Pontifícia Universidade Católica
do Rio de Janeiro

Leonardo Barbosa de Oliveira

Resolução de Correferência utilizando árvores latentes com representação contextual

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Sérgio Colcher

Rio de Janeiro
Novembro de 2020

Pontifícia Universidade Católica
do Rio de Janeiro

Leonardo Barbosa de Oliveira

Resolução de Correferência utilizando árvores latentes com representação contextual

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo.

Prof. Sérgio Colcher
Orientador
Departamento de Informática – PUC-Rio

Prof. Edward Hermann Haeusler
Departamento de Informática – PUC-Rio

Prof. Ruy Luiz Milidiú
Departamento de Informática – PUC-Rio

Rio de Janeiro, 6 de Novembro de 2020

Leonardo Barbosa de Oliveira

Graduado em sistema de informação pela Pontifícia Universidade Católica.

Ficha Catalográfica

Oliveira, Leonardo Barbosa de

Resolução de Correferência utilizando árvores latentes com representação contextual / Leonardo Barbosa de Oliveira; orientador: Sérgio Colcher. – Rio de janeiro: PUC-Rio , Departamento de Informática, 2020.

v., 122 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Árvores Latentes;. 3. Representação Contextual;. 4. Resolução de Correferência;. 5. BERT;. 6. Clusterização;. 7. SpanBERT.. I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao CNPq e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Oliveira, Leonardo Barbosa de; Colcher, Sérgio. **Resolução de Correferência utilizando árvores latentes com representação contextual**. Rio de Janeiro, 2020. 122p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A tarefa de resolução de correferência consiste em identificar e agrupar trechos de um texto de acordo com as entidades do mundo real a que se referem. Apesar de já ter sido abordada em outras conferências, a CoNLL de 2012 é um marco pela qualidade das bases de dados, das métricas e das soluções apresentadas. Naquela edição, o modelo vencedor utilizou um perceptron estruturado para otimizar uma árvore latente de antecedentes, atingindo a pontuação de 63.4 na métrica oficial para o dataset de teste em inglês. Nos anos seguintes, as bases e métricas apresentadas na conferência se tornaram o benchmark para a tarefa de correferência. Com novas técnicas de aprendizado de máquina desenvolvidas, soluções mais elaboradas foram apresentadas. A utilização de redes neurais rasas atingiu a pontuação de 68.8; a adição de representação contextual elevou o estado da arte para 73.0; redes neurais profundas melhoraram o baseline para 76.9 e o estado da arte atual, que é uma combinação de várias dessas técnicas, está em 79.6. Neste trabalho é apresentado uma análise de como as técnicas de representação de palavras Bag of Words, GloVe, BERT e SpanBERT utilizadas com árvores latentes de antecedentes se comparam com o modelo original de 2012. O melhor modelo encontrado foi o que utiliza SpanBERT com uma margem muito larga, o qual atingiu pontuação de 61.3 na métrica da CoNLL 2012, utilizando o dataset de teste. Com estes resultados, mostramos que é possível utilizar técnicas avançadas em estruturas mais simples e ainda obter resultados competitivos na tarefa de correferência. Além disso, melhoramos a performance de um framework de código aberto para correferência, a fim de contemplar soluções com maior demanda de memória e processamento.

Palavras-chave

Árvores Latentes; Representação Contextual; Resolução de Correferência; BERT; Clusterização; SpanBERT.

Abstract

Oliveira, Leonardo Barbosa de; Colcher, Sérgio (Advisor). **Coreference resolution using latent trees with contextual embedding**. Rio de Janeiro, 2020. 122p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The coreference resolution task consists of to identify and group spans of text related to the same real-world entity. Although it has been approached in other conferences, the 2012 CoNLL is a milestone due to the improvement in the quality of its dataset, metrics, and the presented solutions. In that edition, the winning model used a structured perceptron to optimize an antecedent latent tree, achieving 63.4 on the official metric for the English test dataset. During the following years, the metrics and dataset presented in that conference became the benchmark for the coreference task. With new machine learning techniques, more elaborated solutions were presented. The use of shallow neural networks achieved 68.8; adding contextual representation raised the state-of-the-art to 73.0; deep neural networks improved the baseline to 76.9 and the current state-of-the-art, which is a combination of many of these techniques, is at 79.6. This work presents an analysis of how the word embedding mechanisms Bag of Words, GloVe, BERT and SpanBERT, used with antecedent latent trees, are compared to the original model of 2012. The best model found used SpanBERT with a very large margin, achieving 61.3 in the CoNLL 2012 metric using the test dataset. With these results, we show that it is possible to use advanced techniques in simpler structures and still achieve competitive results in the coreference task. Besides that, we improved the performance of an open source framework for coreference, so it can manage solution that demand more memory and processing.

Keywords

Latent Trees; Contextual Embedding; Coreference Resolution; BERT; Clustering; SpanBERT.

Sumário

Lista de figuras

Lista de tabelas

Lista de algoritmos

Lista de símbolos

ACE – *Automatic Content Extraction*
BoW – *Bag of Words*
CBOW – *Continuos Bag of Words*
CoNLL – *Conference on Computational Natural Language Learning*
EFI – *Entropy-guided Feature Induction*
GPT – *Generative Pré-Training*
LSA – *Latent Semantic Analysis*
LSTM – *Long Short-Term Memory*
MLM – *Masked Language Model*
MUC – *Message Understanding Conference*
NER – *Named Entity Recognition*
NLP – *Natural Language Processing*
POS – *Part of Speech*
RNN – *Recurring Neural Network*
SBO – *Span Boundary Objective*
SVD – *Single Value Decomposition*

1
Introdução

Enquanto os primeiros computadores ainda estavam sendo criados, Alan Turing já se perguntava como seria uma máquina capaz de pensar. Em seu famoso Teste de Turing [1] um ser humano entrevista dois indivíduos. Um deles é uma máquina e o outro é um ser humano. Se o entrevistador não conseguir distinguir entre os entrevistados, considera-se que a máquina passou no teste e, portanto, pode-se considerar que possui algum grau de inteligência.

Esse exemplo anedótico mostra como o processamento de linguagem natural (NLP) é antigo na computação e como ele é complexo. Criar um sistema que seja capaz não só de captar os sinais enviados por um humano mas também interpretá-lo e criar uma resposta com o mesmo nível de abstração pode ser confundido facilmente com um ser consciente.

Independente de ser o caso de o Teste de Turing realmente detectar ou não inteligência, ele ainda é um objetivo que linguistas e cientistas da computação vem buscando desde a sua formulação. Como formalizar a linguagem humana e como criar sistemas que consigam utilizar essa linguagem para entender e se comunicar conosco são tópicos frequentes de pesquisa.

A solução para esse tipo de problema tem tido uma abordagem mais ou menos constante, com diferentes tipos de automatização e interferência humana. Assim como outros problemas em engenharia da computação, o processamento da linguagem natural normalmente é subdividido em tarefas menores e mais específicas, de forma que cada uma possa ser evoluída em seu próprio tempo e depois recombinadas em uma solução completa. A figura 1.1, retirada de [2], ilustra bem um sistema genérico de NLP. Nos primeiros trabalhos na área, os componentes representados eram mais distinguíveis entre si. Alguns poderiam ter soluções que eram basicamente um sistema de regras derivado por especialistas, enquanto outros poderiam ter uma abordagem estatística. Com a evolução das redes neurais, esses componentes muitas vezes não estão tão separados e acabam sendo absorvidos em um único elemento de aprendizado profundo[1] que consegue resolvê-los todos de uma única vez, os chamados sistemas de ponta-a-ponta[2].

Uma vez os problemas divididos, é preciso criar formas de medir a eficiência de uma provável solução. Para isso normalmente se constrói um conjunto de dados[3] que possui uma informação a ser aprendida e uma resposta ano-

[1] deep learning
[2] end-to-end
[3] dataset

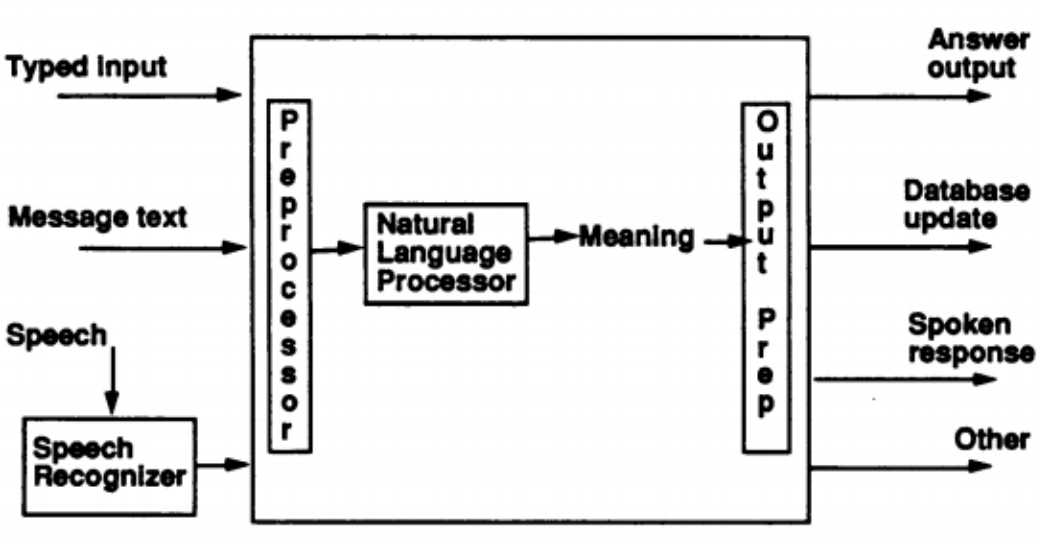


Figura 1.1: Sistema genérico de linguagem natural

tada junto a cada informação. Esse é o processo conhecido como aprendizado supervisionado e tem sido o mais utilizado para se resolver problemas de NLP.

O esforço necessário para se criar esse corpus com um tamanho e qualidade suficientes para se conseguir ter uma amostra do que é representativo de toda uma língua não é pequeno. Por esse motivo, são comuns iniciativas públicas ou privadas de entes com interesses em NLP para criar corpora e métricas, combinando-as em uma competição aberta. Agências como a *Defense Advanced Research Projects Agency* (DARPA) e empresas como Google, Facebook e Alibaba são alguns dos maiores patrocinadores desse tipo de pesquisa.

Exemplos dessas competições são as Message Understanding Conferences (MUC) que teve sete edições entre 1987 e 1997 e as Conference on Computational Natural Language Learning (CoNLL) que possui edições anuais desde 1997. Apesar de terem o nome de conferência, elas funcionam de modo semelhante, com os participantes submetendo soluções a um ou mais problemas propostos. A cada edição, além dos problemas, são definidos um conjunto de textos com anotações específicas de cada tarefa e um conjunto de métricas que serão utilizadas para se avaliar as soluções apresentadas. Entre as tarefas propostas, temos:

- **Reconhecimento de entidades nomeadas** (NER): detectar em um texto quais palavras representam uma pessoa, organização, local, data ou hora, um valor monetário ou um percentual.
- **Segmentação** (Chunking): dividir um texto em partes sintaticamente correlacionadas.

– **Correção de erros gramaticais**: detectar construções gramaticais inválidas e sugerir correções.

– **Resolução de correferência**: agrupar termos que se referem a uma mesma entidade do mundo real.

Esses exemplos ilustram como a interpretação completa de um texto pode ser automatizada em algumas partes. Apesar de não ser simples combiná-las de forma a se criar um programa que consegue entender completamente uma ordem ou mesmo responder perguntas a partir de um texto, é possível perceber que os humanos resolvem essas tarefas em um nível mais ou menos consciente quando se deparam com tarefas semelhantes.

O objetivo das tarefas propostas vai além de uma solução para um problema em especial. Para se resolver cada uma delas acaba fazendo-se necessário resolver uma série de outras subtarefas, que muitas vezes não são problemas para seres humanos. Por exemplo, é preciso definir como representar o texto no programa. Apesar de trivial ao nosso sistema auditivo, essa transformação não tem uma solução única em uma máquina que trabalha apenas com bits. É preciso escolher se serão representados cada um dos caracteres do texto, inclusive as pontuações e espaços em branco, se cada palavra tem uma representação única ou alguma outra solução diferente.

Além disso, as soluções aplicadas a um texto muitas vezes podem ser extrapoladas para outros domínios. Uma vez que se consegue capturar informações de um contexto a partir de uma sequência lógica de elementos como em NLP, é possível aplicar a solução em uma grande família de problemas que possuem a característica de serem séries temporais. Existem trabalhos que utilizam soluções nascidas em problemas de tradução de texto em áreas médicas, como em [3], ou em finanças, como em [4]. Ambos os trabalhos utilizam arquiteturas de redes neurais recorrentes que foram inicialmente criadas para resolver, entre outras coisas, problemas de NLP [5] como correção gramatical e decidir se duas frases ocorrem em sequência em um texto.

1.1 Motivação e Objetivo

A tarefa de correferência consiste em identificar os trechos de um texto que fazem referência a entidades do mundo real e depois agrupá-los de modo que cada entidade esteja representada por apenas um grupo de trechos. Por exemplo, no texto *Você e eu fomos ao laboratório. Naquele local realizamos alguns experimentos*, a tarefa de correferência deve identificar e agrupar os trechos *[Você e eu, nós]* e *[laboratório, naquele local].*

Essa tarefa foi apresentada de maneira formal na sexta edição da MUC [6], sendo que atualmente são utilizados tanto a base de dados como as métricas propostas pelas CoNLL de 2011 [7] e 2012 [8].

A tarefa de correferência é complexa e normalmente exige informações do mundo externo para poder ser completamente resolvida. As soluções encontradas na literatura são bem diversas, variando de sistemas de regras com consulta a bancos de conhecimento até redes neurais com bilhões de parâmetros.

Assim como em outros problemas de NLP, as soluções finais são compostas de partes que podem ser substituídas e é frequente nos trabalhos se encontrar a aplicação de uma nova técnica, algumas vezes provenientes até de outros domínios, em alguma parte de uma solução já conhecida.

Com os avanços em aprendizado profundo em diversas áreas do aprendizado de máquinas, como os trabalhos de [9], [10], [11] e [12] , é natural que as soluções mais recentes se valham de algoritmos que utilizam uma quantidade muito grande de dados para treinar uma quantidade muito grande de parâmetros. Por exemplo, o modelo ELMo [13] possui cerca de 93 milhões de parâmetros e o modelo BERT [12] possui 340 milhões de parâmetros. A desvantagem desse tipo de solução é o custo computacional. Ainda que esse tipo de recurso esteja se tornando cada vez mais acessível, a escalada do número de parâmetros não parece estar acompanhando a queda nos preços. Não são raros trabalhos que reportam resultados utilizando algumas dezenas de placas de vídeo (GPU) que custam alguns milhares de dólares cada. Por exemplo, o modelo BERT [12] foi treinado utilizando 16 máquinas com 8 placas de TPU de 64BG cada uma por 4 dias. Esse hardware é atualmente exclusivo do ambiente da Google e no momento da publicação deste trabalho o custo é de US$1.35[4] por hora/máquina, totalizando US$2073.60 apenas para fazer a otimização final do modelo.

Nos últimos anos houve um desenvolvimento significativo em NLP, em especial na forma de representar as palavras. Trabalhos como [13], [12] e [14] introduziram e avançaram no conceito de representação contextual de palavras. As modificações desses três trabalhos trouxeram avanços em diversas tarefas de NLP, como classificação de validade de sentenças [5] [15], avaliação de similaridade entre sentenças [6] [16] e análise de sentimento [7] [17].

A mais notável talvez seja a evolução da tarefa de perguntas e respostas denominada SQuad [18]. Nesta tarefa um ser humano tem um F1 score de

[4] https://cloud.google.com/tpu/pricing
[5] CoLA
[6] STS-B
[7] SST-2

91.2 e, antes desses modelos serem apresentados, o estado da arte chegavam a 84.4. Os modelos apresentados pelos trabalhos [13], [12] e [14] conseguiram atingir 85.8, 93.2 e 94.6 respectivamente. Ainda que esse avanço tenha sido conseguido utilizando-se deep learning, a maioria deles produz artefatos que podem ser utilizados por soluções mais simples, sem a necessidade de grandes investimentos. O modelo BERT [12] e SpanBERT [14] por exemplo, disponibilizam os seus modelos já pré-treinados para download em repositório aberto. A partir desses modelos já treinados é possível fazer uma transferência de conhecimento para um modelo especializado em outras tarefas e apenas fazer um ajuste fino. De acordo com os trabalhos, esse ajuste fino demanda uma base menor e máquinas menos exigentes.

A motivação deste trabalho é verificar se as técnicas que conseguiram tantos avanços conseguem ajudar uma versão menos exigente de uma tarefa complexa. Se essas técnicas forem eficientes será possível criar uma forma automática de caracterizar um trecho de texto e não mais depender de uma geração manual, que exige mão de obra especializada em linguística e pode não conseguir criar todas as características necessárias para se cumprir a tarefa.

A partir desta motivação este trabalho tem como objetivo **verificar a eficiência de métodos automáticos de geração de features das árvores latentes para a tarefa de correferência**. Para atingir esse objetivo será utilizada uma solução para a resolução de correferência baseada em árvores latentes e serão verificadas quatro formas de representar as palavras:

1. Bag of Words
2. GloVe
3. BERT
4. SpanBERT

Como referência são utilizados os trabalhos de [19] e [20].

Os experimentos mostram que o melhor modelo é o que utiliza a representação contextual do SpanBERT. Este modelo tem score 61.26 no dataset de test. Este resultado é comparável ao segundo colocado na CoNLL 2012, indicando que a substituição por features automáticas é possível.

As contribuições deste trabalho são:

– Uma forma de utilizar representações contextuais criadas por modelos complexos em uma estrutura de árvore latente para resolver correferência
– Melhorias no framework criado por [20] para contemplar soluções que precisam de mais memória e processamento

1.2 Organização do Texto

Este trabalho está dividido em sete capítulos, além desta introdução.

No capítulo 2,é feito um apanhado geral da história da resolução de correferência e alguns modelos notórios que tratam do problema. Também são analisados trabalhos que propõem formas de se representar as palavras em um modelo de aprendizado de máquinas.

No capítulo 3 é detalhado o que é a resolução de correferência e como ela foi definida na CoNLL 2011 e 2012. São descritos o dataset e as métricas adotadas pelas conferencias para esta tarefa em específico.

No capítulo 4 é explicado como funciona o algoritmo de árvore latente e as subtarefas auxiliares que foram utilizadas durante a CoNLL 2012 por [21] e posteriormente por [19] e [20].

No capítulo 5 são explicados em detalhe como funcionam os algoritmos de representação de palavras que foram testados neste trabalho.

Utilizando os elementos explicados dos capítulos anteriores, o capítulo 6 mostra como as diversas partes se combinam para se realizar os testes propostos por este trabalho. Também são abordadas as adaptações e melhorias feitas em cima de outros trabalhos já existentes.

No capítulo 7, são reportados os resultados encontrados, além de uma justificativa experimental para a escolha dos hiperparâmetros e uma análise de erros.

No capítulo 8, é feita uma revisão geral do problema e das soluções analisadas. Também é relatada uma série de melhorias e avanços, que pode ser realizada em trabalhos futuros.

2
Trabalhos Relacionados

A tarefa de resolução de correferência já é tratada de diversas maneiras há muitos anos. Este capítulo apresenta uma evolução histórica da tarefa desde as primeiras soluções baseadas em árvores de decisão até a ascensão dos modelos baseados em redes neurais profundas. Na parte final serão explicadas algumas famílias de soluções e também os trabalhos feitos na área de representação de palavras.

2.1
História da Resolução de Correferência

A resolução de correferência existe como um problema de linguística há muito tempo. Quando o tema começou a ser tratado como um problema computacional, as soluções vieram em formato de sistemas especialistas, ou seja, sistemas que simplesmente automatizavam um conjunto de regras que era definido por um grupo de especialistas na área. Trabalhos como [22], [23] e [24] foram alguns que implementaram essas regras. Esse tipo de solução apresentava uma forte dependência de conhecimentos específicos e o conjunto de regras raramente poderia ser utilizado em mais de uma língua. Além da falta de dados e de uma formalização de métricas de avaliação, as técnicas de aprendizado de máquina ainda eram incipientes na época.

Somente em 1995 a sexta edição da Message Understanding Conferences [6] apresentou uma abordagem mais definitiva do problema. Com forte incentivo da DARPA, a conferência apresentou um corpus anotado e métricas mais adequados ao problema. O dataset era composto de apenas 30 artigos, retirados de jornais. Ainda que um avanço relativo nessa edição, a análise dos resultados dos próprios organizadores [25] mostra que a tarefa ainda não estava bem definida e que mais estudos seriam necessários.

Na edição seguinte o MUC-7 também tratou de correferência e as soluções eram, em sua maioria, sistemas baseados em regras. Os primeiros trabalhos que mudam de paradigma são [26], [27] e [28]. Eles introduzem a ideia de se ter um sistema de pares entre possíveis menções e classificar cada um desses pares utilizando uma árvore de decisão. Essa classificação é feita a partir de algumas características do par, como o número de palavras entre as menções ou se uma menção é um pronome ou não. Com um número grande de menções, os autores aumentaram a chances de se ter duas menções próximas que eram correferentes.

Entre os anos de 2002 e 2005, o programa Automatic Content Extraction (ACE) [29] disponibilizou uma série de corpora consideravelmente maiores que o do MUC-6. Por exemplo o conjunto de 2004 continha mais de 300.000 palavras no conjunto de treino em três línguas diferentes. Apesar do tamanho, esses novos conjuntos de dados não dispunham de muitas anotações de correferência.

Com uma base maior, modelos baseados em aprendizado de máquina começaram a surgir. Muitos se concentravam em avaliar a relação entre os pares de menções e poucos se ocupavam em representar a entidade como um todo. Os primeiros trabalhos nessa direção foram [30] e [31], seguidos por [32] e, finalmente [33].

Apesar de o programa ACE oferecer uma base maior, a avaliação e comparação entre os diversos modelos ainda não era clara. Apesar de se ter métricas definidas, não havia uma implementação de referência e nem mesmo em qual dos datasets disponíveis elas deveriam ser avaliadas. Em [34] é feita uma revisão desses primeiros anos e o autor conclui que é necessário melhorar essas comparações. Em 2011 e 2012 a CoNLL [7], [8] finalmente definiu um padrão para os experimentos, incluindo uma implementação padrão das métricas e um dataset com anotações de tamanho considerável. Na edição de 2011, a maioria dos participantes utilizaram soluções de pares de menções, nos moldes de [28], mas o vencedor foi o modelo apresentado em [35] que era baseado em regras. Na edição de 2012, o vencedor foi o trabalho de [19] que adotava uma abordagem diferente, definindo árvores de antecedentes que representavam as entidades nos documentos.

Com essa mudança de abordagem, novos modelos surgiram utilizando a ideia de se representar o documento e suas entidades como estruturas. Em [36] são utilizados grafos enquanto que em [37] também são utilizadas árvores latentes e um conjunto de features ligeiramente diferente das apresentadas por [19].

Em 2017 o trabalho de [38] adotou uma abordagem nova. Até então a maioria dos modelos separam a tarefa de detectar as menções da tarefa de efetivamente criar os agrupamentos. Neste trabalho o autor propõe uma solução ponta a ponta, ou seja, tanto a detecção de quais palavras integram uma menção quanto a que grupo elas pertencem são resolvidas por um único algoritmo. Essa abordagem definiu um novo estado da arte e também a base de como os modelos seguintes seriam construídos. Apesar da melhoria, este trabalho ainda utilizava um modelo que avaliava apenas os pares de menções. Em [39] o modelo foi melhorado de forma a se ter uma distribuição de probabilidades que pode ser comparada à abordagem de estruturas latentes.

Atualmente o melhor modelo de resolução de correferência é o proposto por [14]. Esse modelo combina em uma solução ponta a ponta o objetivo de detectar as entidades e uma representação de palavras baseadas em contexto. Este trabalho conseguiu definir não só o estado da arte da tarefa de correferência como em outras 14 tarefas de NLP.

Este trabalho é o último de uma sucessão de incrementos significativos na área que, juntos, acabaram por forçar a redefinição de um conjunto de tarefas chamado GLUE [40]. Este conjunto de tarefas guia a evolução da NLP agrupando diversas tarefas e datasets que podem ser utilizados por pesquisadores como uma referência de quais são os problemas que um sistema que se propõe fazer uma interpretação de linguagem deve resolver. A tarefa de correferência depois de tantos avanços e melhorias foi incluída na definição de tarefas nomeadas SuperGLUE [41] devido à sua importância e aos novos avanços.

2.2 Família de Modelos

As diversas soluções apresentadas ao longo dos anos cobrem um grande espectro de técnicas de aprendizado de máquinas. Poucas utilizam técnicas de aprendizado não supervisionado como em [42] e [43] ou mesmo aprendizado por reforço, como em [44] e [45]. A grande maioria dos trabalhos utilizam uma abordagem mais comum de aprendizado supervisionado em que se parte de um conjunto de dados como MUC, ACE ou CoNLL e tenta se obter as respostas anotadas. Neste trabalho serão tratadas soluções deste tipo.

Como uma forma didática de organização, a literatura normalmente agrupa os trabalhos em quatro grandes famílias de soluções. Os trabalhos nem sempre estão perfeitamente encaixados em uma delas e frequentemente se inspiram em soluções de outras famílias. Os trabalhos são organizados em:

- **Pares de menções**: a solução se baseia em determinar para cada possível par de menções em um documento se ele é ou não correferente.
- **Ranqueamento** (Mention Ranking): para cada menção, um conjunto de possíveis pares são ordenados e o par mais provável de ser correferente é utilizado.
- **Baseado em entidades** (Entity-based): a solução se baseia em tentar representar a entidade como um todo, utilizando todo o documento para tomar a decisão.
- **Estrutura de antecedentes**: a solução define uma estrutura ou uma distribuição de probabilidade de como os antecedentes de uma anáfora se agrupam.

2.2.1 Pares de Menções

As soluções baseadas em pares de menções foram as primeiras abordagens a utilizar aprendizado de máquina e foram as principais metodologias adotadas nos trabalhos até a CoNLL de 2011 e 2012. A maioria dos trabalhos desta família formulam o problema principal como uma classificação binária. Ou seja, dado um par de menções extraídos do texto, um classificador é utilizado para definir se o par pertence ou não ao mesmo grupo.

Em [28], o classificador utilizado é uma árvore de decisão [46]. Ela é construída utilizando doze características simples sobre o par, como o número de palavras entre elas, se as duas possuem o mesmo texto ou se uma delas é um pronome. Os pares são gerados utilizando-se uma heurística para tentar diminuir o número de exemplos negativos, ou seja, que não são correferentes. Uma vez que os pares correferentes são definidos, o agrupamento é feito selecionando os pares cuja menções estiverem mais próximas umas das outras dentro do texto (*closest first*).

Uma forma diferente de se agrupar as menções é sugerida por [47]. Nela os pares recebem pontuações e é escolhido o par com melhor pontuação (*best first*). Apesar de os autores mostrarem que esse formato tem um resultado melhor no dataset da MUC, em [48] é mostrado que o inverso acontece nos dados da ACE. Uma outra diferença no trabalho de [47] é que ele utiliza um mecanismo para remover pares que certamente estão errados. Esse mecanismo é uma série de regras linguísticas que podem ser aplicadas, por exemplo, para que duas menções serem correferentes é preciso que elas tenham o mesmo gênero e grau, uma vez que indicam a mesma entidade real.

Uma outra variação do trabalho [28] é mostrada em [33] e [49] com a utilização de um perceptron [50], que é o elemento básico de uma rede neural. Os autores de [33] também aumentaram consideravelmente o número de características do par, utilizando um total de 41 features, incluindo informações externas ao texto, como aquela que identifica se uma menção é sinônimo ou antônimo da outra, retirada de uma base de conhecimento léxico chamada WordNet [51].

Também são encontrados trabalhos que utilizam Support Vector Machines (SVM) [52], uma técnica que cria um separador linear em um espaço multi-dimensional, como em [48] e sistemas baseados em memória, como em [53]. Em sistemas baseados em memória, os exemplos positivos são memorizados e a cada novo par de menção é feito uma comparação com que já se tem em memória para se decidir se ele é ou não correferente.

Alguns outros trabalhos seguem a linha de [33] e adicionam informações

do mundo real em seus modelos. Esse tipo de relacionamento pode ser encontrado também em bases na Wikipedia como YAGO2 [54], Freebase [55] ou WikiNet [56] que, apesar de serem menos precisas, possuem uma ampla cobertura de conhecimento. Além dessas opções, é possível extrair informações de forma automática. Em [57] os autores criam uma taxionomia a partir da Wikipedia utilizando uma rede de categorias e depois criam características para os pares de menções baseadas nessa taxionomia. Em [48] algumas características são baseadas na base YAGO.

De forma geral, esses modelos possuem uma mesma fraqueza: por se concentrarem em olhar pares específicos, soluções mais globais acabam não sendo consideradas. Mesmo quando é utilizado um agrupamento do tipo best first, essa etapa é feita de forma separada, em uma sequência de passos (pipeline) e o aprendizado fica comprometido.

2.2.2 Ranqueamento

Uma das primeiras formas de se tentar tratar os problemas das soluções baseadas em pares de menções foi a utilização do modelo com candidatos gêmeos apresentado por [30]. Nele, para cada anáfora é gerado um par de antecedentes. Um preditor indica qual dos dois antecedentes tem maior chance de ser correferente com a menção. A menção que ganhar a maioria das comparações é escolhida como antecedente. Nessa abordagem todos os possíveis pares de antecedentes são considerados durante a fase de inferência, mas apenas dois candidatos são usados na fase de predição. Para reduzir essa lista é empregado um complexo sistema de competição que é definido a priori.

Para melhorar essa abordagem, [58] propõe criar uma forma de aprender vetores associados a cada antecedente de forma que uma pontuação que seja função desse vetor possa ser usada para ranquear todos os candidatos. Nesta solução todas as menções acabam tendo algum antecedente associado, o que não é verdade para a maioria dos casos. Para minimizar esse problema, os autores propõem aplicar antes um classificador que determina se cada menção possui ou não antecedentes, ou seja, são anafóricas, e utiliza no mecanismo de ranqueamento apenas aquelas que forem classificadas como tendo antecedentes.

Uma outra forma de se resolver essa questão de que muitas menções não são anafóricas é a introdução de uma menção vazia (dummy), como explicado por [59]. Essa menção não existe na realidade e é adicionada em todas as listas de antecedentes. Assim, o vetor associado a ela deverá ser aprendido de forma que menções anafóricas a tenham como antecedente, eliminando a necessidade de se ter um classificador adicional.

Outros trabalhos utilizam abordagens semelhantes a este com algumas variações nas formas de gerar os vetores de características e como gerar os exemplos para o classificador. Muitos modelos usam features baseadas nos trabalhos de [47] e [33]. Em [60] é argumentado que uma combinação de features léxicas e uma heurística linguística pode atingir resultados expressivos na ConLL 2012.

Uma primeira abordagem que tenta fugir um pouco de se ter que estabelecer quais as features dos pares devem ser usadas é mostrada em [61]. Neste trabalho é utilizado uma rede neural [62] que aprende um vetor de features que consegue resolver a subtarefa de dizer se um par é ou não correferente.

2.2.3 Modelos baseados em entidades

Ainda que diversos trabalhos que utilizam o ranqueamento se mostrem competitivos, eles ainda trazem um problema conceitual em sua abordagem que é a falta de informações globais. Uma forma de se resolver esse problema é a construção iterativa dos agrupamentos. Em [31] para cada anáfora é decidido a qual entidade parcialmente construída ela será associada. Em [32] já adiciona informações entre os pares de entidades.

A maioria dos modelos baseados em entidades considera cada par de menção e cada entidade parcial de forma isolada durante o treino: cada um deles será considerado um exemplo positivo ou negativo para o sistema de aprendizado. Em outros modelos, como em [63] e [64], as entidades são construídas a partir da união de outras entidades parcialmente construídas. Desta forma, pode-se levar em consideração as informações entre as duas entidades ao mesmo tempo. Na figura 2.1 é possível ver um exemplo retirado de [31] de como é uma representação dessas entidades. Nesse exemplo, os números entre colchetes representam uma entidade parcial. As linhas contínuas indicam as entidades com foco e as menções marcadas com * são aquelas ativas.

O método de aprendizado adotado por essas famílias de modelos é parecido com a família de pares de menções, mas com um conjunto diferente de features. Por exemplo, [31] verifica se a palavra principal da menção é a mesma de alguma das menções já associada a alguma entidade e também a distância entre as menções será a menor distância da menção e todas as menções de uma entidade.

2.2.4

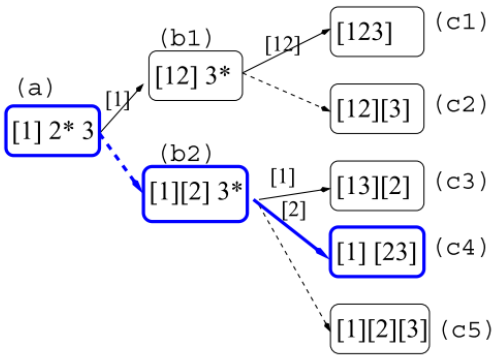


Figura 2.1: Exemplo de representação de uma árvore Bell para três menções

Estruturas de Antecedentes

Uma forma de se tomar a decisão sobre a entidade sem ter que criar características globais é criando uma estrutura de antecedentes. Um dos trabalhos que colaboraram para dar atenção a essa abordagem foi [19] com a proposta de árvores de antecedentes. Esse modelo foi o vencedor da CoNLL 2012 [8]. A estrutura foi proposta inicialmente por [65], mas neste trabalho o algoritmo aprendia um conjunto de árvores, cada uma correspondendo a uma entidade. Em [19] há apenas uma árvore por documento, com uma menção vazia na raiz e cada ramificação a partir desta sendo uma entidade. Uma outra diferença entre esses trabalhos é que em [65] o grafo é não direcionado, enquanto em [19] ele é direcionado do antecedente para a anáfora. Como cada menção só pode estar associada a um antecedente, o grafo se reduzirá a uma árvore.

Outros trabalhos como [66], [37] e [67] utilizam essa mesma estrutura. Na maioria desses é utilizado um perceptron estruturado como definido em [68] e [69]. Nesse mecanismo o perceptron deve aprender um vetor de pesos que, quando multiplicado por um vetor de características de cada arco, permite ordená-los, associando uma menção ao seu antecedente. Se a associação for errada, há uma função de custo que definirá como o vetor de pesos deve ser modificado.

Em 2017 o trabalho de [38] introduziu o conceito de ranqueamento de trecho (*span-ranking*). Nele os trechos do texto que definem uma menção são aprendidos ao mesmo tempo que os vetores de características associados a cada trecho. Ao juntar as duas subtarefas de identificar menções e agrupá-las em um só algoritmo, esse trabalho é um dos primeiros que apresenta uma solução realmente ponta a ponta.

Na figura 2.2 é possível ver o exemplo mostrado por [38]. Nele uma

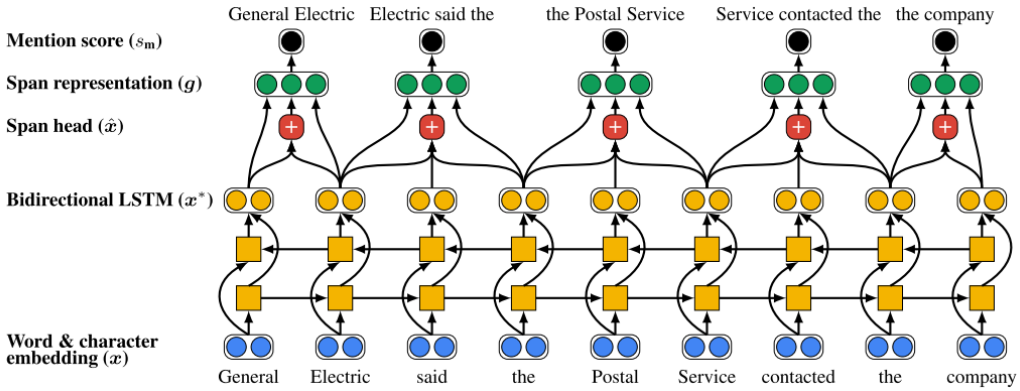


Figura 2.2: Exemplo de span-ranking
Primeiro passo na resolução de correferência, mostrando a representação dos trechos e o score das menções. Trechos com scores baixos são eliminados para se manter um número pequeno de elementos a serem tratados.

sentença possui todas as suas palavras avaliadas e os trechos de até três palavras são ranqueados. Após esse passo ainda é preciso limitar o conjunto dos elementos, eliminando aqueles com score baixo.

Esse aprendizado aplica um tipo especial de rede neural recorrente (RNN) chamada de Long Short-Term Memory (LSTM) [5] que é bem eficaz em problemas de modelagem de linguagens [70] e tradução [71]. A partir deste modelo, técnicas de aprendizado profundo (Deep Learning) passaram a dominar as soluções propostas para correferência e outras tarefas de NLP.

Em 2018, a introdução da técnica ELMo de representações contextuais mostrada por [13] criou uma revolução em diversas tarefas de NLP e a correferência não ficou de fora. Neste trabalho os autores utilizam uma arquitetura baseada no trabalho de [38], mas substituem a LSTM única por uma dupla de LSTMs que trabalham de forma paralela, uma em cada direção do texto. No mesmo ano, o trabalho de [39] conseguiu melhorar ainda mais a mesma arquitetura de ranqueamento de trechos utilizando diversas iterações para determinar a representação de cada possível trecho como uma menção. Como essa tarefa é computacionalmente cara são adotadas heurísticas para se diminuir o número de antecedentes que serão considerados a cada iteração.

Apesar de essa abordagem de span-ranking parecer um tanto distante da proposta de [19], o trabalho de [39] diz que span-ranking pode ser visto como uma forma de prever uma árvore de antecedente em que o antecedente previsto é o pai de um trecho e cada árvore é um agrupamento previsto. Ao se refinar de forma iterativa a representação dos trechos e a distribuição dos antecedentes, a distribuição conjunta de se detectar todos os trechos implicitamente modela todos os caminhos dirigidos em uma árvore de antecedentes.

Seguindo essa mesma linha de trabalhos, a aplicação da próxima geração de representação contextual trazida pela técnica BERT em [12] em um trabalho de correferência foi apresentada por [72]. A arquitetura e premissas são os mesmos daquelas apresentadas em [39], com a substituição da representação dos trechos pelo BERT. Neste trabalho é feito uma transferência de conhecimento do modelo disponibilizado por [12], mas com um refinamento posterior utilizando o dataset do OntoNotes 5.0 [73]. Este modelo já possui requisitos computacionais bastante elevados, com os autores mencionando que são necessárias placas de vídeo como pelo menos 32GB para se fazer o treinamento.

O atual estado da arte em correferência é o trabalho de [14] que propõe uma modificação na forma como o BERT é treinado, mas que ao final utiliza a mesma arquitetura que [72]. Este trabalho também elevou o estado da arte em outras tarefas de NLP e junto de trabalhos como [12] e [13] foi responsável pela definição de um novo parâmetro de comparação de tarefas de NLP chamado SuperGLUE [41].

2.3 Representação de Palavras

Um problema comum a todas as tarefas de NLP é como transformar um texto em alguma representação computacional que possa ser utilizada por um algoritmo. Uma das formas mais simples é associar cada palavra a um número. Normalmente chamada de saco de palavras (Bag of Words - BoW), a ideia é mencionada por [74] e possui algumas variações. O nome deriva do fato de que um documento pode ser representado apenas por suas palavras, não importando a ordem em que aparecem ou mesmo o seu significado.

No trabalho, o autor já aponta algumas limitações do modelo, mas argumenta que documentos semelhantes devem ter distribuições de palavras semelhantes. Por exemplo, se forem consideradas as palavras inteiras ou em partes (grams) de dois textos técnicos sobre um mesmo assunto, é esperado que as frequências dos elementos sejam próximas. Em soluções que utilizam BoW normalmente é definido uma lista de palavras, comumente referido como dicionário ou vocabulário, e cada documento é representado por um vetor de tamanho fixo, com cada posição representando quantas vezes uma palavra do dicionário ocorre naquele documento.

Como mencionado por [75], no BoW cada documento pode ser visto como um ponto em um espaço $d-$dimensional, com d sendo o número de palavras do dicionário. Como é de se esperar essa solução acaba criando um espaço de muitas dimensões. Como mencionado por [76], é a **maldição da dimensionalidade**. Em [74] já são apontadas algumas formas simples de

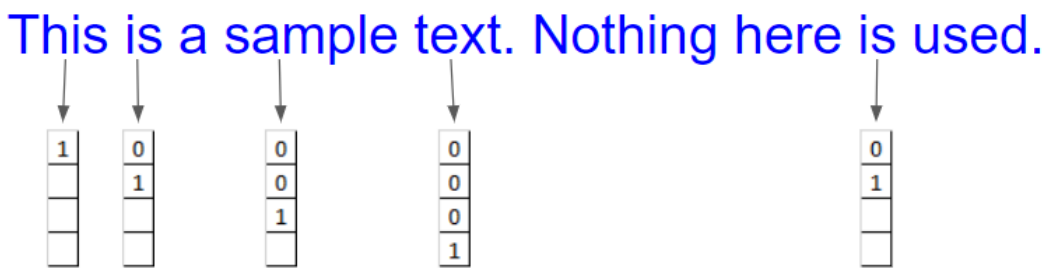


Figura 2.3: Exemplo de BoW
Exemplo de como seria a representação de um texto com apenas 4 palavras em seu vocabulário. Palavras fora do vocabulário não possuem uma representação

diminuir o número de dimensões ao se utilizar a lematização e derivação das palavras. Por exemplo, reduzir os verbos ao infinitivo e utilizar substantivos sempre no singular e no masculino.

Um exemplo fictício é mostrado na figura 2.3. Nela o vocabulário consiste em apenas 4 palavras. Cada uma delas será uma posição no vetor. Para facilitar a visualização, as células com 0 após o elemento 1 foram suprimidas.

Um problema normalmente mencionado com o simples BoW é o fato de que expressões acabam sendo descaracterizadas. Por exemplo, o termo **Federal Reserve** não é capturado de maneira adequada pelas simples ocorrências de suas palavras. Uma solução comum é considerar sequência de termos como os elementos a serem contabilizados. Essa técnica é denominada *n-grams* e ela é a base de uma família de soluções chamada de modelos de linguagem (Language Model). Nesse tipo de solução o objetivo é tentar prever qual a probabilidade de uma sequência de palavras, ou n-gramas.

Por exemplo, em [77] é feita uma comparação utilizando duas ou três palavras e implementa um modelo de linguagem para um reconhecedor de voz utilizando 3-grams. A ideia da solução apontada pode ser simplificada como sendo uma forma de reduzir a probabilidade de estimativas menos confiáveis dadas as frequências observadas e redistribuir a "probabilidade liberada" por todos os n-gramas que não ocorrem no texto, suavizando a distribuição. Em [78] também são utilizados 3-grams, porém é realizada uma comparação para determinar se os elementos devem ser palavras ou classes de palavras. Essas classes de palavras seriam grupos semânticos aprendidos utilizando um corpus de forma a se tentar particionar as palavras de maneira que cada grupo tenha a menor entropia. Os resultados do trabalho, contudo, mostram que não parece existir diferença entre essas abordagens. Em [79] são mostrados outros valores de n-grams e combinações de truques que podem gerar bons resultados

Uma outra técnica de suavização é proposta por [80], onde é utilizado o número de eventos raros de uma população grande para derivar uma estimativa

sem viés das probabilidades desses eventos. Baseado nessa análise, a melhor estimativa de probabilidade para um 3-gram que ocorre apenas uma vez em todo o conjunto de treino de tamanho N não é $1/N$, mas sim apenas uma fração disso. Com essa nova estimativa, a contagem de cada evento raro é descontada para chegar na estimativa sem viés e a soma da massa descontada é alocada nos eventos que não ocorreram.

Uma outra forma de representar as palavras em um documento é a Análise de Semântica Latente (Latent Semantic Analysis - LSA), apresentado por [81]. Nele é criada uma matriz de ocorrências das palavras nos documentos de forma que cada linha corresponde a uma palavra, cada coluna um documento e cada célula, o número de vezes que a palavra ocorre no documento. Através de algumas operações o número de linhas é reduzido de forma a se manter a similaridade entre os documentos utilizando uma análise de componentes principais [82]. Ao final, cada documento é representado pelo vetor coluna das palavras e cada palavra é representada pelo vetor linha. A similaridade entre documentos pode ser calculada como sendo o cosseno do ângulo formado entre eles.

Em [83] é apresentada uma extensão desse método chamado de Análise de Semântica Latente em Multi-visões que aplica o conceito para diversas fontes de dados ao mesmo tempo. Como indicado por [84], esse método consegue fazer um bom levantamento estatístico de informações, mas não vai bem na tarefa de fazer analogia de palavras, indicando uma estrutura sub-ótima.

Segundo [85], hoje em dia é possível treinar modelos estatísticos baseados em n-grams utilizando virtualmente toda a informação disponível, que seria algo da ordem de trilhões de palavras. Mas essa técnica parece já ter chegado ao máximo de sua performance, e para conseguir avançar é necessário utilizar técnicas de aprendizado de máquina. Em [86] já eram utilizadas redes neurais para aprender representações de distribuição, mas ainda sem grandes impactos. Em [76] é utilizada uma rede neural com todos os neurônios entre as camadas interconectados (feedforward) e uma camada de projeção linear com uma camada oculta não linear para aprender ao mesmo tempo a representação vetorial das palavras e um modelo de linguagem estatístico. Essa abordagem também foi seguida por [87], mas com a diferença que os vetores de palavras eram aprendidos antes do modelo de linguagem.

Em [85] são apresentados dois novos modelos: Bag of Words contínuo (Continuous Bag of Words - CBOW) e *skip-gram*. No CBOW é utilizada uma rede similar a [76], mas sem a camada oculta e com a camada de projeção compartilhada entre todas as palavras. Além disso, no modelo são consideradas as quatro palavras anteriores e quatro palavras posteriores a

cada termo para construir um classificador log-linear que fosse capaz de identificar qual era a palavra do meio. A diferença desse modelo para um BoW normal é que este utiliza uma distribuição contínua para representar os termos. O segundo modelo utiliza uma arquitetura similar, mas com um objetivo diferente. Ao invés de tentar classificar uma palavra a partir de um contexto local, o skip-gram tenta classificar palavras que estão a uma distância de uma referência. Segundo o trabalho, aumentar a distância entre as palavras melhora o modelo, mas aumenta a complexidade da solução. Essa segunda arquitetura foi implementada na biblioteca *word2vec*.

O aprendizado da representação das palavras e as tarefas finais foram desacopladas em [88] e permitiu que [89] utilizasse todo o contexto de uma palavra para aprender a representação das palavras.

Uma proposta bem sucedida é chamada de Global Vectors (GloVe) e foi definida em [84]. O argumento principal dos autores é de que o ponto inicial de aprendizado de um vetor de representação deve ser a relação P_ik/P_jk, onde P_ik é a probabilidade de que a palavra k apareça em um contexto da palavra i, e não as probabilidades em si. O argumento é defendido inicialmente através de um exemplo: considere as palavras $i = ice$ e $j = steam$. Para palavras k que são relacionadas a ice e não a $steam$, espera-se que a relação P_ik/P_jk seja grande. De forma análoga, palavras k relacionadas a $steam$, mas não a ice farão com que a a razão seja pequena. Palavras que não são diretamente relacionadas com nenhuma das duas farão com que a razão P_ik/P_jk seja próximo de 1. Assim, a razão é capaz de separar palavras relevantes, que são relacionadas com as palavras de interesse de palavras irrelevantes, melhor do que a simples probabilidade crua. Para resolver esse problema o algoritmo do GloVe utiliza matrizes de co-ocorrências, ou seja, ocorrências de duas palavras em um mesmo contexto, para treinar distancias entre vetores de palavras de forma que a analogia entre elas seja mantida.

Na figura 2.4 é possível ver um exemplo de como as palavras do exemplo anterior seriam representadas. O tamanho e os valores de cada representação dependerá de hiper-parâmetros do modelo e do conjunto de dados utilizado. Uma diferença desses vetores para aqueles criados com o BoW é que estes são densos e suas dimensões possuem maior significância.

Em [90], [91] e em [84] são utilizadas diferentes formas de se fazer o que é comumente classificado como pré-treinamento de vetores de palavras. Ou seja, é possível utilizar um corpus grande de dados para se fazer um aprendizado do tipo semi-supervisionado, como descrito em [90]. A diferença é que, apesar de se utilizar técnicas de aprendizado supervisionado onde há algum tipo de resposta correta, neste tipo de aprendizado essa tarefa é feita de maneira

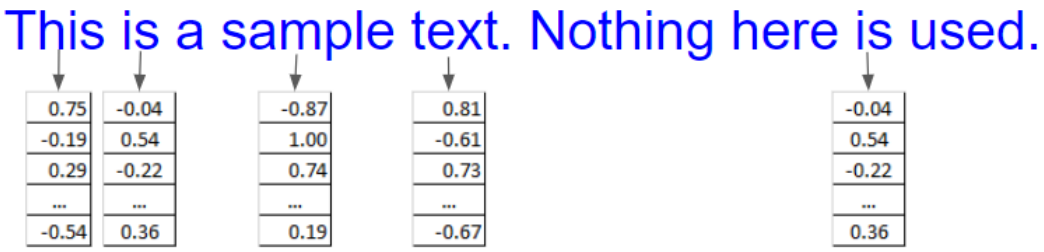


Figura 2.4: Exemplo de GloVe
Representação de algumas palavras criadas utilizando GloVe. O tamanho do vetor depende de um hiper-parâmetro

automática e não é o objetivo principal do modelo. No caso da representação de palavras, o conceito se aplica por se criar tarefas com respostas conhecidas, como definir qual a palavra que aparece em determinado contexto de maneira supervisionada, mas que os dados são gerados de maneira automática e não supervisionada. O objetivo final não é saber classificar a palavra em si, mas sim, obter alguma representação que pode ser utilizada em outras arquiteturas. Essa técnica é um tipo de aplicação do conceito chamado de transferência de conhecimento, como definido em [92] e em [93]. [90] também começa a considerar que cada posição de um vetor que representa uma palavra pode ser considerado uma característica (feature) diferente desta, ou seja, cada dimensão do vetor poderia corresponder a uma interpretação semântica ou gramatical.

Em [94] é apresentado o algoritmo batizado de context2vec. Seu objetivo é otimizar a representação do contexto de uma palavra como um todo e não apenas a palavra. Até então a representação do contexto era feita de maneira simples, utilizando-se apenas a representação de uma palavra-alvo e, possivelmente, alguma ponderação das palavras próximas a ela. Segundo o trabalho, essa abordagem acaba sendo restritiva por não modelar as inter-dependências das palavras e seus contextos sequenciais. A solução proposta é uma rede neural baseada no CBOW [85], mas substituindo a média das palavras por uma LSTM [5] bi-direcional.

No trabalho de [95] é utilizada uma técnica que não precisa de nenhuma informação além de um corpus com um tamanho suficiente. Nela é utilizada também um modelo de linguagem pré-treinado para calcular o uma representação do contexto de uma palavra. Eles mostram que a ideia de se usar duas representações, cada uma em um sentido do texto, é mais eficiente do que utilizar apenas uma.

Um avanço deste trabalho é o modelo chamado de Embeddings for Language Model (ELMo) descrito por [13]. No ELMo as representações são geradas a partir de todas as camandas internas do bi-LSTM como definida

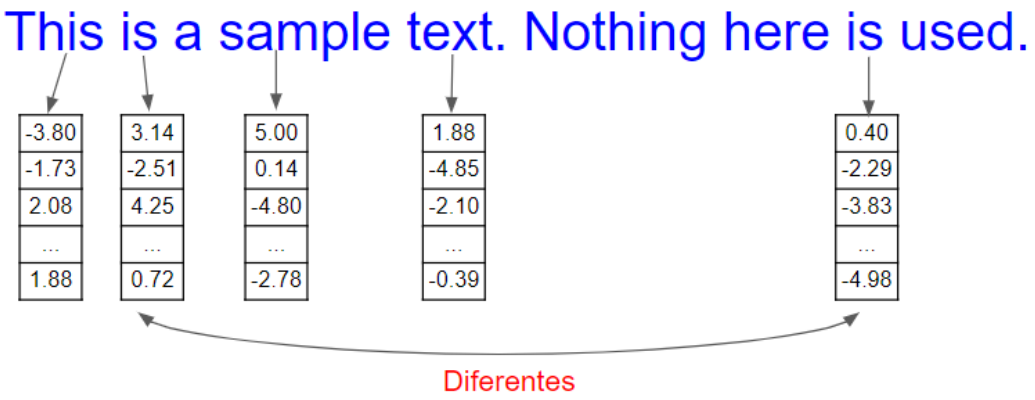


Figura 2.5: Exemplo de Representação Contextual
Representação de algumas palavras criadas utilizando uma representação contextual. Cada ocorrência possui uma representação

em [95]. Mais especificamente, o trabalho utiliza uma combinação linear dos vetores combinados com cada entrada para cada tarefa final. Essa modificação permitiu um avanço significativo em diversas tarefas de NLP, incluindo uma redução de até 20% na taxa de erro. Um exemplo da aplicação da técnica do ELMo é o trabalho de [38] que atinge um novo estado da arte na resolução de correferência, uma das tarefas que não foram tratadas no trabalho original.

Na figura 2.5 é possível ver um exemplo de como são as representações contextuais. A principal diferença é que cada ocorrência de cada palavra possui uma representação específica a ser aprendida de acordo com o seu contexto. Na figura é possível ver que a representação da primeira ocorrência de "is" é diferente da segunda ocorrência.

Seguindo essa linha de combinar um pré-treino não supervisionado com um ajuste fino supervisionado, [96] cria o Generative Pré-Training (GPT). Seguindo a linha de [97] e [98], o GPT captura a representação de um trecho de texto em sua representação. Mas, enquanto trabalhos anteriores se valiam de redes do tipo LSTM, o GPT faz uma escolha diferente e opta por utilizar Transformers [11]. Essa nova arquitetura permite observar um contexto consideravelmente maior do que os modelos anteriores. Durante a fase de pré-treinamento o modelo tem como objetivo maximizar a probabilidade de uma palavra ocorrer após uma sequência de outras palavras de tamanho pré-determinado. Durante a fase de ajuste fino, são utilizados dois objetivos: maximizar a probabilidade de se obter uma resposta pré-definida a partir de uma sequência de palavras; e novamente o objetivo de modelo de linguagem da primeira fase.

Um trabalho intencionalmente feito para ser parecido com o GPT foi o BERT [12]. A diferença principal são duas tarefas auxiliares introduzidas no

BERT durante a fase de pré-treino. Além disso, o BERT foi treinado em um corpus bem maior, e cada etapa do treino possuía quatro vezes mais palavras que o GPT. Há também alguns outros detalhes menores de hiperparâmetros que, segundo [12], podem ter contribuído para uma melhor performance. As duas novas tarefas introduzidas são o modelo de linguagem mascarado (Masked Language Model - MLM), no qual alguns elementos de um texto são mascarados e o modelo deve aprender a descobrir qual elemento foi escondido; e uma tarefa em que são apresentadas duas frases e o modelo deve aprender a classificar se a segunda frase ocorre logo após a primeira (Next Sentence - NS).

O BERT não é apenas um modelo. Ele é uma arquitetura que possui diversos hiperparâmetros que podem ser ajustados e novas partes anexadas. Segundo essa lógica vieram trabalhos como [99] no qual é feita uma extensiva análise de como otimizar os diversos hiperparâmetros do BERT e ainda aumentaram o tamanho do dataset de treino. No trabalho de [100] é apresentado uma forma diferente de se quebrar o texto em tokens melhor do que o trabalho original de [12] e, consequentemente, diminuir o tamanho do modelo. Em [101] são propostas modificações mais estruturais, como a substituição dos Transformers por um modelo parecido, mas bem maior chamado Transformer-XL, apresentado por [102], e um novo objetivo de otimização.

Umas das mais recentes modificações propostas é a feita por [14] e chamada de SpanBERT. O trabalho dos autores é bem extenso, com uma reimplementação completa do BERT original e pequenos ajustes. O modelo SpanBERT tem como motivação tarefas que não foram diretamente atacadas pelo trabalho original, em especial a tarefa de resolução de correferência. Nesse novo modelo foram feitas duas grandes modificações: na tarefa de MLM, ao invés de serem mascarados elementos soltos de um texto, são mascarados trechos seguidos de tamanhos variados; e a tarefa de NS foi substituída pela tarefa de descobrir os tokens mascarados apenas utilizando os tokens nas extremidades do trecho. Essa segunda tarefa foi chamada de Span Boundary Objective (SBO). Com essas modificações o SpanBERT conseguiu atingir pontuações superiores ao antigo estado da arte em diversas tarefas de NLP.

3
Resolução de correferência

A resolução de correferência pode ser entendida como a tarefa de identificar e agrupar os termos de um texto que se referem a uma mesma entidade no mundo real. Um exemplo é o da citação 1, retirada do trabalho [20]. Nela estão destacados os trechos (menções) entre colchetes que representam alguma entidade no mundo real; na parte de fora está indicado a qual entidade cada menção se refere. A tarefa de resolução de correferência equivale a indicar onde estão os colchetes e qual a numeração de cada um deles.

$$\begin{gathered}[Vicente\ del\ Bosque]_1\ admits\ it\ will\ be\ difficult\ for\ [him]_1\ to\ select \\ [David\ de\ Gea]_2\ in\ Spain's\ European\ Championship\ squad\ if \\ [the\ goalkeeper]_2\ remains\ on\ the\ sidelines\ at\ [Manchester\ United]_3. \\ [De\ Gea's]_2\ long\text{-}anticipated\ transfer\ to\ [Real\ Madrid]_4\ fell\ through\ on \\ Monday\ due\ to\ miscommunication\ between\ [the\ Spanish\ club]_4\ and \\ [United]_3\ and\ [he]_2\ will\ stay\ at\ [Old\ Trafford]_3\ until\ at\ least\ January\end{gathered} \tag{1}$$

Apesar de existir variações da tarefa que considerem frases verbais ou causais como em [103], [104] e [105], ou mesmo relações mais fracas do que a identidade de referência (*bridging*), como em [106] e [107], neste trabalho serão tratadas apenas as correferências de frases nominais (*noun phrases*) e pronomes.

Este capítulo apresenta o problema da correferência de maneira formal e explica alguns conceitos linguísticos que são úteis neste contexto. Nele também há um detalhamento do que foi a CoNLL 2011/2012 e quais foram as métricas e os dados que foram definidos durante essas competições. Por fim será apresentado o atual estado na arte na tarefa de correferência.

3.1
Definição Formal

Para se manter uma coerência durante todo o trabalho, serão utilizadas as definições e terminologia de [20]. As definições a seguir foram ligeiramente adaptadas para a realidade deste trabalho.

Definição 3.1 *Seja d um documento e $\mathcal{M}_d$ o conjunto de todas as frases nominais e pronomes em d. Todos os elementos de $\mathcal{M}_d$ serão chamados de menções ou expressões candidatas.*

As menções serão ordenadas de acordo com a sua posição no texto. Essa ordenação será útil quando for definida uma distância entre as menções e quando for definido o grafo na árvore de antecedentes.

Definição 3.2 *Dado duas menções m e n de $\mathcal{M}_d$, é definido que $m < n$ se m inicia antes de n ou ambas iniciam no mesmo termo, mas m termina antes de n. Menções nesta ordem serão descritas como $m_1, m_2, ..., m_k$ para as k primeiras menções.*

Segundo [108], a relação de correferência entre duas menções é reflexiva, simétrica e transitiva, e, portanto, é uma relação de equivalência. Desta forma uma classe de equivalência contém todas as menções que se referem a uma mesma entidade. Utilizando essa relação, é possível então definir correferência da seguinte forma:

Definição 3.3 *A tarefa de correferência é a tarefa de criar um sistema que consiga predizer as classes de equivalência de um documento d a partir do conjunto $\mathcal{M}_d$.*

3.2 Linguística

Ao longo do trabalho serão utilizados alguns termos linguísticos. Assim, é importante ter claro a sua definição no atual contexto. Aqui as definições foram retiradas de [109]

Definição 3.4 *reference (n.) (1) In grammatical analysis, a term often used to state a rela- tionship of identity which exists between grammatical units, e.g. a pronoun refers to a noun or noun phrase. When the reference is to an earlier part of the discourse, it may be called back-reference (or anaphora); correspondingly, reference to a later part of the discourse may be called forward-reference (or cataphora). In switch reference languages, the verb indicates whether the subjects of successive clauses are the same or different. (2) See referent.*

Definição 3.5 *referent (n.) A term used in philosophical linguistics and semantics for the entity (object, state of affairs, etc.) in the external world to which a linguistic expression relates: for example, the referent of the name Bill Clinton is Bill Clinton himself. The term is found both as part of a two-term analysis of meaning (e.g. words things) and in three-term analyses (e.g. words concepts things). In linguistics, care is usually taken to distinguish knowledge of the world from knowledge of language: the extralinguistic notion of reference is contrasted with the intralinguistic notion of sense, a property arising from the meaning relations between lexical items and sentences.*

As definições 3.4 e 3.5 aparecem com frequência durante o estudo de correferência. Enquanto a definição 3.4 de referência indica a ocorrência de uma relação de identidade entre dois termos gramaticais, a definição 3.5 de referente é uma entidade do mundo externo. O texto também cita a distinção que é feita entre a informação restrita ao que é da linguagem e ao que é do mundo externo. Um termo adicional que é citado por [20] é a referência anafórica. Ainda segundo [109], temos:

Definição 3.6 *anaphora (n.) A term used in grammatical description for the process or result of a linguistic unit deriving its interpretation from some previously expressed unit or meaning (the antecedent). Anaphoric reference is one way of marking the identity between what is being expressed and what has already been expressed. In such a sentence as He did that there, each word has an anaphoric reference (i.e. they are anaphoric substitutes, or simply anaphoric words): the previous sentence might have been John painted this picture in Bermuda, for instance, and each word in the response would be anaphorically related to a corresponding unit in the preceding context. Anaphora is often contrasted with cataphora (where the words refer forward), and sometimes with deixis or exophora (where the words refer directly to the extralinguistic situation). It may, however, also be found subsuming both forwards- and backwards-referring functions. The process of establishing the antecedent of an anaphor is called anaphora (or anaphor) resolution, and is an important research aim in psycholinguistics and computational linguistics*

Nesta definição de anáfora 3.6, o autor indica uma relação de interpretação ou sentido. Ou seja, uma anáfora precisa de um antecedente para ser entendido. Desta forma, é possível perceber uma diferença entre a definição de resolução de anáforas e resolução de correferência. Se um nome ocorre duas vezes na mesma frase, como no exemplo da definição 3.5, então não há uma relação anafórica entre as ocorrências, já que o entendimento delas é independente. Porém, há uma correferência, já que ambas dizem respeito ao mesmo ente real. Apesar dessa distinção, a maioria dos trabalhos utiliza os termos anáfora e antecedente segundo a definição 3.6.

Definição 3.7 *Seja $\mathcal{M}_d$ o conjunto de menções de um documento d. Dado duas menções de $\mathcal{M}_d$, m e n com $n < m$, então m será a anáfora e n será o antecedente.*

No trabalho de [20], os pares de menções são geralmente marcados com algo como $pair(m, n)$, ou seja, a anáfora é o primeiro parâmetro e o antecedente é o segundo. Essa notação é diferente do trabalho de [19], onde a notação é do tipo $pair(n, m)$, com as menções na ordem natural. Como a implementação

deste trabalho segue mais a implementação de [20], será seguida a notação $pair(ana, ante)$. Ou seja, será usado o padrão $pair(m, n)$, mas com o cuidado de não deixar a referência ambígua.

3.3 CoNLL 2011/2012

As primeiras vezes que a tarefa de resolução de correferência foi tratada por uma competição de tamanho significativo foi nas MUC-6 [25] e MUC-7 [110]. Apesar do esforço e do avanço relativo que elas proporcionaram, ainda havia a necessidade de datasets melhores e métricas mais bem definidas. Foram as edições de 2011 e 2012 da Conference on Natural Language Learning (CoNLL) [7], [8] que conseguiram realmente atingir o objetivo de definir propriamente a tarefa, incluindo um dataset e métricas.

A edição de 2011 foi motivada por ainda não existir uma competição que utilizasse os dados do OntoNotes [111] que, apesar de existir desde 2007, foi revisado e ampliado por [112] em 2011. A tarefa proposta segue a definição 3.3 e deveria utilizar os dados do OntoNotes 4.0, incluindo as outras camadas existentes no dataset. Nesta tarefa, as menções que não possuem um antecedente (singletons) não devem ser identificadas.

Foram propostas duas versões da tarefa: uma fechada e outra aberta. Na versão fechada, apenas os dados fornecidos, a WordNet [51], que é um banco de conhecimentos e uma tabela de gênero e número levantada por [113] podem ser usados. Na versão aberta, qualquer recurso adicional poderia ser utilizado, incluindo a Wikipedia e outros sistemas pré-existentes.

Em ambas as versões foram fornecidos dados adicionais com anotações de outras camadas, geradas de forma automática, usando ferramentas já prontas. Além disso, foram fornecidas anotações manuais, chamadas de *gold-standard*. Os participantes podiam escolher se utilizariam um ou ambos os tipos de dados para treinar seus modelos.

Para as métricas foram escolhidas a MUC [114], B-CUBED [115], a CEAF [31] e a BLANC [116], sendo que esta última não foi incluída na métrica final, que é a média das três primeiras. Nesta competição já foi fornecida uma implementação de referências para todas as métricas.

Ao final da competição, a organização constatou em [7] que o melhor resultado ainda era baseado em regras e que os sistemas participantes não tentaram utilizar toda a informação presente no OntoNotes para conseguir melhores previsões. No total foram 18 participantes nesta edição.

Na edição de 2012, a CoNLL propõe a mesma tarefa do ano anterior, com a diferença de que foram acrescentados os dados em árabe e chinês, além

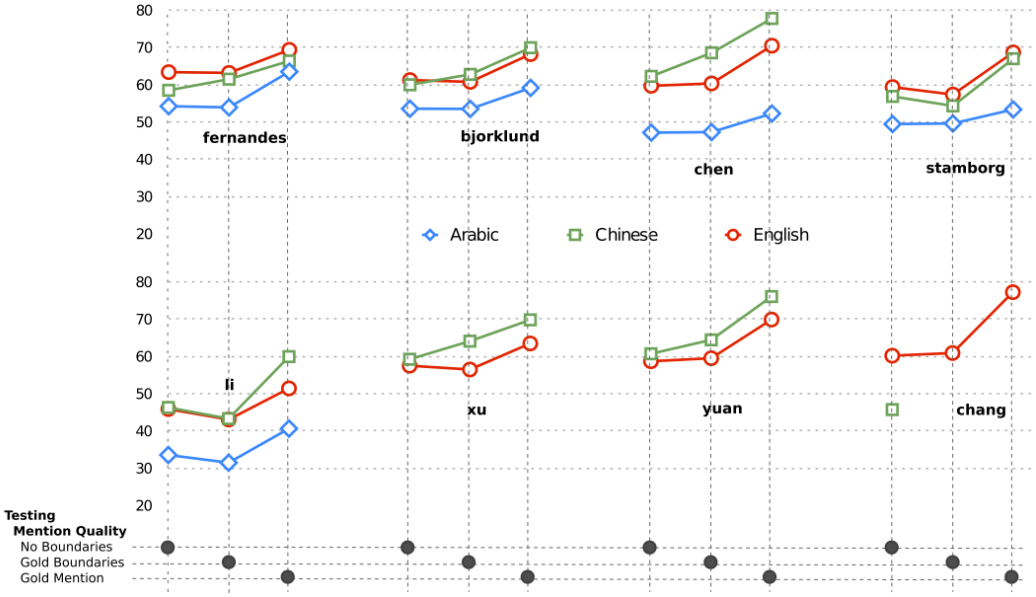


Figura 3.1: Performance de oito participantes da CoNLL 2012
Avaliação utilizando três formas diferentes de se detectar as menções

da versão 5.0 do OntoNotes. Neste ano, o número de participantes foi de 41 e a maioria das melhores soluções utilizaram híbridas, combinando partes baseadas em regras e partes baseadas em alguma técnica de aprendizado de máquina. O melhor modelo foi o de [19], tento a melhor média geral na tarefa fechada e a melhor média no conjunto em inglês e árabe. Olhando apenas para a língua inglesa, os dois melhores trabalhos foram [19] e [20]. Nesta segunda edição a organização fez um trabalho bem elaborado de comparação entre diversas soluções. Na figura 3.1 é possível ver a comparação da performance entre oito participantes da competição. Essa performance foi avaliada no conjunto de treino e foram feitas três variações:

- **No Boundaries**: essa foi a modalidade oficial, em que os sistemas não recebiam nenhuma informação sobre as menções.
- **Gold Boundaries**: nessa modalidade foram adicionadas as informações de todas as possíveis menções que poderiam ser utilizadas.
- **Gold Mention Boundaries**: nessa modalidade foram adicionadas as informações de todas e apenas as menções que estavam anotadas no *gold-standard*.

Apesar de as modalidades que incluem as informações das menções ser um tanto artificiais, elas mostram o quanto os sistemas poderiam ganhar se fosse melhorado o mecanismo de detecção de menções. Em especial, a modalidade *Gold Mention Boundaries* é um tanto artificial, já que inclui uma

parte da informação da resposta, que é identificar e remover os singletons. Segundo a organização, mesmo assim valia a pena a análise, já que poderia gerar alguma informação de quão bom era o mecanismo de agrupamento das soluções.

A figura 3.1 mostra que simplesmente acrescentar a informação de todas as menções não melhora a performance dos participantes mostrados na solução em inglês e árabe. Isso indica que as soluções já possuíam algum mecanismo de detecção de menções com um recall próximo de 100%. Para o chinês já há uma melhoria. A análise feita em [8] pondera que esse efeito se dê pelo fato de a distribuição das menções entre os gêneros no dataset em chinês é diferente e se houvesse mais menções em outros gêneros, provavelmente a performance seria melhor.

3.4 Dataset - OntoNotes

Quando o OntoNotes foi criado em 2007 [111] já existiam outros conjuntos de dados como o MUC e ACE. Mas, segundo [7] eles eram limitados em tamanho escopo, limitando o progresso da pesquisa. O objetivo do OntoNotes é descrito por [112] no trecho 2.

> *Our goal is to provide data in multiple languages and multiple genres (...), richly annotated by a skeletal representation of the literal meaning of sentences, so that a new generation of language understanding would deliver new functional capability.* (2)

Esse objetivo de criar um conjunto de dados que seja não só grande o suficiente, mas também diverso e anotações de diversos tipos permitiu avanços consideráveis em diversas áreas de NLP. Ao adicionar várias camadas com informações diferentes sobre um mesmo texto, os desenvolvedores dizem que é possível fazer verificações de consistência entre as anotações para posterior correções manuais e é possível procurar por relações entre as anotações, o que pode ser objeto de pesquisa.

Na figura 3.2 é esquematizado como as diversas camadas de informação são geradas e compostas para se chegar ao OntoNotes. O Treebank e o PropBank são formas de se representar a sintaxe e a semântica de um texto. No OntoNotes eles foram inspirados pelos trabalhos [117] e [118]. Ambos possuem uma representação em árvore e os termos podem ser alinhados. Os

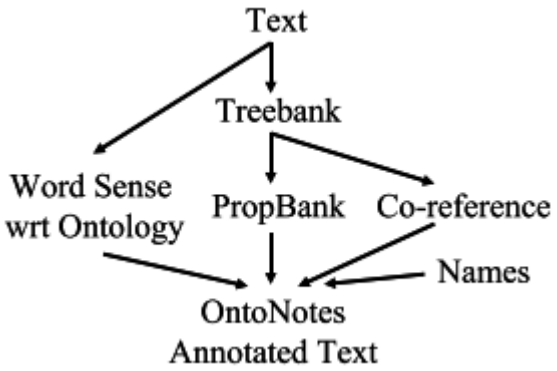


Figura 3.2: Níveis de anotação no OntoNotes

```
#begin document (nw/wsj/07/wsj_0771); part 000
...
...
nw/wsj/07/wsj_0771 0   0         ``   ``  (TOP(S(S*          -    -  -   -         *       *      (ARG1*          *          *    -
nw/wsj/07/wsj_0771 0   1 Vandenberg  NNP       (NP*           -    -  -   -  (PERSON)  (ARG1*          *          *  (8|(0)
nw/wsj/07/wsj_0771 0   2        and   CC        *            -    -  -   -         *       *           *          *    -
nw/wsj/07/wsj_0771 0   3    Rayburn  NNP       *)           -    -  -   -  (PERSON)       *)          *          *  *(23)|8)
nw/wsj/07/wsj_0771 0   4        are  VBP      (VP*        be   01  1   -         *     (V*)          *          *    -
nw/wsj/07/wsj_0771 0   5     heroes  NNS    (NP(NP*)       -    -  -   -         *  (ARG2*           *          *    -
nw/wsj/07/wsj_0771 0   6         of   IN      (PP*         -    -  -   -         *       *           *          *    -
nw/wsj/07/wsj_0771 0   7       mine   NN    (NP*))))       -    -  5   -         *       *)          *          *    (15)
nw/wsj/07/wsj_0771 0   8          ,    ,        *         -    -  -   -         *       *           *          *    -
nw/wsj/07/wsj_0771 0   9          ,    ,        *)        -    -  -   -         *       *)          *          *    -
nw/wsj/07/wsj_0771 0  10        Mr.  NNP       (NP*        -    -  -   -         *       *      (ARG0*     (ARG0*          *    (15
nw/wsj/07/wsj_0771 0  11      Boren  NNP        *)        -    -  -   -  (PERSON)       *)          *)         *    15)
nw/wsj/07/wsj_0771 0  12       says  VBZ      (VP*      say   01  1   -         *       *        (V*)          *          *    -
nw/wsj/07/wsj_0771 0  13          ,    ,        *         -    -  -   -         *       *           *          *          *    -
nw/wsj/07/wsj_0771 0  14  referring  VBG     (S(VP*     refer   01  2   -         *       *  (ARGM-ADV*        (V*)          *    -
nw/wsj/07/wsj_0771 0  15         as   RB     (ADVP*        -    -  -   -         *       *           *  (ARGM-DIS*          *    -
nw/wsj/07/wsj_0771 0  16       well   RB        *)        -    -  -   -         *       *           *)         *    -
nw/wsj/07/wsj_0771 0  17         to   IN      (PP*         -    -  -   -         *       *           *     (ARG1*          *    -
nw/wsj/07/wsj_0771 0  18        Sam  NNP    (NP(NP*        -    -  -   -  (PERSON*       *           *          *          *    (23
nw/wsj/07/wsj_0771 0  19    Rayburn  NNP        *)        -    -  -   -         *)      *           *          *          *    -
nw/wsj/07/wsj_0771 0  20          ,    ,        *         -    -  -   -         *       *           *          *          *    -
nw/wsj/07/wsj_0771 0  21        the   DT    (NP(NP*        -    -  -   -         *       *           *          *     (ARG0*    -
nw/wsj/07/wsj_0771 0  22 Democratic   JJ         *         -    -  -   -    (NORP)       *           *          *          *    -
nw/wsj/07/wsj_0771 0  23      House  NNP         *         -    -  -   -     (ORG)       *           *          *          *    -
nw/wsj/07/wsj_0771 0  24    speaker   NN        *)         -    -  -   -         *       *           *          *       *)    -
nw/wsj/07/wsj_0771 0  25        who   WP  (SBAR(WHNP*)      -    -  -   -         *       *           *          *  (R-ARG0*)    -
nw/wsj/07/wsj_0771 0  26 cooperated  VBD     (S(VP*  cooperate   01  1   -         *       *           *          *       (V*)    -
nw/wsj/07/wsj_0771 0  27       with   IN      (PP*         -    -  -   -         *       *           *          *     (ARG1*    -
nw/wsj/07/wsj_0771 0  28  President  NNP      (NP*         -    -  -   -         *       *           *          *          *    -
nw/wsj/07/wsj_0771 0  29 Eisenhower  NNP *))))))))))      -    -  -   -  (PERSON)       *           *)         *)         23)
nw/wsj/07/wsj_0771 0  30          .    .       *))        -    -  -   -         *       *           *          *          *    -

nw/wsj/07/wsj_0771 0   0         ``   ``  (TOP(S*          -    -  -   -         *       *           *          -
nw/wsj/07/wsj_0771 0   1       They  PRP       (NP*)        -    -  -   -         *  (ARG0*)          *        (8)
nw/wsj/07/wsj_0771 0   2    allowed  VBD       (VP*      allow   01  1   -         *     (V*)          *          -
nw/wsj/07/wsj_0771 0   3       this   DT     (S(NP*         -    -  -   -         *  (ARG1*     (ARG1*         (6
nw/wsj/07/wsj_0771 0   4    country   NN         *)         -    -  3   -         *       *           *)         6)
nw/wsj/07/wsj_0771 0   5         to   TO       (VP*         -    -  -   -         *       *           *          -
nw/wsj/07/wsj_0771 0   6         be   VB       (VP*      be   01  1   -         *       *        (V*)       (16)
nw/wsj/07/wsj_0771 0   7   credible   JJ  (ADJP*)))))      -    -  -   -         *      *)    (ARG2*)         -
nw/wsj/07/wsj_0771 0   8          .    .       *))        -    -  -   -         *       *           *          -
#end document
```

Figura 3.3: Exemplo de um documento no formato CoNLL

desenvolvedores do OntoNotes fizeram algumas adaptações de forma que esse alinhamento pudesse ser feito.

As anotações passaram por um cuidadoso fluxo de verificações e checagens de forma que sempre houvesse um nível de 90% de acordo entre os anotadores.

Para a CoNLL o OntoNotes é formatado em um estilo específico, definido pela competição. Esse formato pode ser visto na tabela 3.1, retirada de [8]. Esse formato é uma compilação de várias camadas do OntoNotes de forma a organizar tudo em um único arquivo. Essa organização é feita utilizando-se rotinas disponíveis pela organização da competição. Um exemplo concreto de como os dados ficam organizados pode ser visto na figura 3.3.

A figura 3.3 mostra um documento completo no formato CoNLL. É possível notar que cada sentença tem um número diferente de colunas, sendo que na última coluna está a anotação dos agrupamentos das menções. Nessa

Coluna	Tipo	Descrição
1	ID do documento	Variação do nome do arquivo do documento
2	Parte	Alguns arquivos são divididos em partes numeradas como 000, 001, 002, etc
3	Número da Palavra	Índice da palavra dentro da frase
4	Palavra	A palavra em si
5	Parte do Discurso	Parte do Discurso (POS) relativo à palavra
6	Trecho da análise	Estrutura da análise sintática separada em colchetes, quebrando-se cada parte antes de abrir o colchete e a palavra/parte do discurso substituído por um *. A análise completa pode ser reconstruída substituindo-se o * por ([posição] [palavra]) e concatenando-se os itens em uma linha daquela coluna.
7	Lema	O lema do predicado é mencionado para as linhas nas quais há um papel semântico ou informação de sentido da palavra. As demais linhas são marcadas com um -
8	ID do conjunto de Predicados	Identificador do conjunto de Predicados (Frameset ID) no PropBank relativo à coluna 7
9	Sentido da Palavra	Sentido da palavra da coluna 4
10	Orador/Autor	Nome do orador ou autor quando disponível
11	Entidade Nomeadas	Essa coluna identifica os trechos que representa as entidades nomeadas
12:N	Argumentos de Predicado	Uma coluna para cada argumento da estrutura de predicados mencionado na coluna 7
N	Correferência	Informação da cadeia de correferência codificada em uma estrutura de parêntesis

Tabela 3.1: Formato CoNLL dos dados

estrutura de parêntesis, há a marcação de quando começa e quando termina uma menção bem como a qual grupo ela pertence. Nesse formato, cada parte é um documento separado, mesmo que elas façam parte de um mesmo texto originalmente. Os números dos agrupamentos são definidos dentro de cada arquivo, mas devem ser considerados como indicadores dentro apenas de cada parte. É possível notar também na figura que algumas palavras fazem parte de mais de uma menção. Neste caso há a indicação de todos as menções a que ela faz parte.

3.5 Métricas

A métrica de avaliação de um algoritmo de resolução de correferência não é bem definida. Atualmente a forma mais adotada é a descrita em [7] que

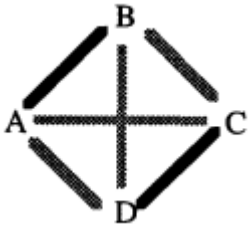


Figura 3.4: Exemplo de cálculo do recall na métrica MUC
Neste exemplo, os arcos escuros indicam as partições da resposta $< A - B,\ C - D >$. Para este exemplo, o recall é de 2/3

utiliza a média do score F1 (média harmônica entre precisão e acurácia) de três métricas: MUC, B-CUBED e CEAF. Cada métrica avalia um aspecto da tarefa principal.

A métrica Message Understanding Conferences (MUC) foi definida em [114] e mede a eficiência da ligação entre as menções. Dado um conjunto de equivalências S gerado a partir de ligações chave, o recall é definido pela equação mostrada em 3-2, sendo que $p(S)$ é a partição de S que está na resposta sendo avaliada , $c(S)$ é o número mínimo de ligações necessárias para se definir S, $m(S)$ é o número de links que estão faltando na resposta.

$$\begin{aligned} Recall &= 1 - \frac{m(S)}{c(S)} \\ &= \frac{c(S) - m(S)}{c(S)} \\ &= \frac{(|S| - 1) - (|p(S)| - 1)}{|S| - 1} \\ Recall &= \frac{|S| - |p(S)|}{|S| - 1} \end{aligned} \tag{3-1}$$

Como exemplo, para o conjunto de ligações chave $< A{-}B, B{-}C, B{-}D >$ e uma resposta $< A - B,\ C - D >$, o conjunto de equivalência fica sendo definido como $S = ABCD$. As partições relativas à resposta são $p(S) = \{\{A,\ B\}\ \{C,\ D\}\}$. O número mínimo de ligações necessárias para se definir o conjunto será o tamanho da árvore geradora mínima do grafo completamente conectado entre todos os elementos, ou seja, $|S| - 1$. A figura 3.4 mostra como seria avaliado este exemplo.

Os erros de precisão para a métrica MUC indicam quantas ligações foram feitas fora do conjunto de ligações chave, como ilustrado pela figura 3.5. O racional é então fazer o contrário do recall, definindo um conjunto S' que é um conjunto de equivalência definido pela resposta e $p'(S')$ é a partição de S' gerado a partir das ligações chave. De forma análoga então, a precisão pode

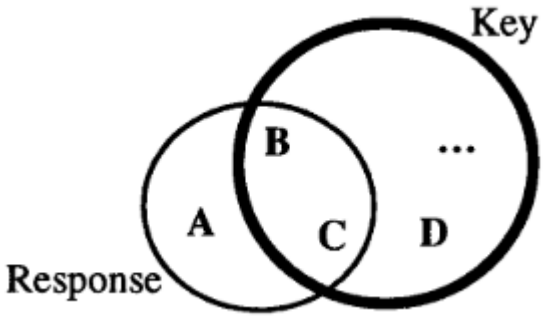


Figura 3.5: Exemplo de um erro de precisão

ser definida de acordo com a equação

$$\begin{aligned} Precisão &= 1 - \frac{m'(S')}{c'(S')} \\ &= \frac{c'(S') - m'(S')}{c'(S')} \\ &= \frac{(|S'| - 1) - (|p'(S')| - 1)}{|S'| - 1} \\ Precisão &= \frac{|S'| - |p'(S')|}{|S'| - 1} \end{aligned} \tag{3-2}$$

A segunda métrica, B-CUBED, se concentra mais nas menções. Ela foi definida em [115] e tenta resolver alguns problemas que aparecem com o MUC. O autor apresenta duas desvantagens da primeira métrica:

- O algoritmo ignora agrupamentos com uma única entidade, já que esses agrupamentos não possuem ligaçoes e essas então não podem ser avaliadas;
- Todos os tipos de erros são iguais. O autor argumenta que alguns erros deveriam ser mais punidos dos que outros.

Para contornar esses problemas, a ideia da métrica é identificar a presença ou ausência das entidades em relação a outras entidades produzidas pelas classes de equivalência. Assim, a precisão e o recall de cada entidade são definidos de acordo com as equações em 3.5. Os valores finais são somados como definido em 3-4, onde o parâmetro w_i é um peso dado para cada entidade. Para a CoNLL este valor é sempre 1.

$$\text{Precision}_i = \frac{\text{número de elementos corretos na saída contendo a entidade i}}{\text{número de elementos na saída contendo a entidade i}} \tag{3-3}$$

$$\begin{aligned} Recall final &= \sum_{i=1}^{N} w_i * Recall_i \\ Precisão final &= \sum_{i=1}^{N} w_i * Precision_i \end{aligned} \tag{3-4}$$

A terceira métrica, Constrined Entity Aligned F-measure (CEAF), aborda o problema das entidades e foi definida em [119]. Existem duas variações, dependendo do tipo de similaridade utilizada. A $CEAF_e$ utiliza a similaridade das entidades e a $CEAF_m$ utiliza a similaridade de menções. Na CoNLL é utilizada a $CEAF_e$. O algoritmo da métrica consiste em alinhar entidade da resposta com, no máximo, uma entidade na anotação, tentando encontrar o melhor mapeamento um-para-um utilizando uma métrica de similaridade. Este é um problema de bi-partição máxima e pode ser resolvido utilizando o algoritmo de Kuhn-Munkres. De forma simplificada, o recall e a precisão são definidos pela equação 3-5, onde $\phi(R, S)$ é uma medida de similaridade entre um conjunto de menções na resposta R e o conjunto de menções na anotação S.

$$\begin{aligned} Recall &= \frac{\phi(R,S)}{\text{número de menções em R}} \\ Precisão &= \frac{\phi(R,S)}{\text{número de menções em S}} \end{aligned} \tag{3-5}$$

Uma quarta métrica que é citada por [7] e [8] é a BiLateral Assessment of Noun-Phrase Coreference (BLANC) [116]. Apesar de estar incluída na implementação do avaliador da CoNLL e ser reportado a pontuação dos participantes no detalhamento dos resultados das duas edições da competição, ela não faz parte da pontuação final que determina o vencedor. Essa métrica é uma variação do índice Rand [120] feita especificamente para a tarefa de resolução de correferência. A motivação da criação dessa quarta métrica é tentar compensar os erros das outras três métricas que eram utilizadas anteriormente. A métrica BLANC é baseada em dois tipos de decisão que compara as soluções anotadas ($GOLD$) daquelas feitas por um sistema (SYS). As decisões são:

– Decisão de correferência, que é feita por um sistema de decisão, que pode ser:

  Uma ligação chamada de correferente entre duas menções correferentes (c); ou

		SYS		Total
		Correferente	Não correferente	
GOLD	Correferente	rc	wn	$rc + wn$
	Não Correferente	wc	rn	$wc + rn$
Total		$rc + wc$	$wn + rn$	L

Tabela 3.2: Matriz de confusão da métrica BLANC

	Correferente	Não correferente	
Precisão	$P_c = \frac{rc}{rc+wc}$	$P_n = \frac{rn}{rn+wn}$	$BLANC - P = \frac{P_c+P_n}{2}$
Recall	$R_c = \frac{rc}{rc+wn}$	$R_n = \frac{rn}{rn+wc}$	$BLANC - R = \frac{R_c+R_n}{2}$
F1	$F_c = \frac{2P_cR_c}{P_c+R_c}$	$F_n = \frac{2P_nR_n}{P_n+R_n}$	$BLANC = \frac{F_c+F_n}{2}$

Tabela 3.3: Definição da fórmula para BLANC

Uma ligação não correferente entre duas menções não correferente (n).

– Decisão de corretura, que é feita por um sistema avaliador, que pode ser:

Um link correto (r) é aquele em que a decisão de correferência é a mesma para GOLD e SYS, ou seja, o sistema está correto naquele link

Um link errado (w) é aquele em que há divergência entre GOLD e SYS, ou seja, o sistema está errado naquele link

Compondo-se os valores c, n, r e w é possível derivar uma matriz de confusão como a tabela 3.2, retirada de [116]. Nela, é mostrado quatro novas variáveis rc, wn, wc e rn que, quando somadas, devem compor todas os possíveis links entre duas menções de um determinado documento (L), sendo que $L = N(N-1)/2$, com N sendo o número total de menções.

A partir das variáveis definidas na tabela 3.2 são definidos os valores de precisão (P), recall (R) e o score F1 (F) para cada tipo de erro. Essas definições foram retiradas de [116] e estão na tabela 3.3. Nessa tabela é possível ver a definição da métrica BLANC bem como outras métricas intermediárias que podem ser calculadas para um conjunto de decisões de correferência.

Em [121] são descritos alguns problemas associados a essas métricas em um cenário mais realista. O autor propõe então uma nova métrica que resolveria os problemas da combinação das métricas atuais. Como o objetivo deste trabalho é fazer uma comparação com as soluções já existentes, manteremos as métricas originais.

Modelo	Ano	CoNLL Score
Joshi, et al. [14]	2019	79.6
Joshi, et al. [72]	2019	76.9
Kantor and Globerson. [122]	2019	76.6
Fei, et al. [45]	2019	73.8
Lee, et al. [39]	2018	73.0
Peters, et al. [13]	2018	70.4
Lee, et al. [38]	2017	68.8
Wiseman, et al. [64]	2016	64.2
Fernandes e Milidiú [19]	2012	63.4

Tabela 3.4: Estado da arte atual
Lista com trabalhos baseados em técnicas de aprendizado supervisionado ou semi-supervisionado

3.6 Estado da Arte

O estado da arte da tarefa de resolução de correferência foi modificado de maneira significativa nos últimos anos. Sua evolução é rastreada por alguns sites [1] e, no momento da publicação deste trabalho, os quatro melhores trabalhos foram feitos no último ano, o que significa uma melhora de mais de 10% na métrica final nos últimos 12 meses.

A tabela 3.4 mostra como se encontra o estado da arte atualmente. O trabalho de [45] é o único baseado em aprendizado por reforço. Todos os demais são sistemas baseados em algum tipo de estrutura latente. Todos os trabalhos, com exceção de [19], se baseiam em algum tipo de rede neural para resolver a correferência ponta a ponta.

[1] $http://nlpprogress.com/english/coreference_resolution.html$

4
Árvores Latentes de Antecedentes

Neste capítulo é detalhado uma solução para o problema de correferência chamado Árvores Latentes de Antecedentes. Em sua maioria são tratadas as formulações propostas por [19] e [20]. É descrito como são feitas as detecções de menções, a geração dos arcos candidatos e como são definidas as features básicas e induzidas. Por fim é detalhado o método de aprendizado do perceptron estruturado.

4.1
Descrição Geral

Uma forma de se resolver a correferência é utilizando uma estrutura latente que represente todo o documento. Essa abordagem foi utilizada com sucesso por [19] na CoNLL 2012, sendo o modelo vencedor da competição. Como indicado por [39], as árvores latentes podem ser reinterpretadas como um ranqueamento de trechos, que é a técnica utilizada nos modelos mais recentes.

Uma vantagem de se ter uma estrutura que representa todo o documento é poder resolver situações como a mostrada na figura 4.1 retirado de [20]. Nesta figura cada nó representa uma menção em um documento. Os arcos são ligações de correferência determinados por um sistema de classificação. As ligações com "+" são correferentes e as ligações com "−" são não correferentes. É possível notar um problema com essa solução porque a menção m_4 é correferente com m_3 e m_1, mas não é correferente com m_2. Porém a menção m_3 é correferente com m_2 mas não com m_1. Um sistema que tomasse decisões locais para cada menção teria que elaborar alguma heurística para avaliar essa situação. Em um sistema que contemple todo o documento é possível criar algum tipo de nota para cada arco e, considerando todos eles, escolher os melhores.

As árvores latentes foram descritas inicialmente por [65] e foram adaptadas por [19]. Uma diferença entre os modelos é que no trabalho de [65] são utilizados grafos não direcionados, enquanto que em [19] os grafos são direcionados. Segundo [19] a adoção da direção é mais apropriada no contexto de resolução de correferência já que indicam algum tipo de dependência entre as menções.

Essa alegação é condizente com a definição 3.6 de anáfora onde há uma relação de dependência semântica entre as menções. Porém, existem termos que não possuem uma dependência clara, por exemplo, quando há duas ocorrências de um mesmo nome próprio. Nestes casos onde a dependência não é clara

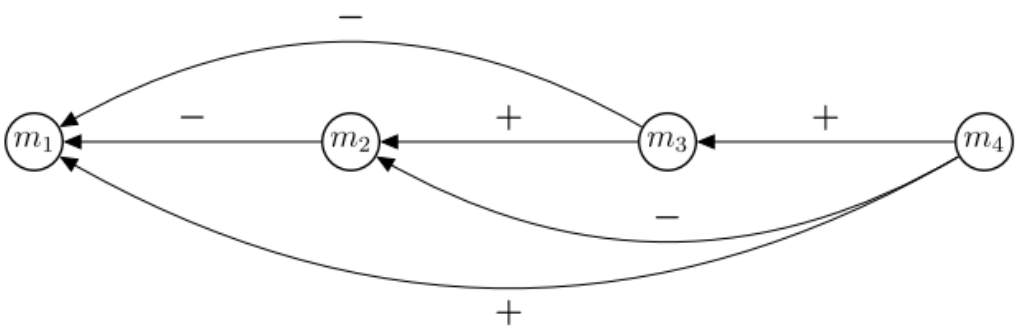


Figura 4.1: Exemplo mais complicado de relacionamento entre menções **Cada nó representa uma menção e cada ligação é uma relação de correferência. Ligações marcadas com "+" são ligações correferentes e ligações com "−" são ligações não correferentes.**

haverá mais de uma versão de árvore latente para representar o documento. De acordo com [19] isso não é um problema, já que a árvore é apenas latente e não há uma preocupação direta com a semântica representada por ela.

O termo *latente* é utilizado, segundo [19], porque não existe uma anotação no conjunto de teste de como essas árvores são organizadas. Elas existem apenas de forma indireta entre a decodificação dos arcos em vetores de características e o agrupamento das menções.

Um exemplo de como seria a representação de uma dessas árvores pode ser vista em [37]. Se for considerado o trecho 3, onde as marcações em colchetes indicam as menções, as letras a qual entidade ela se refere e a numeração a ordem de ocorrência no texto, uma possível árvore latente pode ser representada como na figura 4.2.

$$[DrugEmporiumInc.]_{a1}\ said\ [GaryWilber]_{b1}\ was\ named\ CEO\ of\ [thisdrugstorechain]_{a2}.\ [He]_{b2}\ succeeds\ his\ father,\ Philip\ T.\ Wilber,\ who\ founded\ [thecompany]_{a3}\ and\ remains\ chairman.\ Robert\ E.\ Lyons\ III,\ who\ headed\ the\ [company]_{a4}\ 's\ Philadelphia\ region,\ was\ appointed\ president\ and\ chief\ operating\ officer,\ succeeding\ [GaryWilber]_{b3} \quad (3)$$

A construção desta árvore pode ser resumida nos seguintes passos:

1. **Detecção de menção**: construção dos nós de um grafo para cada menção;

2. **Geração de pares candidatos**: adição de arcos direcionados a partir de uma menção para cada menção anterior, obedecendo um conjunto de regras;

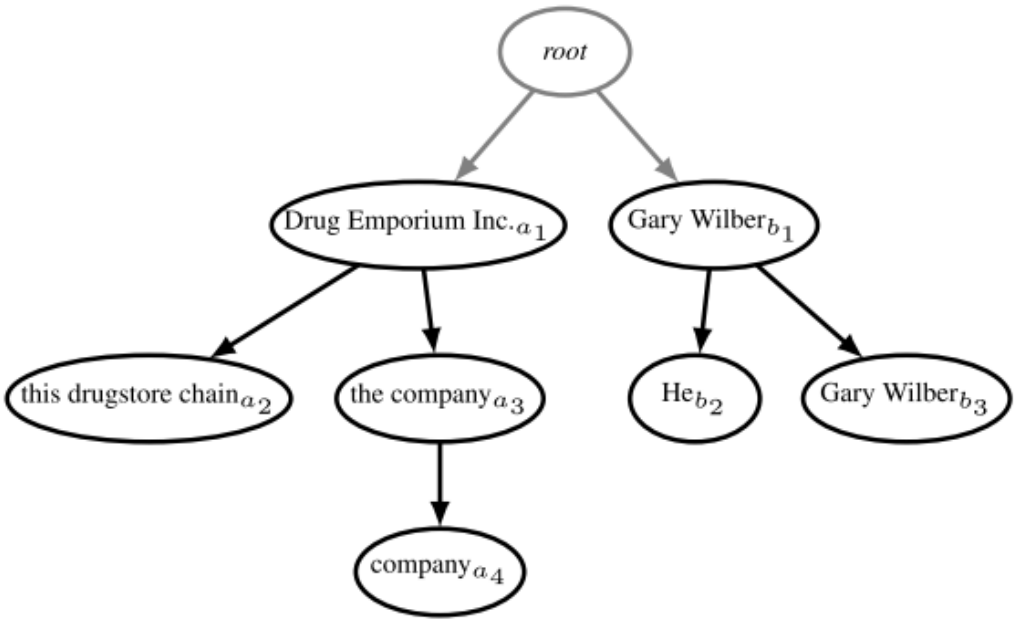


Figura 4.2: Representação de menções em uma árvore
Cada nó é uma menção. As setas partem de uma raiz artificial e seguem para as menções. Cada ramificação a partir da raiz artificial é uma entidade no mundo real

3. **Características básicas**: definição de um conjunto de valores (features) que caracterizam um arco de forma que possa se identificar quais arcos são correferentes (consistentes com a anotação do dataset);

4. **Indução de features**: geração de features induzidas a partir das features básicas de modo a tentar capturar o contexto onde os valores aparecerem;

5. **Aprendizado da árvore de antecedentes**: aprendizado de como extrair as árvores que conectam as menções correferentes a partir do grafo de menções aplicando um algoritmo de perceptron estruturado com margem larga.

No contexto de árvores latentes, os termos arco entre menções e par de menções são equivalentes. Apesar de par de menção ser utilizado na literatura em outros tipos de soluções que não envolvem estruturas latentes, em se tratando de árvores de antecedentes todos os arcos são pares de menções. Mesmo o arco entre a raiz artificial e as demais menções é considerado um par de menções, em que o antecedente é uma menção artificial. Neste trabalho, exceto se explicitado o contrário, os termos podem ser entendidos como sinônimos.

4.2 Detecção de Menções

A detecção de menção em [19] é feita utilizando uma série de regras derivadas inicialmente por [123]. As regras são:

- frases nominais presentes já fornecidas pela anotação do OntoNotes;
- pronomes, mesmo quando ocorrem em frases nominais;
- entidades nomeadas nas categorias Pessoa, Organização e Entidade Geopolítica, mesmo ocorrendo em frases nominais;
- marcações possessivas para manter coerência com as anotações da CoNLL 2012.

Esse conjunto de regras é mais ou menos o mesmo utilizado em outros trabalhos como em [37] e [20]. Em [35] são adicionadas regras para remover algumas ocorrências da palavra *"it"* que não estão marcadas no dataset original. Essas ocorrências são chamadas de *"pleonastic it"*, como em *"It is possible that"*. Outras regras utilizadas estão descritas no documento da tarefa compartilhada da CoNLL 2012 [8].

Em [20] o detector utilizado é uma reimplementação do detector implementado em [124] que é uma remodelagem de [35].

4.3 Geração de Pares Candidatos

A geração de pares candidatos é feita ligando-se cada menção a todos os elementos de um sub-conjunto das menções que ocorrem antes dela. Formalmente, se for considerado um documento d com um conjunto de menções $\mathcal{M}_d$, o conjunto $\mathcal{A}_i$ dos pares candidatos que contém a menção j como anáfora é definido pela equação 4-1. A função F é um filtro aplicado sobre o conjunto de forma a diminuir o número de possíveis pares gerados e, consequentemente, a complexidade computacional. Em [19], a função F é implementada seguindo as regras de [125]. Em [20] são utilizadas todas as anáforas.

$$\mathcal{A}_j := F(\{(ana_j, ante_i)\} \| j > i; ana_j \in \mathcal{M}_d; ante_i \in \mathcal{M}_d \qquad (4\text{-}1)$$

Se forem utilizados os conjuntos $\mathcal{A}_i$ com $i > 0$, ao final do processo de seleção dos pares candidatos será preciso criar um caso especial para quando a menção não possui nenhum antecedente. Além disso, o resultado deste processo será uma floresta com cada árvore representando um agrupamento. Para simplificar os dois processos, [19] utiliza uma menção artificial m_0 que representa a raiz da árvore. Desta forma, todas as menções possuirão exatamente

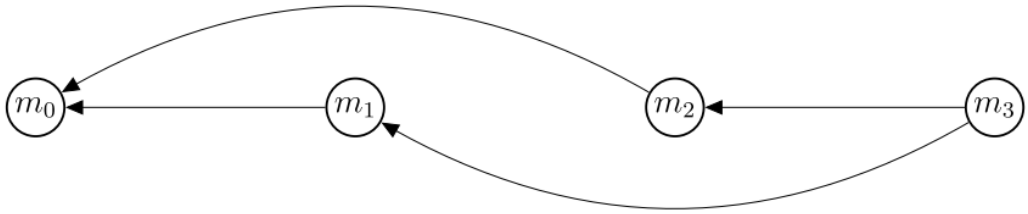


Figura 4.3: Exemplo de pares candidatos

um antecedente. Menções cujo antecedente seja m_0 serão consideradas a raiz da sub-árvore e, portanto, a primeira menção de um novo grupo. Sendo assim, a definição de todos os possíveis pares candidatos é dada pela equação 4-2

$$\mathcal{A} = \bigcup_{j=1}^{\mathcal{M}_d} \mathcal{A}_j \cup \{(ana_j, m_0)\} \tag{4-2}$$

Na estrutura de árvore latentes apenas os links que correspondem a uma correferência são representados. Uma representação de uma árvore latente com seus pares candidatos pode ser vista na figura 4.3, retirado de [20]. O exemplo é de pares candidatos pelo fato de a menção m_3 possuir dois antecedentes.

Na implementação da função de filtro F, [19] escolhe apenas as regras mais gerais de [125] de forma a poder aplicá-las em um mecanismo multilingual. Ao final, qualquer par de menções que atendesse a pelo menos uma das seguintes regras era escolhido como antecedente:

1. **Distância**: $j - i >= k$, sendo k um hiperparâmetro do modelo

2. **Tipo de Entidade Nomeada**: ambas as menções são entidades nomeadas do mesmo tipo, e mais:

   Pessoas: a palavra principal de uma menção é parte da outra, como em *Obama* e *Barack Obama*

   Organizações: a palavra principal de uma menção está contida na outra, ou é um acrônimo da outra

3. **Mesma palavra principal**: a palavra principal das duas menções é a mesma

4. **Discurso raso**: esta regra é um compêndio de algumas regras propostas por [125] baseadas na menção e no orador. De forma simplificada, o gênero e o grau dos oradores e das menções devem seguir um certo padrão.

5. **Pronomes**: a anáfora é um pronome e o antecedente tem o mesmo gênero, grau e animação [126]

6. **Pronomes e Entidades Nomeada**s: a anáfora é um pronome e o antecedente é um pronome compatível ou entidade nomeada

As informações utilizadas para gerar essas regras estão disponíveis nos dados da competição. A maioria está presente no próprio dataset do OntoNotes, além de uma tabela de gênero e número gerada por [113].

4.4 Características Básicas

De acordo com o relatório de resultados da CoNLL 2012 [8], muitos modelos utilizaram conjuntos bem grandes de características (features) dos arcos e das menções. Muitas dessas features são semelhantes e podem ser agrupadas em dois grandes grupos:

– **Características das menções**: informações específicas de cada menção.
– **Características dos arcos**: informações que dependem das duas menções para serem calculadas.

Essa divisão é explicitada por [20], mas ela pode ser encontrada também nos trabalhos de [108] e [127], agrupadas de maneiras diferentes.

Em [19] são utilizadas um total de 71 features básicas, muitas com características binárias, tais como se as o texto das menções é o mesmo ou se uma das menções é um nome próprio. Também são encontradas algumas características bem específicas, como a concatenação dos papéis semânticos das menções para o mesmo predicado, se ambas estiverem na mesma frase. Uma das características utilizadas é a classificação de arco correferente proposto por [123].

4.5 Indução de Features

Em [19] é utilizado uma técnica chamada de indução de features guiada por entropia (Entropy-guided feature induction - EFI). Essa técnica também foi utilizada em [128], [21] e [129] sempre com o objetivo de aumentar o número de features disponíveis para o modelo.

O funcionamento básico é gerar combinações de features básicas através de uma série de padrões (templates). Os templates são gerados a partir dos próprios dados utilizando-se uma árvore de decisão [46] gerada de forma a se obter o maior ganho de informação. Desta forma, os templates acabam sendo mais informativos dos que as features básicas.

e		$\Psi(e)$						$c(e)$
ante	ana	ante-head	ante-pos	ana-head	ana-pos	sameNE	dist	
1	2	Vicente	NOUN	him	PRON	N	0	1
1	3	Vicente	NOUN	David	NOUN	Y	1	0
2	3	him	PRON	David	NOUN	N	0	0
2	4	him	PRON	goalkeeper	NOUN	N	1	0
4	6	goalkeeper	NOUN	Gea	NOUN	Y	1	1
5	11	Manchester	NOUN	Trafford	NOUN	Y	5	1

Tabela 4.1: Exemplo de vetor de features

No contexto das árvores latentes, a geração desses templates segue os seguintes passos:

1. Detectar as menções
2. Gerar os pares candidatos
3. Para cada par de menções, gerar um vetor com as features básicas e mais um indicador de correferência, com 1 se o par está no mesmo cluster ou 0 caso contrário
4. Utilizar o novo conjunto de dados para se treinar uma Árvore de Decisão
5. Para cada caminho entre a raiz e um nó intermediário, criar um template com as características correspondente a cada nó percorrido
6. Remover templates duplicados

Para exemplificar essa geração, a tabela 4.1 ilustra a utilização de cinco features utilizadas por [19], aplicadas ao trecho 1. Essa tabela dá origem à árvore de decisão na figura 4.4. Para a geração dos templates, os valores específicos de cada arco e folha não são importantes, apenas o caminho entre os nós. Assim, considera-se apenas um esqueleto da árvore, como na figura 4.5. Esse esqueleto em particular dá origem a quatro templates, como listado ao lado da figura.

Essa forma de geração pode criar um número muito grande de templates. Para evitar isso, [19] limita o tamanho máximo da árvore e também o número máximo de nós que podem ter em um template.

Esses templates darão origem a features induzidas que refletem arcos que são correferentes no contexto do dataset. Cada template será aplicado a cada arco correferente e a composição dos valores deste arco darão origem a uma feature binária, caso já não exista. Para exemplificar, o template **dist**

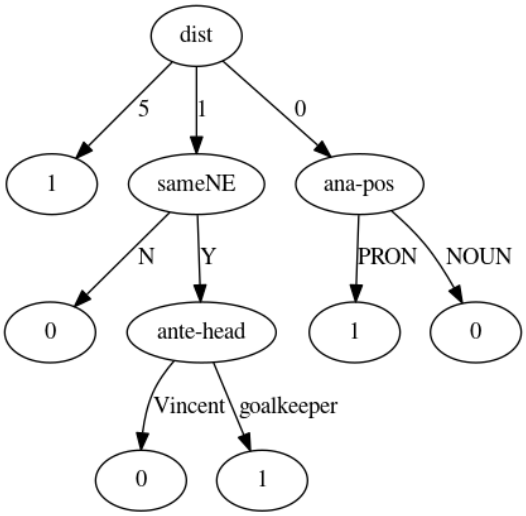
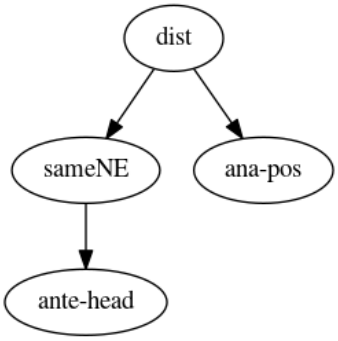


Figura 4.4: Uma árvore de decisão



Templates Gerados

dist
dist & **ana-pos**
dist & **sameNE**
dist & **sameNE** & **ante-head**

Figura 4.5: Esqueleto da árvore

& **sameNE** & **ante-head**, quando aplicado sobre o arco da primeira linha da tabela 4.1, dará origem a uma feature que é gerado segundo a função na equação 4-3. Nessa equação, **x** é o conjunto de todas as menções do documento, **e** é um par de menções e ϕ_m é função geradora da feature m.

$$\phi_m(\mathbf{x}, e) = \begin{cases} 1, & \text{se dist} = 0, \text{ sameNE} = \text{N e ante-head=Vicente} \\ 0, & \text{caso contrário} \end{cases} \quad (4\text{-}3)$$

As features geradas conseguirão capturar um contexto que não foi utilizado pela árvore de decisão, já que os valores utilizados são gerados diretamente a partir dos dados do dataset de treino. Quando todas as features forem geradas, será criado então um vetor de características associado a um par de menções que tem o formato $\mathbf{\Phi}(\mathbf{x}, e) = (\phi_1(\mathbf{x}, e), \phi_2(\mathbf{x}, e), ..., \phi_M(\mathbf{x}, e))$, sendo M o conjunto de todas as features geradas.

O vetor $\phi(\mathbf{x}, e)$ indica quais features derivadas estão ativas, ou seja, que ocorreram alguma vez com um par de menções que era correferente. Apesar de o número total de features ser muito alto, cada par terá pouca informação

ativa. Na verdade, no máximo uma feature por template pode estar ativa.

Para a indução de features, as palavras são utilizadas como um índice em um vocabulário, como no BoW. Valores numéricos são interpretados como categorias.

Uma outra forma de gerar features derivadas é apresentada em [60] e utilizada por [20] chamada de sistema SURFACE, já que explora apenas características superficiais das menções. Nele as features derivadas são geradas apenas pela junção das features básicas que ocorrem na anáfora e no antecedente. Por exemplo, considerando a feature de POS da menção, ela daria origem inicialmente a três features derivadas:

- POS da anáfora, por exemplo NOM
- POS do antecedente, por exemplo PRON
- concatenação da POS da anáfora e do antecedente, NOM+PRON

A partir essas três features, cada uma é concatenada com uma feature especial chamada *fine-grain type*. Essa feature classifica as menções em tipos mais específicos, seguindo algumas regras:

- Nomes próprios recebem sempre o mesmo valor *ProperName*;
- Substantivos comuns são separados entre substantivos definidos e não definidos;
- Pronomes pessoais são mapeados em sua forma canônica, por exemplo o pronome; *him* é mapeado como *he*;
- Pronomes demonstrativos recebem o valor único de *Demonstrative*;
- Todas os demais casos caem na categoria de *Miscellaneous*.

Ao final tem-se então 6 novas features que funcionam como os templates de [19]. Cada template dará origem a diversas features binárias, de acordo com os dados do conjunto de treino. Como esses sistemas utilizam muito menos features básicas e todas são categóricas, o número final de combinações é alto, porém limitado.

Entre os métodos EFI e SURFACE existem mais semelhanças do que diferenças. Ambos partem de um conjunto básico de features que representam quase as mesmas informações, em combinações diferentes. Ambos também geram potencialmente um número exponencial de combinações, mas se limitam a utilizar apenas aquelas que aparecem no dataset de treino. Uma diferença entre eles é que o EFI utiliza uma árvore de decisão limitada e a entropia para gerar um número pequeno de templates, enquanto o SURFACE limita o número de possíveis valores que cada característica pode assumir.

A principal diferença entre os modelos está no tipo de recurso computacional a ser utilizado. O EFI exige mais processamento e o SURFACE exige mais memória. Se por um lado o EFI precisa que seja feito inicialmente uma árvore de decisão no pré-processamento e cada template precisa ser testado em cada arco, por outro cada arco gera apenas um bit por template de informação. Já o SURFACE precisa de pouco processamento pois, além de não exigir um pré-processamento, a informação de cada menção pode ser reutilizada e para um total de N características das menções são geradas $3N$ características apenas por concatenação; porém, cada arco gerará também $3N$ bits de informação o que pode se tornar muito caro no caso de representações contextuais que trabalham com a ordem de 1000 features por menção.

Pelos resultados disponibilizados por [20] nos quais ele compara a sua implementação de árvores latentes com a de [19], é possível ver que os resultados finais são bem semelhantes, ficando a diferença final menor do que 1 ponto percentual no score final da CoNLL.

4.6 Aprendizado da árvore de antecedentes

Segundo [20], a resolução de correferência é, em termos de aprendizado de máquina, um problema de previsão. Ou seja, dado um documento, a tarefa é prever como as menções se encadeiam. Considerando que $\mathcal{X}$ é o espaço de entrada e $\mathcal{Y}$ é o espaço de saída, o objetivo é conseguir uma função f que satisfaça a relação:

$$f : \mathcal{X} \rightarrow \mathcal{Y} \tag{4-4}$$

Sendo mais concreto, $\mathcal{X}$ é o conjunto das menções dos documentos e $\mathcal{Y}$ são os documentos com as marcações da correferência. Quando se utiliza uma estrutura latente como as árvores de antecedentes, o espaço $\mathcal{Y}$ é então fatorado entre um espaço latente $\mathcal{H}$ e um espaço observado $\mathcal{Z}$. Assim, no caso de árvores latentes se procura uma função que satisfaça a relação

$$f : \mathcal{X} \rightarrow \mathcal{H} \times \mathcal{Z} \tag{4-5}$$

Esses novos espaços se concretizam com $\mathcal{H}$ contendo as estruturas latentes e $\mathcal{Z}$ contendo as relações de correferência. Com essa composição é possível então derivar uma função composta da seguinte forma 4-6, com $\mathbf{x}$ sendo um conjunto de menções.

$$F(\mathbf{x}) \equiv F_z(F_h(\mathbf{x})) \tag{4-6}$$

A função F_h deve ser capaz então de receber um conjunto de menções e retornar uma estrutura latente organizada de tal forma que a função

F_z a receba e possa gerar as ligações entre as menções e seus respectivos agrupamentos. Se a função F_h retornar uma árvore de antecedentes de um documento com uma raiz artificial e as ligações das menções correspondendo às correferências entre uma anáfora e seu antecedente, a função F_z se resume em realizar dois passos:

1. Remover a raiz artificial e todos os arcos saindo dela, gerando uma floresta

2. Para cada árvore da floresta, exibir os seus nós como um agrupamento de menções

Em [19], a saída da função F_h é chamada de árvore do documento, a fim de diferenciá-la das sub-árvores que representam os agrupamentos. Também é definido um preditor da árvore do documento $F_h(\mathbf{x})$ como sendo:

$$F_h(\mathbf{x}) = \max_{h \in \mathcal{H}(\mathbf{x})} \langle \mathbf{w}, \mathbf{\Phi}(\mathbf{x}, \mathbf{h}) \rangle \tag{4-7}$$

Onde H(x) é o subconjunto das árvores latentes em H que podem ser derivadas pelo conjunto de menções $\mathbf{x}$; $\mathbf{w}$ é um vetor de pesos a ser aprendido e $\mathbf{\Phi}(\mathbf{x}, \mathbf{h})$ é o vetor de features gerado a partir de $\mathbf{x}$ e uma árvore $\mathbf{h}$, de forma que:

$$\mathbf{\Phi}(\mathbf{x}, \mathbf{h}) = \sum_{e \in \mathbf{h}} \mathbf{\Phi}(\mathbf{x}, e) \tag{4-8}$$

Nessa expressão, a função $\mathbf{\Phi}(\mathbf{x}, e)$ é a função que concatena as features induzidas para um par de menções e.

A partir da definição da função F_h, [19] propõem criar duas estruturas. Uma primeira estrutura $\hat{\mathbf{h}}$ que será aprendida usando uma versão de margem larga da função F_h e uma outra estrutura $\tilde{\mathbf{h}}$ que é a função F_h aplicada a um subconjunto $H(\mathbf{x}, \mathbf{y}) \subseteq H(\mathbf{x})$ que contém apenas estruturas que seguem o agrupamento correto $\mathbf{y}$. Esse agrupamento é a anotação do dataset.

O conjunto $H(\mathbf{x}, \mathbf{y})$ pode ser gerado da seguinte forma: seja $G(\mathbf{x})$ o grafo com todos os pares candidatos gerados para $\mathbf{x}$; deriva-se então uma versão restrita $G(\mathbf{x}, \mathbf{y})$ deste grafo removendo-se os arcos entre menções que estão em grupos diferentes de acordo com y. Os arcos artificiais serão apenas entre a raiz e a primeira menção de cada grupo. Uma ressalva feita por [19] em relação ao grafo $G(\mathbf{x}, \mathbf{y})$ é que ele pode acabar desconectado, se o conjunto x não contiver todas as menções presentes na anotação. No entanto, como o detector de menções utilizado no trabalho tem um recall de cerca de 90%, esses casos são raros. Formalmente, a estrutura $\tilde{\mathbf{h}}$ é definida como:

$$\tilde{\mathbf{h}} = F_h(\mathbf{x}, \mathbf{y}) = \arg \max_{h \in \mathcal{H}(\mathbf{x}, \mathbf{y})} \langle \mathbf{w_t}, \mathbf{\Phi}(\mathbf{x}, \mathbf{h}) \rangle \tag{4-9}$$

É importante notar que o parâmetro $\mathbf{w}$ foi renomeado para $\mathbf{w_t}$, já que esse vetor será aprendido de maneira iterativa e a cada passo ele será modificado.

A estrutura $\hat{\mathbf{h}}$ é gerada utilizando uma versão com uma margem larga [130], [128]. O racional desta versão é, durante o treino, forçar a função F_h a errar mais do que naturalmente o faria. Se o perceptron for considerado como um algoritmo que acha uma forma de separar exemplos em dois grupos em algum espaço, essa versão garantirá que essa separação será feita de maneira a deixar uma "margem larga" entre os exemplos e a fronteira de separação. O efeito prático é que durante a etapa de validação e teste, onde os exemplos são diferentes do treino, há uma maior chance de a fronteira escolhida acertas os novos exemplos. Formalmente, a estrutura $\hat{\mathbf{h}}$ é definida como

$$\hat{\mathbf{h}} = F_h(\mathbf{x}) = \arg\max_{h \in \mathcal{H}(\mathbf{x})} \langle \mathbf{w_t}, \mathbf{\Phi}(\mathbf{x}, \mathbf{h}) \rangle + C \cdot l_r(\mathbf{h}, \tilde{\mathbf{h}}) \tag{4-10}$$

Nessa fórmula, C é um hiperparâmetro do modelo que deve ser ajustado durante o treinamento e $l_r(\mathbf{h}, \tilde{\mathbf{h}})$ é uma função de perda que conta quantos arcos estão diferentes entre uma árvore $\mathbf{h}$ e a atual referência de árvore correta $\tilde{\mathbf{h}}$, ou seja:

$$l_r(\mathbf{h}, \tilde{\mathbf{h}}) = \sum_{j=1}^{N} \mathbf{1}_{[\mathbf{h}(j) \neq \tilde{\mathbf{h}}(j)]} \cdot (1 + r \cdot \mathbf{1}_{[\mathbf{h}(j) \neq raiz]}) \tag{4-11}$$

Nessa equação, a função $\mathbf{1}_{[q]}$ é definida de forma que ela vale 1 se o predicado q é verdadeiro e 0 caso contrário; $\mathbf{h}$(j) é o antecedente da menção j na árvore $\mathbf{h}$; e o valor r é um hiperparâmetro do modelo e é chamado de *valor de perda da raiz*. Esse valor também deve ser calibrado em um conjunto de testes. A lógica para a existência desse parâmetro é penalizar mais quando o modelo selecionar a raiz artificial como antecedente, diminuindo o número total de grupos gerados.

A atualização do vetor de pesos $\mathbf{w_t}$ é feito de acordo com a lei de formação:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{\Phi}(\mathbf{x}, \tilde{\mathbf{h}}) - \mathbf{\Phi}(\mathbf{x}, \hat{\mathbf{h}}) \tag{4-12}$$

Essa forma de atualizar os valores de w é uma forma de representar matematicamente a ideia de se atualizar a cada passo a "distância" entre uma árvore $\tilde{\mathbf{h}}$ coerente com a anotação e a melhor árvore até o momento $\hat{\mathbf{h}}$.

O algoritmo completo do perceptron estruturado médio com margem larga proposto por [19] está representado pelo código do algoritmo 1. Ele é similar ao algoritmo proposto do perceptron simples proposto por [131]. Neste algoritmo é utilizada uma versão com média seguindo a proposta de [68] que, segundo os autores, produz um modelo mais robusto.

É importante notar que a margem larga *aumenta* a pontuação dos arcos

Algoritmo 1: Perceptron estruturado com margem larga

$\mathbf{w}_0 \leftarrow 0$;
t $\leftarrow 0$;
enquanto *não convergiu* **faça**
  para $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ **faça**
    $\tilde{\mathbf{h}} \leftarrow \arg\max_{h \in \mathcal{H}(\mathbf{x},\mathbf{y})} \langle \mathbf{w_t}, \mathbf{\Phi}(\mathbf{x}, \mathbf{h}) \rangle$;
    $\hat{\mathbf{h}} \leftarrow \arg\max_{h \in \mathcal{H}(\mathbf{x})} \langle \mathbf{w_t}, \mathbf{\Phi}(\mathbf{x}, \mathbf{h}) \rangle + C \cdot l_r(\mathbf{h}, \tilde{\mathbf{h}})$;
    $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{\Phi}(\mathbf{x}, \tilde{\mathbf{h}}) - \mathbf{\Phi}(\mathbf{x}, \hat{\mathbf{h}})$;
    $t \leftarrow t + 1$;
$\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \mathbf{w}_i$

errados, fazendo com que o *argmax* na função F_h tenha uma chance maior de selecionar uma estrutura h com mais arcos errados. Esse efeito é proposital, jã que fará com que o perceptron estruturado atualize os pesos $\mathbf{w_t}$ de forma que, na próxima iteração, esses arcos errados estejam mais afastados da fronteira de separação.

A margem larga é utilizada em outros trabalhos, como em [33], [59] e [20], sendo que algumas vezes ela é mencionada como um custo aumentado (cost-augmented).

O problema de se encontrar a melhor árvore (argmax) é resolvido por [19] utilizando-se o algoritmo de Chu-Liu-Edmonds [132] que é um algoritmo para se resolver arborescência mínima. Mas, segundo [20] no caso específico de como essas árvores latentes foram criadas é possível resolver utilizando-se um simples algoritmo guloso que escolhe os arcos com a maior pontuação.

5
Representação Contextual de Palavras

Uma etapa comum a muitos sistemas de NLP é como a informação escrita será lida. Apesar de cada possível símbolo em um texto digital já ter uma representação numérica, essa não é a única opção. Trabalhos que utilizam o texto como uma sequência de caracteres só conseguiram ter uma eficiência maior quando as redes convolucionais [9] estavam bem desenvolvidas. Um exemplo dessa aplicação é [133]. Essa dificuldade vem da necessidade de se precisar modelar toda a cadeia de símbolos, o que não é um problema simples.

Este capítulo faz um apanhado geral das técnicas mais utilizadas para se representar um texto em aprendizado de máquinas. Apesar de o enfoque ser dado na resolução de correferência, essas técnicas são genéricas para qualquer tarefa de NLP.

5.1
Bag of Words

Uma abordagem mais simples é simplesmente fazer uma correspondência entre cada palavra e um número. Uma forma de se fazer isso é simplesmente criar um vocabulário contendo todas as possíveis palavras da língua e numerar as palavras. Cada palavra é associada ao seu índice no vocabulário e um texto pode ser representado por quantas vezes cada palavra ocorre.

Esse processo considera que um texto nada mais é um "saco de palavras" e normalmente é chamada de *Bag of Words* (BoW). Ela foi mencionada inicialmente em [74], no qual o autor reconhece que o modelo tem suas limitações, principalmente no âmbito das tarefas de interpretação de texto, mas que essa abordagem estatística é relativamente eficaz para se comparar dois textos.

Além do problema de não considerar a ordem das palavras, o BoW tem um problema de escalabilidade. O número possível de palavras em uma língua é muito grande. Mesmo se forem considerados apenas as palavras que ocorrem em um corpus, esse número pode crescer rapidamente. Por exemplo, o conjunto de treino do OntoNotes utilizado na CoNLL 2012 tem um total de 62364 "palavras", de acordo com a separação do próprio dataset.

Uma inspeção no conjunto do que seriam as palavras mostra que símbolos de pontuação, números, datas e outras marcações de texto estão sendo marcadas como palavras. Além disso, inflexões verbais, plurais, diferenças ortográfi-

Regra	Exemplo
SSES → SS	caresses → caress
IES → I	ponies → poni
SS → SS	caress → caress
S →	cats → cat

Tabela 5.1: Exemplo da primeira fase da extração de radical

cas e diferença de letras maiúsculas e minúsculas aumentam consideravelmente o tamanho da base sem adicionar muito sentido.

Existem diversos mecanismos linguísticos disponíveis para remediar esse problema. Lematização e extração de radical (*stemming*) são as duas mais utilizadas. Lematização é a técnica linguística para reduzir uma palavra ao seu *lema*, que é a forma como ela ocorre em um dicionário, ou seja, as palavras são reduzidas ao singular e masculino e os verbos são considerados no infinitivo. A extração de radicais é um pouco mais complicada, como indicado por [74]. Nem sempre é claro qual morfema, ou seja, parte da palavra, adiciona significado a ela e esse significado pode ser diferente para cada receptor da mensagem.

Um dos algoritmos de extração de radicais para a língua inglesa mais utilizados é o proposto por [134]. O algoritmo é bem elaborado e é composto por 5 etapas, aplicadas sequencialmente. Em casa etapa, convenções são aplicadas para se selecionar regras de transformação e são aplicadas ao radical mais longo. Uma parte da primeira fase pode ser visto na tabela 5.1, com alguns exemplos de como seriam aplicados na prática.

Uma variação comum do BoW é agrupar palavras e utilizar a junção destas como uma nova palavra. A ideia é chamada de *n-grams* e tenta capturar algum contexto em torno de cada palavra. Com essa técnica, ocorrências como *Federal Reserve* aparecem como uma única entidade e conseguem capturar a informação da entidade melhor do que apenas as ocorrências de *Federal* e *Reserve*. [77], [78] e [79] são alguns trabalhos que mostram como a utilização de n-grams é mais eficaz em diversas tarefas do que o BoW simples. O n-gram mais comum é utilizando sequências de três palavras. O Bag of Words pode ser considerado um caso de 1-gram.

5.2 Representação de Palavras

Uma primeira melhoria em relação à modelagem do BoW e n-grams é dimiuir a dimensionalidade da representação. [76] fala em uma "maldição de dimensionalidade" para se referir ao fato de que as duas técnicas mencionadas são representações em espaços de muitas dimensões. Se for considerado um

Palavra	Índice	Representação
about	1	0 0 0 1
back	2	0 0 1 0
Caesar	3	0 1 0 0
day	4	1 0 0 0

Tabela 5.2: Representação de palavras no BoW

vocabulário com 100.000 palavras, para se modelar uma sequência de 10 palavras serão necessários $100.000^{10} - 1 = 10^{50} - 1$ parâmetros.

O racional nessa redução é imaginar que cada n-gram é um vetor de tamanho 1 ortogonal a todos os demais e a intenção é projetar o mesmo conjunto de pontos em uma base diferente, com menos dimensões. O efeito prático é que cada palavra será representada por um vetor com menos informações. Esse efeito pode não parecer claro inicialmente, já que cada palavra no BoW, por exemplo, é associada a um escalar e, portanto, seria unidimensional. Porém esse número não é a informação completa, mas sim o índice em um vetor binário, como demonstrado na tabela 5.2.

Existem pelo menos três formas de se conseguir reduzir a dimensionalidade da representação de uma palavra: Latente Semantic Analysis (LSA), Word2Vec e Global Vectors (GloVe). Cada uma possui as suas próprias premissas e métodos de atingir um mesmo objetivo: capturar a semântica das palavras e representá-la em um vetor. [135] faz uma estudo comparativo entre esses três modelos e indica que há na literatura trabalhos com conclusões conflitantes, ora apontando um modelo melhor e ora outro. Ao final o estudo conclui que Word2Vec e Glove apresentam representações melhores do que LSA para inglês e árabe. Entre Word2Vec e Glove, os resultados parecem mostrar que há um ligeira vantagem do Word2Vec na tarefa testada, mas o estudo não é categórico em afirmar se um dos dois modelos é melhor que o outro.

5.2.1 LSA

O LSA foi descrito em [136] e consiste em achar uma aproximação de baixo posto para a matriz de termos e documentos. Após a transformação, cada nova característica é uma combinação linear de outras características que já existiam antes. Motivados mais pelo problema de indexar e procurar documentos através de palavras, os autores apresentam três justificativas para o seu método:

– Ao se indexar apenas as palavras que ocorrem em um documento, a

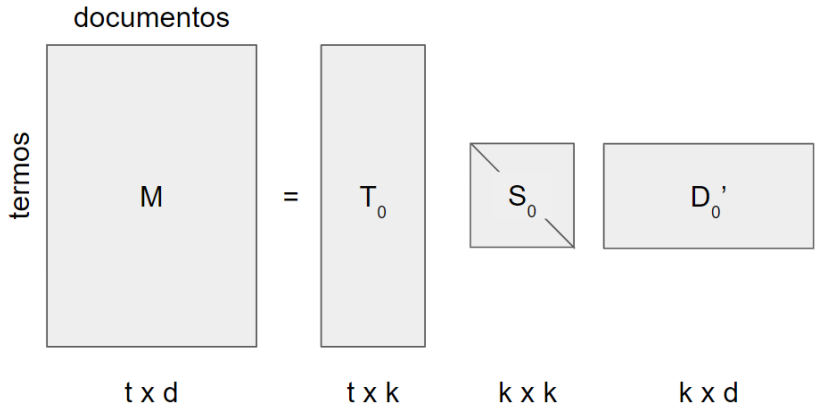


Figura 5.1: Decomposição por SVD

pesquisa conceitual pode acabar utilizando sinônimos ou outros termos parecidos que não aparecem em um documento específico.

– Os métodos existentes para se tratar automaticamente a polissemia, ou seja, o efeito de uma mesma palavra ter diversos sentidos, não eram adequados.

– A implementação corrente das tabelas de ocorrências dificultava uma busca por termos compostos. Isso porque os algoritmos atribuíam o mesmo peso tanto para a ocorrência de duas palavras que sempre apareciam juntas, quanto para a ocorrência de palavras que raramente estavam no mesmo contexto.

O método começa criando uma matriz M em que cada linha corresponde a um termo, cada coluna um documento e cada célula é a frequência do termo no documento. A partir desta matriz é aplicada a técnica de Decomposição Singular de Valor (Single Value Decomposition - SVD) [137] para se construir outras três matrizes T_0, S_0 e D_0 de tal forma que:

$$M = T_0 S_0 D_0' \tag{5-1}$$

Sendo que as matrizes T_0 e D_0 possuem colunas ortonormais e S_0 é uma matriz diagonal. Essa forma é chamada de *decomposição de valor singular de* M. As matrizes T_0 e D_0 são chamadas de *vetor singular esquerdo e direito* e S_0 é chamada de *valoressingulares*. Na figura 5.2 há uma ilustração de como as matrizes se organizam. As dimensões da matriz S_0 (k) é um hiperparâmetro do modelo. É ele quem vai controlar o quanto de informação será reduzida. Idealmente esse valor é grande o suficiente para que nenhuma informação se perca, mas pequeno o suficiente para que não se modele também o ruído.

Após a decomposição, cada palavra é representada por um vetor linha da matriz T_0 e cada documento é representado por um vetor coluna da matriz D'_0

5.2.2 Word2Vec

Uma melhoria a ser feita em relação ao LSA é a utilização de técnicas de machine learning para se aprender uma representação de palavras. Uma arquitetura que permite fazer isso é chamada de Word2Vec e foi introduzida em [85]. O objetivo é criar uma representação de tal maneira que a similaridade das palavras seja mantida. Em termos de aprendizado de máquina, ele tenta maximizar a acurácia de operações vetoriais no formato $vetor("King")$-$vetor("Man") + vetor("Woman") = vetor("Queen")$

Para se obter essa representação são apresentados dois modelos: *Continuous Bag-of-Words (CBOW)* e *Continuos Skip-gram*. No modelo CBOW é utilizada uma rede neural do tipo feedforward aplicada para modelos de linguagem, similar ao descrito em [76]. Na arquitetura da rede, a camada oculta não linear é removida e a camada de projeção é compartilhada entre todas as palavras, fazendo com que todas as palavras sejam projetadas no mesmo vetor médio. A analogia com o modelo BoW [74] é feita porque não é levado em consideração a ordem das palavras. Mesmo palavras que ocorrem após a palavra que está sendo prevista são utilizados. Segundo [85] o melhor resultado foi obtido utilizando-se quatro palavras antes e quatro palavras depois da palavra a ser prevista. O termo *contínuo* se refere ao fato de o modelo criar uma distribuição contínua de probabilidade para cada palavra e não um contador discreto como no BoW.

O segundo modelo é parecido com o BoW, mas com a diferença de se tentar classificar uma palavra a partir de uma outra que ocorra na mesma frase. Mais precisamente, a entrada do modelo é uma palavra e a saída é uma distribuição de probabilidades de todas palavras que ocorrem a uma certa distância da palavra de entrada. O nome *skip-gram* vem do fato de se fazer algo parecido com um 2-gram, mas desconsiderando-se algumas palavras entre os termos utilizados. Segundo [85], quanto maior a distância entre os termos, melhor era a representação das palavras, segundo a sua utilização em tarefas de NLP.

Na figura 5.2, retirada de [85], é possível ver uma comparação entre os dois modelos. No modelo CBOW, vê-se que a palavra a ser predita ($w(t)$) é uma função das palavras em torno dela. No modelo *skip-gram* tem-se uma relação inversa, com uma única palavra sendo utilizada para se tentar prever

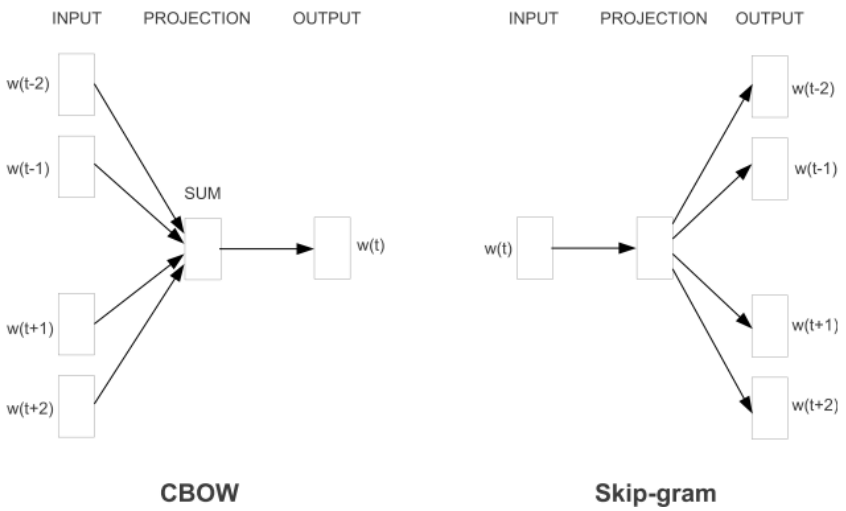


Figura 5.2: Comparação entre CBOW e Continuos Skip-Gram
No CBOW uma palavra é deduzida a partir de um contexto, no Skip-Gram tenta-se prever um contexto a partir de uma palavra

ocorrências próximas.

Ambos os modelos possuem uma série de hiperparâmetros que podem ser ajustados para se ter uma melhor performance. Em [135] são feitos diversos testes com variações possíveis de qual modelo utilizar e como selecionar os dados para treiná-los.

5.2.3 GloVe

Os modelos LSA e word2vec conseguem ter uma performance aceitável nas tarefas que se propõem a resolver, mas, segundo [84], ambos os modelos sofrem de problemas fora das tarefas propostas. O LSA por exemplo, consegue utilizar bem as estatísticas de todo um corpus para representar as suas palavras, mas em tarefas de analogia sua performance é ruim. Já o word2vec foi desenhado para ir bem nas tarefas de analogia, mas não consegue capturar bem as informações de diversos textos ao mesmo tempo, já que cada exemplo se resume a um contexto local.

Para contornar esses problemas, [84] propõem o *Global Vectors (GloVe)* que utiliza todo o corpus disponível para criar uma matriz de co-ocorrências de palavras. Além disso, segundo os autores, os vetores produzidos pelo GloVe são representações que apreendem uma parte do significado das palavras e não apenas estatísticas, já que o modelo vai bem tanto nas tarefas de analogia quanto nas de identificação de entidades nomeadas.

O argumento principal do trabalho é que alguma parte do significado pode ser extraído a partir das probabilidades de co-ocorrência global, ou seja, considerando-se todos os dados disponíveis, qual a probabilidade de uma

palavra ocorrer no contexto de uma outra. Mais especificamente, a relação entre duas palavras i e j, pode ser medida através da razão entre a probabilidade condicional de uma terceira palavra k ocorrer no contexto de cada uma delas. A razão das probabilidades $P(k|i)/P(k|j)$ deve ser alta para palavras k próximas à palavra i e deve ser baixa para palavras k próximas a j. Nos casos em que k não é próxima a nenhuma delas ou próxima às duas, a razão deve ficar próximo de 1.

No trabalho original é utilizado o seguinte exemplo: considerando-se as palavras *steam* e *ice*, espera-se que a razão $P(solid|steam)/P(solid|ice)$ seja grande, enquanto que $P(fashion|steam)/P(fashion|ice)$ seja próxima de 1, assim como $P(water|steam)/P(water|ice)$

Partindo então do racional que a razão das probabilidades depende de três palavras i, j e k, o objetivo do modelo é aprender uma função F tal que:

$$F(\mathbf{w}_i, \mathbf{w}_j, \tilde{\mathbf{w}}_k) = \frac{P(k|i)}{P(k|j)} \tag{5-2}$$

Nessa equação, $\mathbf{w}_i$ e $\mathbf{w}_j$ são representações vetoriais das palavras i e j, e $\tilde{\mathbf{w}}_k$ são vetores de palavras em contextos separados. Essa separação tem por objetivo construir uma segunda representação de cada palavra e a representação final de uma palavra x é dada por $\mathbf{w}_x + \tilde{\mathbf{w}}_x$, diminuindo as chances de se modelar um ruído (overfitting).

Em [84] é argumentado que, como espaços vetoriais são estruturas essencialmente lineares, pode-se restringir apenas a funções F que dependam da diferença $\mathbf{w}_i - \mathbf{w}_j$. Além disso, como em 5-2 os argumentos são vetores e o lado direito é um escalar, é sugerido trocar a entrada da função para o produto escalar dos termos, ou seja, $(\mathbf{w}_i - \mathbf{w}_j)'\tilde{\mathbf{w}}_k$. Por último, um ponto apresentado é que a escolha de qual palavra será representada por $\mathbf{w}$ e qual estará em $\tilde{\mathbf{w}}$ é arbitrária e, portanto a função F deve ser um homomorfismo entre os grupos $(\mathbb{R}, +)$ e $(\mathbb{R}_{>0}, \times)$. Ao aplicar essas restrições na função F, é definido a equação 5-3, no qual os termos b_k e $\tilde{b}_k$ são termos adicionados chamados de *bias* que foram adicionados para se manter a simetria entre w e $\tilde{w}$ e X_{ik} é a frequência da co-ocorrência das palavras i e k:

$$\mathbf{w}_i'\tilde{\mathbf{w}}_k + b_i + \tilde{b}_k = log(X_{ik}) \tag{5-3}$$

A equação 5-3 é uma simplificação da equação 5-2 e não é bem definida quando X_{ik} é 0, ou seja, não há co-ocorrência entre as duas palavras. Para contornar esse problema, os autores propõem uma função de ponderação $f(X_{ij})$ a ser aplicada em uma função de custo J de tal forma que $f(0) = 0$ e que seja construída de forma que não sejam exacerbadas nem co-ocorrências muito raras e nem muito frequentes. A função de custo J é então definida pela equação 5-4

e a função f é definida na equação 5-5, de α é um hiperparâmetro do modelo. Sendo definida desta forma, o termo do somatório em J desaparece quando X_{ij} tende a 0.

$$J = \sum_{i,j=1}^{V} f(X_{ij})\Big(\mathbf{w}_i'\tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - log(X_{ij})\Big)^2 \tag{5-4}$$

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & \text{se } x < x_{max} \\ 1 & \text{caso contrário} \end{cases} \tag{5-5}$$

Para se calcular os vetores das palavras a função de custo J deve ser minimizada para um dado corpus que possui um vocabulário V. No trabalho original foram utilizados cinco diferentes corpora com tamanhos entre 1 bilhão e 6 bilhões de dados e foi feita uma comparação direta com o modelo word2vec. São mostrados resultados que indicam que o GloVe não é só melhor que o word2vec na tarefa de analogia de palavras como também é mais estável ao longo das iterações.

Apesar desse resultado, em [135] é indicado que a comparação feita em [84] não é muito justa, uma vez que os parâmetros do word2vec não foram otimizados de forma adequada. Os autores então fazem um estudo mais elaborado e concluem que a diferença é, pelo menos, bem menor, com uma possível pequena vantagem do word2vec em relaçao ao GloVe. Eles também indicam que há na literatura diversos trabalhos com conclusões não concordantes e sugere que isso se deve ao fato de modelo word2vec possuir muitas formas de ser otimizado e, dependendo de como isso é feito, o resultado indica um ou outro modelo como melhor.

5.3 Representação de Contextos

Os modelos de representação de palavras conseguiram atingir novos patamares no estado da arte em várias tarefas de NLP, em especial naquelas de comparação de documentos, analogia e detecção de entidades nomeadas. Esses modelos conseguem representar muito bem palavras que possuem significado bem definidos e ter conceitos como hierarquia e gênero de alguma forma representados em seus valores. Por exemplo, em [84] o autor diz que analogias como *"king is to queen as man is to woman"* deveriam poder ser codificadas na equação vetorial $king - queen = man - woman$.

Apesar do avanço significativo dessas abordagens, ainda há o limitador implícito de que cada palavra ainda possui apenas uma única representação. Mesmo que essa representação tenha sido calculada a partir de algum contexto, palavras que ocorrem com muita frequência ou que podem assumir significados

muito diferentes dependendo da frase acabam sendo prejudicadas.

A maioria desses trabalhos possui alguma forma de criar um contexto local das palavras, anualizando de alguma forma quais as outras palavras que ocorrem próximas umas das outras a partir de um corpus grande o suficiente. Nesses algoritmos normalmente se tem um modelo que é auto-supervisionado. Ou seja, utilizando um corpo de texto sem nenhum tipo de anotação são definidas tarefas supervisionadas auxiliares que permitem o aprendizado de um modelo que será forçado criar a representação das palavras em algum momento.

Na tarefa de correferência o problema de se ter uma única representação por palavra acada sendo exacerbado, uma vez que muitas das menções são pronomes e, portanto, possuem uma frequência muito alta. Além disso, qual a entidade a que o pronome se refere pode só ficar claro se forem consideradas palavras no final da frase. As sentenças em 4 mostram um exemplo de como a última palavra pode mudar a correferência.

$$\begin{aligned} &\textit{I poured the water from the bottle into the } \textbf{\textit{cup}} \textit{ until } \textbf{\textit{it}} \textit{ was } \underline{\textit{full}} \\ &\textit{I poured the water from the } \textbf{\textit{bottle}} \textit{ into the cup until } \textbf{\textit{it}} \textit{ was } \underline{\textit{empty}} \end{aligned} \tag{4}$$

5.3.1 ELMo

Em 2018, [13] introduz uma forma de representar as palavras que leva em consideração o contexto em que cada palavra ocorre em um texto chamada Embeddings from Language Models (ELMo). Nesta representação, cada ocorrência de cada palavra possui um vetor associado, permitindo que características mais complexas como sintaxe e semântica fizessem parte da representação. Um exemplo é mostrado na figura 5.3. Nela são calculados os vizinhos da palavra play utilizando GloVe e ELMo. Para o embedding contextual o vizinho mais próximo foi calculado inicialmente a representação de todas as palavras do corpus SemCor 3.0 e depois foi calculada o vizinho mais próximo da palavra "play"que ocorria na sentença na primeira coluna. A segunda coluna mostra apenas as palavras da sentença em que o vizinho mais próximo foi encontrado.

A arquitetura apresentada pelo ELMo consiste em ter duas redes do tipo long shor-term memory (LSTM), uma em cada direção do texto, na qual cada uma tenta prever qual a palavra seguinte. Esse tipo de rede tem como característica fundamental conseguir resolver bem problemas em que se tem uma sequência de elementos e se quer determinar qual o próximo elemento. A LSTM é uma rede neural recorrente, ou seja, ela possui etapas recorrentes que, no caso do ELMo, é feita para cada palavra. A figura 5.4 ilustra a arquitetura

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Figura 5.3: Vizinhos mais próximos da palavra *"play"* utilizando GloVe e ELMo (biLM)

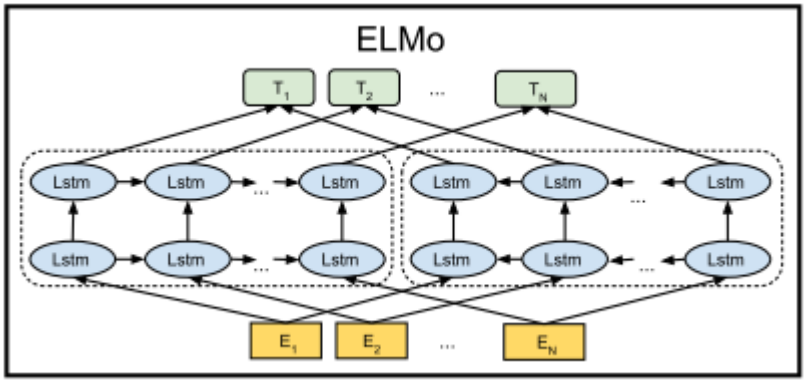


Figura 5.4: Arquitetura ELMo
Utiliza duas LSTM, uma em cada direção do texto. As representações são concatenações das saídas de cada etapa da rede recorrente.

geral desta técnica.

Nessa abordagem apresentada a técnica é utilizada apenas como uma parte de uma rede. Para cada problema é necessário fazer adaptações na arquitetura e fazer o treino do modelo a partir do início. Isso implica não só um custo computacional alto, como também a necessidade de se ter um corpus grande o suficiente para o problema específico de forma que o modelo tenha um bom desempenho.

5.3.2 BERT

Para contornar esse problema, o modelo Bidirectional Encoder Representations from Transformers (BERT) foi apresentado em 2019 por [12]. Nele a premissa é de que o modelo será treinado utilizando um corpus muito grande em uma tarefa auxiliar genérica. Através de transferência de conhecimento (Transfer Learning) esse modelo pode ser utilizado em outras tarefas sem precisar ser treinado do início, passando apenas por um ajuste fino (fine-tuning) que é adaptado para cada tarefa específica.

No BERT, ao invés de se ter uma concatenação das saídas das redes, cada representação é definida a partir de um conjunto de elementos chamados de transformers e que foram apresentados em [11]. A figura 5.6 mostra a arquitetura desse tipo de elemento. Os transformers, assim como as LSTM são utilizados quando se tem a informação apresentada de forma sequencial. A grande diferença entre esses dois elementos é que as LSTM são recorrentes, exigindo que cada passo seja computado individualmente, enquanto os transformers utilizam convoluções, que possibilita o cálculo em paralelo. Uma outra vantagem dos transformers em relação a LSTM é que eles utilizam mecanismos de atenção para aprender não só os pesos de cada entrada, mas também quantos elementos anteriores devem ser analisados.

A arquitetura do BERT pode ser visualizada na figura 5.7. Nessa figura, cada elemento representado em azul é um transformer. O modelo pode chegar a ficar muito grande, sendo reportada a utilização de 12 e 16 blocos no trabalho original, gerando 110M e 340M de parâmetros. Esse tamanho de rede exige uma quantidade muito grande de dados para que o modelo fique bom. Para o treino original são utilizados os corpora BooksCorpus (800M de palavras) e Wikipedia (2500M de palavras). Todo esse esforço acaba sendo compensado uma vez que o modelo pré-treinado pode ser utilizado e então o transfer learning possibilita outras tarefas terem ganho, independente de seus datasets específicos.

Para o treinamento do BERT o paper define duas tarefas auxiliares que serão treinadas ao mesmo tempo. A função de perda no final será definida como a soma das perdas de cada uma dessas tarefas. A entrada para as duas tarefas será uma sequência de palavras divididas em tokens. A primeira tarefa é chamada de Masked Language Model (MLM) e consiste em substituir alguns tokens da entrada por uma máscara e depois definir qual era o valor original do token. A segunda tarefa é chamada de Next Sentence e consiste em definir se duas sentenças que foram colocadas na entrada aparecem em sequência em um texto ou não.

Essas suas tarefas auxiliares forçam a rede a criar uma representação para cada token de entrada forma que ele possa ser identificado pelos demais tokens da entrada através da MLM. Além disso, o conjunto dos tokens precisa reter alguma informação de ordem dentro do texto, de forma que a segunda tarefa consiga ser executada.

Utilizando essa técnica o BERT conseguiu atingir um novo estado da arte em 11 das tarefas mais comuns de NLP. Mas a tarefa de resolução de correferência ainda não era uma delas. Apenas no trabalho de [72] que a tarefa foi abordada e um novo estado da arte foi definido.

Tanto o BERT quanto o ELMo conseguiram tantos avanços em tarefas

tão diversas que eles foram uma das motivações para o trabalho [41], que define tarefas mais difíceis e mais abrangentes, já que o antigo conjunto de benchmark estava ficando quase completamente resolvido.

O BERT propõe mais do que um modelo. É uma forma de abordar um problema. A ideia de se utilizar uma tarefa auxiliar para fazer um pré-treino em um dataset grade e depois utilizar o transfer learning foi utulizadas em trabalhos posteriores, como adaptações da arquitetura original. Por exemplo, em [99] os autores propõem otimizações dos hiperparâmetros do BERT. Eles não utilizam a tarefa de Next Sentence, utilizam sentenças maiores durante o treinamento e adotaram um dicionário maior.

5.3.3 Variações do BERT

A utilização do BERT da forma como foi definida em [12] apresenta alguns problemas para a resolução de correferência de acordo com [14]. Primeiro é que a tarefa de MLM não parece ser muito difícil quando apenas um token é escondido. Pela forma que as palavras são divididas, cada token será uma palavra inteira apenas se ela ocorrer um dicionário pré-estabelecido. Caso não ocorra, ela é quebrada e cada nova parte é então procurada no dicionário. Como todas as letras estão no dicionário, ao final cada token pode ser apenas uma letra. Essa forma de divisão é chamada de word piece-wise tokenization.

Um segundo problema é que a tarefa de Next Sentence não parece ajudar muito a melhorar os resultados, como citado por [99] e [14].

Diante disso, os autores então propõem uma forma diferente de fazer o treino do BERT e a batizam de SpanBERT. O objetivo principal é representações possam ser usadas para trechos de textos e não apenas tokens. Nessa nova forma de se fazer o treino são feitas duas modificações: na tarefa de MLM vários tokens em sequência são mascarados ao mesmo tempo e uma segunda tarefa chamada de Span Boundary Objective que aprende a definir todos os tokens mascarados utilizando apenas os tokens imediatamente anterior e posterior ao trecho mascarado.

Com essas duas modificações, o modelo conseguiu definir o atual estado da arte em 14 diferentes tarefas.

Uma abordagem um pouco diferente do BERT, mas que ainda utiliza o paradigma pré-treino seguido de transfer learning é apresentado em [101] e é chamada de XLNet. Essa abordagem utiliza uma variação dos transformers chamada Transformers-XL [102] para implementar o que os autores chamaram de modelo de permutação de linguagem (Permutation Language Modeling). Nele todas as possíveis permutações da ordem de fatorização dos elementos de

um contexto. Apesar de esse modelo ter conseguido superar o BERT de [12] em algumas tarefas do GLUE, novas implementações, como o SpanBERT já o ultrapassaram.

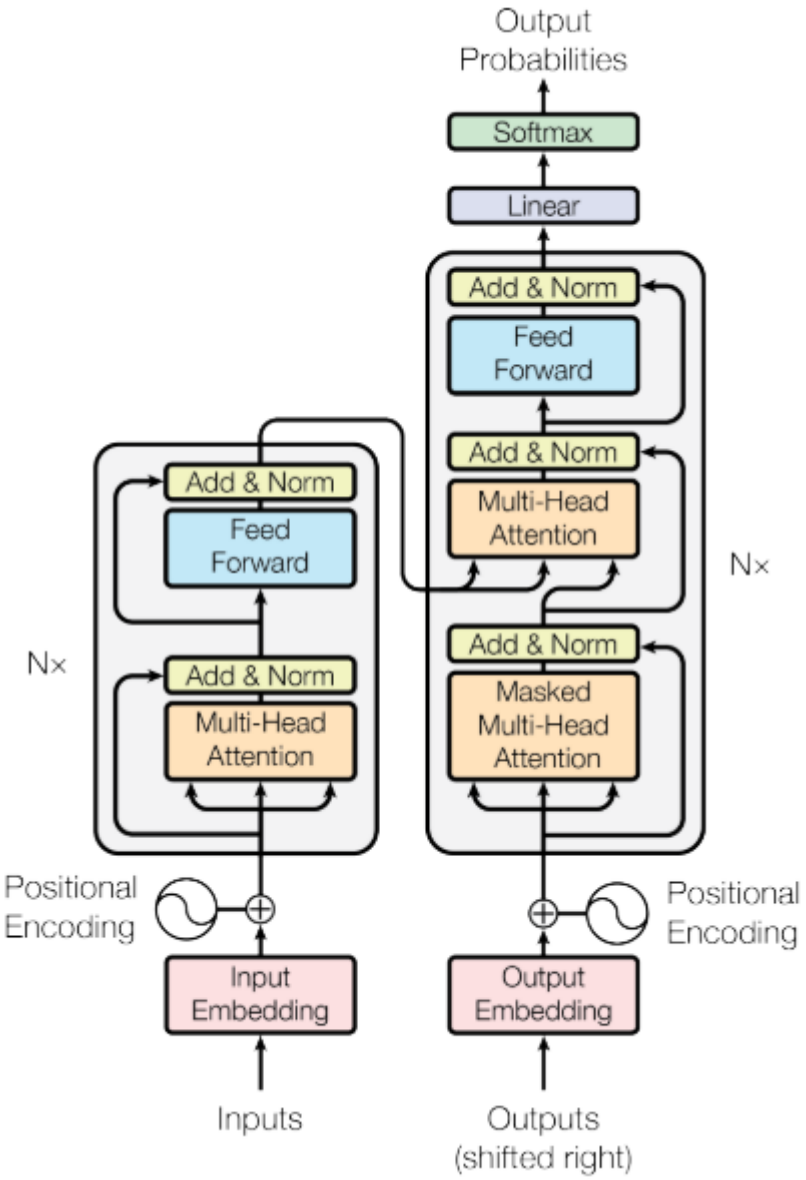


Figura 5.5: Arquitetura geral de um Transformer
À esquerda está a parte encoder e à direita está a parte decoder

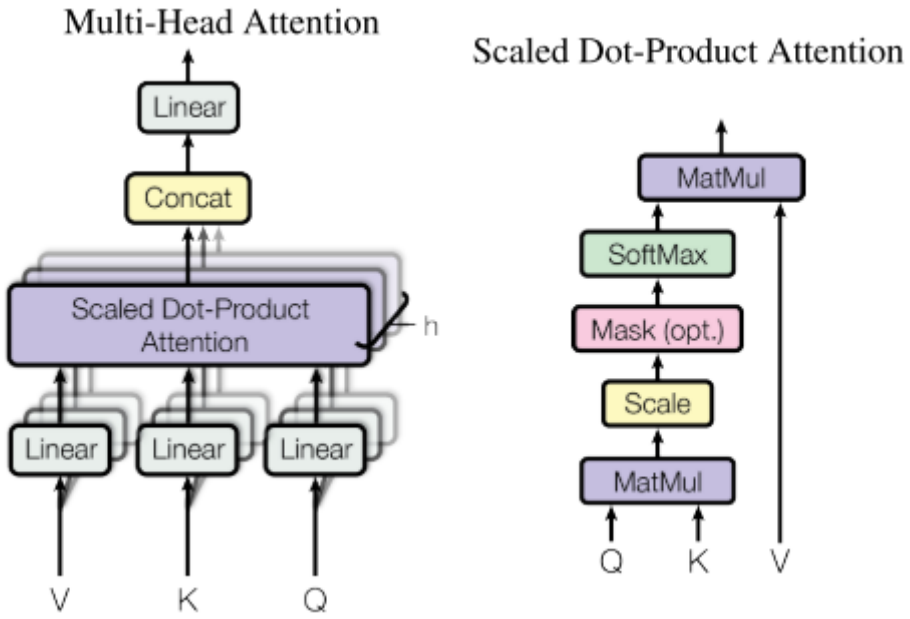


Figura 5.6: Detalhe dos blocos utilizados pelo Transformer
À esquerda está a composição dos blocos (cabeças) de atenção. à direita o detalhe como é feito o produto das entradas, efetivamente calculando a atenção.

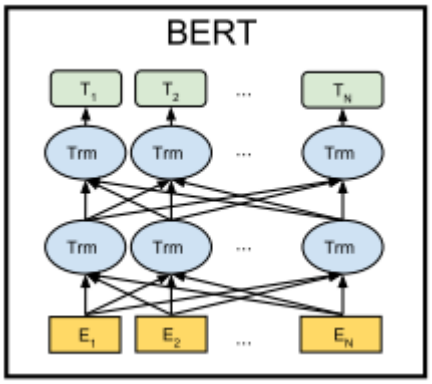


Figura 5.7: Arquitetura BERT como uma série de transformers

6
Resolução de correferência com representação contextual

Neste capítulo é feita a combinação da arquitetura de árvores latentes com a representação contextual das palavras. É detalhada a arquitetura geral dos experimentos, bem como quais são as modificações feitas nas soluções já apresentadas pelas referências. Em especial, são descritos as decisões e o embasamento empírico para se fazer as escolhas do parâmetros e da arquitetura.

6.1
Descrição dos experimentos

Nos trabalhos de correferência que utilizam pares de menções definidos de maneira heurística, há uma etapa posterior de definições de features desses pares. A maioria dessas features derivam de um conjunto pré-definido de regras linguísticas criadas por especialistas ou são estatísticas simples sobre o contexto no qual o par de menções ocorre. Em [33] é demonstrado como um conjunto de features mais bem elaborado pode fazer com que um sistema simples de classificação de pares de menções atinja resultados expressivos na tarefa de correferência. Em [60] é demostrado um sistema com boa performance utilizando poucas features básicas, que foram combinadas em um estilo um pouco diferente do EFI.

Após o trabalho de [38] a maioria dos avanços na tarefa proposta pela CoNLL 2012 se deu no sentido de criar uma solução ponta a ponta, ou seja, que já detectasse as menções, definisse quais seriam as features adequadas para cada par e criasse algum tipo de ranqueamento ou classificação dos possíveis pares utilizados. Apesar de serem soluções ponta a ponta, elas apresentam certos elementos internos inspirados por outros trabalhos. Por exemplo, em [39] os autores aplicam a técnica ELMo em [38] e em [72] é aplicada a técnica BERT baseando-se no mesmo trabalho.

Como as técnicas de representação contextual foram desenvolvidas apenas recentemente, é natural que elas sejam aplicadas no estado da arte mais recente. Porém, uma pergunta válida que fica é se os avanços recentes não poderiam ser conseguidos com arquiteturas mais simples, explorando-se apenas a informação contida nas features das menções de pares. Se for possível conseguir resultado compatíveis com pelo menos o melhor modelo criado a partir das features criadas por especialistas, então pode ser que exista modelos de aprendizado que exijam menos poder computacional que os atuais, mas que consigam resolver o mesmo problema.

Para se testar a hipótese de que apenas uma melhor codificação já é suficiente para capturar a informação utilizada na correferência, este trabalho utiliza o modelo de árvores latentes e varia a forma de codificar a informação dos pares de menções. Esse modelo foi escolhido por ter sido o ganhador da CoNLL 2012 e apresentar uma boa modularidade, facilitando a substituição de apenas um componente e, consequentemente, a comparação entre os resultados.

São feitos quatro testes de codificação de pares de menções: um de referência (baseline) e mais três variações. O baseline será definido conforme os trabalhos de [19] e [20]. Nele as features utilizadas são as definidas manualmente e é utilizado o EFI. Nas variações, as features serão substituídas por representações das palavras das menções. As representações serão feitas utilizando-se o GloVe, BERT e SpanBERT.

O modelo GloVe foi escolhido por dois motivos: primeiro ele possui menos parâmetros a serem otimizados do que o word2vec e tem uma performance superior ao LSA; segundo, ele é um modelo que tem como motivação alguma modelagem contextual considerando todo o corpus, como os demais modelos avaliados e uma questão central para a resolução de correferência.

O modelo BERT foi escolhido por ser o modelo mais estável dentre as atuais propostas de representação contextual. Apesar de ter sido proposto recentemente, o número de variações existentes, otimizações finas propostas e aplicações práticas [1] indicam que ele é um modelo inicial confiável.

O modelo SpanBERT foi escolhido por ser o atual estado da arte e ter sido desenvolvido com a tarefa de correferência entre um de seus objetivos.

Todos os testes seguem o fluxo ilustrado pela figura 6.1. Este fluxo é fortemente inspirado pelos trabalhos [19] e [20], sendo que muitos elementos da implementação são adaptações do código disponibilizado pelo autor do último. Também são utilizados implementações e modelos dos autores de [12] e [14], além de uma biblioteca [2] construída especificamente para este trabalho.

O fluxo consiste em separar o texto em menções, criar os arcos candidatos entre essas menções, fazer a codificação desses arcos e alimentar o perceptron estruturado de forma a se obter os melhores arcos e, consequentemente, os agrupamentos. É o mesmo fluxo observado em trabalhos como o de [19], [20].

As adaptações foram necessárias para que todos os modelos pudessem ser treinados da mesma forma. Apesar de o baseline poder ser treinado nas máquinas disponíveis para este projeto, as representações dos modelos BERT e SpanBERT exigiriam ambientes mais complexos do que os disponíveis.

[1] https://www.blog.google/products/search/search-language-understanding-bert/
[2] https://github.com/leonardoboliveira/conll2012_boilerplate

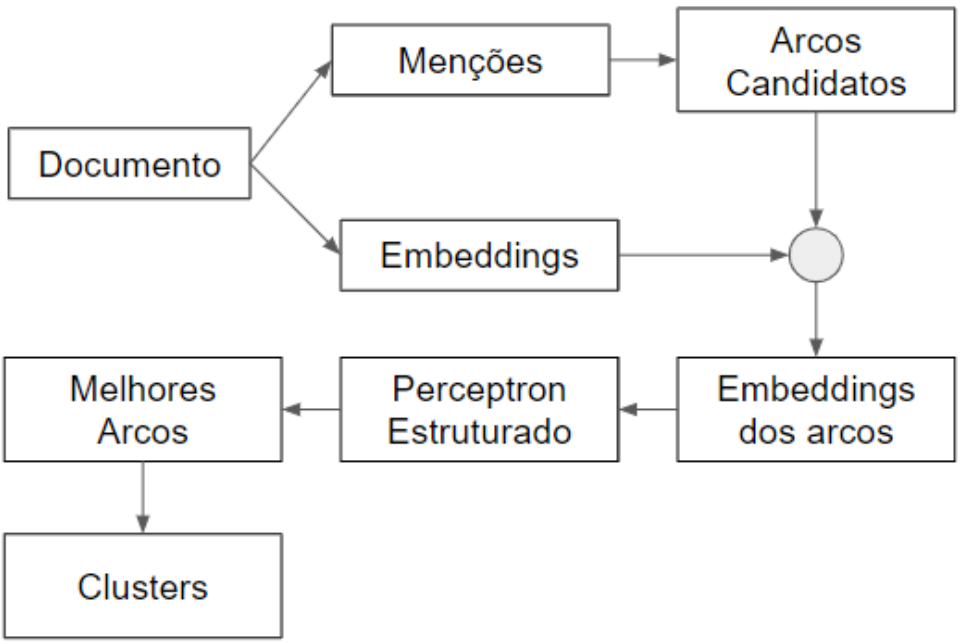


Figura 6.1: Fluxo de informação durante a resolução de correferência

Para se manter uma consistência entre os testes, todos eles utilizam a mesma implementação dos elementos principais do fluxo. Apenas a forma de calcular o vetor representativo de cada palavra que é variada. Como o objetivo deste trabalho é comparar o desempenho da codificação dos arcos, apenas os resultados nos conjuntos de treino e validação são utilizados.

6.2 Arquitetura Geral

Em termos de arquitetura, o sistema segue a organização demostrada pela figura 6.2 que é uma modificação da implementação de [108] para o problema de correferência. Neste diagrama há um módulo principal de treino que recebe uma parte do dataset para ser utilizada no treino. Cada lote é dividido em documentos e cada documento é tratado pela biblioteca StanfordParser [138]. Ela gera as informações de menções, informações linguísticas como a estrutura léxica do texto, palavra principal da menção; e também informações que estão presentes no dataset como o orador, gênero e a anotação de correferência.

As menções e seus atributos são repassados para o módulo de experimentos que utiliza os vetores associados aos arcos candidatos e gera as árvores candidatas a partir dos pesos definidos pelo perceptron estruturado. Esses vetores são gerados no módulo de extração de features destacados na figura. Esse módulo é o único que varia entre os vários experimentos deste trabalho.

Após a convergência do perceptron, o módulo de experimento devolve ao módulo principal uma versão treinada do modelo.

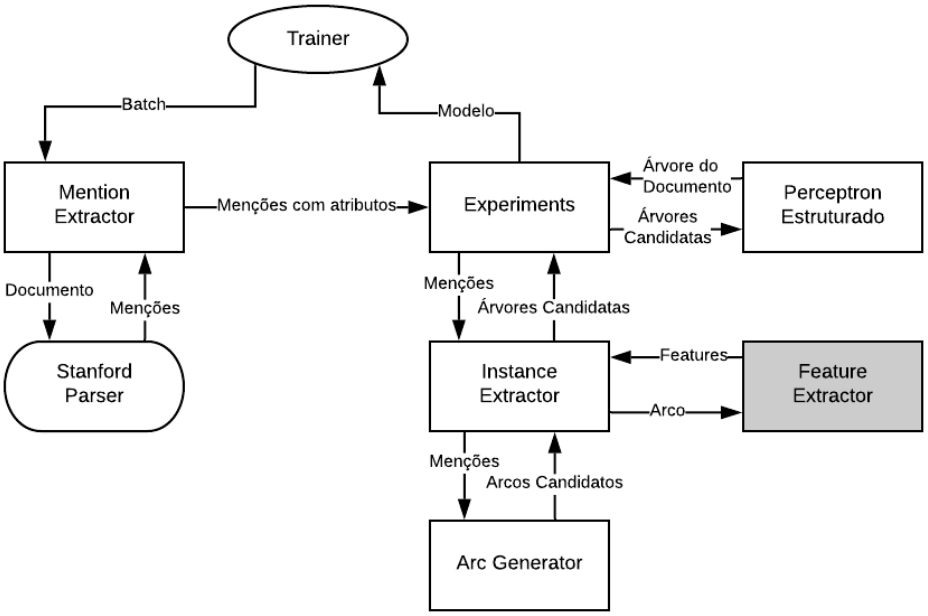


Figura 6.2: Arquitetura básica do sistema de treino
O sistema segue a implementação de [108], com modificações. Em especial, o extrator de features destacado muda para cada tipo de representação testada

Após o treinamento, o modelo é processado pelo módulo de predição. Ele tem uma arquitetura muito parecida, como exibido na figura 6.3. As diferenças estão na entrada do preditor que utilizará o dataset de validação e na saída que utiliza o modelo para definir a árvore do documento

6.3 Conjunto de dados

O OntoNotes 5.0 [73] é um corpus que procura abordar textos das mais diversas origens. Seus textos são agrupados em sete tipos:

- Conversas na televisão (bc)
- Notícias de televisão (bn)
- Revistas (mz)
- Notícias atuais (nw)
- Partes do Novo e Velho Testamento (pt)
- Conversas de telefone (tc)
- Textos retirados da internet (wb)

Cada tipo possui diversas fontes, por exemplo, os dados de notícias de televisão foram obtidos a partir de programas da ABC, CNN, NBC e outros.

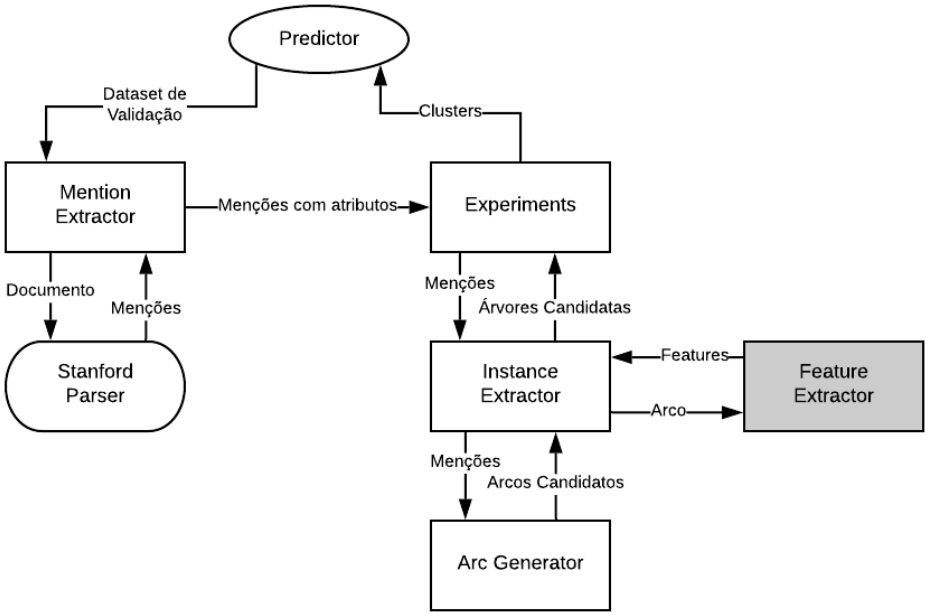


Figura 6.3: Arquitetura básica do sistema de predição

O preditor tem uma arquitetura semelhante ao sistema de treino

Como neste trabalho será necessário criar partições do conjunto de dados original de treino, é importante que a distribuição da natureza dos textos se mantenha mais ou menos a mesma entre o conjunto original e cada uma das partições.

Para se obter esse efeito e ainda ter alguma variabilidade, cada documento do corpus original é colocado em uma partição segundo uma distribuição uniforme. Após o particionamento, uma das partições é aleatoriamente para ser o conjunto de validação durante o treino. Essa partição é chamada de minivalidação para distinguir do conjunto de dados de validação oficial disponibilizado pela CoNLL.

Na tabela 6.1 é possível ver o particionamento dos dados de treino. Foram definidas 73 partições, sendo uma escolhida para a minivalidação. Esse número de divisões é utilizado para manter as partições com dados suficiente de todos os tipos e ainda com um tamanho total pequeno. Isso permite que todos os testes com as diversas formas de representação caibam na memória dos servidores disponíveis para este projeto.

6.4 Detecção de Menções

Neste trabalho utiliza-se o detector de menções implementado para [20] . Antes, porém, foi feita uma validação de sua performance. Essa análise foi feita

Tipo	Total de Documentos	Média por partição	% do Total
bc	368	5.1	1.4%
bn	850	11.6	1.4%
mz	409	5.6	1.4%
nw	1832	25.1	1.4%
pt	403	5.5	1.4%
tc	115	2.0	1.4%
wb	7424	101.7	1.4%

Tabela 6.1: Particionamento do conjunto de treino
Particionamento do dataset de treino em 73 partes

Dataset	Total	Detectadas	Corretas	Precisão	Exatidão
Treino	155558	573891	149695	26%	96%
Validação	19155	78853	18213	23%	95%

Tabela 6.2: Performance do detector de menções

comparando-se quantas menções eram detectadas em relação a quantas menções estavam anotadas em um determinado documento. Foram consideradas menções corretas apenas aquelas que possuíam exatamente o mesmo conjunto de palavras. Marcações parciais foram consideradas erros.

A tabela 6.2 mostra a precisão e exatidão (recall) do detector de menções nos dados de treino. A exatidão é a métrica importante neste caso, já que é preciso que o detector identifique todas as entidades para que o restante do algoritmo possa agrupá-las. Um bom algoritmo de clusterização deve ser capaz de colocar as entidades erroneamente identificadas em grupos isolados, fazendo com que não sejam utilizadas ao final. Em [19] é colocado que o detector de menções utilizado tem um recall acima de 90% e que as menções que não foram detectadas não devem impactar tanto no resultado final.

É importante notar que uma parte das menções que são foram detectadas devido a divergências nas anotações e também ao fato de que uma pequena parcela das frases verbais foi anotada. Como a tabela 6.2 indica, essas menções correspondem a menos de 5% do total e não deve influenciar muito o resultado final do agrupamento.

6.5 Geração de arcos candidatos

Na implementação de [20] a geração dos arcos candidatos é feita de maneira exaustiva. Todos os pares entre as menções detectadas e seus antecedentes são utilizados. Como esse processo gera uma quantidade grande de informações e a forma de codificar os arcos neste trabalho se propõe a ser mais exigente

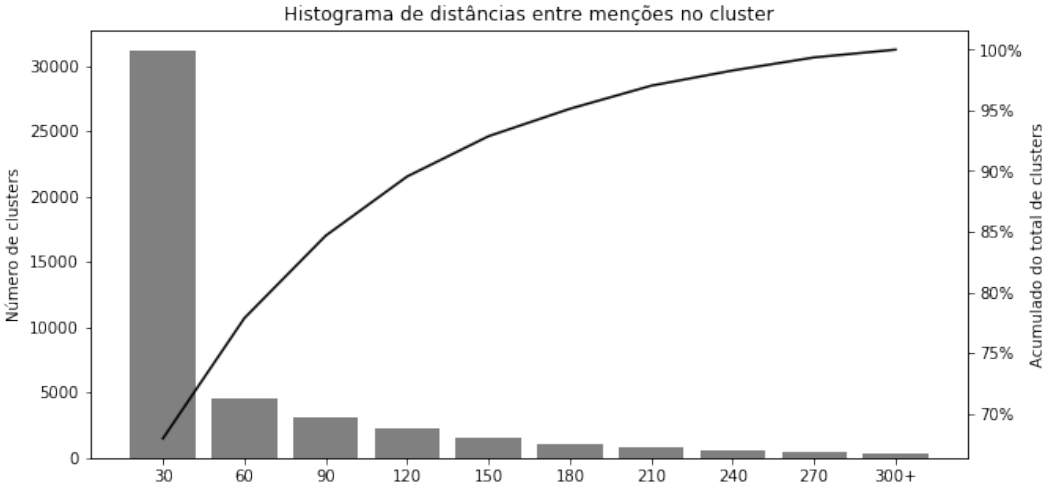


Figura 6.4: Distribuição dos tamanhos dos clusters
As barras indicam o número de clusters que possuem até um determinado número de menções dentro dele. A linha é o percentual acumulado do número de clusters

em termos de memória que a versão original, optou-se por adotar uma abordagem inspirada por [19] e mais próxima da forma descrita na seção 4.3. Porém, ao invés de se gerar os arcos baseados em diversas regras, apenas a regra da distância entre as menções foi utilizada.

A distância máxima entre as menções foi definida através de uma análise dos dados de treino. A figura 6.4 mostra o número de menções que o detector encontrou entre a primeira e a última menção de um cluster, independentemente de a menção estar realmente no cluster ou não. Os dois gráficos juntos indicam que quase 70% dos clusters possuem até 30 menções identificadas entre a primeira e a última menção deste cluster. A figura 6.5 mostra quantas menções existem entre duas menções de um mesmo cluster. Por essa figura é possível concluir que em mais de 95% das vezes duas menções de um mesmo cluster estarão a menos de 50 menções uma da outra. O valor do parâmetro N foi estabelecido em 40, de forma que as chances de se identificar duas menções dentro de um mesmo cluster seja grande e que não haja a necessidade de se explorar todos os arcos possíveis.

6.6 Representação contextual dos arcos

A principal colaboração deste trabalho é a comparação de três formas de se representar o contexto com a forma atualmente utilizada em árvores latentes. Atualmente essa representação é feita utilizando-se features básicas que são definidas por especialistas. Essas features básicas depois são combinadas utilizando uma forma que é guiada pela entropia, como em [19] ou simples-

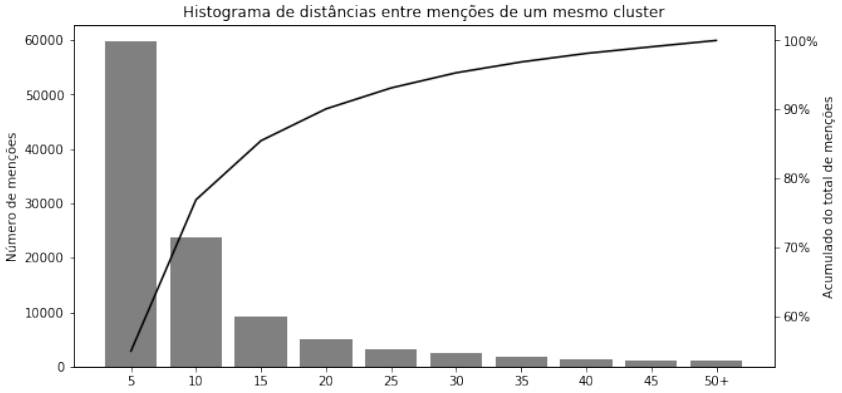


Figura 6.5: Distribuição das distâncias entre as menções dentro de um mesmo cluster

As barras indicam quantas menções estão existem entre duas menções de um mesmo cluster. A linha indica o valor acumulado do total de menções

mente concatenando as informações básicas das menções que compõem cada arco.

Para o modelo de baseline serão utilizadas as 24 features básicas indicadas por [108]:

- Características das menções
  1. Palavra principal (mention head)
  2. Primeiro/Último palavra
  3. Palavras antes/depois
  4. Número de palavras
  5. Classe da entidade nomeada, caso a menção seja um nome próprio
  6. Gênero/Grau
  7. Classe sintática
  8. Tipo fino
  9. Relação de dependência entre a palavra principal da menção e seu regente
  10. Regente da palavra principal
  11. Ancestralidade na árvore sintática da palavra principal
- Características do Par
  1. As duas menções são iguais
  2. A palavra principal das duas menções é a mesma

3. Indicador se a anáfora ou o antecedente estão em uma relação de "apelido" da outra. Esse indicador utiliza uma série de heurísticas para determinar se há uma relação na qual uma das menções é uma substituição clara da outra, como por exemplo se uma é a abreviação da outra
4. Indicador se uma menção está contida na outra
5. Indicador se a palavra principal de uma menção ocorre na outra
6. Indicador se todos os modificadores da anáfora estão presentes no antecedente
7. Número de frases entre as menções, limitado a 5
8. Número de palavras entre as menções, limitado a 10
9. Indicador se o campo *speaker* é o mesmo para as duas menções

Apesar de essa lista ser diferente das features básicas propostas por [19], é possível notar que a combinação delas produzem resultados parecidos. Por exemplo, as features *classe semântica* e *gênero* ao terem seus valores combinados terão uma ocorrência que corresponde à feature Sy5 de [19] que é o indicador de as menções serem pronomes e concordarem em gênero.

Para o baseline é utilizada a derivação de features do sistema SURFACE de [60]. Nesse mecanismo de derivação as features básicas têm seus valores para as menções combinados e concatenados de forma a gerar um novo conjunto de características que depende apenas da ocorrência dos valores no dataset de treino.

6.6.1 Variações das representações

O principal diferencial das representações contextuais é que cada palavra tem um vetor de features associado para cada contexto em que ela ocorre. Para se conseguir essa codificação é preciso inicialmente definir qual é o tamanho do contexto. Idealmente todo o documento seria considerado o contexto. Porém, os modelos do BERT e SpanBert possuem um tamanho máximo de entrada e, mesmo que se utilize o maior tamanho disponibilizado pelos autores de [12] e [14], eles ainda são muito pequenos para alguns documentos do OntoNotes. Além disso, ambos os trabalhos relatam uma perda de performance na utilização de contextos muito longos.

A sugestão apresentada tanto por [12] quanto por [14] é utilizar um contexto deslizante. Neste caso, define-se a priori um tamanho de janela que será utilizado e um passo. Cada janela é fornecida para um codificador

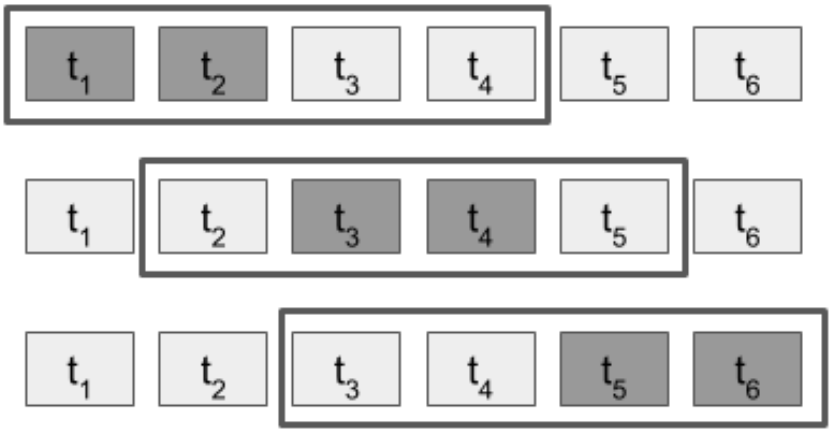


Figura 6.6: Exemplo de contexto deslizante
Neste exemplo, a janela possui tamanho 4 e passo 1. Os tokens destacados são aqueles que serão utilizados no final

(embedder) que retornará uma representação para cada palavra, de acordo com aquele contexto. Em seguida é dado um passo e uma nova janela é definida. É importante notar que nesse formato cada token pode ter mais uma representação, uma para cada contexto no qual ele aparece. Será utilizado a representação no contexto em que o token aparece mais ao centro. Na figura 6.6 há um exemplo de como esse mecanismo funciona.

No primeiro passo, os tokens $t_1, t_2, t_3 et_4$ são utilizados como um contexto a ser codificado. Apesar de todos os tokens terem uma representação, apenas os vetores associados aos tokens t_1 e t_2 serão utilizados. Na figura também é possível notar que os tokens t_3 e t_4 possuem três representações diferentes, mas apenas as representações do segundo passo serão utilizadas.

O tamanho da janela e do passo são hiperparâmetros do modelo. Pare este teste serão seguidos os valores que foram testados pelos trabalhos [12] e [14], ou seja, para o modelo BERT a janela possui tamanho 384 e o passo tamanho 128; para o modelo SpanBERT tanto a janela quanto o passo terão tamanho 256. No trabalho [14] foram testadas janelas com e sem interseção e os resultados apontaram que a utilização de janelas sem interseção era melhor.

Para o modelo GloVe pode-se utilizar qualquer valor de janela e passo, já que as palavras terão a mesma codificação independente do contexto em que elas ocorrem.

Apesar de existirem evidências em [14] de que se fazer um ajuste fino no modelo do BERT utilizando um dataset específico da tarefa em questão, o intuito deste trabalho é manter os requisitos computacionais o mais restrito que for possível. De acordo com [99], esse tipo de ajuste fino (fine-tuning) é medido em unidades de ano-GPU, ou seja, quantos anos levaria para o modelo ser treinado se fosse utilizado apenas uma placa de vídeo de qualidade profissional.

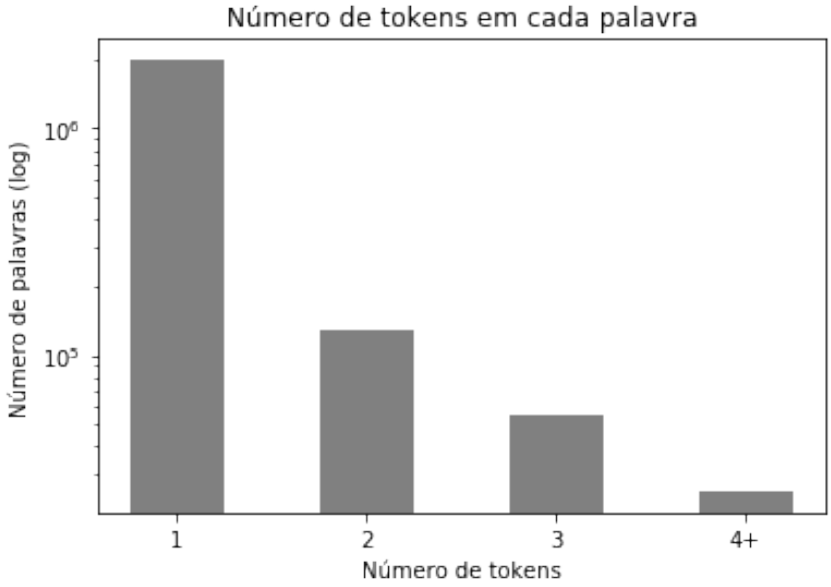


Figura 6.7: Distribuição de tokens por palavra
Número de tokens em cada palavra no conjunto de treino

Desta forma, os modelos são utilizados como disponibilizados pelos autores dos trabalhos originais.

Uma segunda adaptação que é necessária ser definida é como ter um único vetor associado a um par de menções sendo que os modelos retornam vetores de representação de palavras. Nos trabalhos [39], [72] e [14] que utilizam algum tipo de representação contextual o problema não existe de forma direta, já que a detecção de menção está na mesma rede que faz a classificação do par. Como essas soluções utilizam uma rede neural, é razoável esperar que a rede faça algum tipo de soma ponderada entre os tokens para se determinar como eles devem se combinar. No caso deste trabalho optou-se então por utilizar o valor médio dos tokens que compõem a menção e a o par será caracterizado pela concatenação da representação das duas menções.

Apesar de ser uma decisão de arquitetura muito importante, o fato de se utilizar a média dos tokens ao invés de, por exemplo, o primeiro token da palavra, ela não deve influenciar muito para o dataset do OntoNotes. A figura 6.7 mostra que mais de 90% das palavras são compostas de apenas um token. Desta forma, a média das palavras da menção é, aproximadamente, a média dos tokens da menção.

Já o fato de se utilizar a média das palavras para representar a menção não é uma decisão óbvia. A figura 6.8 mostra que menos de 50% das menções possuem apenas uma palavra. Isso significa que, potencialmente, a representação da menção não é única. Para se manter a coerência entre a representação das palavras e das menções, optou-se por utilizar como o vetor representativo

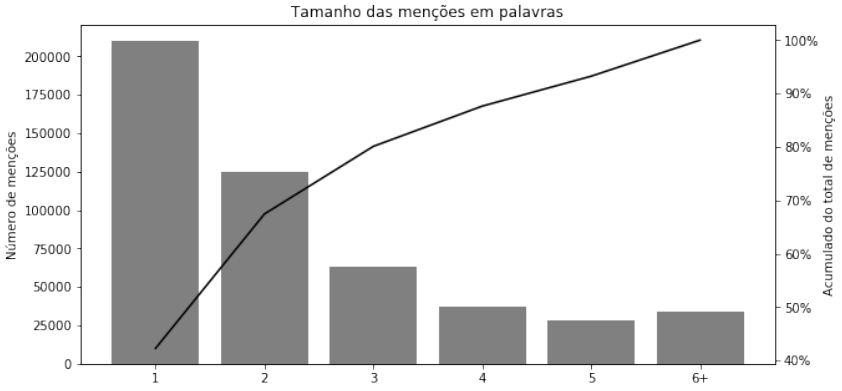


Figura 6.8: Distribuição de palavras por menção
Número de palavras em uma menção no conjunto de treino

de uma menção o vetor médio das representações de suas palavras. Para as menções artificiais o vetor representante será sempre **0**.

Os vetores nas representações utilizando GloVe, BERT e SpanBERT possuem tamanhos diferentes de dimensões, mas cada dimensão possui aproximadamente a mesma variabilidade. Para o GloVe os vetores possuem 300 dimensões com valores reais variando entre -3 e 3; para o BERT foi utilizado os vetores com 768 dimensões e cada um varia entre -5 e 5; para o SpanBERT foi utilizado vetpres cp, 1024 posições, também variando entre -5 e 5.

Apesar de o algoritmo do perceptron estruturado tratar de features com valores reais, optou-se por transformar cada dimensão em uma feature categórica. Desta forma pode-se utilizar técnicas como o EFI na indução de outras features.

Essa categorização se dá transformando cada dimensão real em seu decil na distribuição dentro do conjunto de treino. Ou seja, dado todo o conjunto de treino, cada documento é codificado em sua integralidade e é calculada a distribuição de valores para cada uma das posições do vetor de representação das palavras considerando todas as palavras do corpus. A partir dessa distribuição são calculados os decis, ou seja, quais os valores que dividem o conjunto em 10 partes com o mesmo número de ocorrências. Cada posição do vetor tem a sua distribuição como é possível ver pela figura 6.9. Ou seja, cada posição será níveis específicos de estratificação de seus valores. É possível ver na figura a distribuição de duas das 300 features dos vetores codificados utilizando o GloVe. As distribuições não são iguais e a região destacada mostra que o intervalo de valores que corresponderiam à classe 2 é diferente em cada uma delas.

Após a definição desses níveis, cada posição do vetor terá associado um valor inteiro entre 0 e 9. Esse valor será tratado como as features numéricas básicas, ou seja, como sendo possíveis valores de uma classe.

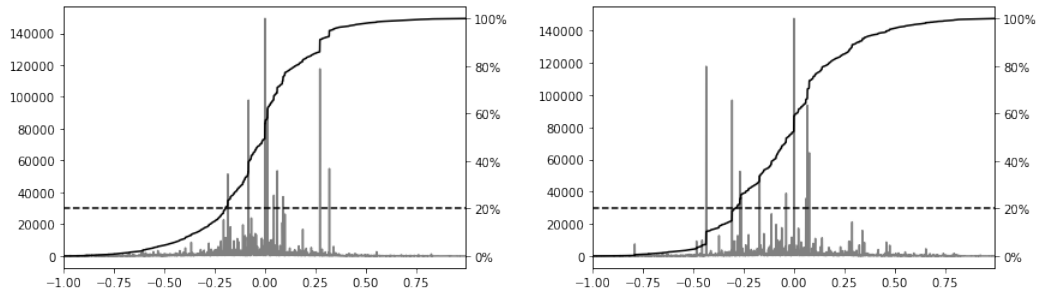


Figura 6.9: Exemplos de distribuição dos valores das features
Estas são duas features codificadas com GloVe com a dispersão dos valores no dataset de treino. No eixo direito está marcado o percentual acumulado das frequências. Em destaque em pontilhado está a região correspondente à classe 2

6.6.2 Perceptron Estruturado

A solução implementada utiliza como base o perceptron para árvores latentes criado para [20], com algumas modificações de performance e adaptações para a nova forma de representar os arcos. Uma modificação um pouco mais significativa na sequência de treinamento é também utilizada. No algoritmo 1 a implementação utiliza uma atualização no estilo gradiente descendente como no algoritmo original do perceptron de [131]. A desvantagem dessa forma de otimizar é que é preciso que todos os dados estejam em memória para se conseguir fazer a convergência. Como neste trabalho são utilizadas mais features e, portanto, mais memória, e necessário utilizar uma versão estocástica, mais próxima do gradiente descendente estocástico, por exemplo como o proposto em [139].

Mais especificamente, para cada parte do conjunto de dados (batch), os pesos são atualizados até que não haja uma melhora nos resultados naquele batch. O código é descrito pelo algoritmo 2. Os batches são definidos particionando-se o dataset $\mathcal{D}$ de forma que cada partição tenha aproximadamente o mesmo número de documentos e que eles sejam escolhidos de forma representativa. Ou seja, define-se um conjunto $\mathcal{B}$ cujos elementos sejam partições de $\mathcal{D}$ e as distribuições de cada elemento seja próxima à distribuição original.

O critério de convergência para os batchs é que o número de arcos corretos parou de aumentar. Não é interessante tentar atingir a melhor marca possível em cada batch, já que isso aumentaria as chances de se ter um overfitting.

Algoritmo 2: Perceptron estruturado estocástico

$\mathbf{w}_0 \leftarrow 0$;
t $\leftarrow 0$;
enquanto *não convergiu* **faça**
  para *batch* $\in \mathcal{B}$ **faça**
    enquanto *não convergiu* **faça**
      para $(\mathbf{x}, \mathbf{y}) \in$ *batch* **faça**
        $\tilde{\mathbf{h}} \leftarrow \arg\max_{h \in \mathcal{H}(\mathbf{x},\mathbf{y})} \langle \mathbf{w_t}, \mathbf{\Phi}(\mathbf{x}, \mathbf{h}) \rangle$;
        $\hat{\mathbf{h}} \leftarrow \arg\max_{h \in \mathcal{H}(\mathbf{x})} \langle \mathbf{w_t}, \mathbf{\Phi}(\mathbf{x}, \mathbf{h}) \rangle + C \cdot l_r(\mathbf{h}, \tilde{\mathbf{h}})$;
        $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{\Phi}(\mathbf{x}, \tilde{\mathbf{h}}) - \mathbf{\Phi}(\mathbf{x}, \hat{\mathbf{h}})$;
        $t \leftarrow t + 1$;
$\mathbf{w} \leftarrow \frac{1}{t} \sum_{i=1}^{t} \mathbf{w}_i$

7
Resultados

Neste capítulo são descritos os resultados dos experimentos. Esses resultados seguem uma ordem de forma evolutiva, partindo-se de uma modificação mais simplista e, então, sendo incorporados elementos mais elaborados. Um *leaderboard*, ou seja, um ranqueamento de todos os experimentos feitos é apresentado e o comparativo com as soluções já propostas por outros estudos. Ao final é feito a avaliação do modelo no dataset de teste bem como onde ele se com as pontuações oficiais de outros modelos. É adotada uma convenção de nomenclatura de arquiteturas. A tabela 7.1 indica o código associado a uma arquitetura específica e todos os gráficos e menções a esse código podem ser entendidos como uma referência explícita a arquitetura designada.

7.1
Ambiente Experimental

Todos os experimentos seguem um mesmo protocolo e os resultados de todos eles são comparáveis. Será sempre utilizado o conjunto de dados de treino da CoNLL 2012 para fazer o treinamento do modelo, dividido em 73 partes de forma uniforme, sendo uma delas para fazer uma minivalidação e 72 para treinamento.

A arquitetura utilizada é a implementação de [20] para árvores latentes, com as modificações feitas para este trabalho. A utilização da indução de features, bem como a forma de codificar os arcos de antecedentes, depende de cada experimento. Em todos os experimentos será utilizada a margem larga. Será adotado o valor 2000 para a constante C, que foi o valor adotado por [19].

As métricas utilizadas serão as mesmas designadas pela CoNLL 2012, ou seja, a média do F1 score das métricas MUC, B-CUBED e CAEF-E. Essa métrica é referenciada nos resultados como *Score CoNLL*.

A evolução da performance dos modelos será mostrada a cada etapa do treinamento. Uma etapa é um conjunto de 6/73 do conjunto total de dados no qual o modelo é treinado e posteriormente tem sua performance avaliada no 1/73 do conjunto que ficou separado para minivalidação. Como a variabilidade do resultado entre uma etapa e a outra é muito grande, os gráficos serão mostrados com uma suavização de média móvel exponencial com $\alpha = 0.1$, ou seja, a cada instante t, a média é dada por $ewm_t = (1 - \alpha) * ewm_{t-1} + \alpha * x_t$, onde x_t é o valor no instante t. Nos gráficos também estarão evidenciadas as

Código	Features	Indução	Margem
lexico	léxicas	nenhuma	2.000
lexico_efi	léxicas	EFI	2.000
glove	GloVe	nenhuma	2.000
glove_efi	GloVe	EFI	2.000
glove_surface	GloVe	SURFACE	2.000
bert	BERT	nenhuma	2.000
bert_efi	BERT	EFI	2.000
bert_surface	BERT	SURFACE	2.000
span	SpanBERT	nenhuma	2.000
span_efi	SpanBERT	EFI	2.000
span_surface	SpanBERT	SURFACE	2.000
span_surface_2E6	SpanBERT	SURFACE	2.000.000
baseline	manuais	SURFACE	2.000

Tabela 7.1: Códigos para os resultados divulgados

épocas, ou seja, uma passagem completa por todo o conjunto de dados, como linhas verticais pontilhadas.

Para se manter uma coerência na divulgação dos resultados e evitar que as imagens fiquem sobrecarregadas de notações, são utilizados códigos únicos para os experimentos, conforme a tabela 7.1. A coluna *features* indica qual representação foi utilizada cada arquitetura. O valor "*léxicas*"indica que o modelo utiliza apenas as features que dependem de menções, definidas de forma manual. Os valores "*GloVe*", "*BERT*"e "*SpanBERT*"indicam qual a técnica de codificação dos arcos foi utilizada para se definir as features. O valor "*manuais*"indica a utilização de todas as featuers derivadas por especialistas utilizadas por [20], ou seja, o conjunto léxico mais as features que dependem de ambas as menções.

7.2 Utilização de representação de palavras

Nos trabalhos de [19] e [20] são utilizadas features derivadas de características definidas por linguistas e utilizados em diversos outros trabalhos de resolução de correferência. Essas features capturam informações básicas das palavras das menções de um arco e algumas características associadas ao texto entre elas.

Um primeiro experimento para se automatizar a geração de features é substituir as features básicas que dependem apenas de menções pela representação GloVe. Essa representação é obtida após fazer a média da representação vetorial das palavras da menção, obtida pela técnica Glove e, posteriormente, fazendo-se a binarização de cada dimensão do vetor.

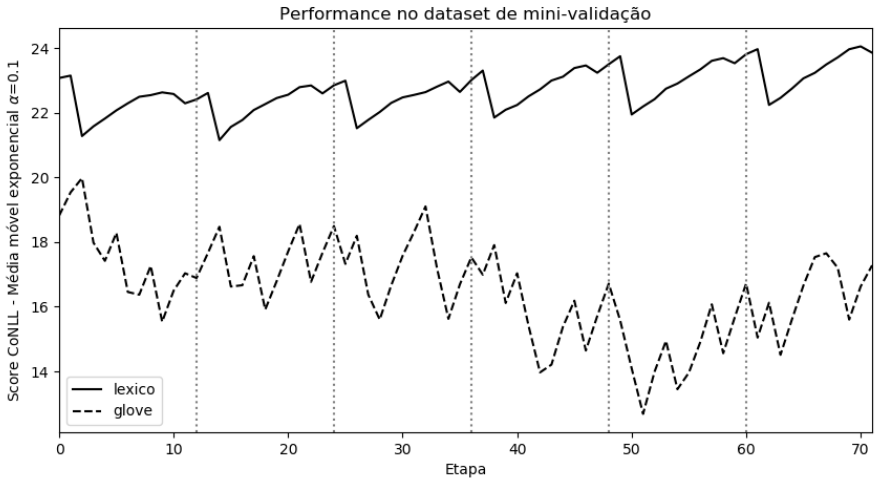


Figura 7.1: Desempenho do Glove contra as features léxicas

Evolução do Score CoNLL do GloVe e das features léxicas durante o treinamento

Código	MUC	B-CUBED	CEAF-E	Score CoNLL
lexico	40.22	22.77	12.92	25.30
glove	39.52	20.63	12.90	24.35

Tabela 7.2: Resultados dos modelos no dataset de validação

Na figura 7.1 é mostrado o resultado do treinamento dos modelos *lexico* e *glove_bins*. O primeiro utiliza como features as características que dependem apenas das menções. O segundo utiliza as features categóricas geradas pelo GloVe. Em nenhuma delas é utiliza indução de features.

Considerando que a métrica da CoNLL pode ser entendido como o percentual das entidades que foram corretamente identificadas e que os trabalhos de [19] e [20] reportam valores da ordem de 60 pontos no dataset de teste, é possível concluir que nenhum desses dois modelos tem um bom desempenho em termos absolutos. Além disso, percebe-se que o GloVe tem um desempenho abaixo das features criadas por especialistas. Na tabela 7.2 é mostrado o resultado do melhor modelo desse treinamento avaliado no dataset de validação. A diferença acaba ficando menor do que durante o treinamento.

Mesmo utilizado um mecanismo de indução como o EFI, o GloVe não tem um desempenho comparável às features léxicas, como pode ser visto na figura 7.2. É possível notar que os modelos tiveram uma melhora da ordem de 50% em relação aos modelos sem o EFI. Esse nível de melhora é compatível com os números divulgados por [19].

A tabela 7.3 mostra o desempenho do melhor modelo de cada arquitetura. A melhora observada no treinamento se reflete nessa validação, dando

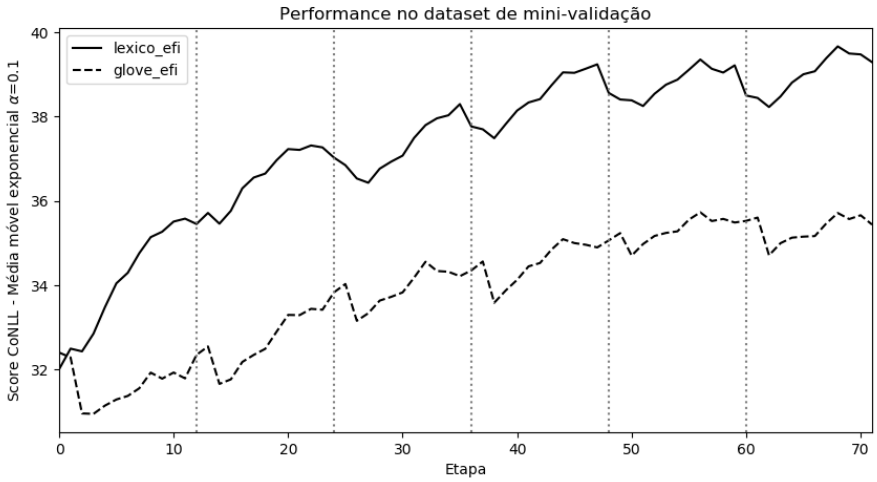


Figura 7.2: Desempenho do Glove contra as features léxicas, utilizando EFI
Evolução do Score CoNLL do GloVe e das features léxicas durante o treinamento, utilizando EFI

Código	MUC	B-CUBED	CEAF-E	Score CoNLL
lexico_efi	48.89	37.01	31.07	38.99
glove_efi	44.23	33.95	28.78	35.65

Tabela 7.3: Resultados dos modelos no dataset de validação utilizando EFI

indicações de que o EFI consegue auxiliar na correferência de maneira real, e não é apenas um overfitting.

Uma outra forma de se aplicar a indução de features é a SURFACE [20]. Nela, as features são emparelhas e novos templates surgem. Diferente do EFI que utiliza uma árvore de decisão limitada, o número de templates gerados pelo SURFACE é diretamente proporcional ao número de features básicas. Isso significa que quando se extrapola das 14 features léxicas e se utiliza as 300 features geradas pelo GloVe, o que signfica um aumento de cerca de 20 vezes no tamanho da representação dos arcos. A figura 7.3 mostra duas distribuições dos tamanhos dos arcos representados pela indução SURFACE e EFI utilizando-se o GloVe em relação às features léxicas. Como esperado, as representações pelo SURFACE são consideravelmente maiores do que aquelas geradas pelo EFI.

Apesar desse aumento significativo de tamanho e características utilizadas, o SURFACE não melhora o resultado do BloVe, como pode-se ver na figura 7.4. O comportamento de ir piorando o resultado à medida que o treinamento avança é um forte indicador de que as features geradas pelo SURFACE no GloVe geram apenas ruído.

Seja porque as features manuais discriminam mais as palavras presentes na menção e a representação no GloVe faz uma média, ou seja porque o GloVe

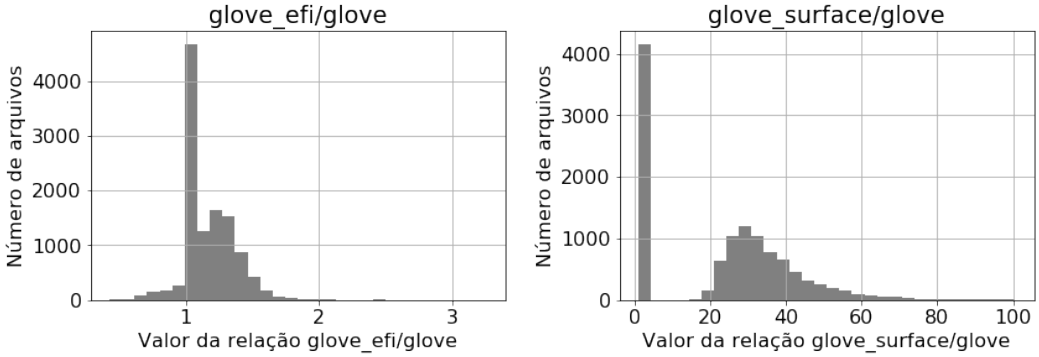


Figura 7.3: Comparação de tamanhos relativos das induções EFI e SURFACE **Na figura da esquerda está a distribuição da relação glove_efi/glove e na figura da direita a distribuição da relação glove_surface/glove**

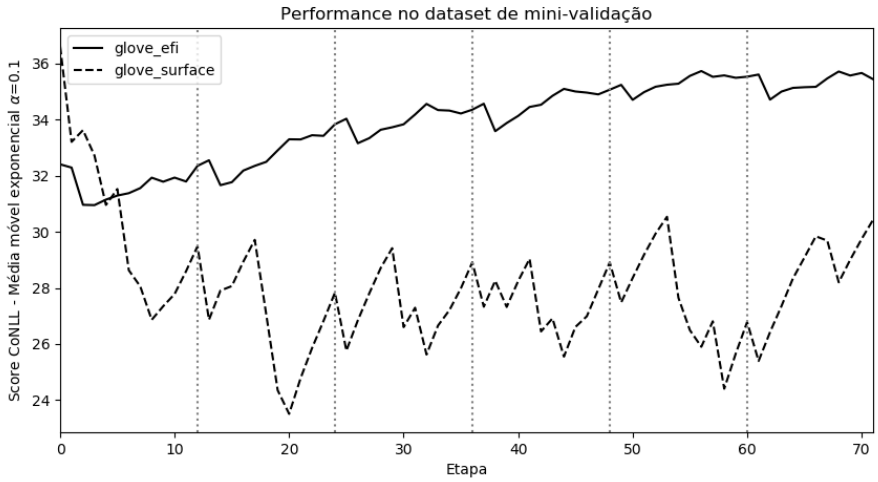


Figura 7.4: Desempenho do Glove utilizando EFI e SURFACE

talvez não capture informações sintáticas específicas como a classe ou o regente da palavra principal, o fato é que para a automatização neste exemplo não é justificada.

7.3 Utilização de representação de contexto

No trabalho de [12] no qual é apresentado o BERT são mostrados resultados em diversas tarefas de NLP que atingiram um novo estado da arte. Muitas dessas tarefas utilizavam anteriormente representações de palavras como o GloVe ou word2vec antes do BERT.

Naquele trabalho não há nenhuma menção sobre como o BERT poderia ser utilizado na tarefa de correferência. Isso abre espaço para um novo experimento: substituir a representação gerada pelo GloVe pela representação gerada pelo BERT. Como o objetivo deste trabalho é manter a utilização de

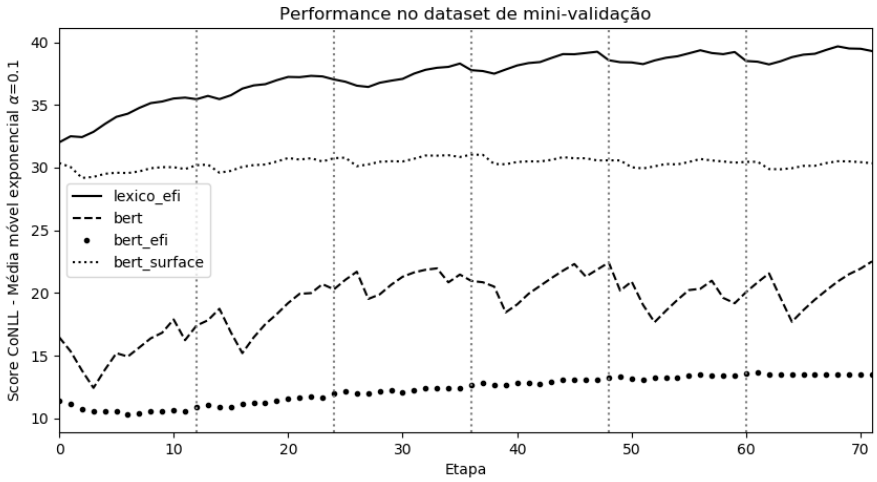


Figura 7.5: Desempenho do BERT

Código	MUC	B-CUBED	CEAF-E	Score CoNLL
bert_surface	24.99	30.88	25.51	27.13
bert	39.83	20.85	11.05	23.91
bert_efi	9.28	12.6	15.08	12.32

Tabela 7.4: Resultados dos modelos codificados com BERT no conjunto de validação

recursos computacionais baixos, não é feito nenhum tipo de ajuste fino no modelo disponibilizado por [12].

Assim como no teste com o GloVe, as features foram transformadas em categóricas e foram utilizadas as técnicas EFI e SURFACE para fazer a indução delas. Neste teste foi utilizado modelo *BERT-Base* disponiblizado por [12], que utiliza vetores de 768 posições. Cada menção é representada pelo vetor médio das palavras que a compõem.

Na figura 7.5 estão mostrados os resultados do treinamento das árvores latentes utilizando-se o BERT como codificador dos arcos.

Apesar de o BERT capturar o contexto e cada ocorrência de cada palavra ter a sua representação única, os resultados obtidos indicam que a utilização simples dessa codificação não é suficiente para resolver se um arco é ou não correferente. O que chama atenção nesse experimento é o fato de a indução SURFACE ser muito melhor do que a EFI, o que é contrário do que acontece com as features básicas e o GloVe.

Na tabela 7.4 estão mostrados os resultados dos modelos codificados pelo BERT. Comparando-se os valores desta tabela com os resultados das tabelas 7.2 e 7.3, é possível ver que os resultados desta arquitetura não são muito animadores.

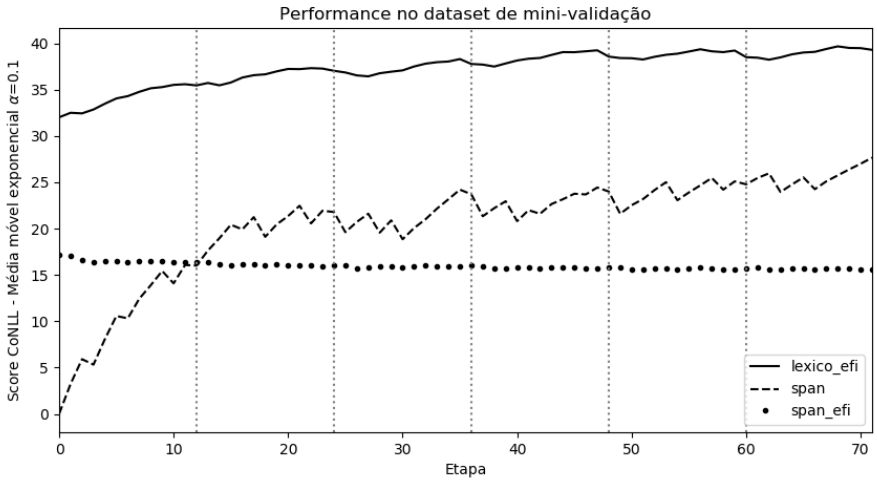


Figura 7.6: Desempenho do SpanBERT

7.4 Utilização de representação de trechos

Em [72] o autor cita que a forma que o BERT é treinado não é muito bom para a tarefa de correferência. Apesar de os autores mostrarem uma forma de integrar a arquitetura do BERT, que possui algumas centenas de milhões de parâmetros, em uma rede ainda maior e conseguirem um resultado muito bom, eles sugeriram uma forma diferente de treinar o BERT.

A nova arquitetura foi chamada de SpanBERT e foi apresentada em [14]. A grande diferença entre o SpanBERT e o BERT original é o fato de o SpanBERT já ser treinado para representar trechos de texto e não partes de palavras. A resolução de correferência era o principal motivador desta nova arquitetura e acabou gerando boas soluções em diversas outras tarefas que não haviam sido tratadas no trabalho original de [12].

O SpanBERT é uma forma diferente de se treinar o BERT. Ou seja, no final do treinamento, o modelo final possui os mesmos tipos de entradas e saídas que o BERT original. Para este último experimento ele recebeu o mesmo tratamento que o experimento do BERT. Ou seja, foi utilizado o modelo já disponibilizado por [14] e os arcos codificados com a menção sendo o vetor médio das palavras codificadas. Para este exemplo foi utilizado o *SpanBERT-Large*, que utiliza vetores de 1024 posições.

A figura 7.6 mostra o desempenho do SpanBERT se comparado às features léxicas. Apesar de ter um score melhor do que o BERT, o SpanBERT não atinge a marca do baseline. O resultado do EFI chama a atenção pela estabilidade. Logo nos primeiros exemplos o modelo converge para um patamar e, mesmo sendo treinado por mais épocas, ele não muda.

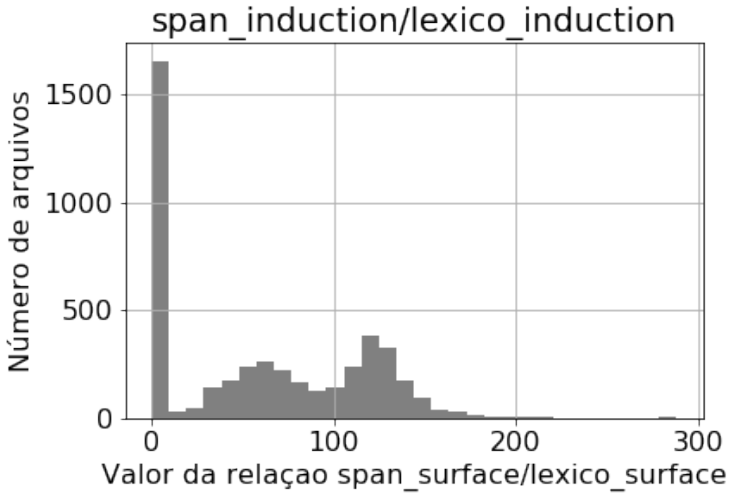


Figura 7.7: Comparação de tamanhos entre span_surface e lexico_surface

7.5 Modelo Final

A última tentativa explorada neste trabalho é aplicar a indução SURFACE ao SpanBERT. Essa solução gera uma quantidade muito grande de informação por arco. Com 1024 features básicas pode-se chegar a ter mais de 3000 bits ligados por arco. Isso significa uma quantidade de memória cerca de 100 vezes maior do que a utilização das features léxicas com a indução SURFACE. A figura 7.7 mostra como é a dispersão da relação dos tamanhos dos arcos induzidos pelo SURFACE utilizando o SpanBERT e as features léxicas.

Na prática isso significa um modelo muito mais caro computacionalmente. Porém, os resultados obtidos são melhores do que os modelos anteriores. A figura 7.8 mostra uma comparação deste modelo utilizando as features léxicas e também utilizando todas as features definidas de forma manual, ou seja, as que dependem de menções e as que dependem de pares de menções.

É possível ver que o desempenho deste modelo está no mesmo nível do modelo derivado por [20], que foi a base da implementação, e também de [19], que foi a base teórica.

Um fato relevante é o desempenho das duas formas de indução ser tão distinto. O EFI com o SpanBERT produz um resultado pior do que o modelo sem indução, enquanto a indução pelo SURFACE produz resultados comparáveis ao baseline. O motivo pode ser explicado pela tabela 7.5. Ela mostra uma busca no estilo *GridSearch* dos valores dos parâmetros da árvore de decisão utilizada para se construir o EFI. Cada coluna representa um possível valor de altura máxima e cada linha um valor possível de número máximo de nós. Para cada combinação desses dois parâmetros o conjunto de dados foi dividido em 5 partes. Em cada etapa foram utilizadas 4 partes dos

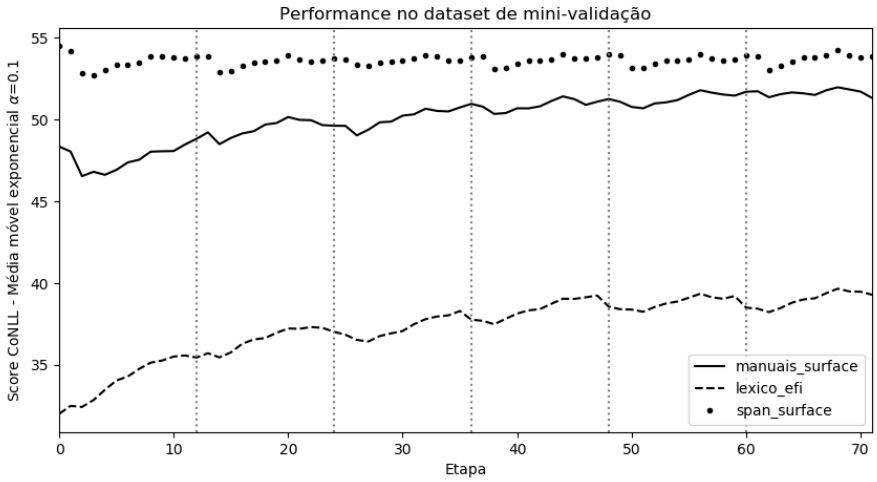


Figura 7.8: Perfomance do span_surface em comparação com as features manuais

dados para fazer um treinamento e na parte restante foi feita uma validação. Os valores da tabela são a média ± 1 desvio padrão dessas 5 etapas para a acurácia balanceada. Nessa métrica, o peso dos exemplos é ponderado pelo número de ocorrências de exemplos positivos e negativos. Desta forma, mesmo um conjunto de dados desbalanceado como este terá uma acurácia de 0.5 para um sistema que aleatoriamente escolhe entre 0 e 1 para cada exemplo.

Pelos resultados da tabela 7.5 é possível ver que a árvore de decisão não consegue aprender nenhuma informação útil dos dados codificados pelo SpanBERT e depois divididos em categorias. Neste caso, o EFI não terá a sua utilidade garantida, já que a entropia do sistema foi comprometida pelo pré-tratamento dos dados.

Uma modificação que pode ser feita em todos os modelos é otimizar os hiperparâmetros. Por exemplo, no modelo do SpanBERT foi utilizada a margem larga com o mesmo valor dos demais. Mas, como esse modelo quando utilizado com a indução do tipo SURFACE gera muito mais features do que os demais, é razoável procurar valores mais altos. Se ao invés do valor 2.000 for utilizado o valor 2.000.000, a performance do modelo melhora ainda mais, como mostrado na figura 7.9. Nela é possível ver o efeito da margem larga de forma evidente, ou seja, no início do treinamento o modelo erra muito mais, mas a cada etapa a melhora constante leva a um modelo melhor do que antes.

Com esses experimentos, os resultados finais obtidos estão mostrados na tabela 7.6. Nela são apresentados os resultados do melhor modelo de cada representação no dataset de validação. É importante notar que os valores aqui mostrados são inferiores aos reportados pelos demais da literatura por não serem modelos otimizados ao máximo. Como a motivação deste trabalho

nós \altura	5	10	20	30	Ilimitado
8	0.50 ±0.01	0.50 ±0.01	0.50 ±0.01	0.50 ±0.01	0.50 ±0.01
16	0.50 ±0.02	0.50 ±0.01	0.50 ±0.01	0.50 ±0.01	0.50 ±0.01
32	0.50 ±0.01	0.49 ±0.02	0.49 ±0.02	0.49 ±0.02	0.49 ±0.02
64	0.50 ±0.01	0.49 ±0.01	0.49 ±0.01	0.49 ±0.01	0.49 ±0.01
128	0.50 ±0.01	0.48 ±0.03	0.48 ±0.03	0.48 ±0.03	0.48 ±0.03
256	0.50 ±0.01	0.45 ±0.08	0.44 ±0.09	0.44 ±0.09	0.44 ±0.09
Ilimitado	0.50 ±0.01	0.44 ±0.09	0.35 ±0.24	0.33 ±0.28	0.33 ±0.28

Tabela 7.5: Resultado da validação cruzada da árvore de decisão para o SpanBERT

Cada coluna corresponde a uma altura máxima que a árvore poderia ter. Cada linha corresponde a um número máximo de nós que a árvore pode ter. Os valores exibidos são a média ± 1 desvio padrão para 5 rodadas de validação cruzada

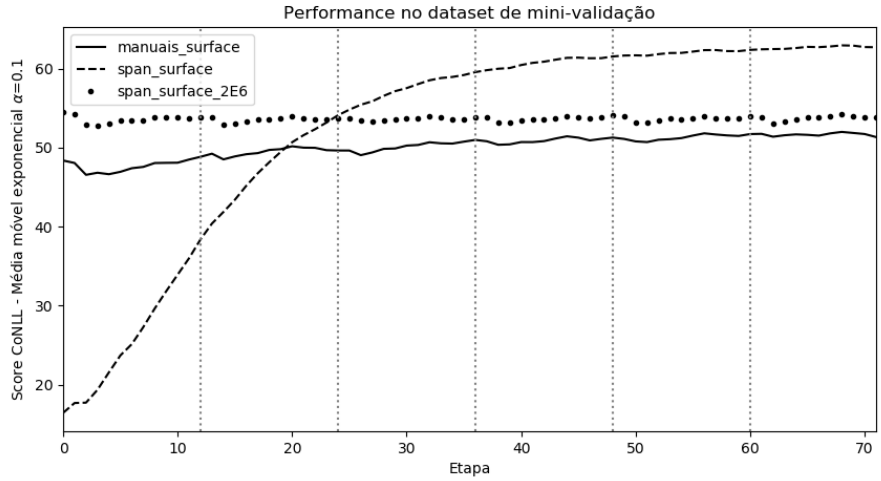


Figura 7.9: Perfomance do span_surface utilizando margem muito larga

é apenas comparar a eficiência da representação automática, a otimização completa dos modelos será tratada em trabalhos futuros. O modelo *baseline* é o mesmo utilizado por [20], no qual é reportado uma performance de 64.22 no score da CoNLL para o dataset de teste, porém sem o treino completo.

Na figura 7.10 são compilados os resultados da evolução da performance dos modelos da tabela 7.6 durante o treinamento. É possível perceber que a maioria dos modelos não sofre muitas mudanças ao longo do treinamento, exceto pelo modelo com margem muito larga.

7.6 Análise de Erros

No trabalho [20] é proposta uma ferramenta de análise de erros para correferência. Essa ferramenta é bem parecida com as métricas MUC e BLANC

Modelo	MUC	B-CUBED	CEAF-E	Score CoNLL
span_surface_2E6	61.72	52.04	45.89	53.2
span_surface	57.85	48.13	42.05	49.34
baseline	56.08	45.41	39.88	47.12
lexico_efi	48.89	37.01	31.07	38.99
glove_efi	44.23	33.95	28.78	35.65
bert_surface	24.99	30.88	25.51	27.13

Tabela 7.6: Resultado final de todos os modelos

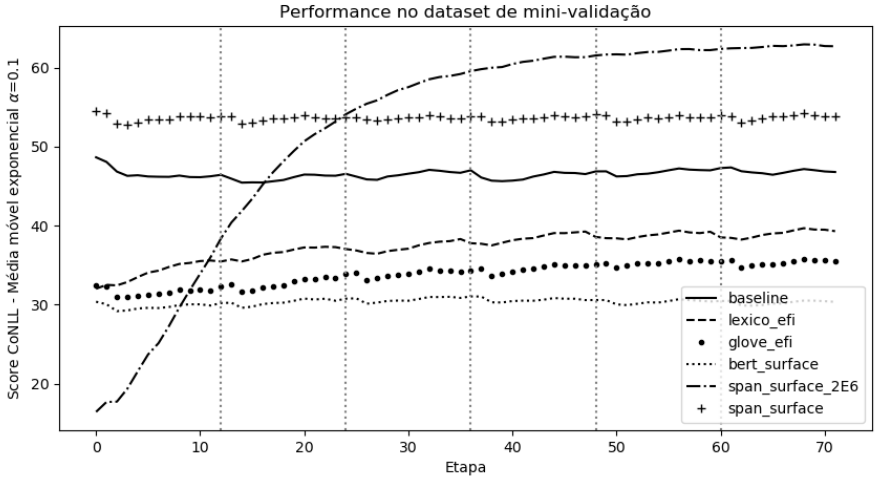


Figura 7.10: Comparativo da performance no dataset de mini validação dos modelos finais

utilizadas pela CoNLL. Porém, enquanto as métricas oficiais se focam em contabilizar certas ou erradas entre as menções, dada uma árvore de antecedentes qualquer, a ferramenta se propõe a analisar os erros apenas de arcos que poderiam ter sido escolhidos.

O recall é calculado como sendo o número de arcos que o sistema de correferência não encontrou e que tornariam uma entidade correta. A precisão é a calculada como o número de arcos que o sistema encontrou, mas que não estão na anotação do dataset.

Cada erro é classificado de acordo com o tipo da primeira menção que ocorre em cada grupo. As classes podem ser:

- **PRO**: Pronome
- **NAM**: Nomes Próprios
- **NOM**: Substantivos
- **DEM**: Pronome demonstrativo
- **VRB**: Verbos

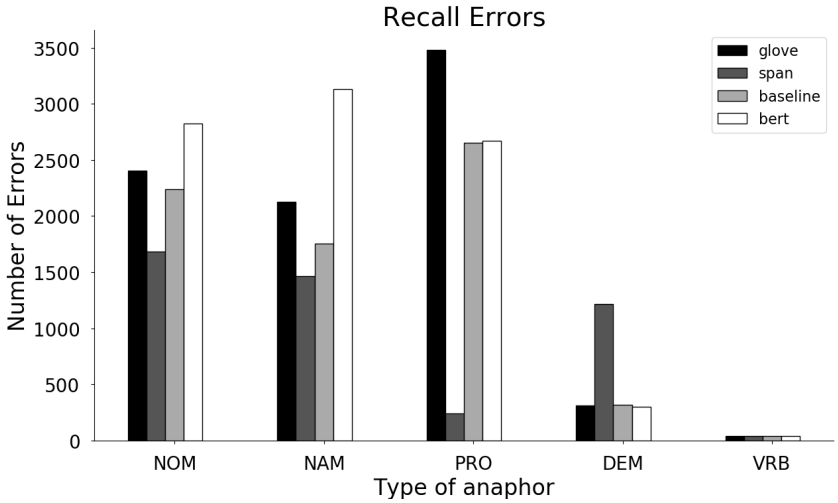


Figura 7.11: Erros de recall entre os modelos

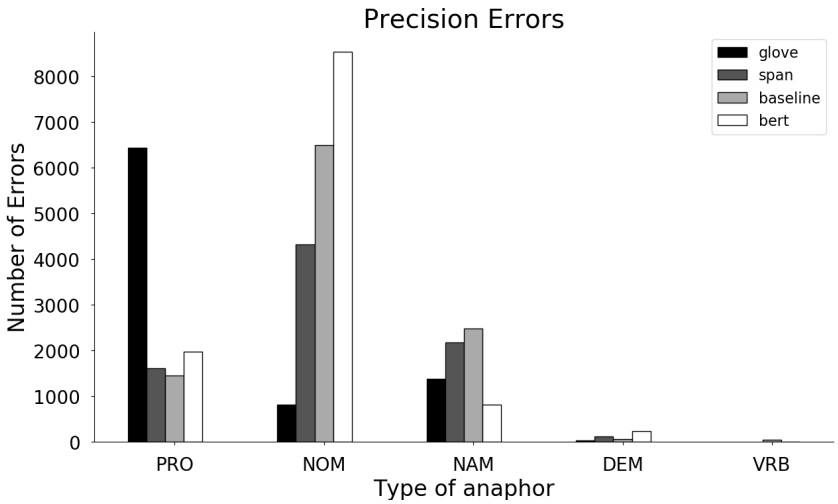


Figura 7.12: Erros de precisão entre os modelos

De acordo com essa classificação e definições de precisão e recall, as figuras 7.11 e 7.12 mostram os erros de recall e precisão que os melhores modelos de cada forma de se representar os arcos cometeeram. É importante notar que as figuras representam o número de erros de cada tipo, ou seja, quanto mais erros, pior o sistema.

Analisando os erros de recall, é possível notar que não há uma diferença muito grande entre os modelos apresentados, excetuando-se os erros cometidos quando as menções são do tipo nome próprio. Nesse caso, tanto as features básicas quanto o GloVe vão melhor do que o BERT, mas o SpanBERT é extremamente eficiente nestes casos. Como nomes próprios geralmente são o principal antecedente de uma menção, faz sentido um sistema que sempre tenha os mesmos valores para as mesmas palavras acertar mais esse tipo de relação.

	MUC	B-CUBED	CEAF-E	Score CoNLL
span_surface_2E6	70.59	59.42	53.77	61.26

Tabela 7.7: Performance do melhor modelo no dataset de teste

Quando cada ocorrência do nome possui sua própria representação, é preciso que exista uma combinação de pesos mais elaborada para se conseguir fazer essa distinção.

Os erros de precisão parecem ser mais distintos entre os sistemas. O GloVe comete muito poucos erros de nomes, enquanto comete muito mais erros de pronomes. Esse comportamento é esperado, já que todas as palavras possuem uma identificação única, os nomes devem ser facilmente identificados, enquanto pronomes, por aparecerem com frequência, acabem sendo confundidos, a depender da situação. Já o BERT possui o comportamento contrário. Muitos erros de nomes e poucos de pronomes. Esse resultado é compatível com a ideia de representação contextual. Como a representação dos pronomes está ligada ao seu contexto, é esperado que eles sejam mais bem identificados, enquanto que os nomes podem aparecer longe de seu contexto correto, sendo mais difícil a sua identificação.

Os modelos de SpanBERT e baseline são mais balanceados, sendo que o SpanBERT comete menos erros de nomes. Essa é uma indicação de que as features geradas automaticamente pelo SpanBERT possuem um poder de selecionar melhor qual pronome está ligado a qual nome.

7.7 Resultados no dataset de teste

O melhor modelo encontrado utilizando-se o dataset de validação teve a sua performance medida no dataset de teste. A tabela 7.7 mostra o desempenho em cada uma das métricas da CoNLL 2012 bem como a pontuação final. Como esperado, o valor de 61.26 no dataset de teste é acima do score de 53.2 obtido no dataset de validação.

Apesar de haver uma melhora no dataset de teste em relação ao dataset de treino, ela não foi tão grande quanto o modelo do baseline. Em [20] é reportada a performance de 64.22 no dataset de teste, enquanto que o modelo atingiu apenas 47.12 no dataset de validação

Comparando-se o resultado obtido com outros modelos, tem-se a tabela 7.8. O melhor modelo é o vencedor da CoNLL 2012 [19], com score 63.37. O segundo colocado nessa competição no dataset em inglês é o trabalho [140] com score 61.31. O modelo baseado no SpanBERT tem performance um pouco

Modelo	Score CoNLL
Fernandes e Milidiú [19]	63.37
Marschat [140]	61.31
span_surface_2E6	61.26
Lee, et. at. [35]	57.79

Tabela 7.8: Comparação com outros trabalhos

abaixo, com 61.26, acima do score de 57.79 do vencedor da CoNLL de 2011 [35].

É possível notar que, mesmo sem fazer os ajustes finos nos hiperparâmetros, o modelo que utiliza o SpanBERT consegue ter uma boa performance. Esse resultado é relevante pois mostra que é possível fazer a substituição de features manuais por features automáticas. Essa substituição permite uma menor dependência de especialistas, além de se poder criar modelos em diversas línguas sem que haja a necessidade de ajustá-lo a cada uma.

8
Conclusão e trabalhos futuros

A tarefa de resolução de correferência pode ser entendida como a tarefa de se detectar e agrupar as palavras de um texto que se referem a uma mesma entidade do mundo real. Essa tarefa é muito complexa pois nem sempre a informação necessária para resolvê-la está presente no texto original. Essa complexidade dificulta a geração de bases de dados anotados muito grandes para serem utilizadas por sistemas automatizados. Um dos maiores esforços nesse sentido foi a criação do OntoNotes, um conjunto de textos com diversas camadas de anotações retiradas de diversas fontes. O seu potencial foi explorado na CoNLL de 2011 e 2012 que trouxe a resolução de correferência como tarefa compartilhada.

Na edição de 2011 os melhores trabalhos eram apenas baseados em regras derivadas por especialistas em linguística, mas na edição seguinte os modelos foram além e começaram a surgir sistemas baseados em aprendizado de máquina que poderiam utilizar uma mesma base de dados e um mesmo conjunto de métricas para serem avaliados. O modelo vencedor desta competição foi o modelo proposto por [21] e se baseava em árvores latentes de antecedentes com suas características induzidas por entropia. Após esse modelo, se seguiram outros que utilizaram arquiteturas cada vez mais elaboradas de redes neurais.

Em paralelo ao desenvolvimento da resolução de correferência, a representação de textos teve um grande avanço com a utilização de aprendizado de máquina para se determinar qual a melhor forma de se transformar palavras em vetores de dimensões menores. Após os avanços trazidos pelo word2vec e GloVe, as representações contextuais implementadas pela técnica BERT criaram novos estado-da-arte em diversas tarefas.

As evoluções das representações contextuais e da correferência se unem na variação do BERT chamada SpanBERT. Essa variação foi criada para ser um elemento de uma solução de correferência que é o atual estado da arte. Seu tamanho e demanda computacional porém ultrapassa o suportado por computadores normais.

Apesar de essas histórias estarem hoje unidas, há uma lacuna: não foi encontrada na literatura uma aplicação dos atuais modelos de representação contextual em soluções mais simples, como a de árvores latentes. Este trabalho se propôs a preencher essa lacuna, tendo como objetivo verificar a eficiência de métodos automáticos de geração de features para a tarefa de correferência.

Neste trabalho apresentamos uma comparação entre quatro formas dis-

tintas de se representar os arcos de uma árvore latente de antecedentes. A primeira forma é a mesma já utilizada em outros trabalhos e usa features desenvolvidas manualmente por especialistas. A segunda forma usa o GloVe, uma técnica de representação de palavras de forma não-contextual. A terceira forma codifica os arcos utilizando o modelo original do BERT e na quarta forma é utilizada a sua versão mais atual, o SpanBERT.

Os experimentos mostram que as features manuais têm um bom poder preditivo para a resolução de correferência, ainda mais quando combinadas com técnicas de indução como o EFI e o SURFACE. No dataset de validação, os arcos codificados com GloVe e o BERT original não atingem o baseline de 47.12 de score CoNLL. O melhor modelo do GloVe é a variação com EFI com o score 35.65 e o melhor modelo com o BERT é a variação com o SURFACE, com 27.13. O melhor modelo é o SpanBERT com a indução utilizando SURFACE e uma margem muito larga, com o score 53.20. Este modelo atinge o score de 61.26 no dataset de teste, um pouco abaixo do vencedor da CoNLL de 2012 que obteve 63.37, mas acima do vencedor da CoNLL de 2011, que obteve 57.79.

A conclusão deste trabalho é a de que as representações contextuais são capazes de criar features automáticas para as árvores latentes na tarefa de correferência. Comparando-se as árvores latentes treinadas com features criadas por especialistas e aquelas treinadas com as features do SpanBERT, é possível ver que os modelos atingem resultados equivalentes. Ao se otimizar alguns hiperparâmetros do SpanBERT, é possível ver no dataset de validação que este modelo consegue uma performance até melhor do que o original. Apesar de o modelo com as features automáticas ter ficado com um score abaixo do modelo com as features manuais no dataset de teste, ainda assim a geração automatizada se mostra interessante. Isso porque a metodologia aqui aplicada não se restringe a uma língua e não há a necessidade de fazer engenharia de features, que pode ter um alto custo com pessoal.

8.1 Contribuições

Este trabalho traz duas contribuições: uma teórica e outra prática. Do ponto de vista teórico é mostrado que as representações contextuais podem ser utilizadas como uma forma automática de se representar arcos em uma árvore latente de antecedentes para se resolver a correferência de um texto. Essa contribuição é relevante pelo fato de que as soluções atuais que utilizam representações contextuais possuem requisitos computacionais muito elevados e a solução apresentada consegue ser resolvida com um servidor de capacidade padrão e sem a utilização de hardware especializado.

Do ponto de vista prático, este trabalho traz melhorias e atualizações a um framework de código aberto que permite explorar diversas famílias de soluções de correferência. Entre as melhorias foi implementado uma versão estocástica do perceptron estruturado, que permite utilizar representações bem maiores do que a versão original, e a adição da forma de indução EFI que não existia.

8.2 Trabalhos Futuros

Os modelos apresentados neste trabalho utilizam em sua grande maioria hiperparâmetros sub-ótimos. Muitos valores foram adotados da literatura e os demais são escolhidos a partir de uma busca com valores discretos, apenas para validar o comportamento geral de cada arquitetura.

Uma possibilidade de extensão deste trabalho é a otimização dos hiperparâmetros das arquiteturas adotadas. Em especial, é possível que se faça a otimização completa dos melhores modelos e se descubra qual a efetiva posição deles dentro do atual estado da arte.

Pelos experimentos notou-se que ainda há uma certa discricionariedade nas escolhas de como as features dos arcos são feitas. Decisões como a forma de representar as menções e qual tipo de indução são alguns desses exemplos. Uma automação completa desse processo poderia levar a resultados melhores. Um dos problemas envolvidos nesse tipo de solução seria como incorporar o perceptron estruturado em algum tipo de rede neural. Trabalhos como o de [141] já estão surgindo nessa direção e poderiam contribuir para um melhor desempenho dos modelos.

9
Referências bibliográficas

[1] TURING, A. M.. **Computing machinery and intelligence**. In: MACHINE INTELLIGENCE: PERSPECTIVES ON THE COMPUTATIONAL MODEL. 2012. 1

[2] BATES, M.. **Models of natural language understanding**. Proceedings of the National Academy of Sciences of the United States of America, 92(22):9977–9982, 10 1995. 1

[3] SONG, H.; RAJAN, D.; THIAGARAJAN, J. J. ; SPANIAS, A.. **Attend and diagnose: Clinical time series analysis using attention models**. 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, p. 4091–4098, 2018. 1

[4] SHIH, S. Y.; SUN, F. K. ; LEE, H. Y.. **Temporal pattern attention for multivariate time series forecasting**. Machine Learning, 108(8-9):1421–1441, 2019. 1

[5] HOCHREITER, S.; SCHMIDHUBER, J.. **Long Short-Term Memory**. Neural Computation, 1997. 1, 2.2.4, 2.3

[6] GRISHMAN, R.; SUNDHEIM, B.. **Message Understanding Conference-6**. In: PROCEEDINGS OF THE 16TH CONFERENCE ON COMPUTATIONAL LINGUISTICS -, volumen 1, p. 466, Morristown, NJ, USA, 1996. Association for Computational Linguistics. 1.1, 2.1

[7] PRADHAN, S.; RAMSHAW, L.; MARCUS, M.; PALMER, M.; WEISCHEDEL, R. ; XUE, N.. **CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes**. In: COMPUTATIONAL NATURAL LANGUAGE LEARNING (CONLL) SHARED TASK, p. 1–27, 2011. 1.1, 2.1, 3.3, 3.4, 3.5, 3.5

[8] PRADHAN, S.; MOSCHITTI, A. ; URYUPINA, O.. **CoNLL-2012 Shared Task : Modeling Multilingual Unrestricted Coreference in OntoNotes**. Conll, (June):1–40, 2012. 1.1, 2.1, 2.2.4, 3.3, 3.3, 3.4, 3.5, 4.2, 4.4

[9] LECUN, Y.; BOTTOU, L.; BENGIO, Y. ; HAFFNER, P.. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE, 1998. 1.1, 5

[10] BAHDANAU, D.; CHO, K. H. ; BENGIO, Y.. **Neural machine translation by jointly learning to align and translate**. In: 3RD INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, ICLR 2015 - CONFERENCE TRACK PROCEEDINGS, 2015. 1.1

[11] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, ; POLOSUKHIN, I.. **Attention is all you need**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, volumen 2017-Decem, p. 5999–6009, 6 2017. 1.1, 2.3, 5.3.2

[12] DEVLIN, J.; CHANG, M.-W.; LEE, K. ; TOUTANOVA, K.. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. 10 2018. 1.1, 2.2.4, 2.3, 5.3.2, 5.3.3, 6.1, 6.6.1, 6.6.1, 7.3, 7.4

[13] PETERS, M.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K. ; ZETTLEMOYER, L.. **Deep Contextualized Word Representations**. p. 2227–2237, 2 2018. 1.1, 2.2.4, 2.3, **??**, 5.3.1

[14] JOSHI, M.; CHEN, D.; LIU, Y.; WELD, D. S.; ZETTLEMOYER, L. ; LEVY, O.. **SpanBERT: Improving Pre-training by Representing and Predicting Spans**. 7 2019. 1.1, 2.1, 2.2.4, 2.3, **??**, 5.3.3, 6.1, 6.6.1, 6.6.1, 7.4

[15] WARSTADT, A.; SINGH, A. ; BOWMAN, S. R.. **Neural Network Acceptability Judgments**. Transactions of the Association for Computational Linguistics, 2019. 1.1

[16] ROSENTHAL, S.; FARRA, N. ; NAKOV, P.. **SemEval-2017 Task 4: Sentiment Analysis in Twitter**. 2018. 1.1

[17] SOCHER, R.; PERELYGIN, A.; WU, J. Y.; CHUANG, J.; MANNING, C. D.; NG, A. Y. ; POTTS, C.. **Recursive deep models for semantic compositionality over a sentiment treebank**. In: EMNLP 2013 - 2013 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE, 2013. 1.1

[18] RAJPURKAR, P.; ZHANG, J.; LOPYREV, K. ; LIANG, P.. **SQuad: 100,000+ questions for machine comprehension of text**. In: EMNLP 2016 - CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS, 2016. 1.1

[19] FERNANDES, E. R.; DOS SANTOS, C. N. ; MILIDIÚ, R. L.. **Latent trees for coreference resolution**. Computational Linguistics, 2014. 1.1, 1.2, 2.1, 2.2.4, 2.2.4, 3.2, 3.3, **??**, 3.6, 4, 4.1, 4.1, 4.2, 4.3, 4.3, 4.3, 4.4, 4.5, 4.5, 4.5, 4.5, 4.6, 4.6, 4.6, 9, 6.1, 6.1, 6.4, 6.5, 6.6, 6.6, 7.1, 7.2, 7.2, 7.2, 7.5, 7.7, **??**

[20] MARTSCHAT, S.. **Structured Representations for Coreference Resolution**. PhD thesis, 2017. 1.1, 1.2, 3, 3.1, 3.2, 3.2, 3.3, 4, 4.1, 4.2, 4.3, 4.3, 4.4, 4.5, 4.6, 9, 6.1, 6.1, 6.4, 6.5, 6.6.2, 7.1, 7.2, 7.2, 7.2, 7.5, 7.5, 7.6, 7.7

[21] FERNANDES, E. R.; DOS SANTOS, C. N. ; MILIDIÚ, R. L.. **Latent Structure Perceptron with Feature Induction for Unrestricted Coreference Resolution**. Proceedings of the Joint Conference on EMNLP and CoNLL: Shared Task, 2012. 1.2, 4.5, 8

[22] **Focusing in the Comprehension of Definite Anaphora**. In: COMPUTATIONAL MODELS OF DISCOURSE. The MIT Press, 1983. 2.1

[23] LAPPIN, S.; LEASS, J.. **Lappin, Leass - An Algorithm for Pronominal Anaphora Resolution**. 1994. 2.1

[24] SIDNER, C. L.. **Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse.** Massachusetts Inst of Tech Cambridge Artificial Intelligence lab, 1979. 2.1

[25] SUNDHEIM, B. M.. **Overview of results of the MUC-6 evaluation**. p. 423, 1996. 2.1, 3.3

[26] MCCARTHY, J. F.; LEHNERT, W. G.. **Using Decision Trees for Coreference Resolution**. 01, 1995. 2.1

[27] SOON, W.. **Corpus-based learning for noun phrase coreference resolution**. Science, p. 285–291, 1999. 2.1

[28] SOON, W. M.; LIM, D. C. Y. ; NG, H. T.. **A machine learning approach to coreference resolution of noun phrases**. Computational Linguistics, 2001. 2.1, 2.2.1

[29] DODDINGTON, G.; MITCHELL, A.; PRZYBOCKI, M.; RAMSHAW, L.; STRASSEL, S. ; WEISCHEDEL, R.. **The automatic content extraction (ACE) program tasks, data, and evaluation**. In: PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE

ON LANGUAGE RESOURCES AND EVALUATION, LREC 2004, 2004. 2.1

[30] YANG, X.; ZHOU, G.; SU, J. ; TAN, C. L.. **Coreference resolution using competition learning approach**. 2003. 2.1, 2.2.2

[31] LUO, X.; ITTYCHERIAH, A.; JING, H.; KAMBHATLA, N. ; ROUKOS, S.. **A mention-synchronous coreference resolution algorithm based on the Bell tree**. 2004. 2.1, 2.2.3, 2.2.3, 3.3

[32] CULOTTA, A.; WICK, M. ; MCCALLUM, A.. **First-order probabilistic models for coreference resolution**. In: NAACL HLT 2007 - HUMAN LANGUAGE TECHNOLOGIES 2007: THE CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, PROCEEDINGS OF THE MAIN CONFERENCE, 2007. 2.1, 2.2.3

[33] BENGTSON, E.; ROTH, D.. **Understanding the value of features for coreference resolution**. In: EMNLP 2008 - 2008 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE: A MEETING OF SIGDAT, A SPECIAL INTEREST GROUP OF THE ACL, 2008. 2.1, 2.2.1, 2.2.2, 9, 6.1

[34] NG, V.. **Supervised noun phrase coreference research: The first fifteen years**. In: ACL 2010 - 48TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, PROCEEDINGS OF THE CONFERENCE, número July, p. 1396–1411, 2010. 2.1

[35] LEE, H.; PEIRSMAN, Y.; CHANG, A.; CHAMBERS, N.; SURDEANU, M. ; JURAFSKY, D.. **Stanford ' s Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task**. In: PROCEEDINGS OF THE FIFTEENTH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING: SHARED TASK. ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 2011. 2.1, 4.2, **??**, 7.7

[36] DURRETT, G.; HALL, D. ; KLEIN, D.. **Decentralized entity-level modeling for coreference resolution**. In: ACL 2013 - 51ST ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, PROCEEDINGS OF THE CONFERENCE, 2013. 2.1

[37] BJÖRKELUND, A.; KUHN, J.. **Learning structured perceptrons for coreference resolution with latent antecedents and non-local features**. In: 52ND ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, ACL 2014 - PROCEEDINGS OF THE CONFERENCE, volumen 1, p. 47–57. Association for Computational Linguistics, 2014. 2.1, 2.2.4, 4.1, 4.2

[38] LEE, K.; HE, L.; LEWIS, M. ; ZETTLEMOYER, L.. **End-to-end neural coreference resolution**. In: EMNLP 2017 - CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS, p. 188–197, 2017. 2.1, 2.2.4, 2.2.4, 2.3, **??**, 6.1

[39] LEE, K.; HE, L. ; ZETTLEMOYER, L.. **Higher-Order Coreference Resolution with Coarse-to-Fine Inference**. p. 687–692, 4 2018. 2.1, 2.2.4, **??**, 4.1, 6.1, 6.6.1

[40] WANG, A.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O. ; BOWMAN, S. R.. **Glue: A multi-task benchmark and analysis platform for natural language understanding**. In: 7TH INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, ICLR 2019, 2019. 2.1

[41] WANG, A.; PRUKSACHATKUN, Y.; NANGIA, N.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O. ; BOWMAN, S. R.. **SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems**. 2019(July):1–30, 2019. 2.1, 2.2.4, 5.3.2

[42] CARDIE, C.; WAGSTAFF, K.. **Noun Phrase Coreference as Clustering**. In: PROCEEDINGS OF THE JOINT CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING AND VERY LARGE CORPORA, 1999. 2.2

[43] MOOSAVI, N. S.; STRUBE, M.. **Unsupervised coreference resolution by utilizing the most informative relations**. In: COLING 2014 - 25TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS, PROCEEDINGS OF COLING 2014: TECHNICAL PAPERS, 2014. 2.2

[44] CLARK, K.; MANNING, C. D.. **Deep reinforcement learning for mention-ranking coreference models**. In: EMNLP 2016 - CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS, p. 2256–2262, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics. 2.2

[45] FEI, H.; LI, X.; LI, D. ; LI, P.. **End-to-end Deep Reinforcement Learning Based Coreference Resolution**. In: PROCEEDINGS OF THE 57TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, p. 660–665, Stroudsburg, PA, USA, 2019. Association for Computational Linguistics. 2.2, **??**, 3.6

[46] QUINLAN, J. R.. **Induction of Decision Trees**. Machine Learning, 1986. 2.2.1, 4.5

[47] NG, V.; CARDIE, C.. **Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution**. p. 1–7, 2002. 2.2.1, 2.2.2

[48] RAHMAN, A.; NG, V.. **Narrowing the modeling gap: A cluster-ranking approach to coreference resolution**. Journal of Artificial Intelligence Research, 2011. 2.2.1

[49] STOYANOV, V.; CARDIE, C.; GILBERT, N.; RILOFF, E.; BUTTLER, D. ; HYSOM, D.. **Reconcile : A Coreference Resolution Research Platform**. Computing, 2010. 2.2.1

[50] HUNT, E.; MINSKY, M. ; PAPERT, S.. **Perceptrons**. The American Journal of Psychology, 1971. 2.2.1

[51] MILLER, G. A.. **WordNet: A Lexical Database for English**. Communications of the ACM, 1995. 2.2.1, 3.3

[52] CORTES, C.; VAPNIK, V.. **Support-Vector Networks**. Machine Learning, 1995. 2.2.1

[53] HOSTE, V.. **The Mention-Pair Model**. p. 269–282. 2016. 2.2.1

[54] HOFFART, J.; SUCHANEK, F. M.; BERBERICH, K.; LEWIS-KELHAM, E.; DE MELO, G. ; WEIKUM, G.. **YAGO2: Exploring and querying world knowledge in time, space, context, and many languages**. In: PROCEEDINGS OF THE 20TH INTERNATIONAL CONFERENCE COMPANION ON WORLD WIDE WEB, WWW 2011, 2011. 2.2.1

[55] BOLLACKER, K.; EVANS, C.; PARITOSH, P.; STURGE, T. ; TAYLOR, J.. **Freebase: A collaboratively created graph database for structuring human knowledge**. In: PROCEEDINGS OF THE ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2008. 2.2.1

[56] NASTASE, V.; STRUBE, M.; BÖRSCHINGER, B.; ZIRN, C. ; ELGHAFARI, A.. **WikiNet: A very large scale multi-lingual concept network**. In: PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON LANGUAGE RESOURCES AND EVALUATION, LREC 2010, 2010. 2.2.1

[57] PONZETTO, S. P.; STRUBE, M.. **Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution**. In: HLT-NAACL 2006 - HUMAN LANGUAGE TECHNOLOGY CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION OF COMPUTATIONAL LINGUISTICS, PROCEEDINGS OF THE MAIN CONFERENCE, 2006. 2.2.1

[58] DENIS, P.; BALDRIDGE, J.. **Specialized models and ranking for coreference resolution**. In: EMNLP 2008 - 2008 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE: A MEETING OF SIGDAT, A SPECIAL INTEREST GROUP OF THE ACL, 2008. 2.2.2

[59] CHANG, K.-W.; SAMDANI, R.; ROZOVSKAYA, A.; SAMMONS, M. ; ROTH, D.. **Illinois-Coref: The UI System in the CoNLL-2012 Shared Task**. CoNLL-Shared Task, 2012. 2.2.2, 9

[60] DURRETT, G.; KLEIN, D.. **Easy victories and uphill battles in coreference resolution**. In: EMNLP 2013 - 2013 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE, p. 1971–1982, 2013. 2.2.2, 4.5, 6.1, 6.6

[61] WISEMAN, S.; RUSH, A. M.; SHIEBER, S. M. ; WESTON, J.. **Learning anaphoricity and antecedent ranking features for coreference resolution**. In: ACL-IJCNLP 2015 - 53RD ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS AND THE 7TH INTERNATIONAL JOINT CONFERENCE ON NATURAL LANGUAGE PROCESSING OF THE ASIAN FEDERATION OF NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE, volumen 1, p. 1416–1426, Stroudsburg, PA, USA, 2015. Association for Computational Linguistics. 2.2.2

[62] DREYFUS, S. E.. **Artificial neural networks, back propagation, and the kelley-bryson gradient procedure**. Journal of Guidance, Control, and Dynamics, 1990. 2.2.2

[63] CLARK, K.; MANNING, C. D.. **Improving coreference resolution by learning entity-level distributed representations**. In: 54TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, ACL 2016 - LONG PAPERS, volumen 2, p. 643–653, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics. 2.2.3

[64] WISEMAN, S.; RUSH, A. M. ; SHIEBER, S. M.. **Learning global features for coreference resolution**. In: 2016 CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, NAACL HLT 2016 - PROCEEDINGS OF THE CONFERENCE, p. 994–1004, 4 2016. 2.2.3, **??**

[65] YU, C. N. J.; JOACHIMS, T.. **Learning structural svms with latent variables**. In: ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES, 2009. 2.2.4, 4.1

[66] CHANG, K. W.; SAMDANI, R. ; ROTH, D.. **A constrained latent variable model for coreference resolution**. In: EMNLP 2013 - 2013 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE, 2013. 2.2.4

[67] LASSALLE, E.; DENIS, P.. **Joint anaphoricity detection and coreference resolution with constrained latent structures**. In: PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2015. 2.2.4

[68] COLLINS, M.. **Discriminative training methods for hidden Markov models**. 2002. 2.2.4, 4.6

[69] SUN, X.; MATSUZAKI, T.; OKANOHARA, D. ; TSUJII, J.. **Latent variable perceptron algorithm for structured classification**. In: IJCAI INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, p. 1236–1242, 2009. 2.2.4

[70] ZAREMBA, W.; SUTSKEVER, I. ; VINYALS, O.. **Recurrent Neural Network Regularization**. (2013):1–8, 2014. 2.2.4

[71] SUTSKEVER, I.; VINYALS, O. ; LE, Q. V.. **Sequence to sequence learning with neural networks**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2014. 2.2.4

[72] JOSHI, M.; LEVY, O.; ZETTLEMOYER, L. ; WELD, D.. **BERT for Coreference Resolution: Baselines and Analysis**. p. 5802–5807, 8 2019. 2.2.4, **??**, 5.3.2, 6.1, 6.6.1, 7.4

[73] AL, R. W. E.. **OntoNotes Release 5.0 LDC2013T19**. Linguistic Data Consortium, 2013. 2.2.4, 6.3

[74] HARRIS, Z. S.. **Distributional Structure**. WORD, 1954. 2.3, 5.1, 5.2.2

[75] SZYMAŃSKI, J.. **Comparative analysis of text representation methods using classification**, 2014. 2.3

[76] BENGIO, Y.; DUCHARME, R.; VINCENT, P. ; JAUVIN, C.. **A neural probabilistic language model**. CrossRef Listing of Deleted DOIs, 1, 2000. 2.3, 2.3, 5.2, 5.2.2

[77] KATZ, S. M.. **Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer**, 1987. 2.3, 5.1

[78] P.˜BROWN; V.˜DELLA PIETRA; DE SOUZA, P.; J.˜LAI ; R.˜MERCER. **Class-based n-gram models of natural language**. Computational Linguistics, 1992. 2.3, 5.1

[79] GOODMAN, J. T.. **A bit of progress in language modeling**. Computer Speech and Language, 2001. 2.3, 5.1

[80] GOOD, I. J.. **The Population Frequencies of Species and the Estimation of Population Parameters**. Biometrika, 1953. 2.3

[81] LANDAUER, T. K.; DUMAIS, S. T.. **A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge**. Psychological Review, 1997. 2.3

[82] TIPPING, M. E.; BISHOP, C. M.. **Probabilistic principal component analysis**. Journal of the Royal Statistical Society. Series B: Statistical Methodology, 1999. 2.3

[83] RASTOGI, P.. **Representation Learning for Words and Entities**. 2019. 2.3

[84] PENNINGTON, J.; SOCHER, R. ; MANNING, C. D.. **GloVe: Global vectors for word representation**. In: EMNLP 2014 - 2014 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE, p. 1532–1543, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics. 2.3, 2.3, 5.2.3, 5.2.3, 5.2.3, 5.3

[85] MIKOLOV, T.; CHEN, K.; CORRADO, G. ; DEAN, J.. **Efficient estimation of word representations in vector space**. In: 1ST INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, ICLR 2013 - WORKSHOP TRACK PROCEEDINGS, 1 2013. 2.3, 2.3, 5.2.2

[86] HINTON, G. E.. **Learning distributed representations of concepts**, 1986. 2.3

[87] MIKOLOV, T.; KOPECKÝ, J.; BURGET, L.; GLEMBEK, O. ; ČERNOCKÝ, J. H.. **Neural network based language models for highly inflective languages**. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, p. 4725–4728, 2009. 2.3

[88] COLLOBERT, R.; WESTON, J.. **A unified architecture for natural language processing: Deep neural networks with multitask learning**. In: PROCEEDINGS OF THE 25TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2008. 2.3

[89] COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K. ; KUKSA, P.. **Natural Language Processing (Almost) from Scratch**. In: JOURNAL OF MACHINE LEARNING RESEARCH, 2011. 2.3

[90] TURIAN, J.; RATINOV, L. ; BENGIO, Y.. **Word representations: A simple and general method for semi-supervised learning**. In: ACL 2010 - 48TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, PROCEEDINGS OF THE CONFERENCE, 2010. 2.3

[91] MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. ; DEAN, J.. **Distributed representations ofwords and phrases and their compositionality**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2013. 2.3

[92] SUN, C.; QIU, X.; XU, Y. ; HUANG, X.. **How to Fine-Tune BERT for Text Classification?** In: LECTURE NOTES IN COMPUTER SCIENCE (INCLUDING SUBSERIES LECTURE NOTES IN ARTIFICIAL INTELLIGENCE AND LECTURE NOTES IN BIOINFORMATICS), volumen 11856 LNAI, p. 194–206, 2019. 2.3

[93] ZOPH, B.; VASUDEVAN, V.; SHLENS, J. ; LE, Q. V.. **Learning Transferable Architectures for Scalable Image Recognition**. In: PROCEEDINGS OF THE IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 8697–8710, 7 2018. 2.3

[94] MELAMUD, O.; GOLDBERGER, J. ; DAGAN, I.. **context2vec: Learning generic context embedding with bidirectional LSTM**. In: CONLL 2016 - 20TH SIGNLL CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING, PROCEEDINGS, 2016. 2.3

[95] PETERS, M. E.; AMMAR, W.; BHAGAVATULA, C. ; POWER, R.. **Semi-supervised sequence tagging with bidirectional language models**. ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 1:1756–1765, 2017. 2.3

[96] RADFORD, A.; SALIMANS, T.. **Improving Language Understanding by Generative Pre-Training**. OpenAI, p. 1–12, 2018. 2.3

[97] DAI, A. M.; LE, Q. V.. **Semi-supervised sequence learning**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2015. 2.3

[98] HOWARD, J.; RUDER, S.. **Universal language model fine-tuning for text classification**. ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 1:328–339, 2018. 2.3

[99] LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L. ; STOYANOV, V.. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. 7 2019. 2.3, 5.3.2, 5.3.3, 6.6.1

[100] LAN, Z.; CHEN, M.; GOODMAN, S.; GIMPEL, K.; SHARMA, P. ; SORICUT, R.. **ALBERT: A Lite BERT for Self-supervised Learning of Language Representations**. 9 2019. 2.3

[101] YANG, Z.; DAI, Z.; YANG, Y.; CARBONELL, J.; SALAKHUTDINOV, R. ; LE, Q. V.. **XLNet: Generalized Autoregressive Pretraining for Language Understanding**. 6 2019. 2.3, 5.3.3

[102] DAI, Z.; YANG, Z.; YANG, Y.; CARBONELL, J.; LE, Q. ; SALAKHUTDINOV, R.. **Transformer-XL: Attentive Language Models beyond a Fixed-Length Context**. 2019. 2.3, 5.3.3

[103] ECKERT, M.; STRUBE, M.. **Dialogue acts, synchronizing units, and anaphora resolution**. Journal of Semantics, 2000. 3

[104] CHEN, B.; SU, J.; PAN, S. J. ; TAN, C. L.. **A Unified Event Coreference Resolution by Integrating Multiple Resolvers**. In: PROCEEDINGS OF 5TH INTERNATIONAL JOINT CONFERENCE ON NATURAL LANGUAGE PROCESSING, 2011. 3

[105] KOLHATKAR, V.; HIRST, G.. **Resolving "This-issue"anaphora**. In: EMNLP-CONLL 2012 - 2012 JOINT CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING AND COMPUTATIONAL NATURAL LANGUAGE LEARNING, PROCEEDINGS OF THE CONFERENCE, 2012. 3

[106] POESIO, M.; MEHTA, R.; MAROUDAS, A. ; HITZEMAN, J.. **Learning to resolve bridging references**. 2004. 3

[107] HOU, Y.; MARKERT, K. ; STRUBE, M.. **Global inference for bridging anaphora resolution**. In: NAACL HLT 2013 - 2013 CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS: HUMAN LANGUAGE TECHNOLOGIES, PROCEEDINGS OF THE MAIN CONFERENCE, 2013. 3

[108] MARTSCHAT, S.; STRUBE, M.. **Latent Structures for Coreference Resolution**. Transactions of the Association for Computational Linguistics, 3:405–418, 2015. 3.1, 4.4, 6.2, 6.2, 6.6

[109] INGEMANN, F.; CRYSTAL, D.. **Dictionary of Linguistics and Phonetics**. Language, 74(1):230, 1998. 3.2, 3.2

[110] CHINCHOR, N.. **OVERVIEW OF MUC-7/MET-2 Nancy**. Proceedings of the 7ˆ{th} Message Understanding Conference (MUC7), 1998. 3.3

[111] HOVY, E.; MARCUS, M.; PALMER, M.; RAMSHAW, L. ; WEISCHEDEL, R.. **OntoNotes: the 90% solution**. In: HUMAN LANGUAGE TECHNOLOGY CONFERENCE OF THE NAACL, SHORT PAPERS, p. 57–60, 2006. 3.3, 3.4

[112] WEISCHEDEL, R.; HOVY, E.; MARCUS, M.; PALMER, M.; BELVIN, R.; PRADHAN, S.; RAMSHAW, L. ; XUE, N.. **OntoNotes: A large training corpus for enhanced processing**. Handbook of Natural Language Processing and Machine Translation. Springer, 2011. 3.3, 3.4

[113] BERGSMA, S.; LIN, D.. **Bootstrapping path-based pronoun resolution**. In: COLING/ACL 2006 - 21ST INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS AND 44TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, PROCEEDINGS OF THE CONFERENCE, 2006. 3.3, 4.3

[114] VILAIN, M.; BURGER, J.; ABERDEEN, J.; CONNOLLY, D. ; HIRSCHMAN, L.. **A model-theoretic coreference scoring scheme**. 1995. 3.3, 3.5

[115] BALDWIN, A. B. B.; BAGGA, A. ; BALDWIN, B.. **Algorithms for scoring coreference chain**. First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference, 1998. 3.3, 3.5

[116] RECASENS, M.; HOVY, E.. **BLANC: Implementing the Rand index for coreference evaluation**. Natural Language Engineering, 2011. 3.3, 3.5, 3.5

[117] MARCUS, M.; SANTORINI, B. ; MARCINKIEWICZ, M.. **Building a Large Annotated Corpus of English: The Penn Treebank**. Computational linguistics, 1993. 3.4

[118] CHEN, J.; PALMER, M.. **Towards robust high performance word sense disambiguation of english verbs using rich linguistic features**. In: LECTURE NOTES IN COMPUTER SCIENCE (INCLUDING SUBSERIES LECTURE NOTES IN ARTIFICIAL INTELLIGENCE AND LECTURE NOTES IN BIOINFORMATICS), 2005. 3.4

[119] LUO, X.. **On coreference resolution performance metrics**. In: HLT/EMNLP 2005 - HUMAN LANGUAGE TECHNOLOGY CONFERENCE AND CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, PROCEEDINGS OF THE CONFERENCE, 2005. 3.5

[120] RAND, W. M.. **Objective criteria for the evaluation of clustering methods**. Journal of the American Statistical Association, 1971. 3.5

[121] MOOSAVI, N. S.; STRUBE, M.. **Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric**. In: 54TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, ACL 2016 - LONG PAPERS, volumen 2, p. 632–642, Stroudsburg, PA, USA, 2016. Association for Computational Linguistics. 3.5

[122] KANTOR, B.; GLOBERSON, A.. **Coreference Resolution with Entity Equalization**. In: PROCEEDINGS OF THE 57TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, p. 673–677, Stroudsburg, PA, USA, 2019. Association for Computational Linguistics. **??**

[123] DOS SANTOS, C. N.; CARVALHO, D. L.. **Rule and Tree Ensembles for Unrestricted Coreference Resolution**. In: PROCEEDINGS OF THE FIFTEENTH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING: SHARED TASK, número June, p. 51–55, 2011. 4.2, 4.4

[124] MARTSCHAT, S.. **Multigraph Clustering for Unsupervised Coreference Resolution**. Acl 2013, (2012):81–88, 2013. 4.2

[125] LEE, H.; CHANG, A.; PEIRSMAN, Y.; CHAMBERS, N.; SURDEANU, M. ; JURAFSKY, D.. **Deterministic Coreference Resolution Based on Entity-Centric, Precision-Ranked Rules**. Computational Linguistics, 2013. 4.3, 4.3, 4

[126] ORÄSAN, C.; EVANS, R.. **NP animacy identification for anaphora resolution**. Journal of Artificial Intelligence Research, 2007. 5

[127] BJÖRKELUND, A.; FARKAS, R.. **Data-driven Multilingual Coreference Resolution using Resolver Stacking**. Joint Conference on EMNLP and CoNLL-Shared Task, 2012. 4.4

[128] FERNANDES, E. R.; MILIDIÚ, R. L.. **Entropy-guided feature generation for structured learning of Portuguese dependency parsing**. In: LECTURE NOTES IN COMPUTER SCIENCE (INCLUDING SUBSERIES LECTURE NOTES IN ARTIFICIAL INTELLIGENCE AND LECTURE NOTES IN BIOINFORMATICS), 2012. 4.5, 4.6

[129] DOS SANTOS, C. N.; MILIDIÚ, R. L.. **Entropy guided transformation learning**. In: SPRINGERBRIEFS IN COMPUTER SCIENCE, número 9781447129776, p. 9–21. 2012. 4.5

[130] TSOCHANTARIDIS, I.; JOACHIMS, T.; HOFMANN, T. ; ALTUN, Y.. **Large margin methods for structured and interdependent output variables**. Journal of Machine Learning Research, 2005. 4.6

[131] ROSENBLATT, F.. **The Perceptron - A Perceiving and Recognizing Automaton**, 1957. 4.6, 6.6.2

[132] EDMONDS, J.. **Optimum branchings**. Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics, 71B(4):233, 1967. 9

[133] ZHANG, X.; LECUN, Y.. **Text Understanding from Scratch**. 2015. 5

[134] PORTER, M. F.. **An algorithm for suffix stripping**, 1980. 5.1

[135] NAILI, M.; CHAIBI, A. H. ; BEN GHEZALA, H. H.. **Comparative study of word embedding methods in topic segmentation**. In: PROCEDIA COMPUTER SCIENCE, 2017. 5.2, 5.2.2, 5.2.3

[136] DEERWESTER, S.; DUMAIS, S. T.; FURNAS, G. W.; LANDAUER, T. K. ; HARSHMAN, R.. **Indexing by latent semantic analysis**. Journal of the American Society for Information Science, 41(6):391–407, 9 1990. 5.2.1

[137] FORSYTHE, G. E.; MALCOMN, M. A. ; MOLER, C. B.. **Least squares and the singular value decomposition**. In: COMPUTER METHODS FOR MATHEMATICAL COMPUTATIONS, chapter 9. Prentice Hall Professional Technical Reference, 1977. 5.2.1

[138] DE MARNEFFE, M.-C.; MANNING, C. D.. **The Stanford typed dependencies representation**. p. 1–8, 2008. 6.2

[139] KINGMA, D. P.; BA, J. L.. **Adam: A method for stochastic optimization**. In: 3RD INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, ICLR 2015 - CONFERENCE TRACK PROCEEDINGS, 12 2015. 6.6.2

[140] MARTSCHAT, S.; CAI, J.; BROSCHEIT, S.; MUJDRICZA-MAYDT, E. ; STRUBE, M.. **A Multigraph Model for Coreference Resolution**. In: PROCEEDINGS OF THE JOINT CONFERENCE ON EMNLP AND CONLL: SHARED TASK, 2012. 7.7, **??**

[141] MILIDIÚ, R. L.; ROCHA, R.. **Structured Prediction Networks through Latent Cost Learning**. In: PROCEEDINGS OF THE 2018 IEEE SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE, SSCI 2018, p. 645–649. IEEE, 11 2019. 8.2