pesadas, exigindo máquinas com poder de processamento considerável. Por exemplo, no artigo (CHEN et al.,2018) é utilizado um computador com um processador Intel (R)CPU Xeon (R) E5-2620 que possui 8 núcleos, podendo utilizar até 16 threads. Já no artigo (PARRAVICINI et al., 2019), o desenvolvimento do trabalho utiliza uma máquina com um processador Intel Xeon CPU E5-2680 v2 com 20 núcleos, 40 threads em 2.80GHz e 378 GB de memória RAM. Trabalhos com redes neurais produzem resultados satisfatórios para textos formais, porém necessitam de máquinas poderosas e custo-sas de manter financeiramente, o que é inviável para organizações com recursos computacionais ou financeiros limitados.

Artigo	Textos anotados	Entradas externas	Métodos
(GUO et al., 2013)	Tweets	Freebase, Wikipedia	Modelo de espaço vetorial (VSM) e Aprendizado de Classificação (LTR)
(FANG; CHANG, 2014)	Tweets	Wikipedia	Extração de informação e Recuperação de informação
(DERCZYNSKI et al., 2015)	Tweets	Lupedia, DBpedia, TextRazor, Zemanta	FFNN (Embedding de palavras)
(FANG et al., 2016)	Noticiários	Wikipedia, Freebase	Regressão logística para o modelo de duas camadas (Embedding de palavras, Embedding de conhecimento)
(YAMADA et al., 2016)	Documento de texto	Wikipedia	Regressão aumentada por gradiente Árvores (Embedding de palavras, Embedding de entidades)
(MORENO et al., 2017)	Noticiários	Wikipedia	Classificadores binários (Embedding de palavras, Embedding de entidades)
(CHEN et al., 2018)	Noticiários	Wikipedia, Freebase	Árvore de regressão aumentando em pares (Embedding de palavras, Embedding de entidades)
(LE; TITOV, 2018)	Documento de texto	Wikipedia, Yago	Campo aleatório condicional, loopy belief propagation (Embedding de palavras, Embedding de entidades)
(KOLITSAS; GANEA; HOFMANN, 2018)	Documento de texto	Wikipedia	Shallow FFNN and LSTM (Embedding de palavras, Embedding de entidades)
(ZHU; IGLESIAS, 2018)	Document de texto	DBpedia	Algoritmo de similaridade contextual semântica (Embedding de palavras, Embedding de categorias)
(MARTINS; MARINHO; MARTINS, 2019)	Noticiários	Wikipedia	Stack-LSTM (Embedding de palavras, Embedding de entidades)
(SEVGILI; PANCHENKO; BIEMANN, 2019)	Noticiários	Wikipedia, DBpedia	FFNN (Embedding de palavras, Embedding de grafos)
(WANG; IWAIHARA, 2019)	Noticiários	Wikipedia	TNN and CNN (Embedding de palavras)
(PARRAVICINI et al., 2019)	Noticiários	DBpedia	Similaridade semântica(Embedding de grafos) e heurística de pesquisa no espaço de estados
(LIU et al., 2019)	Noticiários	Wikipedia	Algoritmo Forward-Backward (Embedding de entidades)
(HOSSEINI et al., 2019)	Tweets	DBpedia	Redes neurais e embedding à estrutura de recuperação baseada em MRF
(HAN et al., 2019)	Tweets	DBpedia	Embedding de entidades
(SENS; PINHEIRO, 2019)	Tweets	DBpedia	Embedding de palavras, Embeddings de conhecimento

Tabela 1 - Comparação de abordagens holísticas para Entity Linking

3. Proposta

Este capítulo apresenta, respectivamente, o método para o treinamento dos modelos de classificação a serem utilizados para desambiguação de anotações da tarefa EL e o método para a etapa de desambiguação da tarefa EL em si.

A Figura 1 exemplifica os passos realizados pela proposta deste trabalho para a anotação semântica de um *tweet*. A proposta deste trabalho considera que a etapa de desambiguação do EL pode ser resumida como um problema de classificação binária. No caso, as entidades que descrevem as menções são classificadas como *positivas* caso façam sentido na postagem ou como *negativas* caso não façam sentido.

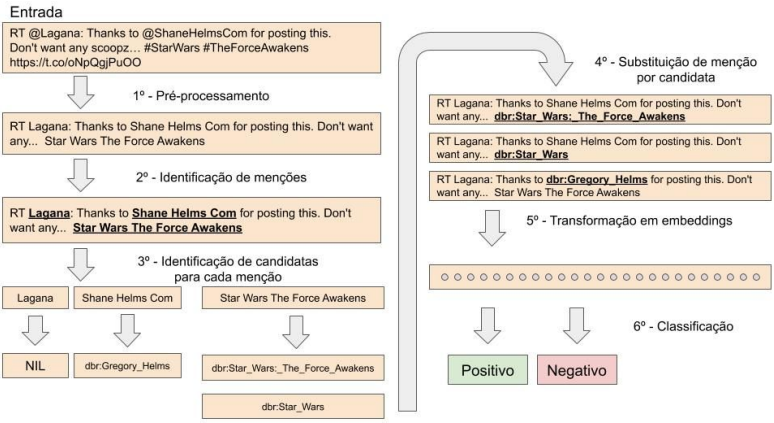

Figura 1: Fluxograma geral da proposta.

3.1. Pré-processamento

O método proposto recebe como entrada uma postagem de mídia social, no caso um *tweet*. O primeiro passo do modo proposto é o pré-processamento, que consiste em: (i) retirar links; (ii) palavras consideradas sem relevância; (ii) remoção de alguns caracteres especiais, como o \# e @; (iv) a quebra de palavras, como apresentado no exemplo onde "StarWars" foi transformado em "Star Wars".

3.2. Reconhecimento de Menções

Após o pré-processamento, usualmente o próximo passo em uma abordagem EL é o reconhecimento de entidades nomeadas no texto. Este passo, no entanto, não é abordado nesta proposta, pois o foco da mesma é tratar a etapa de desambiguação. Devido a isso, é

considerado que as menções já foram identificadas, seja manualmente ou por alguma ferramenta do estado-da-arte.

3.3. Busca de Entidades Candidatas para Menções

Com as menções identificadas, a próxima etapa desta proposta visa encontrar possíveis entidades que descrevam corretamente cada menção, ou seja, suas entidades candidatas. Para encontrar as entidades candidatas, os recursos da DBpedia que descrevem entidades foram transformados em documentos a serem pesquisados na ferramenta ElasticSearch. O Elasticsearch é um indexador de documentos especializado em buscas textuais com alta performance e que fornece inúmeras estratégias para realizar as buscas.

Cada documento indexado possui os seguintes atributos: o endereço da entidade na DBpedia (URI), possíveis nomes relacionados à entidade (nome de superfície) e o texto introdutório da entidade em questão (Contexto). Os documentos inseridos no ElasticSearch foram obtidos através de (MOUSSALLEM et al., 2017)..

A estratégia de busca utilizada no ElasticSearch é a pesquisa por N-Grams, onde a menção é quebrada em caracteres de tamanho n e a busca é realizada por similaridade com os nomes de superfície dos documentos armazenados, que também são quebrados em partes de tamanho n. Caso nenhuma entidade candidata seja encontrada para a menção, como é no caso para a menção Lagana, é atribuída a menção o valor NIL, indicando que não foi encontrada no KG uma entidade que a descreva corretamente.

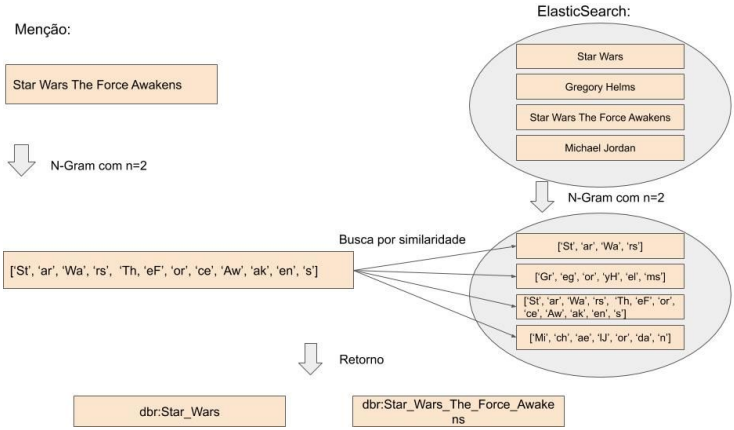

Figura 2: Exemplo de busca por N-Gram com N=2.

A Figura 2 exemplifica a busca por N-Gram, onde é dada uma menção de entrada, e.g. "Star Wars The Force Awakens", a mesma passa pelo processo de quebra do N-Gram e a busca é realizada. Então é retornado os endereços das entidades na DBPedia(URI) de candidatas que foram consideradas similares pelo ElasticSearch.

3.4. Desambiguação de Entidades Candidatas

A quarta etapa do modo proposto é a desambiguação das entidades candidatas. Para um conjunto de menções M, cada menção $m_i \in M$ possui um conjunto de candidatas C_i. Para cada candidata $c_j \in C_i$, é gerado uma sentença onde é utilizado o resultado do primeiro passo, mas cada menção $m_i \in M$ é substituída por cada uma de suas candidatas $c_j \in C_i$. β

3.5. Substituição de Palavras e Entidades Candidatas por Embeddings

O quinto passo é a substituição das palavras e entidades por seus respectivos embeddings, ou seja, pelas suas representações vetoriais. Os embeddings que foram armazenados no MongoDB foram gerados seguindo o seguinte processo: (i) adquirir as triplas da DBpedia, através do dataset Mappingbased Objects; (ii) adquirir os resumos de cada entidade da DBpedia, através do dataset Long Abstracts. Os dois datasets foram alinhados pelas entidades descritas e então transformados em *embeddings* (vetores) para cada entidade através da biblioteca *FastText*, que foi escolhida por ser capaz de criar *embeddings* de palavras e conhecimento.

3.6. Desambiguação de Entidades

O sexto e último passo é onde ocorre a desambiguação de entidades, isto é, a escolha de uma entidade candidata para ligar à sua respectiva menção, sendo definida a partir de classificação binária que irá julgar se uma menção foi corretamente anotada ou não.

Cada instância gerada no quinto passo será classificada dado um modelo de classificação previamente calculado. Tal modelo fornece uma probabilidade de que a candidata faz sentido no contexto em que ela se encontra. Porém, apenas a porcentagem sozinha não é suficiente para garantir que a desambiguação é feita de maneira correta. Uma série de regras são utilizadas para auxiliar no processo de EL, entre elas:

1. Caso uma menção tenha sido relacionada apenas com a entidade candidata *NIL*, ou seja, não foi possível encontrar possíveis candidatas na etapa de Identificação de Candidatas, pode-se pular a etapa de classificação e concluir que não foi possível desambiguar tal menção.
2. É definido um *threshold* que define uma porcentagem mínima aceitável para que uma candidata possa realmente ser considerada uma boa candidata. Se nenhuma

candidata exceder esse limite, o resultado da desambiguação não recebe candidatas retornadas.

3. Se as porcentagens de todas as candidatas para uma dada menção forem iguais, é considerado que não foi possível realizar a desambiguação e nenhuma candidata é retornada.
4. Se uma ou mais candidatas excedem o *threshold*, é retornado a candidata com maior chance de pertencer à classe positiva.

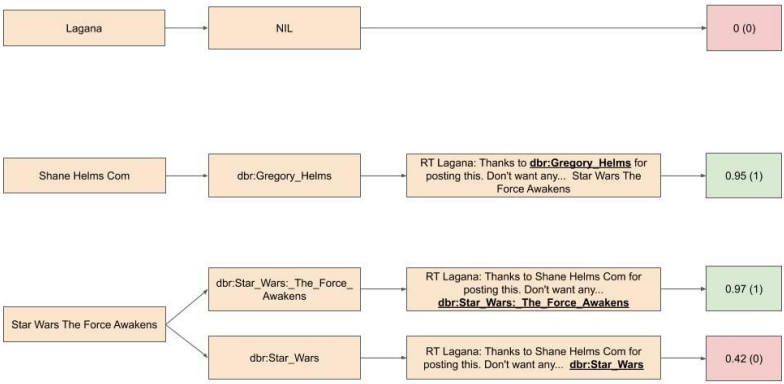

Figura 3: exemplo de classificação.

A Figura 3 demonstra 3 menções, com suas respectivas candidatas, que passam pela etapa de classificação. É demonstrado que cada candidata recebe sua própria porcentagem. Com um threshold igual à 0.9, foi possível realizar a desambiguação das menções "Shane Helms Com" e "Star Wars The Force Awakens", que foram ligadas às entidades "dbr:Gregory_Helms" e "dbr:Star_Wars_The_Force_Awakens" respectivamente. (A etapa de formação do vetor de embeddings foi ocultada no exemplo).

4. Resultados

O resultado dos experimentos é apresentado comparando com os experimentos de outras ferramentas de anotação, na tabela abaixo:

200 dimensões 300 dimensões	Memória RAM	Tempo de treinamento	Curva ROC
Naive Bayes	4.2G 5.3G	26s 55s	0.5 0.5
Decision Tree	4.2G 5.3G	26s 56s	0.5 0.5
K-Nearest Neighbors	4.2G 5.3G	25m 47m	0.75 0.75
Sigmoid SVM	4.4G 5.8G	8h 20m 12h 12m	0.78 0.79
Linear SVM	4.4G 5.8G	6h 22m 9h 7m	0.79 0.79
RBF SVM	4.4G 5.8G	8h 32m 12h 46m	0.87 0.86
Random Forest	4.2G 5.3G	50s 1m 27s	0.89 0.89

Tabela 2 - Macro e Micro F1 de abordagens executadas no sistema de benchmark GERBIL. ERR indica que o anotador gerou muitos erros no GERBIL.

A tabela acima demonstra que a abordagem adotada por este trabalho ultrapassa a performance da maioria das ferramentas atuais no dataset "Microposts2016-Test" e também possui bons resultados no dataset "Microposts2015-Test".

É notável que a ferramenta AGDISTIS/MAG apresenta os melhores resultados nos 3 datasets e isso se deve à utilização de atributos adicionais em seus classificadores, como popularidade da candidata, e uma etapa de pré-processamento mais elaborada.

5. Conclusões

Muitos dados não estruturados e semi-estruturados estão atualmente disponíveis na Web, porém é difícil de determinar sua semântica correta. Uma anotação semântica descreve o sentido de dados (ou partes deles) mediante a associação de tais dados a recursos de informação com semântica bem definida presentes em, por exemplo, grafos de conhecimento.

Entre os diversos processos automáticos de anotação existentes, destaca-se o EL. No entanto, abordagens e ferramentas automáticas atuais para anotação semântica falham em entregar anotações com boa precisão e cobertura, principalmente em dados provenientes de mídias sociais. Uma das razões para a baixa precisão e cobertura é que tais ferramentas não consideram todo o contexto que envolvem as anotações. Uma forma de contornar tal limitação é considerar o contexto existente em dados textuais já anotados, como, por exemplo, considerar as relações entre as palavras e os recursos presentes em grafos de conhecimento. A exploração de tais relações, podem ajudar a desambiguar com maior precisão novas anotações.

Este trabalho apresentou a proposta de utilizar de maneira conjunta embeddings de palavras e de conhecimento em classificadores binários. O classificador binário utilizado nesta proposta, Random Forest, foi selecionado dentre diversos outros classificadores binários devido que o mesmo obteve a melhor pontuação com a métrica AUC-ROC. Com o classificador escolhido e o modelo do mesmo gerado, podemos classificar candidatas para menções de novas publicações e realizar a tarefa de desambiguação para EL.

Comparando os resultados obtidos com outras ferramentas do estado-da-arte, é possível perceber que a abordagem deste trabalho obteve bons resultados com os datasets mais atuais "Microposts2016-Test" e "Microposts2015-Test" e uma pior performance no dataset "Micropost2014-Test".

Apesar dos resultados satisfatórios, a proposta pode ser melhorada em algumas partes. O uso de gírias e abreviações não é tratado de maneira adequada neste trabalho, além de diversos atributos (e.g. localização e data/hora da publicação, popularidade da candidata) que poderiam ser levados em consideração na etapa de classificação são completamente ignorados.

7. Referências

ALMEIDA, E. Classificação ordinal com opção de rejeição. In: . [S.l.:s.n.], 2010.AUER, S. et al. Dbpedia: A nucleus for a web of open data. In:The semantic web. [S.l.]: Springer, 2007. p. 722–735.

BHATTACHARYA, I.; GETOOR, L. Collective entity resolution in relational data.ACM Transactions on Knowledge Discovery fromData (TKDD), ACM, v. 1, n. 1, p. 5, 2007.

BOLLACKER, K. et al. Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM.Proc. of the2008 ACM SIGMOD international conference on Management of data. [S.l.], 2008. p. 1247–1250.

BONTCHEVA, K.; ROUT, D. Making sense of social media streams through semantics: A survey.Semantic Web.

CHEN, H. et al. Bilinear joint learning of word and entity embeddings for entity linking.Neurocomputing, Elsevier, v. 294, p. 12–18, 2018.

CHONG, W.-H.; LIM, E.-P.; COHEN, W. Collective entity linking in tweets over space and time. In: SPRINGER.European Conf. onInformation Retrieval. [S.l.], 2017. p. 82–94.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. In:ICML. [S.l.: s.n.], 2006.

DERCZYNSKI, L. et al. Analysis of named entity recognition and linking for tweets.Information Processing & Management, Elsevier,v. 51, p. 32–49, 2015.

EKELUND, S. Roc curves—what are they and how are they used? In:. [S.l.: s.n.], 2012.

FABIAN, M.; GJERGJI, K.; GERHARD, W. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In:16th Intl.World Wide Web Conf., WWW. [S.l.: s.n.], 2007. p. 697–706.

FAN, J.; UPADHYE, S.; WORSTER, A. Understanding receiver operating characteristic (roc) curves.CJEM, v. 8 1, p. 19–20, 2006.

KOIVUNEN, M.-R. et al. Annotea: An open rdf infrastructure forshared web annotations. 2001.

KOLITSAS, N.; GANEA, O.-E.; HOFMANN, T. End-to-end neural entity linking. In:CoNLL. [S.l.: s.n.], 2018.

LAENDER, A. H. e. a. A brief survey of web data extraction tools.ACM Sigmod Record, ACM, v. 31, n. 2, p. 84–93, 2002.

LE, P.; TITOV, I. Improving entity linking by modeling latent relations between mentions. Association for Computational Linguistics,v. 1, p. 1595–1604, 2018.

LEHMANN, J. et al. DBpedia - a crystallization point for the web of data.Journal of Web Semantics, Elsevier Science Publishers B. V.,v. 7, n. 3, p. 154–165, 2009.

LI, Y. et al. Entity disambiguation with linkless knowledge bases. In:INTL. WORLD WIDE WEB CONF. STEERING COMMITTEE.Proc. of the 25th Intl. Conf. on World Wide Web. [S.l.], 2016. p.1261–1270.LIU, C. et al. Attention-based joint entity linking with entityembedding.Information, v. 10, p. 46, 2019.

LORENA, A. C.; CARVALHO, A. C. P. de Leon Ferreira de.Estratégias para a combinação de classificadores binários em soluções multiclasses.RITA, v. 15, p. 65–86, 2008.

LUO, G. et al. Joint named entity recognition and disambiguation.In:Proc. of the Conf. on Empirical Methods in Natural LanguageProcessing. [S.l.: s.n.], 2015. p. 879–888.

MARTINS, P. H.; MARINHO, Z.; MARTINS, A. F. T. Joint learning of named entity recognition and entity linking. In:ACL. [S.l.: s.n.],2019.

MENDES, P. N. e. a. Dbpedia spotlight: shedding light on the web ofdocuments.In: ACM. Proceedings of the 7th international conference on semantic systems. [S.l.], p. 1–8, 2011.

MIN, J. H.; JEONG, C. A binary classification method for bankruptcyprediction.Expert Syst. Appl., v. 36, p. 5256–5263, 2009.

MORENO, J. G. et al. Combining word and entity embeddings for entity linking. In: SPRINGER.European Semantic Web Conference.[S.l.], 2017. p. 337–352.

MORO, A.; RAGANATO, A.; NAVIGLI, R. Entity linking meets word sense disambiguation: a unified approach.Transactions of theAssociation for Computational Linguistics, v. 2, 2014.

MOUSSALLEM, D. et al. Mag: A multilingual, knowledge-base agnostic and deterministic entity linking approach. In:K-CAP. [S.l.:s.n.], 2017.

NADEAU, D.; SEKINE, S. A survey of named entity recognition andclassification.Lingvisticae Investigationes, John Benjamins publishing company, v. 30, n. 1, p. 3–26, 2007.

NICKEL, M. et al. A review of relational machine learning for knowledge graphs.Proc. of the IEEE, v. 104, n. 1, p. 11–33, 2016.

PARRAVICINI, A. et al. Fast and accurate entity linking via graph embedding. In:GRADES/NDA@SIGMOD/PODS. [S.l.: s.n.], 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python.J.Mach. Learn. Res., v. 12, p. 2825–2830, 2011.

RATINOV L., R. D. D. D. . A. M. Local and global algorithms for disambiguation to wikipedia.In 49th Annual Meeting of theAssociation for Computational Linguistics: Human LanguageTechnologies (HLT '11), p. 1375–1384, 2011.

RENNIE, J. D. M.; RIFKIN, R. Improving multiclass text classification with the support vector machine. In: . [S.l.: s.n.], 2001.

RUDER, S.; VULIĆ, I.; SØGAARD, A. A survey of cross-lingual word embedding models. In: . [S.l.: s.n.], 2017.

SEVGILI, Ö.; PANCHENKO, A.; BIEMANN, C. Improving neural entity disambiguation with graph embeddings. In:ACL. [S.l.: s.n.],2019.SHEN, W.; WANG, J.; HAN, J. Entity linking with a knowledge base:Issues, techniques, and solutions.IEEE Transactions on Knowledge And Data Engineering, IEEE, v. 27, n. 2, p. 443–460, 2015.

SINGLA, P.; DOMINGOS, P. Entity resolution with markov logic. In:IEEE.Data Mining, 2006. ICDM'06. Sixth International Conferenceon. [S.l.], 2006. p. 572–582.

SPECKAXEL, R.; NGOMO, C. N. Ensemble learning for named entity recognition.International Semantic Web Conference, 2014.

USBECK, R. et al. Gerbil: general entity annotator benchmarking framework. In: INTERNATIONAL WORLD WIDE WEBCONFERENCES STEERING COMMITTEE.Proceedings of the24th international conference on World Wide Web. [S.l.], 2015. p.1133–1143.

WANG, Q.; IWAIHARA, M. Deep neural architectures for joint named entity recognition and disambiguation.2019 IEEE InternationalConference on Big Data and Smart Computing (BigComp), p. 1–4,2019.

WANG, Q. et al. Knowledge graph embedding: A survey of approaches and applications.IEEE Transactions on Knowledge andData Engineering, IEEE, v. 29, n. 12, p. 2724–2743, 2017.

WHANG, S. E.; GARCIA-MOLINA, H. Joint entity resolution. In:IEEE.Data Engineering (ICDE), 2012 IEEE 28th InternationalConference on. [S.l.], 2012. p. 294–305.

YAMADA, I. et al. Joint learning of the embedding of words andentities for named entity disambiguation. In:Proceedings of The 20thSIGNLL Conference on Computational Natural Language Learning.[S.l.: s.n.], 2016. p. 250–259.

ZHU, G.; IGLESIAS, C. A. Exploiting semantic similarity for named entity disambiguation in knowledge graphs.Expert Systems WithApplications, Elsevier, v. 101, p. 8–24, 2018.