Reconhecimento de implicação textual em português

Erick Rocha Fonseca

Tese de Doutorado do Programa de Pós-Graduação em Ciências de
Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Erick Rocha Fonseca

Reconhecimento de implicação textual em português

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Sandra Maria Aluísio

USP – São Carlos
Junho de 2018

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

F676r Fonseca, Erick Rocha
 Reconhecimento de implicação textual em português
 / Erick Rocha Fonseca; orientadora Sandra Maria
 Aluísio. -- São Carlos, 2018.
 160 p.

 Tese (Doutorado - Programa de Pós-Graduação em
 Ciências de Computação e Matemática Computacional) --
 Instituto de Ciências Matemáticas e de Computação,
 Universidade de São Paulo, 2018.

 1. Processamento de linguagem natural. 2.
 Reconhecimento de implicação textual. 3. Redes
 neurais. I. Aluísio, Sandra Maria, orient. II.
 Título.

Erick Rocha Fonseca

Recognizing textual entailment in Portuguese

Doctoral dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Sandra Maria Aluísio

USP – São Carlos
June 2018

AGRADECIMENTOS

Gostaria de agradecer às várias pessoas que estiveram próximas e me auxiliaram durante o caminho por vezes cansativo e amargo do doutorado.

A pesquisa em Processamento de Língua Natural, que faz máquinas agirem como se entendessem línguas humanas, me fascina há anos. Foi o que me trouxe ao NILC para o mestrado em 2011, e que continua me despertando interesse com seus novos desdobramentos. Tenho de agradecer aos professores do NILC, que criaram e mantêm vivo este grupo de pesquisa dedicado a um tema que me é muito caro: Graça Nunes, Sandra Aluísio, Thiago Pardo e Diego Amâncio. À Sandra, particularmente, agradeço por toda a dedicação à orientação e por estar muito presente.

No laboratório, vários amigos estiveram próximos durante ao menos algum tempo do doutorado, e lhes agradeço a companhia, as ideias e a ajuda prestada. Ficam também boas recordações das conferências para onde fomos juntos. Lembro em especial dos que por mais tempo estiveram comigo: Nathan, Fernando, Leandro, Christopher, Pedro, Lucas. Outros amigos, que pude ver com maior ou menor frequência, também foram importantes: Livy, Gustavo Zen, Fabiano Berardo, Roberto Gueleri. Também agradeço à Yasmim, que conheci apenas no fim da caminhada, mas que me deu muitas forças para terminá-la.

Agradeço também à minha mãe e ao meu pai, que sempre me deram forças e estimularam o estudo. E à minha irmã, que sempre esteve disponível para me ouvir e conversar sobre tantas coisas, trazendo sempre o humor das discussões fraternais.

Vorrei anche ringraziare gli amici che ho conosciuto in Italia, di tante origini diverse. L'anno che ci ho passato è stato molto piacevole grazie a loro. In particolare quelli con cui ho condiviso l'ufficio in FBK: Anna, Simone e Serra, con cui ho avuto tante conversazioni su tanti temi, e Bernardo Magnini, sempre molto amichevole. Ringrazio anche a tutti gli altri di FBK, a Aya, una cara amica con cui ho condiviso la casa, e i brasiliani con cui sono uscito tante volte.

Agradeço ainda a tantos anônimos que me ajudaram respondendo perguntas na Internet e disponibilizando ferramentas. Tento compensar o favor me juntando a eles e ajudando outras pessoas. Por fim, agradeço à CAPES e à FAPESP pelo apoio financeiro no doutorado, incluindo um ano de doutorado sanduíche na Itália.

“You see, there is a branch of human knowledge known as symbolic logic, which can be used to prune away all sorts of clogging deadwood that clutters up human language.”

— Isaac Asimov, Foundation

RESUMO

FONSECA, E. R. **Reconhecimento de implicação textual em português**. 2018. 160 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

O reconhecimento de implicação textual (RIT) consiste em identificar automaticamente se um trecho de texto em língua natural é verdadeiro baseado no conteúdo de outro. Este problema vem sendo estudado por pesquisadores da área de Processamento de Línguas Naturais (PLN) há alguns anos, e ganhou certo destaque mais recentemente, com a maior disponibilidade de dados anotados e desenvolvimento de métodos baseados em *deep learning*.

Esta pesquisa de doutorado teve como objetivo o desenvolvimento de recursos e métodos computacionais para o RIT, com especial foco em língua portuguesa. Durante sua realização, foi compilado o corpus ASSIN, o primeiro a fornecer dados para treinamento e avaliação de sistemas de RIT em português, e foi organizado o workshop de mesmo nome, que reuniu pesquisadores interessados no tema.

Além disso, foram feitos experimentos computacionais com diferentes tipos de estratégias para o RIT, com dados em inglês e em português. Foi desenvolvido um novo modelo para o RIT, o TEDIN (*Tree Edit Distance Network*). O modelo é baseado no conceito de distância de edição entre árvores sintáticas, já explorado em outros trabalhos de RIT. Seu diferencial é combinar a representação de conhecimento linguístico explícito com a flexibilidade e capacidade representativa de redes neurais. Foi também desenvolvido o Infernal, um modelo para RIT que usa técnicas clássicas de aprendizado de máquina com engenharia de atributos.

Os resultados experimentais do TEDIN ficaram abaixo de outros modelos da literatura, e uma análise cuidadosa de seu comportamento indica a dificuldade de se modelar as diferenças entre árvores sintáticas. Por outro lado, o Infernal teve resultados positivos no ASSIN, definindo o novo estado-da-arte para o RIT em português.

Palavras-chave: Processamento de Línguas Naturais, Reconhecimento de Implicação Textual, redes neurais.

ABSTRACT

FONSECA, E. R. **Recognizing textual entailment in Portuguese**. 2018. 160 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2018.

Recognizing Textual Entailment (RTE) consists in automatically identifying whether a text passage in natural language is true based on the content of another one. This problem has been studied in Natural Language Processing (NLP) for some years, and gained some prominence recently, with the availability of annotated data in larger quantities and the development of deep learning methods.

This doctoral research had the goal of developing resources and methods for RTE, especially for Portuguese. During its execution, the ASSIN corpus was compiled, which is the first to provide data for training and evaluating RTE systems in Portuguese, and the workshop with the same name was organized, gathering researchers interested in this theme.

Moreover, computational experiments were carried out with different techniques for RTE, with English and Portuguese data. A new RTE model, TEDIN (Tree Edit Distance Network), was developed. This model is based on the concept of syntactic tree edit distance, already explored in other RTE works. Its differential is to combine explicit linguistic knowledge representation with the flexibility and representative capacity of neural networks. An RTE model based on classical machine learning and feature engineering, Infernal, was also developed.

TEDIN had experimental results below other models from the literature, and a careful analysis of its behavior shows the difficulty of modelling differences between syntactic trees. On the other hand, Infernal had positive results on ASSIN, setting the new state-of-the-art for RTE in Portuguese.

Keywords: Natural Language Processing, Recognizing Textual Entailment, neural networks.

LISTA DE ILUSTRAÇÕES

Figura 1 – Duas sentenças diferentes com árvores de dependência semelhantes . . .	34
Figura 2 – Exemplo simples de TED	58
Figura 3 – Exemplo de TED com sentenças maiores	58
Figura 4 – Exemplo de um par de sentenças semelhante semanticamente, mas com diferenças sintáticas	59
Figura 5 – Duas árvores sintáticas cuja diferença apresenta o movimento de su- bárvore	60
Figura 6 – Exemplos de transformações do BIUTEE.	70
Figura 7 – Árvores sintáticas de duas sentenças	73
Figura 8 – Ângulos entre vetores. Observe-se que a normalização de vetores man- tém o mesmo ângulo entre os mesmos.	79
Figura 9 – Diferenças e produtos entre vetores normalizados	80
Figura 10 – Arquitetura do modelo BiMPM (imagem adaptada de Wang, Hamza e Florian (2017))	99
Figura 11 – Ilustração da arquitetura do TEDIN para o cálculo do custo de uma operação	117
Figura 12 – Treinamento do TEDIN para comparar pares positivos e negativos . . .	119
Figura 13 – Arquitetura do TEDIN para classificação	121
Figura 14 – Árvores de dependência de um par do conjunto de validação do ASSIN	125
Figura 15 – Árvores de dependência de um par do conjunto de validação do SNLI .	127
Figura 16 – Árvores de dependências das sentenças usadas para ilustrar a extração de atributos do Infernal.	133

LISTA DE TABELAS

Tabela 1	– Exemplo de par de RIT positivo e negativo	24
Tabela 2	– Acurácia e F_1 do sistema baseline, média e mediana dos participantes nas sete primeiras edições do RTE Challenges	40
Tabela 3	– Estatísticas sobre os conjuntos de dados de RIT	49
Tabela 4	– Operações de edição para o par mostrado na Figura 3	58
Tabela 5	– Operações de edição para o par mostrado na Figura 5	59
Tabela 6	– Sumário de sistemas baseados em similaridade usados nos RTE Challenges e ASSIN	74
Tabela 7	– Sumário de sistemas baseados em transformações textuais.	75
Tabela 8	– Comparação entre aprendizado de máquina clássico e <i>deep learning</i>	82
Tabela 9	– Sumário dos modelos analisados de redes neurais para RIT	101
Tabela 10	– Resumo das Diretrizes para Anotação	106
Tabela 11	– Estatísticas da Anotação. Os primeiros dois valores se referem à anotação de similaridade; os dois últimos valores à inferência.	107
Tabela 12	– Estatísticas de similaridade do ASSIN.	108
Tabela 13	– Estatísticas de inferência do ASSIN.	108
Tabela 14	– Resultados oficiais de todas as execuções para a tarefa de similaridade semântica	110
Tabela 15	– Resultados oficiais de todas as execuções para a tarefa de inferência textual	111
Tabela 16	– Performance do TEDIN, DFF e AdArte no ASSIN	123
Tabela 17	– Performance do TEDIN no SNLI	124
Tabela 18	– Performance do TEDIN no SICK	124
Tabela 19	– Sequência de edições encontrada pelo TEDIN	126
Tabela 20	– Transformações do TEDIN para uma sentença do SNLI	127
Tabela 21	– Sequências de letras substituídas ao se consultar o dicionário DELAF	129
Tabela 22	– Alinhamentos encontrados segundo o PPDB	132
Tabela 23	– Valores dos atributos do Infernal para o par da Figura 16	135
Tabela 24	– Performance do Infernal avaliado no ASSIN	138
Tabela 25	– Importância dos atributos do Infernal	139
Tabela 26	– Dificuldades para o Infernal no ASSIN	141

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Contextualização	23
1.2	Breve Histórico do Reconhecimento de Implicação Textual	24
1.3	Objetivos, Lacunas e Hipóteses da Pesquisa	26
1.4	Organização da Tese	27
2	O PROBLEMA DE RECONHECIMENTO DE IMPLICAÇÃO TEX- TUAL	29
2.1	Definição do Problema	29
2.2	Aplicações	31
2.3	Dificuldades da Tarefa	32
2.4	Avaliação	37
2.4.1	<i>Métodos Baselines</i>	37
2.5	Conjuntos de Dados	38
2.5.1	<i>RTE Challenges</i>	38
2.5.2	<i>SICK</i>	41
2.5.3	<i>SNLI</i>	44
2.5.4	<i>MultiNLI</i>	45
2.5.5	<i>Outros Conjuntos</i>	46
2.5.5.1	<i>SciTail</i>	46
2.5.5.2	<i>MPE</i>	47
2.5.5.3	<i>CLTE</i>	48
2.5.5.4	<i>JOCI</i>	48
2.6	Considerações Finais	49
3	ABORDAGENS CLÁSSICAS	51
3.1	Notação	51
3.2	Aprendizado de Máquina	52
3.3	Bases de Conhecimento para RIT	53
3.3.1	<i>Wordnet</i>	53
3.3.2	<i>PPDB</i>	54
3.3.3	<i>CatVar</i>	55
3.3.4	<i>DIRT</i>	55

3.3.5	VerbOcean	56
3.4	Distância de Edição de Árvores	57
3.4.1	Limitações	58
3.5	Abordagens de RIT	60
3.5.1	Similaridade e Alinhamento	60
3.5.1.1	EDITS	61
3.5.1.2	IKOMA (Sobreposição Lexical Ponderada)	62
3.5.1.3	UAIC (Sobreposição Lexical com Heurísticas)	62
3.5.1.4	PKUTM (Sobreposição de Nós em Árvore)	63
3.5.1.5	Similaridade em Duas Etapas	64
3.5.1.6	L2F/INESC-ID	65
3.5.1.7	ASAPP e Reciclagem	65
3.5.1.8	Blue Man Group	66
3.5.2	Lógica Formal	66
3.5.2.1	COGEX	66
3.5.3	Transformações Textuais	68
3.5.3.1	Probabilistically Sound Calculus	68
3.5.3.2	BIUTEE	69
3.5.3.3	Modelos de Edição de Árvore para RIT	71
3.5.3.4	AdArte	72
3.5.3.5	Operações sobre Subárvores e Metaheurísticas para TED	73
3.6	Considerações Finais	74
4	ABORDAGENS DE REPRESENTAÇÕES DISTRIBUÍDAS E REDES NEURAIIS	77
4.1	Motivação	77
4.1.1	Limitações da Engenharia de Atributos	77
4.1.2	Aprendizado de Máquina com Representações Distribuídas	78
4.1.2.1	Representação de Palavras	78
4.1.2.2	Modelos de Aprendizado de Máquina	81
4.2	Conceitos de Redes Neurais para RIT	82
4.2.1	Redes Feedforward	82
4.2.2	Redes Recorrentes	83
4.2.3	Atenção	85
4.2.4	Dropout	87
4.3	Arquiteturas Neurais para RIT	87
4.3.1	Codificação de Sentenças	87
4.3.1.1	LSTM simples	87
4.3.1.2	Neural Semantic Encoders	88
4.3.1.3	Auto-atenção direcional	90

4.3.1.4	<i>Gated Attention</i>	91
4.3.2	<i>Modelagem Conjunta de Sentenças</i>	92
4.3.2.1	<i>Atenção Simples sobre a Premissa</i>	92
4.3.2.2	<i>Redes Feedforward Modulares</i>	93
4.3.2.3	<i>Redes Recorrentes Modulares</i>	95
4.3.2.4	<i>Comparação Bilateral Multiperspectiva</i>	99
4.4	<i>Considerações Finais</i>	101
5	A AVALIAÇÃO DE SIMILARIDADE SEMÂNTICA E INFERÊNCIA TEXTUAL	103
5.1	<i>Criação do Corpus</i>	103
5.1.1	<i>O Conjunto de Dados</i>	103
5.1.2	<i>Criação do Corpus</i>	104
5.1.3	<i>Estatísticas da Anotação</i>	106
5.2	<i>A Avaliação Conjunta</i>	109
5.2.1	<i>Baselines</i>	109
5.2.2	<i>Participantes</i>	109
5.2.3	<i>Resultados</i>	110
5.3	<i>Publicações após a Avaliação Conjunta</i>	111
5.4	<i>Considerações Finais</i>	112
6	MÉTODOS PROPOSTOS PARA RIT	115
6.1	<i>TEDIN</i>	115
6.1.1	<i>Pré-processamento</i>	116
6.1.2	<i>Cálculo de TED</i>	116
6.1.2.1	<i>Treinamento</i>	116
6.1.3	<i>Classificação para RIT</i>	119
6.1.4	<i>Avaliação</i>	121
6.1.4.1	<i>ASSIN</i>	121
6.1.4.2	<i>SNLI</i>	123
6.1.4.3	<i>SICK</i>	124
6.1.5	<i>Análise de Transformações</i>	125
6.2	<i>Infernal</i>	128
6.2.1	<i>Pré-processamento</i>	129
6.2.2	<i>Atributos</i>	131
6.2.3	<i>Avaliação</i>	136
6.2.4	<i>Relevância de Atributos</i>	138
6.2.5	<i>Análise de Erros</i>	140
6.3	<i>Considerações Finais</i>	143

7	CONCLUSÕES	145
7.1	Revisão da Literatura	145
7.2	Criação de Recursos para RIT em Português	145
7.3	Abordagens Computacionais para RIT	146
7.3.1	<i>TEDIN</i>	146
7.3.2	<i>Infernal</i>	146
7.3.3	<i>Outros Modelos Neurais</i>	146
7.4	Trabalhos Futuros	147
7.4.1	<i>Corpora para RIT</i>	147
7.4.2	<i>Dificuldades para RIT</i>	147
7.4.3	<i>TEDIN</i>	148
7.5	Produção Técnica	148
7.6	Produção Acadêmica	149
	REFERÊNCIAS	151

INTRODUÇÃO

1.1 Contextualização

O reconhecimento de implicação textual (RIT), ou inferência de linguagem natural, é uma tarefa dentro do Processamento de Línguas Naturais (PLN) que consiste em determinar se o significado de uma dada passagem de texto está contido, ou implicado, por uma outra passagem (DAGAN *et al.*, 2013). Realizar esta tarefa corretamente é de grande interesse para aplicações de PLN que lidam com textos de várias fontes: por exemplo, um sistema de perguntas e respostas pode checar se uma resposta candidata a responder uma pergunta é embasada por uma base de conhecimento textual (HARABAGIU; HICKL, 2006), ou se já foi respondida previamente (ABACHA; DINA, 2016); sistemas de sumarização podem remover do sumário uma sentença que já seja implicada pelo resto dele (GUPTA *et al.*, 2014).

A definição exata de RIT mais difundida na literatura é dada por Dagan, Glickman e Magnini (2006), que definem implicação textual como uma relação direcional entre um par de passagens, chamadas *texto* (T) e *hipótese* (H). Considera-se que T implica H se um humano, após ler T (e tomá-lo como verdade), diria que H também é verdade¹. Apesar de esta definição ser um pouco subjetiva, a anotação de corpora para RIT mostrou que é possível atingir concordância considerável entre julgamentos humanos (DAGAN *et al.*, 2013). A Tabela 1 mostra um exemplo de um par com relação de implicação e outro sem.

Além da implicação, outras relações entre pares de sentenças comumente exploradas são a paráfrase (quando ambas têm o mesmo conteúdo, havendo uma relação bidirecional de implicação) e contradição (quando ambas as sentenças não podem ser verdadeiras ao mesmo tempo). Assim como a implicação, para se detectar estes tipos de relação é

¹ Bowman *et al.* (2015) introduziram uma nomenclatura alternativa para o problema, chamando as sentenças envolvidas de *premise* e *hypothesis*.

Implicação?	Texto	Hipótese
Sim	Derrota em casa elimina holandeses do maior torneio de seleções da Europa.	A seleção holandesa está fora do maior torneio de seleções da Europa.
Não	De acordo com a PM, por volta das 10h30 havia 2 mil militantes no local.	O protesto encerrou por volta de 12h15 (horário local).

Tabela 1 – Exemplo de par de RIT positivo e negativo

preciso comparar o conteúdo das sentenças e verificar se se referem às mesmas coisas. Alguns corpora para o treinamento de sistemas de RIT apresentam pares de sentenças classificados quanto a mais de duas possíveis relações.

Realizar o RIT de forma precisa envolve muitas dificuldades, relativas tanto à linguagem humana em si como ao conhecimento de mundo em geral. O conhecimento linguístico é necessário para o entendimento dos textos analisados, para que se possa identificar eventos e atributos mencionados em cada um e compará-los.

Classicamente, uma forma de se prover o conhecimento linguístico é em uma etapa de pré-processamento, com técnicas de PLN tais como parsing sintático e semântico, resolução de correferências e detecção de entidades nomeadas (DAGAN *et al.*, 2013). Mais recentemente, estratégias baseadas em *deep learning* têm aberto mão desse tipo de pré-processamento, pois seus modelos são capazes de aprender os padrões linguísticos conforme são treinados para resolver a tarefa de RIT.

Já o problema de conhecimento de mundo, por outro lado, é muito mais difícil. Este envolve praticamente qualquer coisa que um ser humano normalmente saiba: relações familiares, conhecimentos geográficos, funcionamento em geral da sociedade, entre outros. Não há uma solução eficiente para tratar deste problema, e os trabalhos de RIT costumam focar na parte linguística do problema (MARELLI *et al.*, 2014b; BOWMAN *et al.*, 2015).

1.2 Breve Histórico do Reconhecimento de Implicação Textual

As pesquisas na área de RIT são relativamente recentes, com o primeiro evento dedicado exclusivamente ao assunto e uma avaliação padronizada introduzidos em 2005 (DAGAN; GLICKMAN; MAGNINI, 2006) — as *RTE Challenges*. Antes disso, embora já houvesse interesse de pesquisadores de PLN no assunto, os estudos sobre RIT eram direcionados a aplicações específicas. Com o estabelecimento da RIT como uma área avaliada independentemente, houve uma evolução em seu tratamento computacional.

As RTE Challenges tiveram sete edições anuais (BENTIVOGLI *et al.*, 2011), com cada uma fornecendo um novo conjunto de dados e algumas delas introduzindo novas particularidades (como o tratamento de textos de várias sentenças). Os participantes da avaliação exploraram diferentes estratégias em seus sistemas, mas não surgiu nenhuma abordagem que se destacasse claramente como a melhor (DAGAN *et al.*, 2013). Notou-se que a maioria dos participantes, incluindo os que obtiveram os melhores resultados nas avaliações, usavam estratégias bastante superficiais, incapazes de atacar algumas das dificuldades conhecidas do RIT.

Com o advento do *deep learning* ganhando espaço em PLN a partir de meados da década de 2010 (GOLDBERG, 2016), o problema de RIT teve um renascimento com a criação de novos corpora com estilos diferentes das RTE Challenges; a saber, o SICK (MARELLI *et al.*, 2014b) e o SNLI (BOWMAN *et al.*, 2015). O primeiro foi projetado como um meio de avaliar métodos de composicionalidade; isto é, métodos que combinavam unidades linguísticas como palavras em estruturas mais complexas, como sintagmas e orações, até chegar a uma representação de um texto.

O SNLI foi mais impactante para a pesquisa com redes neurais, pois buscou viabilizar o treinamento de modelos com uma grande quantidade de parâmetros, que precisam de grandes quantidades de dados de treino. Além disso, em geral, os seus pares não são tão simples quanto os do SICK, o que aumenta a dificuldade de classificá-los independente da abordagem. Novos métodos baseados em redes neurais têm alcançado resultados cada vez melhores na classificação de pares do SNLI desde sua publicação².

Mais recentemente, foi lançado o corpus MultiNLI (WILLIAMS; NANGIA; BOWMAN, 2017) com o propósito de usar o RIT como uma métrica para avaliar representações vetoriais de sentenças, produzidas por redes neurais. O corpus foi usado em uma avaliação conjunta que exigia que sistemas participantes criassem representações vetoriais independentes para cada sentença, e posteriormente classificassem a relação entre ambas. Além disso, o MultiNLI conta com sentenças de diversos gêneros, como transcrições de conversas ou trechos de guias de viagens. Seu uso estimulou o desenvolvimento de métodos que pudessem lidar bem com estas condições.

Visando trazer a pesquisa em RIT para mais perto de uma aplicação real, o corpus SciTail (KHOT; SABHARWAL; CLARK, 2018) foi recentemente publicado, simulando um cenário de QA (*Question Answering*, ou respostas a perguntas). Os pares do SciTail equivalem a determinar se uma possível resposta a uma dada pergunta é justificada por uma base de conhecimento. Por ter sido lançado pouco tempo antes do tempo de escrita desta tese, ainda não se conhecem pesquisas de RIT com o SciTail, de modo que ainda não se pode determinar seu impacto para a área.

² A listagem atualizada com resultados de modelos publicados no SNLI está disponível em <<https://nlp.stanford.edu/projects/snli/>>

Os conjuntos de dados supracitados são usados como *benchmarks* para o RIT em inglês. Há também conjuntos usados para outras línguas, como em italiano (BOS; ZANZOTTO; PENNACCHIOTTI, 2009) e alemão (ZELLER; PADÓ, 2013), além de abordagens *cross*-linguísticas, em que cada sentença do par está em uma língua (NEGRI *et al.*, 2012). Para o português, existe o ASSIN (FONSECA *et al.*, 2016), criado no escopo do presente projeto.

1.3 Objetivos, Lacunas e Hipóteses da Pesquisa

Esta pesquisa de doutorado partiu do objetivo geral de desenvolver recursos e métodos computacionais para o RIT, com especial foco em língua portuguesa. Seus objetivos pontuais foram os seguintes:

- Desenvolver recursos para possibilitar a pesquisa de RIT em português, especialmente um corpus anotado. Este objetivo foi cumprido com a compilação do corpus ASSIN, divulgado em uma avaliação conjunta de mesmo nome (FONSECA *et al.*, 2016).
- Desenvolver métodos que explorassem estruturas sintáticas das sentenças.

Uma das conclusões obtidas das análises dos trabalhos sobre o ASSIN foi a grande correlação da sobreposição de palavras nas duas sentenças com a presença de implicação. No entanto, sabe-se pela própria definição da tarefa que apenas palavras em comum não são suficientes para o RIT. Buscou-se entender quais outras propriedades linguísticas do par poderiam melhorar a automação da tarefa. Em particular, estruturas sintáticas não foram usadas pelos participantes do ASSIN, e portanto foi decidido explorá-las em sistemas de RIT nesta pesquisa.

O desenvolvimento de métodos para RIT começou com a definição de um conjunto de atributos, aperfeiçoado ao longo da pesquisa, que originou o Infernal, descrito no Capítulo 6. Este modelo foi motivado pela hipótese de que um classificador automático se beneficiaria de atributos referentes às estruturas sintáticas das sentenças envolvidas.

Outra estratégia explorada na pesquisa se baseou em uma forma específica de comparação das estruturas sintáticas de duas sentenças, a distância de edição de árvores (ou *tree edit distance*, TED, explicada no Capítulo 3). A TED teve aplicações razoavelmente bem-sucedidas no RIT, mas carece de flexibilidade para a definição de parâmetros importantes. Mais especificamente, a TED calcula a distância entre duas árvores sintáticas como a soma dos custos de se adicionar, remover ou substituir palavras. Para tal, é necessário que o custo de cada alteração seja previamente definido, o que é particularmente difícil e pouco tratado na literatura.

Formulou-se então a hipótese, mais forte do que a anterior, de que uma forma flexível de determinar custos de edições para TED melhoraria sua performance em geral quando aplicada para RIT. Para testar esta hipótese, foi desenvolvido o TEDIN (*Tree Edit Distance Network*), um modelo de rede neural para o cálculo do custo de cada operação de edição usada pela TED. O TEDIN produz representações vetoriais para cada edição sintática (como a inserção de uma determinada palavra ou substituição de uma por outra), e calcula seu custo por meio de camadas de rede neural.

Em experimentos realizados, a segunda hipótese não se confirmou. O cálculo de TED feito pelo TEDIN muitas vezes não parece fazer sentido, e sua performance em RIT ficou abaixo de outros modelos. Avaliado no ASSIN, atingiu medida F_1 de 0,5, valor abaixo de outros modelos neurais e próximo de outra abordagem mais simples de TED. Já o Infernal, por outro lado, teve bom desempenho, atingindo F_1 de 0,72 no ASSIN e estabelecendo um novo estado-da-arte.

1.4 Organização da Tese

Os próximos capítulo desta tese se organizam da seguinte forma. São apresentados os conceitos de Implicação Textual mais formalmente no Capítulo 2, além de dificuldades inerentes à sua automatização. São também apresentados os conjuntos de dados desenvolvidos para a tarefa, que ditam como é tratada no PLN. Abordagens para o RIT baseadas em técnicas clássicas de aprendizado de máquina são apresentadas no Capítulo 3, bem como conceitos de aprendizado de máquina e de PLN relevantes para seu entendimento. O Capítulo 4 traz conceitos de *deep learning* e representações distribuídas, e trabalhos da literatura que seguem esta linha para executar o RIT.

As contribuições deste trabalho de doutorado vêm em seguida. O ASSIN, nome do corpus para RIT em português e avaliação conjunta, é descrito no Capítulo 5. Modelos propostos para resolver a tarefa são apresentados no Capítulo 6, junto com seus resultados em experimentos. Por fim, o Capítulo 7 traz as conclusões da tese.

O PROBLEMA DE RECONHECIMENTO DE IMPLICAÇÃO TEXTUAL

Neste capítulo, é apresentada uma visão geral do problema de RIT, partindo da sua definição teórica. São mostradas também aplicações práticas de RIT, e dificuldades de realizá-lo corretamente, especialmente ao se lidar com sentenças mais complexas e envolvendo conceitos abstratos.

Apresenta-se também a forma com que é tratado o RIT na prática pela comunidade de PLN. São apresentados os conjuntos de dados compilados para servirem de insumo a sistemas de RIT, suas particularidades e dificuldades, além de métricas de avaliação para os sistemas que realizam a tarefa. Em última instância, estes conjuntos de dados são o que determina o limite da capacidade de sistemas automatizados de realizar o RIT corretamente; portanto, é importante entender como abordam o problema.

2.1 Definição do Problema

Informalmente, o reconhecimento de implicação textual (RIT) é o processo de detectar quando, dado um par $\langle T, H \rangle$, a interpretação de T (texto) permite que se conclua H (hipótese) como verdadeiro (DAGAN; GLICKMAN; MAGNINI, 2006; DAGAN *et al.*, 2013). O par (1) exemplifica um caso de implicação textual, que pode ser expresso por $T \Rightarrow H$.

- (1) a. Chuvas fortes causaram o atraso de vários voos na tarde de ontem.
- b. Ontem, choveu bastante.

Uma definição formal de acarretamento encontrada na linguística é que $T \Rightarrow H$ se e somente se H for verdadeiro em todas as circunstâncias (ou possíveis mundos) em que T for

verdadeiro (CHIERCHIA; MCCONNELL-GINET, 2000). Nesta definição, o par acima ainda contém uma relação de acarretamento, mas o mesmo não se aplica ao par (2):

- (2) a. O embaixador se recuperou da lesão após uma cirurgia bem-sucedida.
- b. O embaixador passa bem.

Ainda assim, para a maioria dos propósitos em aplicações práticas, dever-se-ia considerar que (2-a) $\Rightarrow$ (2-b). Na comunidade de PLN, portanto, é comum se adotar uma definição menos formal e mais subjetiva para a implicação: diz-se que $T \Rightarrow H$ se uma pessoa, após ler T , inferiria que H é muito provavelmente verdade (DAGAN *et al.*, 2009).

Essa definição, apesar de formulada de maneira simples, depende de pelo menos duas premissas, também apontadas por Dagan *et al.* (2009): (i) a pessoa avaliando o par toma T como verdadeiro e (ii) diferentes pessoas lendo T e H compartilham conhecimentos de mundo e da língua em que os textos são formulados. Caso contrário, algumas pessoas poderiam não compreender a relação entre ambos.

Quanto à primeira premissa, avaliar a veracidade de T está fora do escopo da área de RIT; simplesmente assume-se que esse é verdadeiro. Quanto à segunda, normalmente não há preocupação dos trabalhos da área, dado que o conhecimento de mundo exigido por pares usados em avaliações (apresentadas na Seção 2.5) é bastante simples para o entendimento de humanos, embora possam ser muito difíceis de serem reconhecidos por máquinas, como é explicado adiante.

Outra diferença, de natureza teórica, entre a visão clássica da linguística sobre acarretamento e a usada em PLN é mencionada por Dagan *et al.* (2013): a definição clássica considera também *tautologias*, isto é, hipóteses que sempre são verdadeiras sem a necessidade de um texto que as embase. Por exemplo, a afirmação *O céu é azul* é uma tautologia, podendo ser inferida como verdade simplesmente pelo conhecimento de mundo.

Já em PLN, só se considera um caso de implicação textual quando o conteúdo de T é essencial para que se possa concluir que H é verdadeiro, mesmo que apenas T não seja suficiente. No caso de uma tautologia, como no exemplo dado, apenas uma consulta a uma base de conhecimentos seria suficiente para determinar a veracidade de H (BENTIVOGLI *et al.*, 2011).

Um caso particular da implicação textual é a paráfrase, que ocorre quando há uma relação de acarretamento bidirecional entre dois textos, denotado por $T \Leftrightarrow H$. O par (3) exemplifica um caso de paráfrase.

- (3) a. O projeto não é de interesse dos acionistas da empresa.
- b. Os acionistas da empresa não estão interessados no projeto.

Por fim, um ponto relacionado à pesquisa em RIT é a detecção de *contradição*, que corresponde a casos em que dois trechos de textos contêm informações conflitantes, de modo que não podem ambos ser verdade ao mesmo tempo. Pode-se denotar a contradição por $T \Rightarrow \neg H$ ¹. Um exemplo de contradição é mostrado no par (4), extraído e traduzido do SNLI (BOWMAN *et al.*, 2015). Segundo as definições de anotação do corpus, quando as duas sentenças contiverem elementos que possam se referir à mesma entidade, deve-se assumir que este é o caso — como no exemplo, *famílias* e *pessoas*. Dessa forma, diminui-se a ambiguidade e se possibilita que haja mais casos de contradição.

- (4) a. Famílias estão esperando sua vez de entrar em um parque de diversões.
 b. Pessoas estão esperando na fila de um restaurante.

Quanto à nomenclatura, é interessante notar que a partir da publicação do SNLI, difundiu-se em inglês o uso do termo *premise* para se referir ao primeiro componente de um par avaliado quanto ao RIT. Por conta disso, e pela palavra *premissa* evitar ambiguidades em certos contextos, são usados nesta tese *texto* e *premissa* de forma intercambiável para se referir a este elemento.

2.2 Aplicações

Aplicações de RIT podem ser encontradas em várias áreas de PLN. Particularmente, QA (*Question Answering*, ou Respostas a Perguntas) se beneficia de forma bastante direta, havendo métodos de RIT desenvolvidos especificamente para esta área (ANDROUTSOPOULOS; MALAKASIOTIS, 2010). Isto se deve à possibilidade de se validar respostas para uma determinada pergunta por meio de uma base de conhecimento textual. Com efeito, o corpus SciTail (KHOT; SABHARWAL; CLARK, 2018) explora este tipo de relação, contendo pares de sentenças criados a partir de respostas candidatas a uma pergunta e trechos de uma base de conhecimento.

Na área de sumarização automática, em que trechos de um ou mais documentos são coletados para montar um resumo de seu conteúdo, métodos de RIT são interessantes por poderem verificar diferentes trechos que têm o mesmo significado, e assim incluir no resultado apenas um deles (GUPTA *et al.*, 2014). Na compressão de sentenças, uma subetapa da sumarização, é interessante substituir sentenças por paráfrases que ocupem menos espaço (ZHAO *et al.*, 2009).

Em extração de informação, área que trabalha com a extração de informação estruturada a partir de textos em língua natural, é comum a busca por alguns padrões de texto definidos manualmente para se encontrar informação pronta para ser estruturada.

¹ Este tipo de relação não deve ser confundido com casos neutros, em que T ser verdadeiro ou falso não implica nada quanto a H , relação denotada por $T \nRightarrow H$.

Por exemplo, poder-se-ia usar o padrão *X escreveu o livro Y* para se descobrir nomes de autores e seus respectivos livros. O uso de técnicas de RIT para extração de pares relacionados é uma melhoria bem direta para este tipo de técnica (SHINYAMA; SEKINE, 2003).

Outro exemplo de aplicação em que detecção de RIT é útil é a detecção de plágio (BARRÓN-CEDENO *et al.*, 2013). Neste caso, a aplicação é bastante direta: busca-se determinar se um texto suspeito de plágio tem o mesmo conteúdo de um original, escrito de forma diferente.

Por fim, o próprio RIT pode ser usado como uma forma de avaliar a capacidade de modelos computacionais de entender a linguagem humana, dado que exige um bom entendimento das duas sentenças (WILLIAMS; NANGIA; BOWMAN, 2017). Uma forma de fazer com que esse tipo de avaliação realmente tenha um caráter genérico e não seja específico da tarefa é exigir que seja gerada uma representação (normalmente matricial ou vetorial) de cada sentença como parte do processo de decisão; idealmente, estas mesmas representações poderiam ser usadas para outras tarefas. Esta foi uma das motivações da organização do MultiNLI (descrito na Seção 2.5.4).

2.3 Dificuldades da Tarefa

A detecção de implicação textual exige entendimento linguístico e de mundo em geral, algo bastante difícil de ser automatizado. Para atacar esta dificuldade, é comum o uso de recursos como dicionários, ontologias, regras de transformação, entre outros, que constituem o que é chamado *conhecimento de background*.

Dagan *et al.* (2013) afirmam que a aquisição de conhecimento de *background* necessário para resolver casos difíceis é um dos maiores gargalos na pesquisa em RIT. As análises apresentadas a seguir exemplificam e quantificam os problemas.

Clark *et al.* (2007) fazem uma análise de uma amostra de 100 pares do conjunto RTE-3 (descrito na Seção 2.5.1) para determinar que tipo de conhecimento é necessário para identificar as implicações. Os autores listam 13 categorias de conhecimento, que podem ser agrupadas nos seguintes cinco grupos gerais:

1. Alterações sintáticas, como alterações de voz passiva/ativa, remoção de adjetivos, orações subordinadas, etc. Com o uso de ferramentas como um parser sintático, o RIT nesses casos é relativamente fácil.

Por exemplo, considere-se o par de sentenças ilustradas na Figura 1. As únicas diferenças entre elas são a mudança de voz ativa para passiva e o adjetivo *jovem* que aparece somente na primeira, e poderiam ser tratadas da seguinte forma. A etiqueta *auxpass* indica um verbo auxiliar para formar voz passiva (outros formalismos de

anotação usariam etiquetas diferentes), o que permite que um sistema de RIT verificar a variação de voz em relação à segunda sentença. Além disso, o modificador indicado pela etiqueta *amod* poderia ser descartado ainda mantendo a implicação na sentença resultante.

2. Alterações lexicais. Essa categoria pode incluir a substituição de uma palavra por um sinônimo ou hiperônimo (conceito mais abrangente), a substituição de expressões equivalentes, ou a remoção de algumas palavras (por exemplo, *processo de produção de X* por *produção de X*). No caso de substituições, recursos lexicais como tesouros e ontologias podem ser suficientes para a detecção; casos de remoção podem ser mais difíceis de tratar.
3. Conhecimento de mundo. Essa categoria abriga uma vasta gama de noções, normalmente triviais para humanos, necessárias para se compreender certos eventos. Por exemplo, leitores humanos reconhecem que *Ursos podem matar pessoas* $\Rightarrow$ *Ursos atacam pessoas*, ou que *Durante o julgamento de X por assassinato (...)* $\Rightarrow$ *X foi acusado de assassinato*. Isso se deve ao entendimento de como um urso poderia matar alguém, ou das condições em que um julgamento acontece.

No primeiro caso, este tipo de conhecimento não se resume a uma simples relação lexical entre *matar* e *atacar*, pois há contextos em que a relação não se manifesta: *Pistolas podem matar pessoas* $\nRightarrow$ *Pistolas atacam pessoas*. De fato, não é de se esperar que um recurso lexical liste *atacar* como ação implicada por *matar*.

O segundo caso é ainda mais difícil, pois as estruturas sintáticas das orações não são semelhantes. O processo de entendimento passa por inferir, a partir da menção do julgamento, que *X foi julgado*, e em seguida que *X foi acusado*. O segundo passo pode ser resolvido por uma relação lexical razoavelmente simples, mas o primeiro apresenta uma dificuldade maior.

4. Reconhecer expressões que implicam na ocorrência de um evento, expressando-o de forma indireta. Por exemplo, *Y conseguiu X*, *Foi confirmado que X*, *Y disse que X*. Nessas expressões, pode-se tomar X como um novo fato (embora o problema da veracidade seja agravado em alguns casos).
5. Expressões idiomáticas que não possuem significado literal. Em muitos casos, podem ser tratadas como casos de substituição multipalavras (por exemplo, *abrir mão* $\Rightarrow$ *desistir*), mas exigem também que o sistema de RIT não faça uma interpretação literal do significado, pois o levaria a tirar conclusões erradas.

Clark *et al.* (2007) mostram que a maioria dos pares necessita de algum tipo de conhecimento de mundo para que possa ser corretamente classificada, o que impõe um

Figura 1 – Duas sentenças diferentes com árvores de dependência semelhantes

limite superior razoavelmente baixo para a performance de sistemas baseados apenas em estruturas sintáticas e recursos lexicais.

LoBue e Yates (2011) fazem uma análise semelhante à de Clark *et al.* (2007) no corpus do RTE-5. Os autores consideraram os pares que exibiam relação de implicação ou contradição (também anotado no conjunto de dados) e cuja identificação não poderia ser obtida apenas com conhecimento linguístico. Foi feita uma análise de 108 pares e, para cada um, descritos os passos de inferência necessários para se chegar à conclusão final. Em seguida, o conhecimento requerido para tal foi organizado em 20 categorias.

Algumas delas se referem a conhecimentos acessíveis em bases de dados: por exemplo, há uma categoria para geografia, que inclui saber que Austrália é um país ou que Sydney é uma cidade na Austrália. Outros exemplos de categorias são relações de todo-parte (por exemplo, saber que uma floresta tem árvores) ou nomes de figuras públicas (como chefes de estado).

Já outras categorias necessitam de tratamento mais cuidadoso para serem compreendidas corretamente. Por exemplo, há a categoria aritmética, que inclui casos em que é necessário somar, subtrair ou ainda arredondar certos valores. Tal tipo de raciocínio não é comumente explorado em sistemas de PLN.

A análise mostra que o conhecimento geográfico é o mais comumente necessário para RIT (aparecendo em 16,5% da amostra analisada), enquanto cada uma das outras categorias corresponde a menos de 9% da amostra. Ou seja, não há um único tipo de conhecimento suficiente para resolver a maior parte dos casos de RIT.

No caso de pares que envolvem textos com mais de uma sentença, há ainda a análise com respeito à coesão textual. A coesão textual é o que une uma sequência de sentenças e parágrafos para formar um texto que possa ser entendido como um todo (Koch, 2013). Um mecanismo comum da coesão textual é a referência a alguma entidade ou evento mencionado anteriormente no texto por meio de diferentes palavras, para evitar a repetição. A seguir são mostrados alguns exemplos de pares $\langle T, H \rangle$ que apresentam esse

fenômeno (expressões que fazem referência entre si em **negrito**):

- **Casamentos homossexuais** são permitidos na Espanha, Holanda e Bélgica. **Tais uniões** também são legais em seis províncias canadenses e no estado norte-americano de Massachusetts. $\Rightarrow$ Massachusetts permite **casamentos homossexuais**.
- A **China** busca soluções para a segurança em suas minas de carvão. **Um acidente recente** custou a vida de mais de uma dúzia de mineiros. $\Rightarrow$ **Um acidente em uma mina na China** matou vários mineiros.
- **Elizabeth II** é a monarca reinante do Reino Unido desde 1952. **Ela** foi proclamada rainha em seis de fevereiro de 1952, em seguida à morte de **seu** pai, George VI. $\Rightarrow$ O pai de **Elizabeth Segunda** foi George VI.

Esse tipo de correferência (duas ou mais expressões, não necessariamente iguais, que se referem ao mesmo conceito) é mais um problema que precisa ser tratado por sistemas de RIT. [Mirkin, Dagan e Padó \(2010\)](#) apresentam uma análise quantitativa de sua importância, analisando 120 pares do conjunto de desenvolvimento do RTE-5.

Os autores anotaram manualmente os casos de correferência cuja resolução era necessária para se chegar à resposta correta. O estudo revelou que, para 44% dos pares, a resolução de referência era obrigatória, ou seja, não seria possível detectar a relação de implicação caso não se conhecessem as entidades envolvidas. Para outros 28%, embora não obrigatória, a referência discursiva poderia ajudar a detecção. Além disso, 27% dos pares analisados continham mais de uma referência.

Os autores apontam também que alguns estudos sobre RIT usaram ferramentas para a resolução automática de anáforas como parte do pré-processamento. No entanto, não se chegou a uma conclusão sobre os benefícios trazidos: houve casos com ganhos mínimos de performance ou mesmo de piora, devido a erros introduzidos pelas ferramentas ([BAR-HAIM et al., 2009](#)).

A explicação para o funcionamento de sistemas de RIT e paráfrase sem a capacidade de resolução de correferência é que raramente se chega a uma compreensão profunda do texto que pudesse demonstrar a prova de inferência formalmente. As estratégias gerais para RIT são mostradas nos Capítulos 3 e 4.

[Dagan et al. \(2013\)](#) listam uma série de conhecimentos e capacidades de interpretação necessários para a RIT. Os autores não contabilizam ocorrências em corpus, apenas dão breves descrições e exemplos de cada fenômeno. Apesar de bem extensa e detalhada, a lista delineada pelos autores não fica clara em alguns pontos, pois alguns dos fenômenos descritos são muito semelhantes entre si. A lista apresenta quatro categorias principais, listadas a seguir.

Comparação Esta categoria é definida, de forma bastante abstrata, como um conjunto de fenômenos que surgem ao se comparar duas sentenças. Inclui fenômenos lexicais como a sinonímia e antonímia, e também implicações escalares (como determinar a maioria de um grupo quando se menciona algo como 70%) e papéis semânticos. Não ficou clara a conexão entre todos estes conceitos sob o rótulo dado.

Elipse Inclui diversos tipos de elipse, ou seja, construções linguísticas em que um termo que pode ser subentendido é omitido. É interessante notar que, ao se trabalhar com português, há mais casos de elipse do que em inglês, como em sujeitos ocultos (*O pesquisador escreve bastante. ~~O pesquisador~~ está prestes a publicar um artigo.*) e substantivos subentendidos em adjuntos (*O primeiro artigo tem muitas citações, mas o segundo ~~artigo~~ tem poucas ~~citações~~.*).

Interpretação Esta categoria inclui diversos fenômenos relativos à interpretação do significado de um texto. Por exemplo, há o entendimento de expressões idiomáticas, resolução correta de anáforas, metáforas, entendimento de quando um fato é relatado como apenas possível ou hipotético etc. Novamente, há algumas particularidades quanto ao idioma: a categoria lista o entendimento correto do pronome *it* em inglês quando não tem referente (cumprindo apenas função de sujeito). Além disso, menciona a compreensão da extensão temporal de eventos baseado no tempo verbal, o que funciona de forma diferente em português, dadas as diferenças nos tempos verbais.

Conhecimento de Mundo Os autores dividem esta grande categoria em outras duas: conceitos centrais, que são recorrentes em muitas situações, e domínios populares, em que o conhecimento associado não é generalizável, mas que ocorrem com frequência.

Na primeira subcategoria, incluem fenômenos como raciocínio espacial, temporal, numérico, entendimento de causa e correlação, etc. Já na segunda, são descritos de forma genérica conceitos como parentesco, comércio, competições, esportes, entre outros.

As dificuldades apontadas nesta seção alertam para a necessidade do uso de ferramentas robustas de PLN para processar os textos com os quais se pretende trabalhar. Dagan *et al.* (2009) destacam que o uso de boas ferramentas e bons recursos foi fundamental para a obtenção de bons resultados nos melhores sistemas participantes dos eventos RTE Challenges.

2.4 Avaliação

Para avaliar sistemas de RIT, usam-se métricas típicas de problemas de classificação, como acurácia, precisão, cobertura e F_1 . O sistema deve classificar pares em um conjunto (chamado *conjunto de teste*), para os quais se conhece a classificação correta, mas que não foi visto pelo sistema em seu treinamento. As definições das métricas usadas são as seguintes:

Acurácia é a proporção de pares classificados corretamente.

Precisão diz respeito a uma categoria em particular (como implicação, contradição ou neutro). É a proporção de pares corretamente classificados pelo sistema como pertencendo a uma dada categoria.

Cobertura também diz respeito a uma categoria em particular. É a proporção de pares pertencentes a uma dada categoria que foi corretamente classificada pelo sistema.

F_1 é a média harmônica entre precisão (P) e cobertura (C):

$$F_1 = \frac{2 \cdot P \cdot C}{P + C}$$

A F_1 de um sistema com respeito a todas as classes pode ser agregada pela média aritmética. A F_1 é especialmente útil em casos em que uma ou poucas classes sejam muito mais comuns que as demais. Nestes casos, um sistema enviesado para responder com a classe mais comum terá uma acurácia alta, mas uma baixa capacidade real de distinguir ocorrências das classes mais raras. A performance de um sistema desse tipo, medida pela F_1 , seria mais baixa que sua acurácia, o que fornece uma avaliação mais realista.

Por outro lado, quando um conjunto de dados é bem balanceado (isto é, possui a mesma quantidade de pares de todas as classes), apenas a acurácia já é um bom indicador de performance. Este é o caso com o SNLI e o MultiNLI (descritos nas Seções 2.5.3 e 2.5.4). Mesmo assim, em alguns conjuntos não balanceados é praxe usar a acurácia como única métrica de performance, como acontece com o SICK e o SciTail (descritos nas Seções 2.5.2 e 2.5.5.1). Já no ASSIN (descrito no Capítulo 5), que também conta com classes desbalanceadas, usou-se a F_1 como métrica padrão.

2.4.1 Métodos Baselines

Um *baseline* é um método razoavelmente simples para resolver algum problema computacional. Diferentes problemas têm baselines diferentes entre si, e um mesmo problema pode ter mais de um baseline. O seu propósito normalmente não é ser de fato uma solução, mas servir de comparação com métodos mais complexos, para avaliar se a

complexidade de fato é vantajosa para a resolução do problema. Para RIT, se baseiam em verificar de forma simplificada as semelhanças entre T e H .

No extremo mais simples, pode ser estimado um limiar de decisão baseado na proporção de palavras em comum entre as duas sentenças. Este foi o método usado nos RTE Challenges, descritos na Seção 2.5.1.

Com o SNLI, descrito na Seção 2.5.3, foram usados dois baselines mais elaborados: um classificador linear com atributos comumente empregados na literatura e outro classificador treinado também com atributos lexicais (explicado mais detalhadamente no Capítulo 3). Apesar de serem mais complexos do que comumente escolhido para baselines, seu uso se justifica por serem fundamentalmente diferentes dos métodos que se esperavam usar no SNLI.

No ASSIN, descrito no Capítulo 5, foram usados dois baselines. Um classificava todos os pares com a classe majoritária enquanto o outro foi um classificador linear treinado apenas com a proporção de palavras em comum das duas sentenças.

2.5 Conjuntos de Dados

A seguir, são listados os principais conjuntos de dados elaborados para treinamento e avaliação de sistemas para RIT em inglês: os RTE Challenges, o SICK, o SNLI e o MultiNLI. Além dos conjuntos de dados em si, são descritas as particularidades do RIT em cada um dos casos, bem como uma visão geral dos resultados de avaliações que os usaram.

2.5.1 RTE Challenges

Os eventos *Recognizing Textual Entailment (RTE) Challenge* aconteceram de 2004 a 2013, visando a avaliação de técnicas de RIT². O surgimento do evento foi motivado pelo fato de haver interesse de pesquisadores de diversas áreas de PLN em RIT, como QA e sumarização automática, que muitas vezes desenvolviam métodos específicos para suas aplicações. Nesse cenário, os avanços obtidos frequentemente ficavam restritos às comunidades específicas (DAGAN *et al.*, 2009).

Em cada edição dos RTE Challenges, foi disponibilizado um corpus com pares $\langle T, H \rangle$ e a indicação de se apresentam relação de implicação. Além disso, cada edição focou em algum diferente aspecto para a RIT, introduzindo novidades na tarefa. A seguir, apresenta-se um resumo do que foi proposto em cada edição do evento.

RTE-1 A primeira edição do evento apresentou pares coletados manualmente e separados

² <https://aclweb.org/aclwiki/Recognizing_Textual_Entailment>

em categorias de acordo com o cenário em que sua identificação seria útil, como QA, tradução automática, recuperação de informação, extração de informação, entre outros.

RTE-2 A segunda edição usou sentenças tiradas da saída de sistemas de PLN (como um trecho de sumarização ou uma informação pontual extraída do texto), de modo a refletir um cenário mais realista para a aplicação de RIT.

RTE-3 A terceira edição contou com parágrafos compostos de várias sentenças no componente T de alguns dos pares.

RTE-4 A quarta edição trouxe a classificação de alguns pares como contradição.

RTE-5 A quinta edição contou com textos mais longos e sem pré-edição, de modo que poderiam conter erros ortográficos e gramaticais.

RTE-6 Na sexta edição, em vez de classificar pares independentes, foi proposto classificar a relação de diversas passagens de texto retornadas por um motor de busca para uma consulta com cada hipótese. Em outras palavras, para cada hipótese, havia diversos pares $\langle T, H \rangle$, com vários ou mesmo nenhum tendo relação de implicação. Uma tarefa secundária da edição foi a detecção de novidades: detectar se uma dada hipótese H contém informações que podem ser inferidas a partir de um corpus ou não (caso em que H traria *novidades*). As duas modalidades são bastante semelhantes; ainda assim, o ajuste de parâmetros de alguns sistemas pode privilegiar a performance em uma ou outra tarefa.

RTE-7 A sétima edição deu continuidade ao modelo anterior.

RTE-8 A oitava e última edição do evento trouxe respostas dadas por estudantes a perguntas que exigiam explicação e elaboração. Os pares para a RIT eram compostos por uma resposta de referência e a resposta dada por um aluno, de modo que quando houvesse implicação a resposta estaria correta.

Resultados de Avaliações

Os dados do RTE Challenges são particularmente difíceis de se classificar. Essa conclusão se baseia na comparação da performance de sistemas desenvolvidos para tratá-los com i) os baselines preparados pelos organizadores e ii) resultados obtidos em outros conjuntos, como o SICK e SNLI.

Na Tabela 2, são apresentadas as médias e medianas da performance dos sistemas participantes das sete primeiras edições do RTE Challenges, além do resultado de baselines para cada uma. Os valores do primeiro grupo são a acurácia, enquanto que do segundo são a medida F_1 .

O baseline para as cinco primeiras edições, reportado por [Bentivogli et al. \(2009\)](#), consiste em um classificador linear que considera apenas a quantidade de palavras em comum de T e H. Nas duas edições seguintes, que tratavam de textos retornados por motores de busca, o baseline considera que há implicação nos cinco primeiros textos para cada hipótese.

Edição	Treino	Treino Acumulado	Teste	Baseline	Média	Mediana	Melhor
RTE-1	567	567	800	55,37	56,20	56,45	70,00
RTE-2	800	2.167	800	54,4	59,00	59,87	75,38
RTE-3	800	3.767	800	62,4	61,75	61,97	80,00
RTE-4	0	4.567	1.000	56,6	58,30	59,41	74,60
RTE-5	600	6.167	600	57,5	61,50	61,52	73,50
RTE-6	221 H*	6.767	243 H*	34,63	33,77	36,14	48,01
RTE-7	284 H*	6.767	269 H*	37,41	35,95	41,90	48,00

* Os números dizem respeito à quantidade de hipóteses nesses conjuntos, não pares; vide texto para maiores detalhes.

Tabela 2 – Valores de acurácia (para as cinco primeiras linhas) e F_1 (para as duas últimas) do sistema baseline, média, mediana e melhor dos participantes nas sete primeiras edições do RTE Challenges. Todos os resultados são para o problema com duas classes (implicação e neutro).

Também é indicado na tabela o tamanho dos conjuntos de dados disponibilizados a cada edição. O uso dos dados de edições anteriores, inclusive das seções de teste, para complementar o conjunto de treino de novas edições era estimulado pelos organizadores; de fato, não foi disponibilizado nenhum novo conjunto de treino no RTE-4. A terceira coluna da tabela indica a quantidade total de pares que poderiam ser utilizados para o treinamento de sistemas em cada edição.

No RTE-6 e RTE-7, os conjuntos de dados eram diferentes. Cada hipótese era relacionada a até 100 textos recuperados por um motor de busca, e cada par recebia anotação de RIT. Embora o número total de pares nesses casos seja bastante grande, há de se observar algumas particularidades que os tornam menos úteis:

- Muitos pares têm componentes T bastante similares entre si, e pareados com o mesmo H (com efeito, são retornados pelo motor de busca por essa razão), o que leva a pouca variabilidade nos dados.
- Muitos pares têm o componente T muito diferente de H, tornando a decisão negativa (ausência de implicação) trivial; isso acontece quando há poucas sentenças semelhantes a T na coleção.

Em geral, tanto a média como a mediana dos participantes está bastante próxima do baseline, chegando a ser superada por esses no RTE-3 — essa edição do evento foi a

que teve maior diferença da quantidade média de palavras em comum entre as sentenças de pares positivos e negativos. O formato das sexta e sétima edições do RTE resultou em performances ainda piores que as anteriores.

Alguns participantes chegaram a obter resultados bastante superiores ao baseline, com [Hickl *et al.* \(2006\)](#) obtendo 75% de acurácia no RTE-2 e [Hickl e Bensley \(2007\)](#) 80% de acurácia no RTE-3. Ainda assim, são valores relativamente baixos em comparação com os obtidos no SNLI (mais detalhes na Seção 2.5.3).

A dificuldade dos RTE Challenges pode ser atribuída à natureza dos textos que compõem seus conjuntos de dados: envolvem conhecimento extra-linguístico, como referências a entidades nomeadas, conhecimento de mundo etc., como listado na Seção 2.3. Além disso, os conjuntos oferecem uma quantidade relativamente pequena de dados em comparação com o SICK ou o SNLI.

Os exemplos a seguir ilustram algumas dificuldades. O par (5), extraído do RTE-6, é classificado como positivo; essa decisão depende de entender que *28-year-old reporter* se refere a *Jill Carroll*, o que não é uma relação óbvia. Já o par (6) ilustra um caso em que para chegar à decisão correta, é necessário saber que o Cairo fica no Egito. Por fim, o par (7) trata de um caso negativo, mas com grande sobreposição de palavras entre as duas sentenças.

- (5)
 - a. The 28-year-old reporter was seized by gunmen on Saturday after calling by the office of a prominent Sunni politician in the neighbourhood.
 - b. Jill Carroll was seized by gunmen.
- (6)
 - a. El-Nashar was detained July 14 in Cairo after Britain notified Egyptian authorities that it suspected he may have had links to some of the attackers.
 - b. El-Nashar was arrested in Egypt.
- (7)
 - a. The chapters voluntarily transferred their right of electing the bishop to Emperor Charles V, and Pope Clement VII gave his consent to these proceedings.
 - b. Emperor Charles V was elected by Clement VII.

Ao longo das avaliações, foram propostas diversas estratégias para a RIT (detalhadas no Capítulo 3), mas no RTE-7, já não havia mais inovação ou melhoria significativa dos sistemas ou das estratégias propostas ([BENTIVOGLI *et al.*, 2011](#)).

2.5.2 SICK

O corpus SICK (Sentences Involving Compositional Knowledge) foi disponibilizado por [Marelli *et al.* \(2014b\)](#) com o objetivo de servir para a avaliação de sistemas que

trabalhem com composicionalidade, isto é, métodos que combinam unidades linguísticas como palavras em estruturas mais complexas, como sintagmas e orações, até chegar a uma representação de um texto. O SICK apresenta pares de sentenças anotados quanto a dois fenômenos linguísticos: a presença de implicação textual (cada par pode ter uma relação de implicação, contradição ou nenhuma das duas) e o nível de relação semântica (cada par é avaliado com um número real de 1 a 5, indicando o quão relacionado semanticamente é o conteúdo das duas sentenças).

De fato, em meados de 2014, quando do lançamento do corpus, a discussão sobre representações vetoriais de unidades linguísticas maiores que as palavras era bastante presente (SOCHER *et al.*, 2012; SOCHER *et al.*, 2013; LE; MIKOLOV, 2014), e ainda incipiente³.

A criação do SICK evitou conteúdo que precisasse de conhecimento extra-linguístico para ser conhecido, como reconhecimento de entidades nomeadas, conhecimento de mundo em geral ou expressões idiomáticas não-composicionais (expressões cujo significado não é resultado do significado das partes; p. ex. *abrir mão* → *desistir*), já que saber lidar com estes tipos de fenômeno não diz respeito à capacidade de gerar representações composicionais. Em vez disso, o corpus inclui fenômenos como variações sintáticas, uso de quantificadores, negações, sinônimos contextuais, etc.

Sua criação tomou como base conjuntos de dados contendo fotografias acompanhadas de descrições escritas por pessoas. Tais descrições têm a vantagem de serem sentenças razoavelmente simples, usando um vocabulário bastante genérico. As sentenças foram então normalizadas, com a conversão de alguns tempos verbais, remoção de entidades nomeadas e outros fenômenos que os criadores do SICK queriam evitar.

Em seguida, novas versões de cada sentença foram geradas automaticamente. O processo de geração incluiu mecanismos como acrescentar uma negação, trocar o sujeito com o objeto, trocar voz ativa para passiva, substituir palavras por antônimos ou sinônimos, entre outros. As sentenças geradas foram verificadas por um anotador, que descartou as que tivessem erros gramaticais ou não fizessem sentido. Por fim, as sentenças originais e as geradas foram pareadas de modo a gerar 10 mil pares, que foram por fim anotados quanto a implicação e similaridade. Além de remover a necessidade de conhecimento extra-linguístico, este processo também produziu muitos pares com grande quantidade de palavras em comum, mas pertencentes a diferentes classes, o que penaliza estratégias lexicais simplistas.

³ Ainda hoje não há um consenso sobre o melhor método para compor representações vetoriais de sentenças ou textos, mas em uma grande quantidade de problemas, redes neurais recorrentes – particularmente LSTMs (HOCHREITER; SCHMIDHUBER, 1997) – têm obtido bons resultados (MUNKHDALAI; YU, 2016). Esse assunto é discutido com mais detalhes no Capítulo 4

Resultados de Avaliações

O SICK foi usado em uma das avaliações do SemEval 2014 ⁴ (MARELLI *et al.*, 2014a), que contou com 21 participantes. Desde então, alguns outros trabalhos de RIT também reportaram sua performance no SICK; no entanto, o corpus foi rapidamente eclipsado pelo SNLI (descrito na Seção 2.5.3), lançado pouco tempo depois e que se tornou o benchmark *de facto* para RIT.

Apesar do propósito inicial do SICK, não foi imposta nenhuma restrição quanto aos sistemas dos participantes do SemEval serem baseados em composicionalidade – o uso de outros tipos de abordagens permitiria uma comparação mais rica. Nos resultados do SemEval, sistemas baseados em composicionalidade ou não tiveram resultados variados, dos melhores aos piores.

Embora vários participantes tenham usado estratégias envolvendo representações vetoriais para sentenças e sintagmas, estas eram usadas como mais um componente de engenharia de atributos⁵. Isto é, os vetores não eram dados de entrada para modelos de aprendizado de máquina, mas sim algumas medidas eram tiradas deles (como similaridade do cosseno).

Os resultados obtidos para RIT no SICK confirmam que este é um conjunto de dados mais fácil que os RTE Challenges. Dentre os sistemas avaliados no SemEval 2014, a mediana da acurácia foi de 77,1%, com média de 75,4%, e o melhor sistema chegou a 84,6%.

Críticas

Apesar de útil para a tarefa de RIT, alguns aspectos do SICK podem ser criticados. Um deles é que pares contraditórios são muito facilmente identificáveis: Lai e Hockenmaier (2014), na descrição de sua participação na avaliação, mostram que 86,4% desses pares podem ser identificados apenas pela presença de palavras de negação. Este é um resultado do processo semi-automatizado da criação do corpus, e não corresponde à dificuldade de se detectar contradições em exemplos com textos reais.

Em uma análise mais aprofundada do corpus e dos resultados do SemEval, Bentivogli *et al.* (2016) conjecturam que muitos participantes podem ter sobreajustado (*overfit*) seus modelos às idiossincrasias do SICK.

Outro problema do corpus diz respeito à qualidade das anotações. Bentivogli *et al.* (2016) mencionam que os anotadores deveriam assumir que as duas sentenças se referiam

⁴ SemEval é uma conferência anual de PLN que propõe diversas avaliações conjuntas.

⁵ Por *engenharia de atributos* se entende a prática de extrair diversas medidas indicadoras para serem usados como representação dos dados para um algoritmo de aprendizado de máquina. Exemplos de atributos comuns nesse tipo de problema são a quantidade ou proporção de palavras em comum, quantidade de sinônimos, distância de edição, etc.

aos mesmos eventos e situações, mas muitas vezes essa norma não foi observada. Isto resultou em anotações inconsistentes; por exemplo, o par com T: “*A couple is not looking at a map*” e H: “*A couple is looking at a map*” foi atribuído à classe neutra. A explicação para esse fato é que os anotadores teriam entendido que cada sentença se referia a um casal diferente, e que, portanto, poderiam ser verdadeiras ou não independentemente uma da outra.

2.5.3 SNLI

O corpus SNLI (*Stanford Natural Language Inference*) (BOWMAN *et al.*, 2015) foi compilado com o intuito de fornecer à comunidade de PLN um recurso grande o suficiente para o treinamento de redes neurais profundas para o RIT. Essa motivação é consoante com os avanços de técnicas neurais para diversos problemas de PLN, que ainda esbarrava na quantidade razoavelmente pequena de dados disponíveis para RIT. Com efeito, o SICK conta com aproximadamente 570 mil pares, e pesquisas com o uso de modelos profundos no corpus obtiveram resultados positivos (ROCKTÄSCHEL *et al.*, 2015; WANG; JIANG, 2015).

Com o lançamento do SNLI, popularizou-se uma nomenclatura diferente para o problema em inglês. A tarefa até então conhecida apenas como *Recognizing Textual Entailment* passou a ser chamada de *Natural Language Inference*, e o *element text* (isto é, a primeira sentença de um par) passou a ser chamada de *premise*.

Similarmente ao SICK, os seus criadores veem o RIT não apenas como um fim em si, mas também como forma de avaliar a capacidade de sistemas computacionais de criar representações semânticas para os textos que processam.

A anotação do corpus se deu da seguinte forma. Inicialmente, foi tomado um conjunto de fotografias com descrições simples do seu conteúdo. As descrições, sem as fotos, foram apresentadas para os anotadores, e eles deveriam então escrever três novas sentenças: uma que necessariamente fosse verdade baseada na descrição, uma que poderia ser verdade ou não, e uma necessariamente falsa. As três novas sentenças foram pareadas com a descrição original para criar exemplos de implicação, ausência de implicação ou contradição, e contradição. Ao serem impedidos de ver as fotos originais, os anotadores tinham que restringir suas sentenças ao conteúdo da descrição.

Após a escrita de sentenças para formar os pares, um subconjunto de cerca de 10% dos pares foi validado por outros anotadores. Cada par deste subconjunto foi mostrado para quatro novos anotadores, que deveriam decidir qual das três categorias (implicação, contradição ou neutro) se aplicava ao par. Os criadores tomaram a decisão de três dos cinco anotadores (o autor do par mais os quatro avaliadores) como o rótulo correto para aqueles pares, com apenas 2% dos casos em que não houve consenso (tais pares foram

descartados). O alto nível de concordância entre anotadores deve ser atribuído ao fato de que os pares foram escritos deliberadamente visando uma das três relações em particular.

Ao contrário dos outros conjuntos apresentados nesta seção, o SNLI não foi usado em nenhuma avaliação conjunta. Ainda assim, diversos pesquisadores treinaram sistemas de RIT no corpus e publicaram seus resultados (SHEN *et al.*, 2017; ROCKTÄSCHEL *et al.*, 2015; PARIKH *et al.*, 2016; CHEN *et al.*, 2017a).

2.5.4 MultiNLI

Os melhores resultados no SNLI foram obtidos por modelos de redes neurais que modelavam relações entre duas sentenças, isto é, modelos que, de alguma forma, comparavam T com H, para enfim emitir uma resposta quanto à presença de implicação ou contradição. Um outro tipo de abordagem neural também explorado por alguns pesquisadores, mais genérico, é a geração de uma representação vetorial para cada sentença sem levar a outra em conta. O modelo, então, aplica alguma função sobre as duas representações e emite a resposta.

Embora o segundo tipo de abordagem tenha apresentado resultados um pouco inferiores no RIT, tem a vantagem de ser mais genérico — idealmente, a mesma representação gerada para uma sentença pode ser usada para diversos fins (possivelmente combinada com as de outras sentenças em um mesmo texto): classificação de textos, tradução, análise de sentimentos, busca por conteúdos semelhantes, entre outros. De fato, pesquisadores que desenvolvem novos métodos de representação sentencial comumente reportam performance de seus modelos em diversas tarefas (YU, 2017; KIROS *et al.*, 2015; SHEN *et al.*, 2017).

Essa foi a motivação para a criação do corpus MultiNLI (*Multi-Genre Natural Language Inference*) (WILLIAMS; NANGIA; BOWMAN, 2017) e seu uso na avaliação conjunta do evento RepEval 2017⁶. Na avaliação, os participantes deveriam usar estratégias do segundo tipo mencionado acima — que partisse da geração de representações de sentenças independentes uma da outra.

Os textos do MultiNLI vêm de vários domínios e gêneros diferentes: conversas por telefone e pessoalmente, guias de viagem, cartas de instituições, livros de ficção, entre outros. Isso permite uma avaliação mais abrangente dos modelos, que precisam aprender idiossincrasias de cada domínio, em contraste com o SNLI em que há apenas descrições de imagens. Além disso, avaliações cross-domínio são outro importante aspecto para ferramentas de PLN: dos dez domínios presentes no conjunto de testes, apenas cinco fazem parte do conjunto de treino.

O MultiNLI tem 433 mil pares, que foram anotados de forma semelhante ao SNLI:

⁶ Mais informações em <<https://repeval2017.github.io/>>

anotadores recebiam uma sentença e deveriam escrever três outras que seriam pareadas com esta, de modo a ter um par com relação de implicação, um com contradição e outro neutro. Por serem escritos pelos anotadores, os componentes H de cada par diferem em estilo dos componentes T na maioria dos gêneros.

2.5.5 Outros Conjuntos

Os conjuntos descritos nas seções anteriores têm posição de destaque na pesquisa de RIT, devido às suas dimensões, seu tempo de existência e/ou ao seu uso em avaliações conjuntas. Além destes, são encontrados também outros conjuntos de dados para RIT na literatura que tiveram impacto menor, seja por tratarem de nichos específicos ou por terem sido publicados mais recentemente. São tratados aqui brevemente alguns destes conjuntos: o SciTail, o MPE, o CLTE e o JOCI.

2.5.5.1 SciTail

O SciTail (KHOT; SABHARWAL; CLARK, 2018) contém pares criados a partir de perguntas de múltipla escolha sobre ciências e textos que potencialmente justificam as respostas. Foi proposto com o objetivo de aproximar o RIT ao cenário de aplicações práticas de PLN, em contraste com o SICK e SNLI. Contém 27 mil pares divididos em relação de implicação ou neutros.

Neste corpus, o componente H de cada par é formado pela combinação de uma pergunta com uma resposta, de modo a formar uma sentença afirmativa. O exemplo (8) ilustra a formação de uma destas sentenças.

- (8) a. *Because trees add water vapor to air, cutting down forests leads to longer periods of what?*
 b. *Drought*
 c. *Because trees add water vapor to air, cutting down forests leads to longer periods of drought.*

Já o componente T de cada par é uma sentença retornada por um motor de busca que recebe a pergunta e a resposta candidata como consulta. Como resultado, os pares do SciTail são bem mais complexos do que aqueles encontrados no SICK ou SNLI. Os pares (9) e (10) ilustram, respectivamente, um caso neutro e outro de implicação.

- (9) a. *Because trees add water vapor to air, cutting down forests leads to longer periods of drought.*
 b. *During periods of drought, trees died and prairie plants took over previously forested regions.*

- (10) a. *When waves of two different frequencies interfere, beating occurs.*
 b. *Beats are the periodic and repeating fluctuations heard in the intensity of a sound when two sound waves of very similar frequencies interfere with one another.*

Para a criação do corpus, partiu-se de um conjunto de cerca de 3.200 perguntas de múltipla escolha, com quatro opções cada. Foram então geradas sentenças afirmativas para cada combinação de pergunta e resposta, conforme exemplificado acima. Em seguida, a pergunta e a resposta são dadas como consulta para um motor de busca, que vasculha uma coleção de textos extraídos da Internet. As primeiras 40 sentenças retornadas pelo motor de busca são pareadas com a pergunta/resposta, gerando um par $\langle T, H \rangle$ que passa por anotação manual.

Cada par poderia ser julgado em três categorias: implicação, neutro e implicação parcial. O terceiro caso, o mais comum, ocorre quando T implica parte de H , mas não totalmente. Cada par foi avaliado por cinco anotadores, e foram mantidos para o conjunto final apenas os pares que obtivessem 80% de concordância e contivessem ou uma relação de implicação ou neutra. O descarte de pares com implicação parcial se justifica pelo objetivo dos autores de coletar casos claros de implicação, que já são suficientemente difíceis de serem detectados automaticamente.

Os autores descrevem alguns experimentos realizados com modelos da literatura que apresentaram bons resultados no SNLI e os comparam com uma nova abordagem. Sua conclusão indica que as diferenças deste corpus em relação ao SNLI levam a uma perda de performance para tais modelos. Por outro lado, sua nova abordagem apresenta resultados um pouco melhores.

2.5.5.2 MPE

O conjunto MPE (*Multiple Premise Entailment*) (LAI; HOCKENMAIER, 2017) contém, em vez de pares $\langle T, H \rangle$, tuplas com quatro premissas e uma hipótese. Cada hipótese é classificada como tendo uma relação de implicação, contradição ou neutra com as outras quatro sentenças. Sua motivação é simular cenários em que se tem acesso a diversos documentos descrevendo algum evento, e se deseja determinar se uma conclusão é verdadeira ou não.

As sentenças que compõem o corpus vêm de descrições de imagens, similarmente ao SICK e SNLI. Cada tupla foi julgada por cinco anotadores, e o conjunto total tem 10 mil tuplas.

Este tipo de formulação da RIT tem algumas propriedades interessantes. Os autores coletaram julgamentos quanto ao RIT de pares individuais de parte do conjunto (pareando cada premissa de uma tupla com a respectiva hipótese) e os compararam com

a classe da tupla. A classe majoritária dentre os quatro julgamentos de pares corresponde à classe da tupla em apenas 34,6% dos casos, indicando a importância de se considerar todo o conjunto de informações para se chegar a uma decisão sobre a hipótese.

2.5.5.3 CLTE

CLTE (*Cross-Lingual Textual Entailment*)⁷ é o nome dado à tarefa, introduzida no evento SEMEVAL 2012 (NEGRI *et al.*, 2012) e continuada no SEMEVAL 2013, de realizar o RIT em pares cujas sentenças estão em línguas diferentes. A proposta do CLTE teve interesse na *sincronização de conteúdo*, isto é, dados dois documentos online em línguas diferentes sobre um mesmo assunto, detectar quando um deles foi atualizado, de modo a se poder sincronizar o outro com as novas informações.

Com esse objetivo, o conjunto de dados da avaliação contém pares em que a relação de implicação pode se dar tanto da primeira sentença para a segunda (forward) como da segunda para a primeira (backward), além de casos de paráfrases e neutros. Em 2012, foram criadas versões do conjunto de dados em espanhol/inglês, francês/inglês, italiano/inglês e alemão/inglês. Cada conjunto é pequeno para os padrões da área, contendo 1.000 pares, na versão inicial de 2012⁸, sendo esse um conjunto balanceado com relação aos tipos de implicação (paráfrase, forward, backward e neutro).

Para a criação do corpus, foram coletadas sentenças de textos de notícias em inglês, que serviriam como o segundo componente dos pares. Em seguida, estas sentenças foram modificadas manualmente com adição e remoção de trechos, em um processo de *crowdsourcing*, para se gerar o primeiro componente, também em inglês. Estes foram então traduzidos para cada uma das línguas usadas no conjunto. Por fim, os pares foram revisados manualmente para se certificar de estarem consistentes com seus rótulos.

Naturalmente, o CLTE introduz uma dificuldade relacionada à tradução automática, o que não foi de interesse da presente pesquisa de doutorado nem de muitos pesquisadores da área de RIT, embora tenham interesse na área de tradução automática.

2.5.5.4 JOCI

O JOCI (*JHU Ordinal Common-sense Inference*) (ZHANG *et al.*, 2017) é um corpus de pares de sentenças anotadas quanto a uma versão mais subjetiva do RIT. Em vez de categorias mutuamente exclusivas comumente usadas, o JOCI apresenta cinco níveis de implicação entre as sentenças: muito provável, provável, plausível, tecnicamente possível e impossível.

⁷ <https://hlt-mt.fbk.eu/technologies/clte-benchmark>

⁸ <https://www.cs.york.ac.uk/semeval-2012/task8/>

Foi o objetivo de seus criadores possibilitar o desenvolvimento de um sistema capaz de fazer inferências de senso comum, isto é, devidas ao conhecimento de mundo que praticamente todas as pessoas possuem. Como explicado na Seção 2.3, são implicações particularmente difíceis de se automatizar, pois muitas vezes envolvem conceitos não explícitos no texto. Além disso, visavam o componente especulativo, que se traduz na presença de diferentes níveis de confiança na anotação dos pares.

2.6 Considerações Finais

Foi apresentada neste capítulo a definição do reconhecimento de implicação textual, suas aplicações, e um levantamento de dificuldades para seu processamento computacional. Também foram apresentados conjuntos de dados anotados para possibilitar o treinamento e avaliação de sistemas para realizar a tarefa, sendo explicados brevemente seus processos de criação e particularidades.

A Tabela 3 mostra informações sobre cinco conjuntos de dados tratados aqui. Para os RTE Challenges, foram apenas consideradas as edições RTE-1 a RTE-5, que consistem de pares simples de textos. A coluna *Tamanho* refere-se à quantidade de pares do corpus.

Corpus	Tamanho	Balanceado	Gêneros/Tipo de Texto	Ano
RTE Challenges	6.767	Não	Notícias	2005 – 2009
SICK	10.000	Não	Descrições de imagens	2014
SNLI	570.000	Sim	Descrições de imagens	2015
MultiNLI	433.000	Sim	Diversos	2017
SciTail	27.026	Não	Perguntas sobre ciências	2018

Tabela 3 – Estatísticas sobre os conjuntos de dados de RIT. A coluna balanceado indica se pares de diferentes classes aparecem na mesma quantidade.

A criação do primeiro conjunto de dados e avaliação conjunta para o RIT em 2005 indicou um reconhecimento por parte da comunidade de PLN da importância do problema. Edições subsequentes refinaram a tarefa e forneceram mais dados, mantendo ativo o interesse no RIT. O esforço para a compilação de recursos de larga escala, como o SNLI e o MultiNLI, mostram que de fato se trata de um problema relevante.

Para a língua portuguesa, há um único recurso disponível nos mesmos moldes que os supracitados: o ASSIN, que será apresentado no Capítulo 5, fruto do presente trabalho de doutorado.

ABORDAGENS CLÁSSICAS

Neste capítulo, são apresentadas abordagens da literatura para lidar com o problema de RIT usando engenharia de atributos e algoritmos clássicos da literatura de aprendizado de máquina. Ainda que as abordagens apresentadas aqui tenham diferenças marcantes entre si, o seu conjunto como um todo é muito mais homogêneo quando comparado aos métodos neurais apresentados no Capítulo 4.

Duas revisões da literatura da área ([DAGAN *et al.*, 2013](#); [ANDROUTSOPOULOS; MALAKASIOTIS, 2010](#)) listam e descrevem as principais abordagens de RIT, mas diferem na forma como as organizam. Aqui, deu-se preferência à nomenclatura adotada por [Dagan *et al.* \(2013\)](#), pois se baseia no raciocínio subjacente a cada uma das abordagens, enquanto [Androutsopoulos e Malakasiotis \(2010\)](#) dão muito peso às técnicas empregadas, que nem sempre definem bem as abordagens.

Por exemplo, [Androutsopoulos e Malakasiotis \(2010\)](#) classificam abordagens quanto a serem baseadas em análise sintática, em representação semântica ou em aprendizado de máquina. No entanto, há na literatura trabalhos que combinem duas dessas ou mesmo as três técnicas. Já quanto à classificação de [Dagan *et al.* \(2013\)](#), a separação é mais clara.

3.1 Notação

Neste capítulo e nos subsequentes, é usada a seguinte notação algébrica. Vetores numéricos são indicados por letras minúsculas em negrito, como $\mathbf{x}, \mathbf{y}, \mathbf{h}$, ou letras gregas: α, β . Matrizes são indicadas por letras maiúsculas em negrito, como $\mathbf{M}, \mathbf{W}$. Valores escalares são denotados por letras minúsculas sem negrito, como k, d . Letras minúsculas também podem representar outros conceitos, como palavras; o texto explicita quando se trata de elementos numéricos ou não.

O i -ésimo elemento de um vetor é denotado por $\mathbf{x}_{[i]}$, e o elemento na linha i e coluna

j de uma matriz por $\mathbf{M}_{i,j}$. Para indicar o vetor que compõe a i -ésima linha de uma matriz, é usado $\mathbf{M}_{[i,*]}$ e, analogamente para a j -ésima coluna, $\mathbf{M}_{[:,j]}$. Em oposição, a notação sem colchetes $\mathbf{x}_i$ se refere ao i -ésimo elemento dentro de alguma sequência $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ mencionada no texto.

3.2 Aprendizado de Máquina

Nesta seção, são expostos de forma breve conceitos gerais de classificação automática, uma subárea do aprendizado de máquina. Tais conceitos, aqui referidos como de *aprendizado de máquina clássico*, serão contrastados com os métodos de *deep learning* e representações distribuídas no Capítulo 4.

Um modelo de classificação automática, ou classificador, têm como objetivo produzir uma função $f(x) = y$, onde x é a representação de alguma instância de dados, e y representa a classe, ou categoria, à qual aquela instância pertence. Normalmente, a representação x se dá por um vetor numérico $\mathbf{x} \in \mathbb{R}^d$, em que cada valor representa um determinado atributo.

No caso do RIT, cada instância x é um par $\langle T, H \rangle$, e o vetor $\mathbf{x}$ costuma incluir atributos como a proporção de palavras em comum entre as sentenças ou presença de negação, como será mostrado neste Capítulo. Atributos como o primeiro, assim como outros valores numéricos, são trivialmente codificados em $\mathbf{x}$ com o seu próprio valor¹. Atributos binários são normalmente representados com um componente que pode ter o valor de 0 ou 1. Por exemplo, se o i -ésimo atributo definido para representar o par é a presença de negação em T , $\mathbf{x}_i$ será 1 caso haja negação e 0 caso contrário. Para representar quais transformações textuais foram realizadas de T para H , é possível considerar cada posição i em $\mathbf{x}$ como um indicativo de se a i -ésima operação de transformação (como inversão de voz ativa para passiva ou remoção de um adjunto) foi realizada.

O vetor $\mathbf{x}$ é, portanto, uma abstração de $\langle T, H \rangle$. Assim, é de grande importância que seja bem representativo do par com respeito à existência de uma relação de implicação, paráfrase, contradição, ou outra que se queira predizer. Chama-se *engenharia de atributos* o processo de determinar os atributos que compõem $\mathbf{x}$, e que é de extrema importância para as abordagens listadas neste capítulo.

Os classificadores empregados para o RIT (e em PLN em geral) em conjunto com engenharia de atributos são normalmente lineares. Tais classificadores determinam a classe de um par $\langle T, H \rangle$ em função do resultado de uma combinação linear $y = \mathbf{w} \cdot \mathbf{x} + \mathbf{b}$, em que $\mathbf{x}$ é o vetor de atributos e w e b são, respectivamente, um vetor de parâmetros e uma

¹ Uma técnica comum em aprendizado de máquina é normalizar atributos para que seus valores estejam em um intervalo limitado como entre 0 e 1. No entanto, isto não altera como o valor original do atributo é obtido.

constante de viés determinados durante o treinamento do classificador². No caso de se trabalhar apenas com duas classes, se y for maior do que 0, é atribuída uma classe (como implicação positiva), caso contrário, a outra classe. Quando se lida com mais de uma classe, é computado um valor $y_j = \mathbf{w}_j \cdot \mathbf{x} + \mathbf{b}_j$ para cada classe c_j , e é escolhida aquela com o maior valor y_j correspondente; i.e. $\text{argmax}_j y_j$.

Na prática, determinar os valores de $\mathbf{w}$ corresponde a quantificar a importância de cada atributo para a decisão e sua orientação para classe positiva ou negativa, e o valor $\mathbf{b}$ corresponde ao viés do classificador (o quanto este tende a uma determinada classe na ausência de mais informações).

3.3 Bases de Conhecimento para RIT

Como será visto, algumas bases de dados linguísticos são comumente usadas para fornecer conhecimento para sistemas de RIT. Estas bases são recursos que contêm algum tipo de conhecimento geral ou linguístico — como taxonomias, expressões equivalentes, ou mesmo bases estruturadas. Esta seção revisa brevemente as que são empregadas nos trabalhos da literatura.

3.3.1 Wordnet

A Wordnet (FELLBAUM, 1998), construída por pesquisadores da Universidade de Princeton para inglês, é um dos recursos lexicais mais conhecidos no PLN. Trata-se de uma estrutura hierárquica em forma de árvore, em que cada nó, chamado *synset*, contém palavras com o mesmo significado. Synsets são conectados de modo que aqueles com conceitos mais específicos sejam filhos de conceitos mais genéricos.

Palavras polissêmicas podem aparecer em vários synsets, com um sentido específico indicado em cada um deles. Por isto, os synsets são mais especificamente conjuntos de *sentidos* de palavras, não simplesmente palavras. No momento da escrita desta tese, a Wordnet conta com mais de 155 mil palavras, 117 mil synsets e 206 mil sentidos de palavras. Algumas relações lexicais são codificadas na estrutura da Wordnet:

Sinonímia Sentidos que ocorram no mesmo synset são ditos sinônimos. Por exemplo, *cachorro* e *cão*.

Antonímia Sentidos de palavras contidos em synsets listados como opostos são ditos antônimos. Por exemplo, *quente* e *frio*.

² Para alcançar um maior poder preditivo, é possível a aplicação de uma função kernel $k(\cdot)$ sobre $\mathbf{x}$, especialmente em conjunto com classificadores SVM (*support vector machines*). Dessa forma, a função aplicada pelo classificador se torna $\mathbf{w} \cdot k(\mathbf{x}) + \mathbf{b}$ (BAUDAT; ANOUAR, 2001).

Hiperonímia Um sentido que ocorra num synset pai é dito hiperônimo de um synset filho. Por exemplo, *animal* é um hiperônimo de *cachorro*.

Hiponímia É o inverso de hiperonímia.

Com a popularidade da Wordnet original, o termo *wordnet* passou a ser usado para se referir a outros recursos com estrutura similar em outras línguas. Para o português, há diferentes projetos similares, dentre os quais se destacam a OpenWordNet-PT (PAIVA; RADEMAKER; MELO, 2012), como a mais completa feita sob revisão manual, e o CONTO.PT (GONÇALO OLIVEIRA, 2016) construída de forma semi-automática de modo a ter uma abrangência maior de palavras, e que se descreve como uma wordnet *difusa*.

3.3.2 PPDB

O PPDB³ (*Paraphrase Database*) é um recurso que contém paráfrases em diversas línguas, incluindo o português, extraídas automaticamente de corpora bilíngues (GANITKEVITCH; CALLISON-BURSCH, 2014). *Paráfrase* no contexto do PPDB se refere a expressões ou palavras equivalentes, não sentenças inteiras, como no corpus ASSIN (descrito no Capítulo 5).

O processo de extração foi baseado no raciocínio de que duas expressões O_1 e O_2 , que têm relação de paráfrase em uma língua origem, podem ser traduzidas para uma mesma expressão D em uma língua destino, chamada *língua pivô*. Ganitkevitch e Callison-Bursch (2014) usaram o Europarl⁴, um corpus de traduções alinhadas dos anais do parlamento europeu. O inglês foi usado como pivô para extrair paráfrases de outras línguas, por contar com um parser sintático mais preciso, o que permitiu agregar mais informações.

Cada item do PPDB é chamado de regra de paráfrase, e descreve uma palavra ou expressão f e sua paráfrase e . As regras têm ainda uma série de atributos que descrevem sua natureza e o quão confiáveis são, como, por exemplo, estimativas da substituição de e por f segundo métricas de tradução automática; a diferença da quantidade de caracteres e palavras entre ambas etc.

As paráfrases extraídas aparecem em três tipos: *lexicais*, em que se trata de um par de palavras; *multipalavras* (*phrasal* no original), que incluem mapeamento de uma palavra para várias ou de várias para várias; e *sintáticas*, que incluem não apenas palavras em e ou f , mas também variáveis representando categorias sintáticas. Estas categorias podem

³ Acessível em <<http://paraphrase.org>>

⁴ Acessível em <<http://www.statmt.org/europarl>>

indicar tanto nós não-terminais em uma árvore de constituintes, como NP (sintagma nominal), VP (sintagma verbal), etc., ou terminais, por meio de etiquetas morfossintáticas⁵.

Uma inconsistência do PPDB é incluir uma grande quantidade de regras envolvendo pares *e* e *f* que são mapeados para as mesmas palavras em inglês devido a diferenças gramaticais com o português. Por exemplo, adjetivos em português em diferentes flexões de gênero e número aparecem como paráfrases no PPDB, assim como diferentes formas do mesmo verbo. Como o processo de construção do PPDB não empregou nenhum tipo de conhecimento morfológico específico para cada língua, este fenômeno já era esperado.

Como foi gerado automaticamente, o PPDB contém também alguns falsos positivos, ou seja, pares em que *e* e *f* têm significados distintos. O recurso está disponível em seis diferentes tamanhos, com os conjuntos maiores tendo regras de paráfrase extraídas com menor confiança.

3.3.3 CatVar

O CatVar (*Categorical Variation*)⁶ (HABASH; DORR, 2003) é uma base de dados em inglês que contém variantes de palavras em diferentes classes gramaticais. Por exemplo, a palavra *hunger* está relacionada a *hungry*, *hungriness*, além de poder ser substantivo ou verbo. O CatVar relaciona estas palavras, bem como as possíveis classes gramaticais de cada uma.

O CatVar contém cerca de 62 mil grupos de palavras, num total de 96 mil formas diferentes. Sua criação se baseou em consultas a dicionários, corpora e outros recursos lexicais como a WordNet.

3.3.4 DIRT

O DIRT (Discovering Inference Rules from Text) (LIN; PANTEL, 2001) é o nome tanto de um algoritmo para descoberta de padrões de inferência textual como da base de dados resultante produzida por seus autores. Seu funcionamento é baseado em analisar grandes corpora sem anotação em busca de padrões de ocorrência das mesmas palavras.

Por exemplo, o DIRT poderia encontrar ocorrências de sentenças como as seguintes:

- houve aumento na produção de **leite** em **janeiro**
- **janeiro** teve aumento na produção de **leite**

⁵ Etiquetas morfossintáticas indicam a classe gramatical a qual uma palavra pertence, como substantivo, verbo, adjetivo, etc.

⁶ Acessível em <<https://clipdemos.umiacs.umd.edu/catvar/>>

- foi produzido mais **leite** em **janeiro**

Ao encontrar uma grande quantidade de padrões como os acima, o DIRT cria padrões genéricos substituindo as palavras recorrentes por variáveis:

- houve aumento na produção de **X** em **Y**
- **Y** teve aumento na produção de **X**
- foi produzido mais **X** em **Y**

Na prática, o algoritmo é um pouco mais complexo, levando em consideração as árvores sintáticas de dependência das sentenças para extrair padrões válidos. No entanto, por se tratar de um algoritmo não-supervisionado, o DIRT eventualmente extrai algumas equivalências incorretas. Os autores indicam que relações de significados opostos frequentemente são extraídas, justamente por costumarem ter os mesmos argumentos. Como exemplo, dão os padrões *X solves Y* e *X worsens Y*. Ainda assim, o DIRT foi usado por vários trabalhos para o RIT.

3.3.5 VerbOcean

O VerbOcean (CHKLOVSKI; PANTEL, 2004) é um recurso que contém verbos interligados por cinco relações semânticas: *similaridade* (*similarity*; indica verbos com significados parecidos, mas não necessariamente sinônimos), *antonímia* (*antonymy*; sentidos opostos), *força* (*strength*; indica quando um verbo é uma versão mais intensa de outro), *capacitação* (*enablement*; indica quando um verbo descreve como se realiza a ação descrita por outro) e *acontece-antes* (*happens-before*; indica quando a ação de um verbo precisa acontecer temporalmente antes da ação descrita pelo outro).

Algumas das relações se sobrepõem um pouco. (CHKLOVSKI; PANTEL, 2004) explicam que *força* é uma relação mais específica que *similaridade*, e *capacitação* é mais específica que *acontece-antes*.

Sua organização difere de recursos como a WordNet por não organizar verbos em hierarquias ou classes. Em vez disso, o VerbOcean apenas apresenta relações semânticas independentes.

Para cada relação, foram usados alguns padrões de texto, criados manualmente, com duas variáveis que deveriam ser instanciadas pelos verbos. Os padrões, num total de 35, foram usados em consultas a um motor de busca na Internet. Por exemplo, para se buscar verbos que tivessem relação de força entre si, foi usado (entre outros) o padrão *not only X, but also Y*, e para a relação de acontece-antes, *to X and then Y*.

O retorno da consulta passa ainda por um processo de refinamento, em que alguns padrões são filtrados dependendo de valores de probabilidade estimada para coocorrência dos verbos envolvidos. Uma análise manual feita pelos autores em uma amostra de relações capturadas pelo VerbOcean para 100 verbos encontrou 65,5% de relações corretas.

3.4 Distância de Edição de Árvores

A Distância de Edição de Árvores, ou *Tree Edit Distance* (TED), é uma medida da diferença entre duas estruturas arbóreas. Em PLN, é comumente usada para comparar árvores sintáticas, mas também tem aplicações em outras áreas (PAWLIK; AUGSTEN, 2011).

A ideia básica da TED é considerar a distância entre as duas árvores, ou seu *custo* de edição, como a soma dos custos de operações de edição que podem ser: (i) a remoção de um nó⁷ (passando os nós filhos para o nó pai), (ii) a inserção de um novo nó na árvore, ou substituição de um nó por outro. O custo de cada operação deve ser definido a priori.

Uma solução trivial para a TED é remover todos os nós de uma árvore e em seguida adicionar todos os da segunda. No entanto, na prática é mais interessante se calcular o custo mínimo possível de edição entre as árvores. Há diferentes algoritmos para calculá-lo, que usam estratégias diferentes para minimizar a complexidade no tempo em diferentes casos (ZHANG; SHASHA, 1989; DEMAINE *et al.*, ; KLEIN, 1998).

Dentre eles, o de Zhang-Shasha (ZHANG; SHASHA, 1989) é um dos mais comumente usados, tendo boa performance no caso médio e tendo implementações disponíveis em diferentes linguagens. O algoritmo se vale de técnicas de programação dinâmica: a cada etapa de sua execução, são avaliadas diferentes possibilidades de edição, e escolhida a de menor custo. Descrever o seu funcionamento em detalhes está fora do escopo desta tese; para os propósitos abordados aqui, basta saber que o algoritmo encontra em tempo polinomial a menor TED para um par de árvores, dados os custos de cada operação.

A Figura 2 ilustra um exemplo simples de se calcular a TED em que duas sentenças diferem apenas em uma palavra. A transformação da primeira na segunda pode ser efetuada pela remoção da palavra *vermelhos* seguida pela adição de *azuis*, ou apenas com uma operação de substituição. Caso se defina um custo de uma unidade para cada operação, a segunda solução apresenta o menor custo total.

Já a Figura 3 e a Tabela 4 ilustram o caso de duas sentenças um pouco maiores e as operações necessárias. Assume-se aqui que todas as operações de edição tenham custo 1.

⁷ No contexto de TED, aqui será usado *nó* como equivalente a uma palavra em uma árvore dependências.

Figura 2 – Exemplo simples de TED

Figura 3 – Exemplo de TED com sentenças maiores. Palavras presentes apenas na primeira sentença estão em azul, e as presentes apenas na segunda estão em vermelho.

Operação	Nós	Resultado
S	A → O	O seleção brasileira está no grupo 3 , junto com Colômbia, Peru e Venezuela .
R	seleção	O ∅ brasileira está no grupo 3 , junto com Colômbia, Peru e Venezuela .
S	brasileira → Brasil	O Brasil está no grupo 3 , junto com Colômbia, Peru e Venezuela .
S	junto → juntamente	O Brasil está no grupo 3 , juntamente com Colômbia, Peru e Venezuela .

Tabela 4 – Operações de edição para o par mostrado na Figura 3. S indica substituição e R, remoção.

3.4.1 Limitações

A TED para duas sentenças que sejam sintaticamente diversas é mais alta, mesmo com trechos semanticamente equivalentes, dada a natureza puramente sintática da medida. Por exemplo, considere-se a situação das sentenças da Figura 4. A expressão *tem uma reunião* tem o mesmo significado de *se reúne*; no entanto, são construções diferentes.

Outra dificuldade no cálculo de TED diz respeito a mover subárvores. Este tipo de operação não é previsto na definição padrão de TED, pois incluir seu tratamento teria tempo exponencial (HEILMAN; SMITH, 2010).

Considere-se, por exemplo, o par ilustrado na Figura 5. As subárvores *na Praça*

Figura 4 – Exemplo de um par de sentenças semelhante semanticamente, mas com diferenças sintáticas

Seca e *sua infância* estão distantes entre si na primeira árvore, mas ambas estão conectadas ao mesmo verbo na segunda. Ainda assim, um algoritmo de TED teria que considerar a remoção de pelo menos uma das subárvores por inteiro e a posterior inserção de seus tokens na posição apropriada, conforme ilustrado na Tabela 5. O token *casa* não pode ser substituído por *infância*, já que o segundo está conectado a outro nó (*passara*) que é removido nas operações de edição.

Op	Nós	Resultado
S	comprou → passou	Luís passou uma casa na Praça Seca, onde passara sua infância.
R	casa	Luís passou uma ∅ na Praça Seca, onde passara sua infância.
R	passara	Luís passou uma na Praça Seca, onde ∅ sua infância.
R	infância	Luís passou uma na Praça Seca, onde sua ∅.
R	sua	Luís passou uma na Praça Seca, onde ∅.
R	onde	Luís passou uma na Praça Seca, ∅.
R	,	Luís passou uma na Praça Seca ∅.
I	infância	Luís passou uma infância na Praça Seca.
S	uma → sua	Luís passou sua infância na Praça Seca.

Tabela 5 – Operações de edição para o par mostrado na Figura 5. S indica substituição; R, remoção e I, inserção. Observe-se que o sintagma *sua infância* precisou ser removido para em seguida ser reinserido.

Figura 5 – Duas árvores sintáticas cuja diferença apresenta o movimento de subárvore

3.5 Abordagens de RIT

3.5.1 Similaridade e Alinhamento

Abordagens baseadas em similaridade partem do raciocínio de que a presença de relação de implicação ou paráfrase tende a ocorrer quando H é mais semelhante a T . Seu principal desafio é determinar critérios de semelhança adequados.

Já métodos calcados em alinhamento são um refinamento de técnicas de RIT por similaridade (DAGAN *et al.*, 2013), pois buscam explicitar alinhamentos entre palavras ou trechos de T e de H que apresentem similaridades locais. A presença de alinhamento entre T e H não necessariamente corresponde à decisão final em sistemas que seguem essa abordagem; ele pode ser um passo intermediário, fornecendo métricas para um módulo responsável pela decisão.

Apesar disso, os métodos ditos de alinhamento e de similaridade são muito similares entre si, consistindo em extrair métricas da similaridade do conteúdo das sentenças. Portanto, optou-se aqui por não fazer distinção entre ambos.

No extremo mais simples, a similaridade pode ser mensurada pelo alinhamento de palavras em comum entre T e H . Uma extensão natural é considerar recursos como a WordNet, e tratar como equivalentes (possivelmente com peso reduzido) sinônimos ou hipônimos (o hiperônimo, termo mais geral, implica no hipônimo, termo mais específico). Modificadores como modalidade (uso do futuro do pretérito ou de auxiliares como *poder*

e *dever*) e negação também podem ser levados em conta.

Outras alternativas podem ser usadas para identificar palavras de significado próximo. Um exemplo são modelos vetoriais (TURNERY; PANTEL, 2010), que codificam em vetores numéricos o tipo de contexto em que cada palavra costuma ocorrer.

Apenas a ocorrência de termos, no entanto, é insuficiente para um bom desempenho no RIT: pares $\langle T, H \rangle$ podem conter muitas palavras em comum e não apresentarem relação de implicação, ou vice-versa. Considere-se, por exemplo, o par de sentenças (1), em que as mesmas palavras ocorrem em ambas, mas que possuem significados opostos.

- (1) a. O ladrão avistou o segurança, mas não foi visto por ele.
 b. O segurança avistou o ladrão, mas não foi visto por ele.

Uma análise mais elaborada leva em conta também as árvores sintáticas de cada sentença. Para mensurar a similaridade entre duas árvores, podem ser contabilizadas diferenças estruturais, como a ocorrência de uma dada palavra ora como sujeito e ora como objeto de um dado verbo.

A seguir, são listados alguns métodos de similaridade e alinhamento. Muitos destes métodos apresentam diversas semelhanças entre si; buscou-se aqui focar em pontos particulares e relevantes deles.

3.5.1.1 EDITS

O EDITS (KOUYLEKOV; NEGRI, 2010), um sistema disponibilizado como código aberto, implementa funcionalidades típicas da estratégia de avaliar as diferenças entre T e H . Seu funcionamento considera o *custo* de cada diferença avaliada entre as duas sentenças, que podem ser referentes a palavras e construções sintáticas diferentes. No extremo mais simples, calcula simplesmente a quantidade de palavras diferentes entre T e H . Uma melhoria comum é consultar recursos como a WordNet para se checar sinônimos, que podem ser considerados palavras iguais para efeitos de distância. Além disso, como o texto normalmente é mais longo que a hipótese, o custo de se remover palavras dele é normalmente bem mais baixo do que a inserção de novas palavras.

Uma forma mais sofisticada de se calcular o custo de palavras diferentes é checar o valor IDF⁸ das que só ocorram na hipótese; dessa forma, uma palavra comum tem menos peso do que uma palavra rara. Além do nível lexical, o EDITS permite atribuir custos

⁸ O IDF (*Inverse Document Frequency*) é uma métrica comum na área de Recuperação de Informação que mede o quão rara é uma palavra. O IDF de uma palavra w é calculado como $\frac{1}{df(w)}$, onde $df(w)$ é a quantidade de documentos dentro de uma grande coleção onde w aparece. Dessa forma, quanto menos comum for uma palavra, maior seu valor de IDF e, de modo geral, mais informativa ela é.

para diferenças sintáticas, calculando a distância de edição de árvores.

Dados todos os custos (tanto lexicais quanto sintáticos), um classificador linear ajusta pesos de acordo com um conjunto de treino e é usado para emitir a decisão final. Avaliado no conjunto do RTE-5, o EDITS obteve uma acurácia de 61,83%, um pouco acima da média dos participantes.

3.5.1.2 IKOMA (*Sobreposição Lexical Ponderada*)

No RTE-7, os melhores resultados foram alcançados pela equipe IKOMA (TSUCHIDA; ISHIKAWA, 2011), com um sistema baseado principalmente na semelhança lexical. Seu método consiste em calcular a proporção de palavras de H que ocorrem em T , ponderadas por um peso inversamente proporcional à sua frequência (de modo a dar mais importância a palavras raras). Além disso, o sistema consulta a WordNet, o CatVar e uma lista de acrônimos (nomes de entidades com suas respectivas iniciais) em busca de termos equivalentes nas duas sentenças. Tais termos são considerados como ocorrendo em ambas as sentenças.

Experimentou-se também o uso de um módulo de filtragem, que examina a sobreposição de palavras de forma mais localizada. Este módulo busca alinhar chunks ou argumentos de APS (Anotação de papéis Semânticos) entre as duas sentenças, e em seguida verifica quantos deles possuem pelo menos 70% (valor determinado empiricamente) de sobreposição de palavras, usando os recursos listados acima. Quando mais de 80% (valor também determinado empiricamente) dos pares de chunks ou argumentos não atingem este valor, é decidido que não há relação.

Nos casos restantes, um limiar de sobreposição determina se há ou não a relação de implicação (este valor não é descrito pelos autores). Com o módulo de filtragem, o sistema proposto atingiu F_1 de 48,00, enquanto sem o mesmo, sua performance chegou a 47,94 no conjunto de testes. Percebe-se, portanto, que a sobreposição de palavras no âmbito da sentença é um bom indicador para o RIT, e que o mecanismo um pouco mais complexo de sobreposição localizada praticamente não trouxe melhorias.

3.5.1.3 UAIC (*Sobreposição Lexical com Heurísticas*)

No RTE-5, os melhores resultados foram obtidos por Iftene e Moruz (2009), com uma abordagem razoavelmente simples, baseada em alinhamento lexical, com a mesma ideia usada anteriormente no RTE-4 (IFTENE, 2008). Seu método busca mapear os nós da árvore de dependência de H (que são palavras) para nós da árvore de T , exceto por *stopwords*.

Inicialmente, o sistema busca por mapeamentos exatos; isto é, ocorrências da mesma palavra em T e H . Em seguida, recursos lexicais são consultados para substi-

tuir verbos de *H* por outros de *T* que possuam significado próximo, possivelmente com algumas alterações em sua árvore sintática. Os verbos substituídos dessa forma podem então ser alinhados. Outras substituições são aplicadas sobre *H* de modo a tentar alinhar suas palavras para *T*:

- Entidades nomeadas podem ser substituídas por acrônimos (por exemplo, *European Union* para *EU*).
- Números escritos por extenso por algarismos.
- Substantivos e adjetivos listados como sinônimos na WordNet.
- Palavras relacionadas, com conhecimento externo extraído da Wikipedia. Os autores não entram em detalhes sobre as substituições, apenas dão exemplos como *Canadian Prime Minister* e *Prime Minister of Canada*, e *Basel in Switzerland* e *European city*.

Para a decisão final, o sistema considera a proporção de palavras de *H* para as quais houve mapeamento e, se estiver acima de um dado limiar, é emitida decisão positiva. O sistema obteve acurácia de 68,33% no RTE-5.

3.5.1.4 PKUTM (*Sobreposição de Nós em Árvore*)

Jia *et al.* (2010) obtiveram os melhores resultados no RTE-6 com um método bastante parecido com o de Iftene e Moruz (2009) no RTE-5. Seu sistema pré-processa todos os pares candidatos com um módulo resolvidor de correferências (voltado para descobrir os referentes de pronomes em textos com várias sentenças), parser de dependências, detecção de entidades nomeadas e um módulo de expressões regulares para identificar e classificar expressões numéricas (podendo se referir a períodos de tempo, hora, dinheiro, grandezas físicas, etc.).

O funcionamento do sistema é baseado em detectar *nós* correspondentes entre a hipótese e cada texto (lembre-se que no RTE-6, havia vários textos para cada hipótese). São considerados nós as expressões multipalavras, as entidades nomeadas e, caso não sejam parte de nenhum dos dois, as palavras isoladas. Dois nós são considerados correspondentes se a distância de edição entre ambos for menor que um dado limiar (proporcional ao tamanho das palavras comparadas); outras palavras que estejam numa relação de acarretamento lexical também podem ser consideradas para a correspondência. Além disso, algumas regras lexicais foram incorporadas ao sistema para agregar conhecimento de mundo; por exemplo, a correspondência entre *30 anos* e *três décadas*.

O sistema checa negações na árvore de dependências, que pode desencadear o uso de antônimos para a correspondência de nós. A decisão final do sistema é dada verificando-se se a proporção de nós da hipótese possui nós correspondentes no texto. O método

privilegia a precisão, que chega a 68,57%, enquanto a cobertura é de apenas 36,94%, resultando em 48,01 de F_1 . A precisão ser bem mais alta que a cobertura provavelmente se deve ao fato de o sistema depender de regras produzidas manualmente.

3.5.1.5 Similaridade em Duas Etapas

Wang, Zhang e Neumann (2009) usam um modelo que combina representações sintática e semântica, e buscam resolver não apenas casos de inferência, mas também de contradição textual.

Visando evitar o problema observado em sistemas que classificam muitos casos de contradição como acarretamento (já que em ambos os casos, T e H costumam falar sobre as mesmas entidades), seu sistema procede em duas etapas: inicialmente, analisa o par T e H em termos de “similaridade textual” (*textual relatedness*). Nessa dimensão, são distinguidos os pares de textos que apresentam implicação ou contradição dos que não possuem nenhuma das duas relações. Em caso positivo, o sistema classifica numa segunda etapa se se trata de acarretamento ou contradição.

Em seguida, é criada uma representação conjunta das sentenças de T e H na forma de um grafo sintático e semântico. Cada palavra de cada sentença é tratada como um nó no grafo, e relações sintáticas (obtidas de uma árvore de dependência) são arestas, assim como relações semânticas (obtidas por um analisador de papéis semânticos). Além disso, é usado um resolvedor de correferências: cada relação que envolva uma palavra inclui também seus correferentes.

A partir desta representação, o sistema tenta alinhar tanto palavras como relações de T e H . Palavras de duas sentenças são alinhadas quando são idênticas ou se possuem alguma relação segundo recursos como a WordNet e o VerbOcean.

Quando o conteúdo de H pode ser alinhado com T , o sistema emite uma resposta positiva quanto à inferência textual. No entanto, muitas vezes isso não é possível, e nesses casos o sistema recorre a uma estratégia de *backup*: um classificador linear é treinado com atributos extraídos das anotações linguísticas geradas para formar o gráfico, assim como uma *bag-of-words*⁹.

Wang, Zhang e Neumann (2009) obtiveram bons resultados no RTE-5, alcançando a segunda melhor posição na tarefa de classificação ternária (acarretamento, contradição ou nenhum dos dois) com uma precisão de 63.7%.

⁹ *Bag-of-words* é o conjunto de todas as palavras que ocorrem em um dado texto, sem levar sua ordem em consideração. Apesar de ser uma representação bastante superficial, sua simplicidade a tornou bastante comum em algumas tarefas de classificação em PLN, muitas vezes obtendo bons resultados.

3.5.1.6 L2F/INESC-ID

Fialho *et al.* (2016) descrevem a participação da equipe L2F/INESC-ID no ASSIN¹⁰, onde obtiveram os melhores resultados para RIT em português europeu e, posteriormente à divulgação dos resultados do evento, também em português brasileiro. Sua abordagem consistiu em extrair diversas métricas dos pares de sentenças, como distância de edição, palavras em comum (incluindo métricas separadas para entidades nomeadas ou verbos modais), BLEU, ROUGE, etc.

Tais métricas foram computadas tanto das sentenças originais como de outras versões, que poderiam estar em caixa baixa, com palavras radicalizadas, usando clusters de palavras (TURIAN; RATINOV; BENGIO, 2010), entre outras modificações. A combinação de diferentes versões das sentenças com as diferentes métricas gerou mais de 90 atributos para descrever cada par, que são então usados para treinar um Kernel Ridge Regression (para similaridade) e um SVM (para inferência).

Fialho *et al.* (2016) experimentaram ainda aumentar o conjunto de treinamento com uma versão do corpus SICK traduzida automaticamente para o português. No entanto, os resultados obtidos ao se treinar o regressor na versão aumentada foram inferiores, provavelmente devido aos erros de tradução. Por fim, os autores avaliam seus modelos quando treinados em uma variante do português e testados na outra. Apesar da queda de performance, os resultados ainda foram superiores ao uso do SICK traduzido.

3.5.1.7 ASAPP e Reciclagem

As equipes ASAPP e Reciclagem (ALVES; GONÇALO OLIVEIRA; RODRIGUES, 2016) participaram do ASSIN e compartilharam um módulo de análises de relações lexicais baseado em redes semânticas (como tesauros e wordnets). Diversas métricas baseadas em tais relações foram extraídas dessas redes.

O Reciclagem não conta com nenhum módulo de aprendizado de máquina, empregando apenas métricas de similaridade baseadas nas relações semânticas entre as palavras das duas sentenças. Nesse sentido, o método teve um caráter exploratório quanto à capacidade de diferentes redes semânticas contribuírem para a tarefa de avaliação de similaridade semântica (descrita no Capítulo 5) e do quanto um sistema sem treinamento poderia alcançar em termos de performance.

Já o ASAPP emprega, além das métricas usadas pelo Reciclagem, atributos como contagem de tokens de cada sentença, orações nominais, tipos de entidades nomeadas, etc., todos dados como entrada para classificadores e regressores. Em suas três execuções submetidas para a avaliação conjunta, foram exploradas formas de partição de dados, combinação de modelos e redução da quantidade de atributos.

¹⁰ Mais detalhes sobre a avaliação conjunta ASSIN são apresentados no Capítulo 5.

3.5.1.8 Blue Man Group

Barbosa *et al.* (2016) descrevem sua participação no ASSIN, em que utilizaram a estratégia proposta por Kenter e Rijke (2015): são obtidas representações vetoriais das palavras (no caso, foi usado o word2vec) e, em seguida, os vetores de uma sentença são comparados com os da outra, obtendo-se medidas baseadas no cosseno e a distância euclidiana.

Todas as medidas obtidas são então agrupadas em histogramas, com intervalos pré-definidos. São usados diferentes histogramas para cada medida, e as suas contagens são dadas como entrada para os modelos de aprendizado de máquina. Para a tarefa de similaridade, foram usados SVR e o método Lasso, e para a inferência, apenas um SVM.

Também foram explorados métodos baseados em redes neurais recorrentes e convolucionais. No entanto, a despeito dos bons resultados reportados na literatura recente em PLN, as redes neurais obtiveram resultados muito abaixo dos outros métodos usados pela equipe (os valores não foram informados). A provável causa desta disparidade é a quantidade relativamente limitada de dados disponíveis no ASSIN.

3.5.2 Lógica Formal

Sistemas baseados em lógica formal são menos comum na área de RIT. De modo geral, tais sistemas mapeiam o conteúdo de T e de H para algum tipo de forma lógica, e então aplicam técnicas de prova de teoremas.

O principal problema desse tipo de abordagem é a conversão de texto em língua natural para uma forma lógica. Além disso, é necessária também uma base de conhecimentos também codificada na representação formal usada. Aqui são apresentados alguns exemplos da literatura.

3.5.2.1 COGEX

Tatu *et al.* (2006) participaram do RTE-2 com o sistema COGEX, um provador lógico para PLN. Como a maioria dos sistemas desta abordagem de lógica formal, o COGEX recorre às próprias palavras usadas no texto para indicar predicados.

O sistema toma como entrada sentenças pré-processadas de duas formas distintas. A primeira usa a saída de um parser sintático de constituintes, enquanto a segunda usa um parser de dependências, e ambas incluem a identificação de entidades nomeadas. Em sua participação no RTE-2, a tokenização de cada representação era diferente, de modo que as formas lógicas extraídas diferiam entre si.

Para exemplificar sua representação interna, considere-se o caso de textos analisados em árvores de constituintes, em que cada terminal e não-terminal da árvore é

mapeado para uma variável em uma conjunção lógica. No exemplo, um sintagma nominal (SN) contendo o nome *Gilda Flores* é mapeado para a seguinte forma:

Gilda_NN(x1) & Flores_NN(x2) & nn_NNC(x3,x1,x2) & _human_NE(x3)

As variáveis *x1* e *x2* referem-se aos terminais usados como predicados lógicos; a variável *x3* corresponde a um nó não-terminal na árvore sintática. Esse nó corresponde a uma unidade que se refere a uma pessoa, e portanto o último predicado lógico indica se tratar de uma entidade nomeada do tipo humano.

Em casos de negação, em vez de introduzir um predicado para a palavra negadora (como *no* ou *not*), o COGEX nega o predicado associado ao verbo correspondente.

Após a análise sintática, o COGEX usa um módulo de análise semântica, que determina papéis semânticos (como o agente, paciente, tempo ou lugar de uma ação) e algumas relações nominais, como posse e relações presentes em substantivos compostos. Um módulo de raciocínio temporal tenta determinar a ordem dos eventos descritos na sentença, bem como a janela de tempo onde aconteceram.

Para realizar a inferência, o COGEX recorre à WordNet para consultar relações entre palavras, além de uma base de axiomas de diversas naturezas, criada manualmente. Na primeira categoria, por exemplo, para entender quando se refere a uma pessoa por seu último sobrenome, há um axioma que atribui o predicado do token do sobrenome para o não-terminal que governa o nome completo. Considerando o fragmento exemplificado anteriormente:

Flores_NN(x2) $\rightarrow$ Flores_NN(x3)

Dessa forma, caso *H* mencione apenas “*Flores*”, pode ser entendido que se refere à pessoa *Gilda Flores*. Outra categoria foi criada a partir de definições da WordNet, e são mais específicas. Um exemplo de axioma dessa categoria é indicar que o papa é o chefe da Igreja Católica Romana (e vice-versa):

Pope_NN(x1) *rightarrow* head_NN(x1) & of_IN(x1,x2) &
Roman_Catholic_Church_NN(x2) & _organization_NE(x2)

Outros axiomas ainda são usados para detectar relações de parentesco, todo-parte, localização, duração de eventos, etc. O COGEX usa mais de 450 axiomas.

Dada a representação das sentenças de entrada e todos os axiomas disponíveis, o COGEX parte do conteúdo de *T* e a forma negada de *H*, buscando uma prova por refutação, isto é, refutar as hipóteses iniciais, o que implicaria em que *H* pode ser inferido a partir de *T*.

O sistema teve a segunda melhor performance no RTE-2, com 73,75% de acurácia, 10% a mais que o terceiro colocado. A principal desvantagem da abordagem é a necessidade

de uma grande base de conhecimento codificada em axiomas lógicos.

3.5.3 Transformações Textuais

Métodos de transformações textuais transformam pouco a pouco o conteúdo de T em H (ou vice-versa), por meio de substituição, adição e remoção de palavras e expressões. Dependendo do tipo de transformação usada, uma relação de implicação se mantém ou é descartada. Por exemplo, a substituição de sinônimos teria pouco impacto para a implicação, enquanto a introdução de uma negação tende a levar a uma contradição. Nas próximas seções são apresentados alguns exemplos da literatura.

3.5.3.1 Probabilistically Sound Calculus

[Harmeling \(2009\)](#) emprega um conceito que chama de cálculo probabilisticamente livre de erros (*probabilistically sound calculus*), que consiste em aplicar transformações no texto T para se chegar a H e, em seguida, calcular a probabilidade de que as alterações feitas mantenham a relação de acarretamento.

Para fazer as alterações, [Harmeling \(2009\)](#) usa um conjunto de 28 regras definidas manualmente que operam em nível léxico ou sintático. Alguns exemplos de regras são a troca de uma palavra por um sinônimo, hiperônimo, ou hipônimo (uma regra para cada); mudar da voz ativa para passiva, ou vice-versa; adicionar ou remover palavras (uma regra para cada classe gramatical).

O processo de aplicação de regras é regido por uma heurística que busca aplicar o mínimo de transformações possíveis. O sistema mantém uma contagem de cada regra que foi aplicada, e o processo termina quando T é transformado em H .

Em seguida, um classificador automático determina se a relação de acarretamento foi mantida. Os atributos de entrada do classificador são as contagens de quantas aplicações de cada regra foram necessárias para a transformação. Para cada instância, são avaliadas as oito sequências mais curtas de regras e, se qualquer uma for classificada como positiva, considera-se que há acarretamento.

[Harmeling \(2009\)](#) reporta que os resultados de seu método ficaram abaixo do estado-da-arte ao ser avaliado com dados do RTE-2 e RTE-3, com acurácia de 56,39% e 57,88%, respectivamente. Apesar de o método apresentar uma boa motivação teórica, uma possível razão para o baixo desempenho é o fato de o classificador tratar cada exemplo como o conjunto de todas as regras aplicadas, e não cada uma isoladamente. Por exemplo, caso as regras aplicadas em uma instância forem $\{R1, R2, R5\}$, o classificador trata o conjunto como um único item. Isso pode ser problemático em casos que apenas uma regra é responsável pela perda da relação de acarretamento.

Uma das dificuldades desta abordagem é o fato de que a aplicação de regras de transformação sobre T pode gerar um número de variantes que cresce exponencialmente. Harmeling (2009), assim como outros pesquisadores, restringiu o processo por meio de heurísticas que minimizam a geração de novas árvores, conforme explicado anteriormente. Já Bar-Haim *et al.* (2009) introduziram uma estrutura de dados chamada floresta compacta (*compact forest*) para contornar o problema.

A floresta compacta trabalha com árvores de dependência, e reaproveita os mesmos nós (palavras ou pontuação) que são usados em diferentes versões de T . Cada relação de dependência na floresta (uma aresta na árvore) tem a indicação de qual árvore pertence.

3.5.3.2 BIUTEE

O BIUTEE (STERN; DAGAN, 2011; STERN; DAGAN, 2012) é outro método que deriva o conteúdo de H por meio de transformações em T . Sua ideia central é semelhante ao cálculo probabilístico de Harmeling (2009): aplicar transformações parciais em T até chegar a H , e determinar se o valor verdade da proposição inicial foi mantido. Stern e Dagan (2011) chamam o processo de transformações de prova (*proof*), em analogia às provas de lógica formal.

O sistema inicialmente aplica transformações que possuem alta probabilidade de manter o significado do texto (e, conseqüentemente, a relação de acarretamento). Essas se dividem em dois grupos, ambos comuns nos trabalhos da área: regras de acarretamento (*entailment rules*) e substituições de correferência.

Regras de acarretamento são basicamente substituição de palavras ou expressões do texto por outras equivalentes. Podem funcionar no nível lexical (como *viajar* $\rightarrow$ *ir*) ou no nível sintático. No nível sintático, se dividem ainda em regras específicas para certos verbos, aprendidas pelo DIRT¹¹ (LIN; PANTEL, 2001); ou regras genéricas, como a substituição de voz ativa por passiva.

Já as correferências são resolvidas por uma ferramenta externa. Uma vez identificadas duas menções a uma dada entidade, não necessariamente na mesma sentença, o sistema experimenta substituir uma pela outra. A Figura 6 ilustra exemplos de aplicações de regras do BIUTEE, com pares extraídos e adaptados do RTE-3.

Apenas a aplicação destas transformações normalmente não é suficiente para que o conteúdo de T se torne igual ao de H . Nesse caso, podem ser feitas novas alterações, conforme necessário, sobre a árvore sintática de T , chamadas de operações de tempo real (*on the fly*).

¹¹ O DIRT é uma base de dados de ocorrências de diversos verbos em inglês com diferentes argumentos (sujeito, objeto e alguns modificadores) que pode ser usado para encontrar verbos com padrões de uso semelhante e, supostamente, significado semelhante.

[Original]	O dono original, Henry Borski, abriu a taverna em 1945. Mais tarde, ele a passou para seu filho Jake Borski (...)
[Hipótese]	Henry Borski e Jake Borski são parentes.
Substituição Lexical	filho $\rightarrow$ parente
Correferência	ele $\rightarrow$ Henry Borski
[Final]	O dono original, Henry Borski, abriu a taverna em 1945. Mais tarde, Henry Borski a passou para seu parente Jake Borski (...)
[Original]	A Nival foi fundada in 1996 por Sergey Orlovskiy. No começo de 2005, a empresa foi comprada pelo grupo Ener1, uma holding da Flórida, por US\$10 milhões.
[Hipótese]	A Nival foi vendida em 2005.
Correferência	a empresa $\rightarrow$ a Nival
Substituição Léxico-sintática	X foi comprada por Y $\rightarrow$ X foi vendida para Y
[Final]	Nival foi fundada in 1996 por Sergey Orlovskiy. No começo de 2005, a Nival foi vendida para o grupo Ener1, uma holding da Flórida, por US\$10 milhões.

Figura 6 – Exemplos de transformações do BIUTEE.

Enquanto as regras ilustradas na Figura 6 são baseadas em conhecimento linguístico extraído de bases como a WordNet e o DIRT, as operações de tempo real são bastante genéricas e, portanto, possuem menor chance de manter a relação de acarretamento. Exemplo de tais operações são a inserção de nós na árvore sintática, o movimento de um nó na árvore, mudança de classe gramatical de uma palavra, etc.

Após todas as transformações serem feitas, seu impacto sobre o significado de T é mensurado. Cada transformação feita sobre o texto é associada a um valor de penalidade que reflete o quanto sua aplicação pode causar perda da relação de acarretamento. Por exemplo, regras do DIRT possuem uma pontuação de 0 a 1 que reflete o quão confiável a regra é, e o BIUTEE usa o logaritmo deste valor como penalidade para transformações baseadas nelas. Um raciocínio semelhante é empregado para dados provenientes de outros recursos; quando não há uma pontuação, é usado -1.

As operações de tempo real possuem penalidades maiores, visto que podem trazer alterações substanciais no significado da sentença. Por exemplo, a inserção de uma palavra tem como custo o logaritmo da probabilidade de sua ocorrência em um grande corpus;

normalmente, esses valores são próximos a -3 para palavras comuns, podendo chegar a -10 ou menos para outras mais raras.

As penalidades de cada transformação são então tratadas como atributos de entrada para um classificador automático. Se o resultado estiver acima de um dado limiar, considera-se que a relação de acarretamento foi mantida. O BIUTEE foi avaliado no RTE-6 e RTE-7 (posteriormente à realização dos eventos), obtendo acurácias de respectivamente 49,09% e 42,93%. No RTE-6, o valor supera todos os participantes do evento, enquanto no RTE-7 está somente acima da mediana.

3.5.3.3 Modelos de Edição de Árvore para RIT

Heilman e Smith (2010) apresentam uma melhoria nos algoritmos de edição de árvore para dar conta de certas deficiências, como mostrado na Seção 3.4. Uma das principais limitações do TED original é a falta de um mecanismo para mover uma subárvore. Nestes casos, o TED contabiliza a distância de remover todos os nós da subárvore e readicioná-los no ponto apropriado, o que se reflete em um custo mais alto do que o intuitivamente esperado. Para contornar esta limitação, os autores definem as seguintes novas operações de TED:

Relabel Node Troca a palavra associada a um nó, efetivamente mudando seu lema e POS tag.

Relabel Edge Troca a etiqueta de um arco de dependência.

Move Subtree Move uma subárvore para ser um novo filho de outro nó.

New Root Determina um novo nó raiz para a sentença, e coloca o antigo como seu filho.

Move Sibling Muda a ordenação dos filhos de um nó.

As melhorias propostas, no entanto, vêm com o custo de aumentar significativamente o espaço de busca para se encontrar a menor sequência de transformações de uma árvore em outra. Enquanto para o TED original há algoritmos com tempo de execução polinomial, os mesmos não podem ser aplicados com as novas operações.

Portanto, é usada uma busca gulosa guiada por uma heurística que estima o custo de edição restante para se realizar as transformações faltantes. Os autores reportam que na maioria dos casos, a busca chega a uma sequência de transformações razoáveis, mas impõem um limite máximo de tentativas de transformação para as exceções em que o algoritmo se aprofunda demais na busca sem encontrar uma solução.

Para classificar pares quanto ao RIT, agregam à sequência de transformações obtida pelo TED a contagem de atributos como a quantidade de nomes próprios inseridos ou removidos.

As operações aplicadas para transformar T em H são agrupadas em um vetor de 33 atributos, usado para treinar um classificador linear. Avaliado com dados do RTE-3, o modelo obtém acurácia de 62,8%, precisão de 61,9% e cobertura de 71,2%. Apesar de não superar o estado da arte, seus criadores atentam para o fato de não possuir uma engenharia de atributos extensiva e especializada para o problema, além de servir para outras tarefas que lidem com pares de sentenças.

3.5.3.4 AdArte

Zanoli e Colombo (2016) apresentam o AdArte (*a transformation-driven approach for RTE*), um sistema para RIT baseado em transformações sintáticas em T para chegar até H . O sistema foi desenvolvido tendo em mente sentenças mais simples e em maior quantidade que os conjuntos dos RTE Challenge, como as que aparecem no SICK.

Em contraste com as abordagens de Harmeling (2009) e Stern e Dagan (2012), o AdArte aprende transformações sintáticas automaticamente, em vez de trabalhar com um conjunto pré-definido. Outra diferença é que este sistema gera uma representação vetorial para a sequência de transformações realizadas, que é dada para um classificador automático, em vez de estimar a probabilidade de cada transformação manter a relação de implicação.

O AdArte trabalha com as três operações básicas de TED: inserção, remoção ou substituição de um nó na árvore sintática (no caso da remoção, os filhos de um nó passam a ser conectados com seu pai). Quando dois nós são equivalentes nas duas árvores, é aplicada uma operação de *match*, equivalente a uma substituição com custo zero. Embora não tenha impacto para o TED, contabilizar operações de *match* indica a quantidade de palavras em comum, e portanto é útil para o RIT. Para todos os efeitos, dois nós são considerados equivalentes quando representam (i) palavras com o mesmo lema, sinônimos, ou casos em que a palavra de T seja um conceito mais específico que a de H ; e (ii) a mesma relação de dependência com o nó pai. Para consulta de sinonímia e palavras mais específicas, são usadas a WordNet, o CatVar e VerbOcean.

Cada alteração feita em uma árvore é representada para o classificador final apenas como o nome da operação e as relações de dependência dos nós envolvidos. Dessa forma, cada transformação é vista de forma bastante genérica, o que é justificado como uma tentativa de diminuir o *overfitting*¹² aos dados. No entanto, é uma representação muito pouco informativa. Considere-se o exemplo a seguir, com as transformações extraídas para transformar a primeira sentença ilustrada na Figura 7 na segunda (os índices subscritos servem para facilitar a visualização).

¹² Overfitting acontece quando um classificador lida com representações muito específicas de uma quantidade relativamente pequena de dados, o que prejudica sua capacidade de generalização.

1. Substituir(T_2 :nsubj, H_2 :nsubjpass)
2. Substituir(T_1 :det, H_1 :det)
3. Inserir(H_3 :auxpass)
4. Inserir(H_6 :case)
5. Substituir(T_6 :dobj, H_8 :pobj)

Figura 7 – Árvore sintáticas de duas sentenças

3.5.3.5 Operações sobre Subárvores e Metaheurísticas para TED

Alabbas e Ramsay (2013) apresentam um modelo que tenta corrigir a limitação dos algoritmos de TED para lidar com subárvores. Sua proposta consiste em executar o algoritmo de Zhang-Shasha verificando a cada passo se há uma sequência de inserções que corresponda a uma única subárvore da segunda sentença, ou remoções de uma subárvore da primeira, ou substituições de subárvores nas duas. Em caso positivo, em vez de serem tratadas como operações independentes sobre vários nós, é considerada uma única operação especial, com custo reduzido.

Para determinar os custos das operações, tanto sobre tokens como sobre subárvores, os autores experimentaram o uso de meta-heurísticas. Foram diferenciadas operações que envolvessem stopwords ou não, além de operações de substituição de sinônimos e de antônimos. A decisão quanto à presença de implicação é tomada baseada em um limiar do valor de TED, aprendido automaticamente. No caso de decisão com três classes, são considerados dois limiares, que definem três intervalos.

O método proposto foi avaliado em um corpus de RIT em Árabe, relativamente pequeno, e no RTE-2. Tanto o uso de operações especiais para subárvores quanto o uso de metaheurísticas para determinar custos rendeu melhorias de performance em comparação com versões que não os usassem. No RTE-2, obtiveram 0.56 de F_1 .

3.6 Considerações Finais

Neste capítulo, foram apresentados conceitos de aprendizado de máquina usados para o RIT, além de bases de conhecimento que auxiliam sistemas a encontrar alinhamentos entre sentenças. Foi também apresentado o conceito de TED, usado por alguns sistemas de RIT e também na pesquisa da presente tese, como será descrito no Capítulo 6.

Foram listados vários trabalhos da literatura para tratar o RIT, agrupados em três categorias principais, trazendo uma visão ampla das técnicas utilizadas e tipos de atributos definidos para aprendizado de máquina. Das duas maiores categorias, a saber, similaridade/alinhamento e transformações textuais, são recapitulados alguns trabalhos nas Tabelas 6 e 7, que mostram um comparativo de suas características.

Modelo	Recursos externos		Ferramentas	Distâncias	Ponderação	Performance
EDITS	Wordnet		parser	Lexical, TED	IDF	61,4% ac. RTE-5
IKOMA	Acrônimos, Wordnet, CatVar		POS tagger, APS, REN, chunker	Lexical	IDF	0,48 F ₁ , RTE-7
UAIC	Wordnet, Wiki- pedia, DIRT		POS tagger, REN, parser	Lexical	Filtragem de stopwords	68,3% ac. RTE-5
PKUTM	Wordnet, Verbocean		parser, RCR	REN, Lexical / multi- palavra	—	0,48 F ₁ , RTE-6
Wang et al.	Wordnet, Verbocean		parser, RCR	APS, Lexical	—	63,7% ac. RTE-5
INESC-ID	Clusters de pala- vras (não super- visionado)		REN	Entidades no- meadas, verbos modais, nega- ção, palavras, n-gramas	—	0,71 (PT-PT) 0,66 (PT-BR)
Reciclagem	Wordnets tesauros	e	POS tagger, le- matizador, REN	Palavras, lemas, negação	vizinhança, distância na wordnet	0,43 (PT-PT) 0,39 (PT-BR)
ASAPP	Wordnets tesauros	e	POS tagger, le- matizador, REN	Palavras, lemas, negação	vizinhança, distância na wordnet	0,59 (PT-PT) 0,50 (PT-BR)
Blue Group	Man Word embeddings		—	Palavras	IDF, cosseno de embeddings	0,61 (PT-PT) 0,52 (PT-BR)

Tabela 6 – Sumário de sistemas baseados em similaridade usados nos RTE Challenges (parte superior) e ASSIN (parte inferior). A sigla APS indica anotação de papéis semânticos; POS indica part-of-speech; RCR indica reconhecimento de correferências; REN significa reconhecimento de entidades nomeadas. Na coluna de Performance, *ac.* indica acurácia. Todos os valores no ASSIN são de F₁

Modelo	Recursos externos	Comparação	Custos	Decisão	Performance
BIUTEE	DIRT, WordNet, CatVar	Operações próprias	Dependente de operação	Classificador sobre vetor de atributos com custos das transformações usadas	65,4% (RTE-3), 49,09% (RTE-6), 42,93% (RTE-7)
Cálculo probabilístico	WordNet	Operações próprias	—	Classificador sobre vetor de atributos com transformações usadas	56,38% (RTE-2), 57,88% (RTE-3)
AdArte	WordNet, CatVar, VerbOcean	TED	Fixos em 1	Classificador sobre vetor de atributos com transformações usadas	82,4% (SICK)
Alabbas e Ramsay	WordNet	TED	Divididos em categorias e otimizados por metaheurísticas	Limiar da TED	56,8% (RTE-2)

Tabela 7 – Sumário de sistemas baseados em transformações textuais.

ABORDAGENS DE REPRESENTAÇÕES DISTRIBUÍDAS E REDES NEURAIS

Representações distribuídas de palavras e textos, bem como redes neurais com milhares de parâmetros, ganharam muito espaço na comunidade de PLN nos últimos anos (GOLDBERG, 2016). A partir da publicação do dataset SNLI (vide Seção 2.5.3), em 2015, esta linha de pesquisa também chegou ao RIT. Neste capítulo, são apresentados alguns conceitos de *deep learning* e abordagens que os utilizam para o RIT.

4.1 Motivação

4.1.1 Limitações da Engenharia de Atributos

Como mencionado na Seção 3.2, a representação dos dados é fundamental para o bom funcionamento de algoritmos de aprendizado de máquina. Os métodos apresentados no Capítulo 3 têm diferentes formas de representar os pares $\langle T, H \rangle$: alguns usam uma série de atributos particulares, como quantidade de palavras em comum e distância de edição; outros, apenas a proporção de palavras alinhadas entre as duas sentenças; outros ainda, uma sequência de transformações nas sentenças.

Todos estes têm em comum o fato de usarem critérios explicitamente definidos para chegar às representações, como a contagem de palavras, o que buscar nas ontologias, as estratégias de alinhamento, as sequências de transformações válidas, etc. Embora funcionem razoavelmente bem para o problema de RIT, estas representações sofrem de uma limitação fundamental: a necessidade de serem definidas manualmente.

Inevitavelmente, isto esbarra no problema de que não sabemos explicar precisamente o que é necessário observar em um par de sentenças para determinar a presença ou não de implicação (vide Seção 2.1). Ao se pensar sobre a implicação textual em um par

de sentenças, ainda que possamos explicar porque chegamos a determinada decisão, não conseguimos elaborar uma explicação formal e genérica que se aplique a qualquer caso.

4.1.2 Aprendizado de Máquina com Representações Distribuídas

Nesta seção, são descritas formas alternativas para a representação de dados textuais em problemas de PLN, bem como algumas características desejadas nos modelos que os processam.

4.1.2.1 Representação de Palavras

A alternativa à engenharia de atributos é fazer com que o próprio modelo de aprendizado de máquina crie uma representação para os dados. Este é o princípio do aprendizado de atributos (*feature learning*) ou aprendizado de representação (*representation learning*) (GOODFELLOW; BENGIO; COURVILLE, 2016, cap. 1).

Neste cenário, os dados são apresentados ao modelo sem se explicitar atributos. Em vez destes, é definida uma forma simples e genérica de converter os dados em representações numéricas. No caso de textos, uma abordagem comum é o uso de *word embeddings*, que são a projeção de palavras para um espaço multidimensional¹. Isto significa que cada palavra do vocabulário é associada a um vetor de números reais, podendo ter de algumas dezenas a centenas de dimensões. Na prática, não apenas palavras, mas qualquer *token*, como números, abreviações, sinais de pontuação, etc. são projetados.

Representações distribuídas são assim chamadas porque todos os elementos dos vetores que representam as unidades básicas (no caso, palavras) contém um valor independente e com algum significado para o modelo (LECUN; BENGIO; HINTON, 2015). Em contraste, representações definidas manualmente muitas vezes usam atributos binários, implementados como um vetor que possui o valor 1 para indicar a presença de um atributo (como uma classe gramatical) e 0 em todas as outras posições (nas outras classes gramaticais).

Formalmente, um *token* w é mapeado para um vetor $\mathbf{w}_e \in \mathbb{R}^d$. Este mapeamento indexa cada palavra ou símbolo de um vocabulário² V a uma linha de uma matriz $\mathbf{W}^{|V| \times d}$. Assim, uma sentença S composta por uma sequência de *tokens* $\{w_1, w_2, \dots, w_m\}$ é mapeada para uma sequência de m vetores de dimensão d , podendo ser representada como uma matriz $\mathbf{M}^{m \times d}$.

¹ Outra abordagem possível são *character embeddings*, em que cada caractere é mapeado para o espaço multidimensional (SUTSKEVER; MARTENS; HINTON, 2011).

² Palavras ou símbolos fora do vocabulário previamente definido, encontrados quando o sistema é usado com novos textos, exigem um tratamento especial. Uma abordagem simples é criar uma pseudo-palavra w_{UNK} para representar qualquer símbolo desconhecido, mas há outros métodos mais elaborados (BOJANOWSKI *et al.*, 2016).

A principal propriedade do espaço onde os *tokens* são projetados é a similaridade entre vetores de *tokens* relacionados — espera-se que palavras com significado ou uso próximo tenham também vetores próximos. A similaridade entre dois vetores pode ser mensurada pelo ângulo entre ambos, como ilustrado na Figura 8a. Vê-se que, *carro* e *veículo*, conceitos similares, possuem um ângulo θ pequeno entre si, enquanto o ângulo ϕ entre *veículo* e *casa* é bem maior.

Figura 8 – Ângulos entre vetores. Observe-se que a normalização de vetores mantém o mesmo ângulo entre os mesmos.

Na prática, em vez de se calcular o ângulo entre dois vetores, é mais comum calcular seu cosseno, por ser computacionalmente mais simples. O valor do cosseno varia entre -1 e 1, e quanto maior for, maior é a similaridade dos vetores. O cosseno entre dois vetores é obtido da seguinte forma:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}| |\mathbf{y}|} = \frac{\sum_i \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_i \mathbf{x}_i^2} \sqrt{\sum_i \mathbf{y}_i^2}} \quad (4.1)$$

O denominador da equação normaliza o produto interno dos dois vetores pelo produto das suas normas, e pode ser omitido quando ambos têm norma igual a 1 (ditos normalizados). A norma de um vetor pode ser entendida como o seu comprimento; o efeito da normalização é ilustrado na Figura 8.

Outras propriedades relevantes de espaços vetoriais frequentemente exploradas por modelos neurais de PLN são as relações de diferença e produto elemento a elemento entre dois vetores normalizados. A primeira indica similaridade entre os vetores na medida em que os valores resultantes se aproximam de zero (posições em que ambos os vetores têm

valores muito próximos terão diferença próxima a zero). Em outras palavras, a norma do vetor de diferença é inversamente proporcional à semelhança entre os vetores.

Já o produto, denotado pelo operador $\odot$, funciona de modo oposto: quanto maior for o valor de uma dimensão, maior a similaridade entre os vetores na mesma. Além disso, um valor negativo indica que os vetores têm valores de sinais opostos naquela dimensão, independente da sua magnitude. Vetores de diferença e produto são ilustrados na Figura 9.

Figura 9 – Diferenças e produtos entre vetores normalizados

Há diversos métodos para geração de embeddings, que buscam otimizar a similaridade entre palavras seguindo diferentes critérios (MIKOLOV *et al.*, 2013; PENNINGTON; SOCHER; MANNING, 2014; BOJANOWSKI *et al.*, 2016). Uma descrição de tais métodos está além do escopo do presente doutorado.

O uso bem sucedido de word embeddings em sistemas de PLN remonta a Collobert e Weston (2008), que tratavam de etiquetagem morfosintática (ou *POS tagging*)³, detecção de entidades nomeadas⁴ e anotação de papéis semânticos⁵. Desde então, diversas outras

³ Etiquetagem morfosintática é a classificação de palavras quanto à sua classe gramatical, como substantivo, adjetivo, verbo, etc.

⁴ Entidades nomeadas são nomes no texto que se referem a pessoas, instituições, cidades, regiões, etc.

⁵ Papéis semânticos são elementos em sentenças que podem ser classificados como agente,

aplicações de PLN se beneficiaram de word embeddings (COLLOBERT, 2011; SOCHER *et al.*, 2012; SOCHER *et al.*, 2013; CHO *et al.*, 2014; TANG *et al.*, 2014, *inter alia*).

Com este tipo de representação, o sistema não fica preso a atributos previamente definidos, podendo descobrir quaisquer relações entre as palavras de $\langle T, H \rangle$ e a presença de implicação. Por outro lado, perde-se o conhecimento codificado em atributos pré-definidos — ainda que estes sejam indicadores limitados, são escolhidos por terem uma boa correlação com a classe que se deseja prever. Além disso, as decisões do modelo se tornam menos transparentes.

4.1.2.2 Modelos de Aprendizado de Máquina

Por conta da supracitada maior sutileza das representações baseadas em embeddings, são necessários modelos de aprendizado de máquina mais poderosos do que os usados com atributos definidos manualmente. Entendem-se por modelos *mais poderosos* os que tenham mais parâmetros (ou pesos) treináveis, e estruturados de forma razoável. Comumente, são implementados como redes neurais, uma família de modelos de aprendizado de máquina que podem assumir diversas formas e propriedades, tendo em seu cerne a aplicação de transformações lineares (multiplicação de matrizes de pesos).

Por terem uma grande quantidade de parâmetros treináveis, redes neurais necessitam de grandes quantidades de dados para obterem boa performance. Ao serem treinadas em conjuntos de dados relativamente pequenos, é comum que seus parâmetros se superajustem aos dados de treinamento (i.e., haja *overfitting*), e o modelo não seja capaz de efetivamente aprender a resolver aquela tarefa de modo generalizável.

Portanto, como regra geral, a engenharia de atributos é mais eficaz que o aprendizado de representações quando há uma quantidade pequena de dados; conforme a quantidade de dados aumenta, a situação se inverte, pois torna-se possível uma rede neural inferir mais generalizações a partir dos dados que dificilmente estariam visíveis em atributos pré-definidos. Ainda assim, com uma boa configuração de hiperparâmetros, redes neurais profundas podem alcançar ou superar modelos mais simples mesmo com poucos dados em certas condições (BEAM, 2017).

Finalmente, outro fator a ser considerado é o tempo de treinamento, que pode ser muito demorado para redes neurais profundas devido à quantidade de parâmetros a serem ajustados. Os últimos anos têm visto inovações tanto de hardware, com o aprimoramento GPUs (NVIDIA, 2015), como de software, com o desenvolvimento de bibliotecas otimizadas (ABADI *et al.*, 2015; AL-RFOU *et al.*, 2016), para treinamento mais rápido de redes neurais em larga escala; mesmo assim, o treinamento de modelos com grandes quantidades de parâmetros pode levar muitas horas. Em contraste, modelos lineares sim-

ples podem ser treinados em alguns segundos ou minutos. A Tabela 8 resume diferenças entre métodos de aprendizado de máquina clássico e deep learning.

No caso de RIT, pesquisadores passaram a usar estratégias baseadas em representações distribuídas e redes neurais a partir da disponibilização do SNLI, que conta com 550 mil exemplos de treinamento. Os modelos que alcançaram bons resultados têm uma quantidade de parâmetros que varia desde 100 mil parâmetros a mais de 15 milhões.

	Aprendizado de máquina clássico	Deep learning
Complexidade dos modelos	Simples; modelos lineares são comuns	Complexa; várias não-linearidades conectadas
Dados de entrada	Atributos definidos manualmente	Apenas as palavras ou caracteres
Tempo de treinamento	Rápido	Demorado
Performance com poucos dados	Tende a ser boa	Tende a ser ruim; hiperparametrização cuidadosa pode ajudar
Performance com muitos dados	Deixa de melhorar a partir de certo limiar	Tende a ser muito boa
Principal limitação de performance	Qualidade dos atributos	Arquitetura do modelo

Tabela 8 – Comparação entre aprendizado de máquina clássico e *deep learning*

4.2 Conceitos de Redes Neurais para RIT

Nesta seção, são apresentados brevemente alguns conceitos de redes neurais e deep learning relevantes para a tarefa de RIT. Uma visão aprofundada do uso de redes neurais em PLN pode ser encontrada em [Goldberg \(2016\)](#); conceitos mais aprofundados e em diversas aplicações são bem descritos por [Goodfellow, Bengio e Courville \(2016\)](#).

4.2.1 Redes Feedforward

A ideia básica de redes neurais é a aplicação de transformações não-lineares sobre dados. Dado um vetor numérico de entrada $\mathbf{x} \in \mathbb{R}^d$, é aplicada uma transformação linear (multiplicação por uma matriz $\mathbf{W} \in \mathbb{R}^{d \times k}$ e soma com um vetor $\mathbf{b} \in \mathbb{R}^k$), seguida pela aplicação de uma função não-linear:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (4.2)$$

Esta transformação representa uma *camada* da rede neural, e k é dito o número de *neurônios*, ou *unidades*, da camada. Funções não-lineares comumente usadas são a tangente hiperbólica, $\tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$, e a relu (*rectified linear unit*), $\text{relu}(x) = \max(0, x)$. A aplicação de tais funções permite encontrar mais padrões nos dados, o que não seria possível apenas com transformações lineares. Este tipo de camada também é chamado de camada *densa*, já que todas as suas unidades dependem de todas as entradas para calcular seus respectivos valores; ou seja, as unidades são *densamente* conectadas.

Para aumentar o poder preditivo do modelo, mais camadas de transformações podem ser postas em sequência, aumentando a sua profundidade — o que inspirou o nome *deep learning* (GOODFELLOW; BENGIO; COURVILLE, 2016, cap. 6). Um modelo composto por uma sequência de transformações não-lineares é chamado de *multilayer perceptron* (MLP), ou de rede *feedforward*:

$$\text{MLP}(\mathbf{x}) = f_N(\mathbf{W}_N \cdot \dots \cdot f_2(\mathbf{W}_2 \cdot f_1(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_N) \quad (4.3)$$

É muito comum que mesmo modelos mais complexos (como os de redes recorrentes) empreguem pelo menos uma camada feedforward para a decisão final sobre um problema de classificação. Esta camada final tem um neurônio para cada classe do problema (no caso do RIT clássico, três: implicação, contradição e neutro), e aquele que tiver o maior valor após o processamento da entrada representa a resposta da rede. Na camada final, normalmente é empregada a função *softmax* para transformar os valores de cada unidade, que variam de $-\infty$ a ∞ em probabilidades:

$$\text{softmax}(\mathbf{y})_i = \frac{e^{y_i}}{\sum_j e^{y_j}} \quad (4.4)$$

Redes neurais são treinadas com o algoritmo de *backpropagation*, que usa cálculo diferencial para determinar ajustes em todas as matrizes de pesos. Dessa forma, os resultados produzidos pela rede se adequam às classes dos exemplos de treinamento.

4.2.2 Redes Recorrentes

Apesar de eficiente, uma rede MLP lida apenas com vetores de entrada de tamanho fixo, enquanto sentenças em língua natural têm tamanhos variáveis. Por causa disso, outros tipos de arquiteturas neurais são necessários para o seu tratamento.

O tipo de arquitetura mais usada em RIT são as redes neurais recorrentes (RNR), que tratam uma palavra a cada passo de execução. A forma mais simples de uma RNR é

a seguinte:

$$\begin{aligned} \mathbf{h}_0 &= \{0\}^d \\ \mathbf{h}_{t+1} &= f(\mathbf{W}_x \mathbf{x}_{t+1} + \mathbf{W}_h \mathbf{h}_t + \mathbf{b}) \end{aligned} \quad (4.5)$$

Nesta formulação, uma camada de rede neural combina o vetor de entrada correspondente ao t -ésimo token da sentença com a sua saída $\mathbf{h}_{t-1}$ produzida no passo anterior. No primeiro passo de execução, o vetor $\mathbf{h}_0$ é inicializado com zeros. No entanto, o treinamento deste tipo de modelo não é eficiente, pois sofre de grandes limitações para aprender a influência de cada palavra em sequências longas (BENGIO; SIMARD; FRASCONI, 1994).

Uma melhoria de RNR largamente usada em PLN são as LSTMs (*long short-term memory*) (HOCHREITER; SCHMIDHUBER, 1997), um tipo de camada de rede neural capaz de ser treinada eficientemente mesmo com sequências longas. Uma LSTM é uma extensão da RNR básica que mantém um vetor numérico c armazenando seu estado interno (podendo representar a *memória* da camada), e incorpora portões (*gates*) que controlam o quanto cada nova entrada x_t altera seu estado interno ou influencia seu vetor de saída no passo t .

$$\begin{aligned} \mathbf{h}_0 &= \{0\}^d \\ \mathbf{c}_0 &= \{0\}^d \\ \mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\ \mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\ \tilde{\mathbf{c}}_t &= f(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\ \mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t \\ \mathbf{h}_t &= \mathbf{o}_t \circ f(\mathbf{c}_t) \end{aligned} \quad (4.6)$$

Nas equações acima, $\circ$ denota a multiplicação de vetores elemento a elemento, e σ é a função sigmoide, cuja saída está faixa $[0, 1]$. O vetor $\mathbf{f}_t$, chamado *forget gate*, controla o quanto do conteúdo de $\mathbf{c}_{t-1}$ — o estado interno no passo anterior — será esquecido ou lembrado no passo t . Similarmente, $\mathbf{i}_t$, chamado *input gate*, controla o quanto dos novos valores $\tilde{\mathbf{c}}_t$ serão incorporados a $\mathbf{c}_t$. Já $\mathbf{o}_t$, chamado *output gate*, determina o quanto do valor de $f(\mathbf{c}_t)$ será emitido como a saída $\mathbf{h}_t$ da LSTM.

O vetor $\mathbf{h}_t$ é usado para os cálculos do passo seguinte $t+1$ desta camada e, caso haja outra camada na rede neural, será a sua entrada no mesmo passo t . Alguns modelos de RIT usam apenas o último vetor de saída da rede (isto é, correspondente ao último token

da sentença, mas que teve influência de todos os anteriores), enquanto outros combinam as saídas de cada passo. Além disso, é possível usar de formas diferentes os vetores produzidos para T e para H , além de usar a mesma rede neural (isto é, com os mesmos pesos) para ambas ou redes diferentes.

Por fim, uma técnica que permite aumentar o poder preditivo de LSTMs é combinar uma camada que lê a sentença de entrada na ordem normal com outra que a lê de trás para frente. Isto soluciona o viés de o estado interno da LSTM conter pouca informação a respeito dos últimos tokens de uma sentença. Nesta técnica, chamada de LSTM bidirecional (ou BiLSTM), os vetores h_t de cada camada são concatenados com a representação do t -ésimo *token*.

4.2.3 Atenção

Uma dificuldade do treinamento de RNRs, incluindo LSTMs, é o tratamento de relações de longa distância; isto é, tokens que não estejam próximos entre si mas que têm algum tipo de relação importante para o entendimento da sentença. Por exemplo, considere o par a seguir.

- (1)
 - a. Um homem caminha ao lado da esposa enquanto carrega uma criança no colo.
 - b. Um homem está carregando uma criança.

Para analisar o par corretamente, a ponto de identificar a relação de implicação, o modelo deveria considerar alguns tokens relativamente distantes na primeira sentença: *homem*, *carrega* e *criança*, pelo menos. Um uso simplificado de RNRs (incluindo LSTMs) considera apenas o conteúdo do seu último vetor de saída para processar a sentença, o que tende a criar um gargalo de armazenamento de informação.

O mecanismo de atenção é um artifício usado em RNRs que permite ao modelo rever os vetores de tokens já processados ou consultar os tokens de uma outra sentença, mitigando o efeito de gargalo. A atenção é utilizada em outros problemas de PLN, com um dos seus usos pioneiros na tradução automática (BAHDANAU; CHO; BENGIO, 2014), em que o modelo consulta tokens da sentença da língua origem enquanto gera uma tradução. A sua aplicação sobre vetores já processados, para a geração de uma melhor representação sentencial, também já foi realizada com sucesso (ZHANG; YANG, 2017). Por fim, a atenção não é limitada a RNRs, podendo ser aplicada a modelos compostos apenas por redes feedforward. É o caso de Vaswani *et al.* (2017), que alcançam resultados de estado-da-arte para tradução automática.

Dada uma sentença S e um *token* w , que pode ser parte de S ou não, a atenção consiste em calcular a distribuição de relevância de cada token $s_i \in S$ para w ⁶. Sendo

⁶ Esta distribuição também pode ser entendida como o alinhamento entre w e S

$\mathbf{x} \in \mathbb{R}^k$ a representação vetorial de w (como um vetor de embedding ou a saída de uma LSTM para a posição de x) e $\mathbf{M}_S \in \mathbb{R}^{|S| \times k}$ a representação matricial de S (a combinação dos vetores de todos os seus tokens), o cálculo de atenção é o seguinte:

$$\mathbf{z} = f_\alpha(\mathbf{x}, \mathbf{M}_S) \quad \mathbf{z} \in \mathbb{R}^{|S|} \quad (4.7)$$

$$\alpha = \text{softmax}(\mathbf{z}) \quad \alpha \in \mathbb{R}^{|S|}, \sum_i \alpha_{[i]} = 1 \quad (4.8)$$

$$\mathbf{r} = \alpha \mathbf{M}_S \quad \mathbf{r} \in \mathbb{R}^k \quad (4.9)$$

A notação $\alpha_{[i]}$ indica a i -ésima dimensão do vetor α . A função f_α é responsável por determinar a importância não-normalizada de cada *token* de S para w . Sua implementação varia, havendo duas famílias principais (SHEN *et al.*, 2017). A primeira emprega uma combinação não-linear de $\mathbf{x}$ e $\mathbf{M}_S$:

$$f_\alpha(\mathbf{x}, \mathbf{M}_S) = \mathbf{w} f(\mathbf{x} \otimes \mathbf{ones}^n \mathbf{W}_x + \mathbf{M}_S \mathbf{W}_S)^T \quad (4.10)$$

onde há os parâmetros ajustáveis das matrizes $\mathbf{W}_x, \mathbf{W}_S \in \mathbb{R}^{k \times k}$ e do vetor $\mathbf{w} \in \mathbb{R}^k$, a função $f(\cdot)$ é uma não-linearidade como tangente hiperbólica ou relu, e $\mathbf{ones}^{|S|}$ é um vetor de $|S|$ posições composto apenas do valor 1, usado como artifício para replicar o vetor $\mathbf{x}$ de modo que tenha o mesmo formato de $\mathbf{M}_S$. Este tipo de atenção foi introduzido em RIT por um dos primeiros trabalhos da área envolvendo redes neurais (ROCKTÄSCHEL *et al.*, 2015). Já o segundo tipo é baseado na multiplicação de valores de $\mathbf{x}$ e $\mathbf{M}_S$:

$$f_\alpha(\mathbf{x}, \mathbf{M}_S) = \mathbf{W}_S \mathbf{M}_S \cdot \mathbf{W}_x \mathbf{x} \quad (4.11)$$

De modo geral, o primeiro tipo de atenção obtém melhores resultados, ao custo de uma maior quantidade de parâmetros e custo computacional. Frequentemente, as matrizes $\mathbf{W}_x$ e $\mathbf{W}_S$ são ignoradas no segundo tipo de atenção, simplificando-o ainda mais e aproximando-o do cálculo de similaridade de cosseno.

Nos cálculos demonstrados acima, não é levada em conta a posição de cada *token* dentro da sentença, o que pode limitar o poder da atenção. Uma solução que demonstra bons resultados é a concatenação de uma representação da posição relativa de cada palavra ao seu vetor (VASWANI *et al.*, 2017).

Porém, nos modelos neurais para RIT encontrados na literatura, essa não tem sido uma preocupação, o que se pode explicar pelos seguintes fatos: (i) na maioria das vezes, as representações sentenciais $\mathbf{M}_S$ são geradas por RNRs, e assim codificam inerentemente informação sobre o posicionamento de cada palavra; e (ii) alguns modelos são deliberadamente muito simplificados, visando a redução computacional e de parâmetros (PARIKH *et al.*, 2016).

4.2.4 Dropout

O dropout é uma técnica bastante simples introduzida por [Srivastava et al. \(2014\)](#). Consiste em zerar os valores produzidos por uma certa proporção p das unidades de uma camada de uma rede neural. Seu raciocínio é de evitar o superajuste aos dados, fazendo com que a camada seguinte aprenda a depender não de poucos valores de entrada, mas de todos.

Durante o treinamento, cada unidade de uma camada tem probabilidade p de ter seu valor zerado. Já durante a execução de um modelo treinado, deseja-se que seu funcionamento seja determinístico; portanto, a saída de cada unidade é multiplicada por $1 - p$. Desta forma, a soma esperada das saídas de uma camada permanece a mesma.

4.3 Arquiteturas Neurais para RIT

Nesta seção, são descritos métodos baseados em redes neurais desenvolvidos para o RIT. São divididos em duas categorias principais: os métodos de codificação de sentenças isoladas e os de codificação conjunta.

A primeira família codifica cada sentença separadamente em um único vetor ou matriz, para em seguida combinar as representações de T e de H como entrada para um classificador. Naturalmente, esta estratégia não se limita ao RIT, sendo flexível o suficiente para poder ser empregada em outras tarefas que envolvam classificação de sentenças (em pares ou isoladas).

A segunda família de modelos processa as duas sentenças levando em conta a relação entre seus conteúdos de alguma forma. Por explorarem melhor a interação entre T e H , estes métodos são os que têm os melhores resultados, ainda que abram mão da flexibilidade de representação.

4.3.1 Codificação de Sentenças

4.3.1.1 LSTM simples

Ao publicar o corpus SNLI, [Bowman et al. \(2015\)](#) avaliaram também a performance de modelos neurais razoavelmente simples de codificação de sentenças para RIT. Foram usados três modelos de codificação: o primeiro apenas soma os vetores de embeddings de todas as palavras de uma sentença; o segundo e o terceiro usam, respectivamente, uma RNR simples e uma LSTM, e representam a sentença como o vetor de saída do último passo das redes. O mesmo processo (com os mesmos pesos, no caso das redes neurais) é usado para codificar T e H .

Em seguida, as representações das duas sentenças são concatenadas e dadas a uma

rede MLP de três camadas de 200 dimensões cada, seguidas pela camada final com softmax sobre as três classes. A acurácia obtida pelos modelos ficou abaixo do classificador lexicalizado usado pelos autores no mesmo trabalho: o modelo baseado em soma de embeddings obteve 75,3%, o de RNR simples, 72,2%, e o de LSTM 77,6%. Estes resultados ilustram também a limitação de RNRs simples em comparação com o ganho significativo de performance com LSTMs.

4.3.1.2 Neural Semantic Encoders

Munkhdalai e Yu (2016) apresentam o NSE (*Neural Semantic Encoders*), aplicado em RIT e outras tarefas de classificação de sentenças (tanto em pares como individualmente). Neste modelo, cada sentença é codificada em uma matriz $\mathbf{M} \in \mathbb{R}^{k \times n}$, onde k é um parâmetro e n é a quantidade de *tokens*.

Neste modelo, a matriz $\mathbf{M}$, chamada também de memória, é inicializada com os vetores de embeddings de cada palavra da sentença, e então atualizada durante cada passo t do algoritmo. Uma LSTM processa a sentença, e é calculada a atenção $\mathbf{r}_t$ de cada vetor de saída $\mathbf{h}_t$ sobre $\mathbf{M}_{t-1}$ (a matriz $\mathbf{M}$ em seu estado anterior), usando o método baseado em multiplicação (vide Equação 4.11) sem matrizes de pesos.

Em seguida, $\mathbf{r}_t$ e $\mathbf{h}_t$ são compostos por uma rede MLP (os vetores são concatenados e processados pela rede), gerando uma representação $\mathbf{c}_t$ para o token:

$$\mathbf{c}_t = \text{MLP}([\mathbf{h}_t, \mathbf{r}_t]) \quad (4.12)$$

$\mathbf{c}_t$ é dado como entrada do passo t para uma segunda LSTM, que produz a representação final do t -ésimo token w_t e que será escrita na memória $\mathbf{M}_t$. Esta segunda LSTM mantém seu estado interno até que o próximo vetor $\mathbf{c}_{t+1}$ seja calculado. A atualização da memória reescreve os valores de cada token da sentença proporcionalmente ao quanto foram relevantes para w_t ; isto é, proporcionalmente ao vetor de atenção α_t :

$$\begin{aligned} \mathbf{g}_t &= \text{LSTM}(\mathbf{c}_t) \\ \mathbf{M}_t &= (\mathbf{g}_t \otimes \mathbf{ones}^n)(\alpha_t \otimes \mathbf{ones}^k) + \mathbf{M}_{t-1} \cdot (\mathbf{ones}^{k \times n} - \alpha_t) \end{aligned} \quad (4.13)$$

$\mathbf{ones}^d$ denota um vetor de valores 1 com d dimensões, e em conjunto com a operação $\otimes$ (produto diádico) é usado para replicar outro vetor em d linhas de uma matriz. n denota o número de *tokens*, e $\mathbf{M}_n$ é a representação final da sentença.

Quando se trabalha com duas sentenças, a memória final de uma sentença pode ser usada para gerar a representação da segunda. Para tal, é empregado um mecanismo

análogo ao da atenção sobre a própria sentença:

$$\begin{aligned}\beta_t &= \text{softmax}(\mathbf{h}_t^T \mathbf{M}'_{t-1}) \\ \mathbf{s}_t &= \beta_t \mathbf{M}'_{t-1}\end{aligned}\tag{4.14}$$

Onde $\mathbf{M}'$ é a matriz de memória da primeira sentença. A codificação das sentenças pode ocorrer simultaneamente, de modo que cada passo t de H afete a memória de T — efetivamente, enquadrando este método no paradigma de modelagem conjunta de sentenças. No entanto, também é possível evitar a interferência de H usando apenas os valores finais $\mathbf{M}'_m$. Os valores para a representação da segunda sentença passam a incluir $\mathbf{s}_t$:

$$\begin{aligned}\mathbf{c}_t &= \text{MLP}([\mathbf{h}_t, \mathbf{r}_t, \mathbf{s}_t]) \\ \mathbf{g}_t &= \text{LSTM}(\mathbf{c}_t)\end{aligned}\tag{4.15}$$

No caso da codificação simultânea, $\mathbf{M}'$ é atualizada de forma análoga a $\mathbf{M}$:

$$\mathbf{M}_t = (\mathbf{g}_t \otimes \mathbf{ones}^n)(\beta_t \otimes \mathbf{ones}^k) + \mathbf{M}_{t-1} \cdot (\mathbf{ones}^{k \times n} - \beta_t)\tag{4.16}$$

Para a decisão final, são tomados os vetores finais das LSTMs que processam as duas sentenças, $\mathbf{g}_m^t$ e $\mathbf{g}_n^h$. Ambos são combinados de três formas: concatenados e calculando-se sua diferença e produto elemento a elemento. A combinação destas representações é enfim dada a uma rede MLP que emite a decisão final:

$$y = \text{softmax}(\text{MLP}([\mathbf{g}_m^t, \mathbf{g}_n^h, \mathbf{g}_m^t - \mathbf{g}_n^h, \mathbf{g}_m^t \odot \mathbf{g}_n^h]))\tag{4.17}$$

Todos os parâmetros do modelo são treinados minimizando o erro supervisionado de RIT. O NSE simples, em que cada sentença é codificada independentemente, alcança 84,6% de acurácia, enquanto o modelo que usa o estado final da memória de T chega a 84,8%. Já o modelo de memória compartilhada atinge 85,4%. No entanto, os dois últimos usam quase o dobro de parâmetros treináveis do primeiro (6,5 e 6,3 milhões contra 3,4 milhões).

Um ponto que poderia ser melhorado nesta arquitetura é o fato de que a representação inicial da memória (embeddings) não leva a ordem das palavras em conta. Sua influência sobre a memória só existe na medida em que a mesma é atualizada, já que recebe valores (em parte) resultantes da primeira LSTM. Uma alternativa seria inicializar a memória com as saídas $\mathbf{h}_t$ da própria LSTM.

4.3.1.3 Auto-atenção direcional

Shen *et al.* (2017) apresentam um modelo neural que evita o uso de redes convolucionais ou recorrentes, intitulado DiSAN (*Directional Self-Attention Network*). Esta arquitetura codifica sentenças em vetores empregando apenas camadas MLP e atenção sobre a própria sentença, e traz dois conceitos novos: o primeiro é calcular atenção em termos de dimensão vetorial, em vez de palavra; o segundo é calcular separadamente a atenção de um token em relação aos que vêm antes dele e aos que vêm depois.

Inicialmente, todas as embeddings das palavras w_i são processadas por uma rede MLP de uma camada, gerando vetores $\mathbf{h}_i$, que compõem uma representação matricial $\mathbf{M}$ para a sentença. Em seguida, é computada a atenção de cada palavra w_i sobre a própria sentença usando o vetor $\mathbf{h}_i$.

A atenção em termos de dimensão vetorial, calculada com o método de transformação não-linear (vide Equação 4.10), é obtida ao se substituir o vetor de pesos $\mathbf{w}$ por uma matriz $\mathbf{W}$, gerando uma matriz $\mathbf{Z} \in \mathbb{R}^{|S| \times k}$ em vez do vetor $\mathbf{z}$. Desta forma, obtém-se uma distribuição de atenção para cada dimensão das colunas da matriz $\mathbf{M}_S$; ou seja, dependendo da dimensão do vetor $\mathbf{x}$ que se observe, diferentes palavras de S podem ser mais relevantes.

Para calcular a atenção somente para parte dos tokens de uma sentença, basta mascarar a matriz de valores não-normalizados $\mathbf{Z}$ com valores $-\infty$, o que faz com que tais posições tenham valor zero após a aplicação do softmax. Em seus experimentos, os autores treinam em conjunto duas redes, uma que usa atenção apenas sobre os tokens posteriores e outra apenas sobre os anteriores.

O resultado de cada rede que usa atenção sobre a própria sentença é dado da seguinte forma. É calculada uma combinação não-linear entre o vetor $\mathbf{h}_i$, produzido pela camada feedforward para o token w_i , e o vetor $\mathbf{r}_i$, seu respectivo vetor com valores de atenção:

$$\begin{aligned} \mathbf{f}_i &= \sigma(\mathbf{W}_R \mathbf{r}_i + \mathbf{W}_H \mathbf{h}_i + \mathbf{b}) \\ \mathbf{u}_i &= \mathbf{f}_i \odot \mathbf{h}_i + (1 - \mathbf{f}_i) \odot \mathbf{r}_i \end{aligned} \tag{4.18}$$

onde $\mathbf{b} \in \mathbb{R}^k$ e $\mathbf{W}_R, \mathbf{W}_H \in \mathbb{R}^{k \times k}$ são parâmetros treináveis, e σ é a função sigmoide, cujo domínio está no intervalo $[0, 1]$. Ao se usar uma rede que aplica atenção somente sobre os tokens posteriores e outra que faz o mesmo com os anteriores, são obtidos dois vetores $\mathbf{u}_i^{fw}$ e $\mathbf{u}_i^{bw}$ para cada token w_i , que são concatenados em um vetor $\mathbf{u}_i^{bi} = [\mathbf{u}_i^{fw}; \mathbf{u}_i^{bw}] \in \mathbb{R}^{2k}$.

Em seguida, é aplicada a chamada atenção *source2token*, implementada como uma composição não-linear sobre uma única sentença:

$$\mathbf{v}_i = \mathbf{W}f(u_i^{bi}\mathbf{W}_u + \mathbf{b}_1) + \mathbf{b}_2 \quad \mathbf{v}_i \in \mathbb{R}^{2k} \quad (4.19)$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{|S|} \end{bmatrix} \quad \mathbf{V} \in \mathbb{R}^{|S| \times 2k} \quad (4.20)$$

$$\mathbf{s}_{[j]} = \sum_i \text{softmax}(\mathbf{V}_{[* , j]}) \odot u_{i[j]}^{bi} \quad \mathbf{s} \in \mathbb{R}^{2k} \quad (4.21)$$

onde $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^{2k}$ e $\mathbf{W}, \mathbf{W}_u \in \mathbb{R}^{2k \times 2k}$ são parâmetros treináveis, e $\mathbf{s}$ é a representação final da sentença. As notações $\mathbf{s}_{[j]}$ e $\mathbf{V}_{[* , j]}$ indicam, respectivamente, a j -ésima dimensão do vetor $\mathbf{s}$ e a j -ésima coluna da matriz $\mathbf{V}$. O softmax computado em cada coluna da matriz calcula a atenção sobre cada dimensão separadamente.

Nota-se aqui que, apesar de ser considerado pelos autores como um mecanismo de auto-atenção, o *source2token* não faz qualquer tipo de alinhamento de um token com os outros da mesma sentença. Em vez disso, cada vetor $\mathbf{v}_i$ é obtido como uma função apenas do vetor u_i^{bi} do seu token correspondente. Por conta disso, considera-se mais adequado chamar este método de *compressão* de representação, em vez de atenção.

Os autores reportam experimentos com diferentes tarefas que envolvem classificação de sentenças ou de pares de sentenças. Para o RIT, a representação das duas sentenças é concatenada e passada por mais uma MLP com uma camada oculta e uma camada softmax. Os autores reportam 85,57% de acurácia no SNLI, usando cerca de 2,4 milhões de parâmetros. Já no MultiNLI, atingem 70,98% e 71,40% nos subconjuntos de mesmo domínio e cross-domínio, respectivamente.

4.3.1.4 Gated Attention

Chen *et al.* (2017c) apresentam um modelo com inovações relativamente simples, mas com melhorias técnicas bastante eficientes para RIT. Sua estratégia consiste em processar cada sentença com três camadas de BiLSTM, e compor a representação sentencial pela concatenação de três vetores: a média de todos os vetores de saída da última camada, o máximo de cada posição vetorial, e a *gated attention*, uma forma de auto-atenção introduzida com o modelo.

Na gated attention, os vetores do *input gate* ($\mathbf{i}_t$ na Equação 4.6) são usados para medir a importância da saída de cada passo da RNR. O raciocínio é o mesmo da definição do input gate — quanto maiores seus valores, maior a proporção de uma determinada entrada ser incorporada à memória da rede. Com efeito, os outros *gates* (*forget gate* e *output gate*) também podem ser utilizados para o mesmo fim, com a diferença de que no

caso do *forget gate*, usa-se 1 subtraído do seu valor, dado seu significado oposto ao *input gate*.

A cada passo t , é calculada a norma l^2 do input gate:

$$||\mathbf{i}_t||_2 = \sqrt{\sum_k \mathbf{i}_{t_k}^2} \quad (4.22)$$

É então composto um vetor $\mathbf{m}$ com as normas de todos os input gates calculados ao longo da sentença. Na prática, este vetor contém a contribuição de cada token para a memória da RNR. Em seguida, é calculado o softmax sobre $\mathbf{m}$, usado para ponderar as saídas da RNR:

$$\mathbf{v}_g = \sum_t \text{softmax}(\mathbf{m}) \mathbf{h}_t \quad (4.23)$$

onde $\mathbf{h}_t$ é o vetor de saída da última camada BiLSTM. O vetor $\mathbf{v}_g$ é concatenado à média e aos máximos vetoriais, mencionados anteriormente, formando a representação final da sentença $\mathbf{v}$. Uma dupla de sentenças, com seus respectivos vetores $\mathbf{v}^t$ e $\mathbf{v}^h$, é representada por:

$$\mathbf{v} = [\mathbf{v}^t, \mathbf{v}^h, |\mathbf{v}^t - \mathbf{v}^h|, \mathbf{v}^t \odot \mathbf{v}^h] \quad (4.24)$$

onde $||$ indica o valor absoluto. O vetor $\mathbf{v}$ passa ainda por duas camadas feedforward e, por fim, a um softmax com as três classes.

Este modelo obteve resultados de estado-da-arte no MultiNLI, com 73,5% de acurácia no subconjunto de mesmo domínio e 73,6% no conjunto cross-domínio. Testado no SNLI, obteve 85,5% de acurácia, o segundo melhor resultado publicado dentre modelos de codificação sentencial. No entanto, sua quantidade de parâmetros é bastante grande, com cerca de 11,6 milhões (devidos principalmente às três camadas de BiLSTM).

4.3.2 Modelagem Conjunta de Sentenças

4.3.2.1 Atenção Simples sobre a Premissa

A primeira publicação de resultados com modelos neurais no SNLI que superou o baseline de um classificador com atributos definidos manualmente foi a de [Rocktäschel et al. \(2015\)](#), que usaram redes LSTM com um mecanismo de atenção da hipótese sobre a premissa.

Seu modelo processa a premissa com uma rede LSTM e, em seguida, processa a hipótese com outra rede (que tem pesos distintos), mas mantendo o mesmo estado interno. Em uma versão mais simples do modelo, é computada a atenção apenas do último vetor

$\mathbf{h}_n$ da hipótese sobre a premissa (usando o método de soma), gerando o vetor $\mathbf{r}_n$. O modelo cria uma representação vetorial para o par combinando estes dois vetores:

$$\mathbf{h}^* = \tanh(\mathbf{W}_r \mathbf{r}_n + \mathbf{W}_h \mathbf{h}_n) \quad (4.25)$$

onde $\mathbf{W}_r, \mathbf{W}_h \in \mathbb{R}^{k \times k}$ são parâmetros treináveis. O vetor $\mathbf{h}^*$ é então projetado por uma camada softmax para o espaço de três classes.

Outra versão mais robusta do modelo computa a atenção de cada vetor $\mathbf{h}_t$ emitido pela LSTM da hipótese sobre a premissa. Neste caso, o cálculo a cada passo considera também a atenção do passo anterior, diferenciando-se da Equação 4.10:

$$f_\alpha(\mathbf{h}_t, \mathbf{M}_S) = \mathbf{w} \tanh(\mathbf{h}_t \mathbf{W}_x + \mathbf{M}_S \mathbf{W}_S + \mathbf{W}_z \mathbf{r}_{t-1})^T \quad (4.26)$$

Desta forma, o modelo carrega a cada passo a informação sobre onde recaiu a atenção no passo anterior. No entanto, os autores não explicam qual a vantagem do seu uso, ou fazem algum teste sem incluí-lo.

Tendo sido um experimento incipiente no desenvolvimento de arquiteturas neurais para RIT, seus resultados do modelo já foram superados até por métodos baseados em codificação de sentenças. Sua acurácia no SNLI foi de 83,5%, usando 3,9 milhões de parâmetros treináveis.

4.3.2.2 Redes Feedforward Modulares

Parikh *et al.* (2016) apresenta um modelo bastante leve para RIT, usando apenas redes feedforward e uma forma bastante simplificada de atenção. Sua arquitetura é dividida em três módulos: atenção, comparação e agregação.

Inicialmente, os vetores de embeddings de cada token são multiplicados por uma matriz de projeção $\mathbf{W}_e \in \mathbb{R}^{e \times k}$, onde e é a dimensionalidade original dos vetores de embeddings e k é a dimensionalidade usada no resto das camadas da rede. Assim, considerem-se os vetores projetados de $T = \{\mathbf{w}_1^t, \mathbf{w}_2^t, \dots, \mathbf{w}_m^t\}$ e de $H = \{\mathbf{w}_1^h, \mathbf{w}_2^h, \dots, \mathbf{w}_n^h\}$.

No módulo de atenção, cada $\mathbf{w}_i^t$ e $\mathbf{w}_j^h$ é processado (separadamente) por uma rede feedforward de duas camadas, gerando novos vetores $F(\mathbf{w}_i^t)$ and $F(\mathbf{w}_j^h)$. Então, é construída uma matriz de similaridade $\mathbf{E} \in \mathbb{R}^{m \times n}$ tal que cada celda e_{ij} contém o produto escalar $F(\mathbf{w}_i^t) \cdot F(\mathbf{w}_j^h)$. Esta operação é uma forma de avaliar a similaridade entre os dois vetores, sendo essencialmente o mesmo cálculo do cosseno (vide Equação 4.1) sem a normalização.

Para calcular a distribuição de atenção de $\mathbf{w}_i^t$ sobre H , é calculado o softmax sobre a linha i da matriz $\mathbf{E}$. Analogamente, a distribuição de $\mathbf{w}_j^h$ sobre T é calculada como o softmax sobre a coluna j . Os valores de atenção são então obtidos multiplicando-se as distribuições pelos valores projetados $\mathbf{w}_i^t$ e $\mathbf{w}_j^h$:

$$\beta_i = \sum_j \text{softmax}(e_{i*}) \mathbf{w}_j^h \quad \beta_i \in \mathbb{R}^k \quad (4.27)$$

$$\alpha_j = \sum_i \text{softmax}(e_{*j}) \mathbf{w}_i^t \quad \alpha_j \in \mathbb{R}^k \quad (4.28)$$

O vetor β_i representa a combinação das palavras de H proporcionalmente ao quão bem se alinham com $\mathbf{w}_i^t$; o análogo ocorre com α_j , T e $\mathbf{w}_j^h$. Em seguida, o módulo de comparação aplica outra rede feedforward, representada por $G(\cdot)$, à concatenação de cada token com seu vetor de alinhamento:

$$\begin{aligned} \mathbf{v}_{1,i} &= G([\mathbf{w}_i^t, \beta_i]) \\ \mathbf{v}_{2,j} &= G([\mathbf{w}_j^h, \alpha_j]) \end{aligned} \quad (4.29)$$

A ideia da camada de comparação é detectar as semelhanças e oposições entre palavras alinhadas. Lembre-se que, pela definição da matriz E , os maiores alinhamentos ocorrem entre vetores de palavras semelhantes. A semelhança entre os vetores $F(\mathbf{w}_i^t)$ e $F(\mathbf{w}_j^h)$, no entanto, não necessariamente reflete a semelhança de significado lexical codificada em embeddings — a transformação neural $F(\cdot)$ pode fazer com que palavras como antônimos sejam mapeados para vetores próximos, de modo a facilitar a detecção de contradição⁷.

Por fim, o módulo de agregação soma o conteúdo dos valores comparados, aplica mais uma rede feedforward $H(\cdot)$ de duas camadas, e uma camada final com o softmax sobre as três classes.

$$\begin{aligned} \mathbf{v}_1 &= \sum_i \mathbf{v}_{1,i} \\ \mathbf{v}_2 &= \sum_j \mathbf{v}_{2,j} \\ \mathbf{y} &= \text{softmax}(H([\mathbf{v}_1, \mathbf{v}_2])) \end{aligned} \quad (4.30)$$

A limitação mais evidente do modelo é o seu foco em similaridades lexicais. Para mitigá-la, os autores propõem uma etapa de auto-atenção antes da fase de atenção inter-sentencial, à qual funciona de forma similar:

$$e_{ij}^{intra} = F_{intra}(\mathbf{w}_i)^T F_{intra}(\mathbf{w}_j) \quad (4.31)$$

⁷ Não se devem confundir os vetores $F(\mathbf{w}_i^t)$ e $F(\mathbf{w}_j^h)$, usados na geração da matriz de alinhamento E , com os vetores $\mathbf{w}_i^t$ e $\mathbf{w}_j^h$. Os primeiros podem estar mapeados em um espaço em que antônimos (ou palavras tendo outro tipo de relação que favoreça contradição entre sentenças) são muito semelhantes entre si; já os segundos estão em um espaço que tende a ser próximo daquele das embeddings.

onde $F_{intra}(\cdot)$ é outra rede feedforward de duas camadas, e uma matriz E^{intra} acumula os valores de similaridade entre seus tokens. A atenção $\mathbf{w}'_i$ observada por cada token w_i é calculada da seguinte forma:

$$\mathbf{w}'_i = \sum_j \frac{\exp(e_{ij}^{intra} + d_{i-j})}{\sum_l \exp(e_{il}^{intra} + d_{i-l})} \mathbf{w}_j \quad (4.32)$$

Neste cálculo, é incorporado um termo treinável $d_{i-j} \in \mathbb{R}$ que agrega alguma informação sobre a distância entre dois tokens. A distribuição de atenção é, portanto, calculada com um softmax sobre a matriz E^{intra} adicionada dos termos de distância apropriados, que variam para cada par de palavras. Após o cálculo de $\mathbf{w}'_i$, a representação vetorial de w_i usada no modelo é a concatenação $[\mathbf{w}_i, \mathbf{w}'_i]$.

A principal vantagem deste modelo é sua simplicidade em termos de quantidade de parâmetros treináveis (cerca de 582 mil na versão com auto-atenção) e da arquitetura utilizada. O modelo atinge resultados bastante próximos do estado-da-arte no SNLI, com 86,8% de acurácia.

4.3.2.3 Redes Recorrentes Modulares

A separação do problema em três estágios ilustrada na seção anterior pode ser implementada com redes recorrentes, como foi proposto por [Chen et al. \(2017a\)](#). Seu raciocínio foi de que, apesar da simplicidade das redes neurais usadas por [Parikh et al. \(2016\)](#), a arquitetura modular conseguiu resultados próximos daqueles obtidos por modelos muito mais complexos e com enorme quantidade de parâmetros. Portanto, o uso de LSTMs, redes mais poderosas que MLPs, na mesma estrutura, poderia potencializar os resultados de [Parikh et al. \(2016\)](#), possivelmente superando resultados de modelos neurais de grande complexidade.

Os autores separam sua arquitetura, chamada ESIM (*Enhanced Sequential Inference Model*), em três módulos ligeiramente diferentes dos de [Parikh et al. \(2016\)](#): codificação, inferência local e composição de inferência. No primeiro módulo, as duas sentenças são processadas por uma rede recorrente, e os vetores de saída da rede $\mathbf{h}_t$ são usados para representar cada token w_t .

Foram experimentadas duas variantes de LSTM para a codificação: LSTM bidirecionais e LSTM arbóreas (TreeLSTM) ([TAI](#); [SOCHER](#); [MANNING, 2015](#)). O segundo tipo lê uma árvore sintática de constituintes da sentença de entrada, e cada passo do seu algoritmo se refere a um nó da árvore, em contraste com LSTMs normais que leem os tokens na ordem que estão na sentença. O vetor de entrada de um nó terminal ($\mathbf{x}$ na Equação 4.6) é a própria representação da palavra — seu vetor de embeddings ou vetor previamente processado por outras camadas —, enquanto nós não-terminais usam um vetor especial $\mathbf{x}'$ representando a ausência de um token de entrada. A característica mais

importante da LSTM arbórea é que nós não-terminais podem combinar os vetores de saída $\mathbf{h}_{t-1}^i$ de seus nós filhos na árvore, tendo uma matriz de pesos $\mathbf{U}^i$ para multiplicar o vetor do i -ésimo filho. Por simplicidade, normalmente são usadas apenas árvores binárias com este tipo de RNR, como é o caso de [Chen et al. \(2017a\)](#). Todos os vetores gerados pela LSTM são combinados em uma matriz $\mathbf{H}^t$ (para T) e $\mathbf{H}^h$ (para H).

Na etapa de inferência local, os vetores computados pela BiLSTM ou TreeLSTM para cada sentença são comparados entre si, visando encontrar palavras ou expressões que tenham alguma relação de implicação ou contradição. Retomando o exemplo do par (1), adicionando mais uma sentença:

- (2) a. Um homem caminha ao lado da esposa enquanto carrega uma criança no colo.
 b. Um homem está carregando uma criança.
 c. Um homem está carregando uma caixa.

É de se esperar que este modelo encontre certos alinhamentos em pares compostos com as sentenças acima. Por exemplo, as ocorrências de *um homem* são trivialmente alinháveis; mais interessante é detectar o alinhamento equivalente entre *carrega uma criança* e *está carregando uma criança*, ou o alinhamento contraditório entre qualquer um destes e *está carregando uma caixa*. Um modelo baseado apenas em alinhamentos lexicais não seria capaz de encontrar a relação entre esses trechos.

O modelo constrói uma matriz de alinhamentos $\mathbf{E}$ similarmente à de [Parikh et al. \(2016\)](#), com o produto vetorial das saídas das LSTMs a cada passo. Note-se apenas que, enquanto a BiLSTM gera uma saída para cada palavra da sentença, a TreeLSTM gera uma para cada nó (terminal ou não) da árvore sintática, totalizando $2n - 1$ vetores para uma sentença de tamanho n . Por isso, a matriz $\mathbf{E}$ é maior quando se emprega uma TreeLSTM.

Novamente, um softmax sobre as linhas de $\mathbf{E}$ indica a distribuição de atenção de T sobre H , e sobre as colunas a de H sobre T . A multiplicação da distribuição de atenção pela matriz $\mathbf{H}^t$ ou $\mathbf{H}^h$ gera os valores de atenção análogos a α_j e β_i da seção anterior. Note-se ainda que, quando a TreeLSTM é usada, um token pode ter parte de sua atenção voltada para não-terminais e vice-versa.

Na etapa de composição de inferência, os vetores $\mathbf{h}_i^t$ e $\mathbf{h}_j^h$ (incluindo vetores correspondentes a nós não-terminais, no caso da TreeLSTM) são pareados com β_i e α_j , respectivamente:

$$\begin{aligned}\mathbf{m}_i &= [\mathbf{h}_i^t, \beta_i, \mathbf{h}_i^t - \beta_i, \mathbf{h}_i^t \odot \beta_i] \\ \mathbf{n}_j &= [\mathbf{h}_j^h, \alpha_j, \mathbf{h}_j^h - \alpha_j, \mathbf{h}_j^h \odot \alpha_j]\end{aligned}\tag{4.33}$$

Os vetores $\mathbf{m}_i$ e $\mathbf{n}_j$ agregam à concatenação dos tokens e valores de atenção sua diferença e produto vetorial, cujas propriedades foram discutidas na Seção 4.1.2.1. Esta é uma representação mais rica do que a usada por Parikh *et al.* (2016). Visando diminuir a quantidade de parâmetros do modelo, estes vetores são projetados para uma dimensionalidade menor por uma camada feedforward $F(\cdot)$. Os valores resultantes são então dados a uma rede BiLSTM ou TreeLSTM, conforme a estratégia utilizada. Nesta etapa, no caso de uma TreeLSTM, haverá vetores de entrada mesmo para as posições correspondentes a nós não-terminais, pois foram codificados na matriz $\mathbf{E}$.

A representação final de cada sentença é composta por dois vetores: a média de todas as saídas da última RNR e o seu valor máximo por elemento.

$$\begin{aligned}
 \mathbf{v}_{a,i} &= \text{LSTM}(F(\mathbf{m}_i)) \\
 \mathbf{v}_{b,j} &= \text{LSTM}(F(\mathbf{n}_j)) \\
 \mathbf{v}_{a,\text{avg}} &= \sum_i \frac{\mathbf{v}_{a,i}}{m} \\
 \mathbf{v}_{b,\text{avg}} &= \sum_j \frac{\mathbf{v}_{b,j}}{n} \\
 \mathbf{v}_{a,\text{max}} &= \max_i \mathbf{v}_{a,i} \\
 \mathbf{v}_{b,\text{max}} &= \max_j \mathbf{v}_{b,j}
 \end{aligned} \tag{4.34}$$

onde m e n são a quantidade total de entradas para a LSTM em T e em H , respectivamente, ou seja, o número de tokens no caso da BiLSTM e o número de nós na árvore no caso da TreeLSTM.

A representação final do par, composta pela concatenação $[\mathbf{v}_{a,\text{avg}}, \mathbf{v}_{b,\text{avg}}, \mathbf{v}_{a,\text{max}}, \mathbf{v}_{b,\text{max}}]$ é processada por uma camada oculta feedforward e, por fim, pelo softmax entre as três classes. Os autores ainda propõem uma combinação do modelo que usa BiLSTM com o de TreeLSTM, simplesmente tirando a média dos valores finais do softmax de ambos.

Chen *et al.* (2017a) apresentam resultados com diversas modificações em seu modelo, como removendo o cálculo de médias ou o de máximos da Equação 4.34 ou usando apenas um dos tipos de LSTM. Interessantemente, quando usados de forma isolada, o modelo de BiLSTM tem performance ligeiramente superior à do de TreeLSTM, indicando que o conhecimento linguístico externo (na forma de árvore sintática) não necessariamente é útil para RNRs. Os melhores resultados encontrados, utilizando todos os mecanismos apresentados aqui, foi de 88,6% no SNLI, e com 7,7 milhões de parâmetros. Já o modelo com BiLSTM, sozinho, alcança 88% de acurácia, com 4 milhões de parâmetros.

Foi desenvolvida também uma melhoria sobre a ESIM original, que pode incluir conhecimento externo proveniente da WordNet ou um recurso de semântica lexical similar

(CHEN *et al.*, 2017b). Neste novo trabalho, os autores propõem o uso de um vetor de conhecimento externo $\mathbf{r}_{ij} \in \mathbb{R}^8$ para cada par de palavras (w_i, w_j) , codificando oito atributos relativos às suas relações na WordNet de Princeton: sinonímia, anotonímia, hiperonímia, hipernímia e hiperônimo comum (os últimos três atributos são direcionais, e portanto são repetidos com o par de palavras invertido). Os atributos assumem um valor binário para indicar sinonímia e antonímia, enquanto que um valor de número real indica a distância no grafo da WordNet para hiperônimos/hipônimos.

O vetor $\mathbf{r}_{ij}$ é usado em diferentes módulos da rede neural. No cálculo da matriz de alinhamentos $\mathbf{E}$, é adicionado um termo $\lambda \mathbb{1}(r_{ij})$, onde λ é um hiperparâmetro da rede e $\mathbb{1}$ é uma função que retorna 0 para um vetor de zeros e 1 caso contrário. Em outras palavras, esta formulação simplesmente adiciona uma constante λ ao cálculo de alinhamento entre w_i e w_j caso os tokens tenham algum tipo de relação na WordNet.

Na etapa de alinhamento local, onde o ESIM computa os valores $\mathbf{m}_i$ e $\mathbf{n}_j$ (vide Equação 4.33), é incluído mais um vetor na operação de concatenação:

$$\begin{aligned}\mathbf{m}_i &= [\mathbf{h}_i^t, \beta_i, \mathbf{h}_i^t - \beta_i, \mathbf{h}_i^t \odot \beta_i, \text{softmax}(e_{i*})\mathbf{r}_{i*}] \\ \mathbf{n}_j &= [\mathbf{h}_j^h, \alpha_j, \mathbf{h}_j^h - \alpha_j, \mathbf{h}_j^h \odot \alpha_j, \text{softmax}(e_{*j})\mathbf{r}_{*j}]\end{aligned}\tag{4.35}$$

O produto do softmax de atenção de cada token com seus respectivos vetores de conhecimento externo indica o quanto aquele token está alinhado com outro(s), de acordo com a WordNet. Uma palavra em T sem nenhuma outra relacionada pela WordNet em H teria estes valores zerados.

Por fim, na composição final da representação do par, enquanto a ESIM original extrai e média e os máximos das saídas de uma camada de LSTM (vide Equação 4.34), a nova arquitetura inclui uma nova medida agregada:

$$\begin{aligned}\mathbf{ext}_{a_i} &= \text{softmax}(\text{softmax}(e_{i*})\mathbf{r}_{i*}) \\ \mathbf{ext}_{b_j} &= \text{softmax}(\text{softmax}(e_{*j})\mathbf{r}_{*j}) \\ \mathbf{v}_{a,\text{ext}} &= \text{softmax}_i(H(\mathbf{ext}_a))\mathbf{v}_{a,i} \\ \mathbf{v}_{b,\text{ext}} &= \text{softmax}_j(H(\mathbf{ext}_b))\mathbf{v}_{b,j}\end{aligned}\tag{4.36}$$

onde $H(\cdot)$ é a aplicação de uma rede feedforward com uma camada. Esta medida engloba os alinhamentos, segundo a WordNet, de todas as palavras de cada sentença, incluindo-as na representação final da rede.

A nova versão do modelo foi implementada apenas com a BiLSTM. Os autores apresentam resultados para diversas mudanças em sua configuração: variando o valor de λ , aplicando o vetor de conhecimento externo em apenas um ou dois dos módulos, e treinando

Figura 10 – Arquitetura do modelo BiMPM (imagem adaptada de Wang, Hamza e Florian (2017))

o modelo em um subconjunto do SNLI. Por fim, apresentam as seguintes conclusões empíricas. Valores maiores de λ são mais úteis com conjuntos de treino menores, quando o modelo tem menos dados para aprender e conhecimento externo é mais importante. A etapa em que o vetor $\mathbf{r}_{ij}$ traz maior contribuição é na de inferência local (descrita na Equação 4.35). Isso indica que codificar o quão bem cada token se alinha com a outra sentença é importante para o problema. Sobre o SNLI original, o modelo atinge 88,6% de acurácia (mesma performance do modelo ESIM sem consulta à WordNet), enquanto um ensemble de dez instâncias com diferentes inicializações, treinadas independentemente, atinge 89,1%

4.3.2.4 Comparação Bilateral Multiperspectiva

Wang, Hamza e Florian (2017) apresentam uma arquitetura neural baseada na comparação de tokens de uma sentença com a outra sob diferentes perspectivas, chamada BiMPM (*Bilateral Multi-Perspective Matching*). Cada *perspectiva* é a projeção da representação de uma sentença com diferentes parâmetros, e o modelo agrega quatro formas diferentes de comparação. A arquitetura é ilustrada na Figura 10.

O BiMPM inicia seu funcionamento aplicando uma mesma BiLSTM sobre as sentenças do par, gerando os vetores $\mathbf{h}_i^t$ e $\mathbf{h}_j^h$ para o i -ésimo par de T e o j -ésimo de H , respectivamente.

Em seguida, há o módulo de comparação, onde está o cerne do modelo. Aqui, é usada uma função de comparações multi-perspectiva $f_m(\cdot)$, que calcula l valores diferentes de similaridade entre dois vetores, ditos similaridades em diferentes perspectivas. Todos são baseados no cosseno entre os dois vetores após serem multiplicados por diferentes valores:

$$\mathbf{m} = f_m(\mathbf{v}_1, \mathbf{v}_2, \mathbf{W}) \quad \mathbf{m} \in \mathbb{R}^l \quad (4.37)$$

$$\mathbf{m}_k = \cos(\mathbf{W}_{[k]} \odot \mathbf{v}_1, \mathbf{W}_k \odot \mathbf{v}_2) \quad 1 \leq k \leq l \quad (4.38)$$

$\mathbf{W} \in \mathbb{R}^{l \times d}$ é uma matriz de pesos, da qual cada linha $\mathbf{W}_{[k]}$ representa uma perspectiva. A função $f_m(\cdot)$ é aplicada de quatro formas distintas, cada uma delas comparando palavras de T com H e vice-versa. Além disso, são comparados separadamente os vetores produzidos pelos componentes de frente para trás e de trás para frente da BiLSTM, de modo que cada token tem 8 vetores de similaridade. São feitas as seguintes comparações:

Comparação completa cada vetor $\mathbf{h}_i$ de uma palavra é comparado com o último vetor emitido pela BiLSTM para a outra sentença.

Comparação *max-pooling* cada vetor $\mathbf{h}_i$ de uma palavra é comparado com os vetores de todas as palavras da outra sentença e, para cada dimensão, é selecionado o maior valor dentre os computados.

Comparação com atenção primeiramente, o cosseno c_{ij} entre o vetor $\mathbf{h}_i$ de uma palavra e cada vetor $\mathbf{h}_j$ da outra sentença é calculado. É feita uma soma ponderada dos vetores da outra sentença, multiplicando cada $\mathbf{h}_j$ por c_{ij} , e a soma é normalizada dividindo-se pela soma de todos os cossenos. A similaridade multi-perspectiva é então calculada entre este resultado e $\mathbf{h}_i$.

Comparação com atenção máxima cada vetor de uma palavra é comparado com o vetor de maior similaridade na outra sentença.

Os oito vetores de similaridade de cada token são concatenados, e uma segunda BiLSTM os processa (separadamente os de T e os de H). Os vetores finais das duas redes são concatenados e passam por mais duas camadas feedforward, e em seguida pelo softmax das três classes.

Os autores comparam versões do BiMPM que compara apenas palavras de H com T ou apenas de T com H , e concluem que o primeiro sentido obtém resultados melhores. Este resultado já era esperado — H é, via de regra, uma sentença menor que T , e encontrar relações de todas as suas palavras com T é mais importante que o inverso. De qualquer forma, resultados ainda melhores são obtidos com as comparações bidirecionais.

O BiMPM como descrito aqui atinge 86,9% de acurácia no SNLI, com o número relativamente pequeno de 1,6 milhões de parâmetros treináveis. Os autores ainda treinaram um ensemble de quatro modelos iguais, cada um inicializado com uma semente aleatória diferente e treinado independentemente, e tiraram a média do softmax de todos. Esta combinação atingiu 88,8% de acurácia, com quatro vezes o número de parâmetros original.

4.4 Considerações Finais

Foram apresentados aqui alguns dos principais modelos neurais empregados para RIT, sumarizados na Tabela 9. Por brevidade, alguns não foram incluídos; uma lista completa e frequentemente atualizada com acurácia e quantidade de parâmetros de trabalhos publicados pode ser encontrada em <https://nlp.stanford.edu/projects/snli/>.

Arquitetura	Atenção	Tipo de Redes	Agregação de Vetores	Parâmetros treináveis	Acurácia (SNLI)
LSTM simples	-	LSTM	saída final da RNR	220k	77,6%
Neural Semantic Encoders	Auto-atenção multiplicativa	LSTM	atenção	3m	84,6%
DiSAN	Auto-atenção aditiva multidimensional	MLP	atenção (compressão)	2,4m	85,6%
Gated Attention	Auto-atenção multiplicativa com input gate	Três camadas BiLSTM	média, máximo e atenção	12m	85,5%
Atenção Simples	Atenção aditiva	LSTM	saída final da RNR	250k	83,5%
Neural Semantic Encoders (inter-atenção)	Auto-atenção e inter-atenção multiplicativa	LSTM	saída final da RNR	3,2m	85,4%
Feedforward modular	Atenção e auto-atenção multiplicativa	MLP	soma	580k	86,8%
RNR modular	Atenção multiplicativa	BiLSTM e TreeLSTM	média e máximo	7,7m	88,6%
RNR modular com wordnet	Atenção multiplicativa	BiLSTM	média e máximo	4,3m (43m com ensemble)	88,6% (89,1% com ensemble)
BiMPM	Atenção multiplicativa simplificada (cossenos)	BiLSTM	saída final da RNR	1,6m (6,4m com ensemble)	87,5% (88,8% com ensemble)

Tabela 9 – Sumário dos modelos analisados de redes neurais para RIT. Nos modelos de ensemble, os parâmetros não são todos treinados juntos.

Os resultados aqui descritos manifestam o potencial preditivo de modelos neurais treinados com grandes volumes de dados, bem como o da importância de suas arquiteturas. Diferentes formas de estruturar redes neurais podem levar a grandes diferenças de performance. Nota-se também que ensembles, ou combinações de modelos, podem melhorar um pouco mais a performance, ao custo de se multiplicar seus parâmetros e tempo computacional, já que todos os modelos precisam ser rodados separadamente para em seguida se combinarem suas decisões.

A AVALIAÇÃO DE SIMILARIDADE SEMÂNTICA E INFERÊNCIA TEXTUAL

Neste capítulo, é descrito o corpus ASSIN, desenvolvido como parte do projeto de doutorado, bem como a avaliação conjunta homônima. O ASSIN é o primeiro conjunto de dados em português anotado quanto a similaridade semântica e inferência textual.

5.1 Criação do Corpus

Nesta seção, descrevemos como se deu a criação do conjunto de dados e apresentamos as estatísticas da anotação.

5.1.1 O Conjunto de Dados

O corpus ASSIN (Avaliação de Similaridade Semântica e Inferência Textual) contém 10 mil pares de sentenças anotados quanto a dois fenômenos relacionadas: a presença de implicação textual e a similaridade semântica (STS, *Semantic Textual Similarity*) (AGIRRE *et al.*, 2015), uma medida numérica do quão similar é o conteúdo das duas sentenças. A anotação dos dois fenômenos foi inspirada pelo corpus SICK, discutido na Seção 2.5.2.

A definição exata destas tarefas não é universal. Quanto ao RIT, é comum o uso das categorias de implicação, contradição e neutro; no entanto, o ASSIN conta com implicação, paráfrase e neutro. Já quanto à similaridade semântica, há diferentes escalas numéricas (AGIRRE *et al.*, 2015), e a usada no ASSIN varia de 1 a 5. Tais escolhas se deram após análise de amostras dos pares que viriam a ser anotados.

A avaliação ASSIN trouxe o primeiro corpus anotado para as duas tarefas em português, incluindo as variantes brasileira e europeia — cada variante corresponde a

metade do corpus. Foram aproveitados os agrupamentos de notícias por assunto fornecidos pelo *Google News*¹ para extrair pares de sentenças, que passaram por um processo de filtragem manual (em que foram excluídos pares considerados ruidosos) e, por fim, foram anotados por juízes humanos. Cada par foi anotado por quatro pessoas, de um conjunto de 36 anotadores, com respeito às duas tarefas.

Este processo ocorreu em contraste com as abordagens utilizadas para a construção de corpora similares em inglês, como SICK (MARELLI *et al.*, 2014a) e SNLI (BOWMAN *et al.*, 2015). Em primeiro lugar, ambos contêm apenas descrições de imagens, enquanto o ASSIN contém notícias sobre variados temas, e frequentemente conceitos abstratos. Além disso, o processo semi-automático do SICK gerou muitos padrões facilmente predizíveis, enfraquecendo a sua representatividade de textos reais. O SNLI, embora tenha todas as suas sentenças escritas por humanos, apresenta muitos pares em que a segunda sentença é apenas uma versão com poucas palavras a menos da outra.

5.1.2 Criação do Corpus

A exploração de agrupamentos de notícias para aquisição de pares de sentenças similares não é uma ideia nova; já foi explorada com sucesso em vários trabalhos da literatura (DOLAN; QUIRK; BROCKETT, 2004; DAGAN; GLICKMAN; MAGNINI, 2006). Entretanto, em vez de anotadores humanos selecionarem pares com base na sobreposição de palavras, empregou-se no ASSIN o *Latent Dirichlet Allocation* (LDA) (BLEI; NG; JORDAN, 2003) para selecionar pares similares.

O LDA, um método de modelagem de espaços vetoriais, atribui uma pontuação para pares de documentos, refletindo o quão similares são entre si. Em um experimento piloto reportado em Fonseca e Aluísio (2015), notou-se que, em comparação com outros métodos de espaço vetorial, o LDA fornecia os pares mais interessantes para inferência textual, pois recuperava o menor número de sentenças sem relação de inferência (que costumam ser a maioria) e era eficiente em detectar similaridades além da sobreposição de palavras.

Usou-se um modelo diferente de LDA para cada variante do português, ambos treinados em grandes corpora de notícias. O modelo para o português do Brasil foi treinado em um corpus coletado do site de notícias brasileiro G1², e o modelo para português europeu com textos do jornal português Público³. Estes corpora foram somente usados para gerar os modelos LDA, não para coletar os pares de sentenças do corpus ASSIN.

Grupos de notícias sobre o mesmo evento foram coletados do Google News em

¹ <<https://news.google.com/>>

² <<http://g1.globo.com/>>

³ <<http://www.publico.pt/>>

suas versões específicas para Brasil e Portugal⁴. Dados os grupos de notícia coletados e um modelo de espaço vetorial treinado, a criação do ASSIN seguiu um processo de três etapas:

1. LDA encontra pares de sentenças similares dentro de cada grupo. Esse passo pode ser parametrizado fixando os valores mínimo e máximo de similaridade s_{max} e s_{min} : fixando um valor máximo evita pares de sentenças quase iguais, que seriam classificados trivialmente como paráfrases, e fixando um mínimo evita pares muito dissimilares que são facilmente classificados como sem relação. Também se fixou a proporção α de tokens que são encontrados em uma sentença mas não em outra (sem contar *stopwords*). Finalmente, sentenças podem ser limitadas por um tamanho máximo; em uma análise preliminar, observou-se que sentenças muito longas têm muita informação e dificilmente podem ser completamente implicadas por outra.
2. Um grupo de quatro revisores analisa os pares coletados em um processo manual. Se um par contivesse uma sentença sem sentido, era descartado. Sentenças foram também editadas para correção de erros ortográficos e gramaticais, ou para alterar casos em que a presença de implicação era pouco clara.
3. Cada par é anotado por quatro pessoas, selecionadas aleatoriamente pelo sistema de anotação dentre todos os anotadores participantes. Cada anotador seleciona um valor de similaridade de 1 a 5, e também uma das quatro opções para inferência: a primeira sentença implica a segunda; a segunda implica a primeira; paráfrase, ou nenhuma relação.

Este processo foi realizado em vários lotes, variando os parâmetros. Usamos os valores de s_{min} de 0,65 e 0,6, sem obter diferença observável no resultado. s_{max} foi fixado em 0,9. A proporção de tokens exclusivos para cada sentença foi fixada em 0,1 como mínimo e valores máximos variando entre 0,7 ou 0,8. Com o último valor, notamos um aumento considerável de pares de sentenças com valor de similaridade baixo.

Dada a subjetividade da anotação, algumas diretrizes se fizeram necessárias para lidar com alguns fenômenos linguísticos recorrentes que tinham diferentes interpretações por parte dos anotadores. As diretrizes são voltadas especialmente para a anotação de inferência, e estão listadas na Tabela 10.

Pares que não tinham concordância de pelo menos três votos para a tarefa de inferência textual foram descartados. O entendimento foi que esses pares eram controversos e portanto não seriam bons exemplos para o conjunto final. Note-se que os anotadores poderiam indicar implicação tanto da primeira para a segunda sentença como da segunda

⁴ Alguns domínios foram filtrados para evitar sites de notícias brasileiros erroneamente listados na seção de Portugal e vice-versa

Conceito	Explicação
Atemporalidade	A interpretação das sentenças não deveria levar em conta a data corrente, de modo que a anotação fizesse sentido no futuro. Assim, embora <i>há 70 anos atrás</i> e <i>em 1945</i> fossem equivalentes em 2015, deveriam ser considerados distintos pelos anotadores.
Entidades Nomeadas	Entidades nomeadas que aparecessem nas duas sentenças, tendo um apostro ou adjetivo em uma delas, deveriam ser consideradas equivalentes. <i>Florianópolis</i> , <i>em Santa Catarina</i> é equivalente a apenas <i>Florianópolis</i> .
Discurso Indireto	Uma sentença com discurso indireto (i.e., <i>O embaixador disse que (...)</i>) pode implicar outra que contenha apenas a fala atribuída. O contrário, no entanto, não é possível.
Quantidades	Valores numéricos diferentes só podem ser aceitos para paráfrase/implicação se tiverem indicadores explícitos de serem aproximações: <i>acerca de</i> , <i>pelo menos</i> , <i>quase</i> , <i>perto de</i> , etc. Por exemplo, <i>arrecadou 7 milhões</i> não implica <i>arrecadou 6 milhões</i> pois, mesmo sendo uma quantia menor, é possível que se refira a outro evento.

Tabela 10 – Resumo das Diretrizes para Anotação

para a primeira; porém, no conjunto final, ajustaram-se os pares de modo que todos os casos de inferência fossem da primeira sentença para a segunda. O valor final de similaridade para cada par é a média das quatro pontuações — números reais separados por intervalos de 0,25.

A anotação foi realizada via uma interface Web construída especialmente para a tarefa, mas flexível o bastante para permitir customizações em futuras anotações. Os anotadores receberam treinamento para calibrar os conceitos das tarefas a serem realizadas, com ajuda de um conjunto de 18 pares exemplificando todos os fenômenos tratados. Em caso de dúvidas, perguntas poderiam ser enviadas via e-mail para a equipe de anotadores, o que permitia discutir casos muito difíceis de decisão, principalmente no começo da anotação.

5.1.3 Estatísticas da Anotação

O corpus foi anotado por um total de 36 pessoas, que participaram em diferentes quantidades: o anotador com menor participação julgou 25 pares, enquanto o com maior participação julgou 6.740.

Do total de pares anotados, 11.3% foram descartados por não terem três julgamentos iguais quanto à implicação, uma proporção um pouco menor do que as reportadas na criação dos conjuntos RTE Challenge (DAGAN; GLICKMAN; MAGNINI, 2006; GLAMPICCOLO *et al.*, 2007). No total, o ASSIN tem 10 mil pares, sendo cinco mil em português brasileiro e cinco mil em português europeu.

A Tabela 11 sumariza estatísticas da anotação. A correlação de Pearson⁵ apre-

⁵ A correlação de Pearson é uma métrica estatística da correlação linear entre duas variáveis,

Métrica	Valor
Correlação de Pearson	0,74
Desvio Padrão Médio	0,49
κ de Fleiss	0,61
Concordância	0,80

Tabela 11 – Estatísticas da Anotação. Os primeiros dois valores se referem à anotação de similaridade; os dois últimos valores à inferência.

sentada na tabela se refere à média das correlações calculadas entre todos os anotadores, ponderada pela quantidade de pares que cada um anotou. Para cada anotador, calculamos a correlação das suas pontuações de similaridade com as médias das pontuações dos pares que ele ou ela anotou (excluindo a sua anotação do cômputo). Para efeito de comparação, a anotação do STS 2015 obteve valores entre 0.65 e 0.85, o que mostra que alcançamos boa concordância entre anotadores quanto à similaridade.

O desvio padrão médio avalia a divergência dos julgamentos de similaridade dos pares. É calculado como a média dos desvios padrão de todos os pares no conjunto; esses, por sua vez, são calculados como o desvio padrão das quatro pontuações em relação à média do par. O valor reportado na anotação do SICK é de 0,76, indicando que as pontuações dos nossos anotadores divergiram menos.

Com relação à inferência, o valor da concordância κ de Fleiss foi relativamente baixo, o que indica que a anotação desta tarefa de fato envolveu boa quantidade de subjetividade. Os conjuntos dos desafios RTE, por exemplo, tiveram uma taxa de concordância maior: 0,6 na primeira edição (DAGAN; GLICKMAN; MAGNINI, 2006), mas chegando a 0,75 ou mais nas subsequentes (GIAMPICCOLO *et al.*, 2007). Entretanto, deve ser notado que esses conjuntos tratam de sentenças curtas como segundo componente do par (a sentença implicada), o que torna a decisão mais fácil.

A última linha da tabela se refere à concordância simples entre os anotadores. Isso significa que, em 80% dos casos, dois anotadores que julgaram o mesmo par escolheram a mesma categoria de inferência.

As tabelas 12 e 13 mostram estatísticas sobre as anotações de similaridade e inferência, respectivamente. Pode-se ver que as pontuações de similaridade mais comuns estão no intervalo entre 2 e 3. Já quanto a inferência, percebe-se que a relação neutra é a classe majoritária, enquanto as paráfrases são uma porção pequena do conjunto.

A pouca quantidade de pares com relação de inferência foi notada já durante nossa análise de um conjunto piloto de 100 pares, que não foi incluído no corpus final. Isso se

indo de 1 (totalmente correlacionadas) a -1 (inversamente correlacionadas), com 0 indicando ausência de correlação. No caso, as duas variáveis consideradas são as similaridades anotadas por um anotador e as médias das similaridades dos outros anotadores para os mesmos pares.

Similaridade	PB	PE	Total
4,0 – 5,00	1.074	1.336	2.410
3,0 – 3,75	1.591	1.281	2.872
2,0 – 2,75	1.986	1.828	3.814
1,0 – 1,75	349	555	904
Média	3,05	3,05	3,05

Tabela 12 – Estatísticas de similaridade do ASSIN.

Relação	PB	PE	Total
Sem relação	3.884	3.432	7.316
Implicação	870	1.210	2.080
Paráfrase	246	358	604

Tabela 13 – Estatísticas de inferência do ASSIN.

devia ao fato de que, em muitos casos, apenas alguns detalhes impediam que houvesse tal relação: a menção a um local, tempo, propósito, entre outros. Essa situação é ilustrada no exemplo a seguir.

- (1) a. O Internacional manteve a boa fase e venceu o Strongest por 1 a 0 nesta quarta-feira, garantindo a liderança do Grupo 4 da Libertadores.
- b. Em casa, a equipe gaúcha derrotou o The Strongest, por 1 a 0, e garantiu a primeira colocação do Grupo 4 da Copa Libertadores.

Apesar de as duas sentenças compartilharem a maior parte do conteúdo, cada uma tem alguma informação específica que não é implicada pela outra. A primeira menciona o nome do time, além de que estava em boa fase e que o jogo foi na quarta-feira. Já a segunda diz que o jogo foi na casa do time, sem explicitar seu nome. Esse tipo de fenômeno é particularmente comum quando se tratam de sentenças longas.

Visando aumentar a proporção de pares com inferência, foram realizadas pequenas mudanças nas sentenças durante a segunda etapa do processo listado na Seção 5.1.1. Assim, passou-se a remover pequenos trechos de uma ou ambas as sentenças, caso as alterações possibilitassem a inferência. Apesar da proporção final estar menos desequilibrada que o observado no conjunto piloto, ainda há muito menos pares com inferência e especialmente paráfrases do que neutros.

5.2 A Avaliação Conjunta

O workshop ASSIN teve lugar em paralelo com o PROPOR 2016, no dia 13 de julho de 2016⁶, onde os participantes apresentaram os métodos que usaram. O evento também promoveu a discussão sobre a metodologia de criação de um corpus desta natureza, já que se mostrou haver interesse na comunidade de PLN em português no tema. Embora não se tenha chegado a algum consenso, a discussão motivou outros pesquisadores que podem organizar esforços para a anotação de novos corpora para RIT.

O corpus ASSIN foi dividido em seções de treinamento (com três mil pares de cada variante) e teste (com os dois mil restantes de cada). A metade brasileira do corpus de treinamento foi disponibilizada em 20 de novembro de 2015, e a metade portuguesa foi disponibilizada dois meses depois. Os participantes receberam o conjunto de teste (sem os rótulos corretos dos pares) em 4 de março de 2016, e tiveram 8 dias para enviar aos organizadores os arquivos com as respostas produzidas por seus sistemas. Cada participante pôde enviar até três resultados. A seguir, são descritos brevemente os baselines da tarefa e os resultados obtidos. As descrições dos sistemas participantes foram feitas na Seção 3.5.

5.2.1 Baselines

Foram usadas duas estratégias como baseline para o ASSIN: a primeira memoriza a média das similaridades do corpus de treino e a classe de inferência mais comum, e emite esses valores para todos os pares de teste. A segunda, um pouco mais sofisticada, consiste no treinamento de um classificador de regressão logística e um regressor linear para as tarefas de RIT e similaridade semântica, respectivamente. Estes dois modelos são treinados com apenas dois atributos: a proporção de tokens exclusivos da primeira e da segunda sentença.

5.2.2 Participantes

Participaram do ASSIN seis equipes, sendo três brasileiras e três portuguesas. Cada uma pôde enviar até três saídas dos seus sistemas para cada combinação de variante e subtarefa. As seis equipes participaram da tarefa de similaridade semântica, e quatro delas participaram da inferência textual. Foram exploradas diferentes abordagens para tratar os problemas, mas nem todas foram capazes de superar os baselines. A descrição das estratégias elaboradas pelos participantes foi mostrada na Seção 3.5.

⁶ A descrição completa do workshop está em http://propor2016.di.fc.ul.pt/?page_id=381

5.2.3 Resultados

As métricas usadas na avaliação das duas tarefas são consoantes com as usadas em avaliações conjuntas internacionais. Na tarefa de similaridade textual, foi usada a correlação de Pearson, tendo o erro quadrático médio (MSE, *mean square error*) como medida secundária. Idealmente, os sistemas devem ter a maior correlação possível e o menor MSE possível. Para a inferência, foi usada a medida F_1 , tendo a acurácia como medida secundária.

As Tabelas 14 e 15 listam os resultados das tarefas de similaridade e inferência, respectivamente, obtidos por cada participante em suas três execuções. São também apresentados resultados de sistemas baseline, descritos a seguir.

Equipe	Exec.	PB		PE		Geral	
		Pearson	MSE	Pearson	MSE	Pearson	MSE
Solo Queue	1	0,58	0,50	0,55	0,83	0,56	0,66
	2	0,68	0,41	0,00	1,55	0,29	0,98
	3	0,70	0,38	0,70	0,66	0,68	0,52
Reciclagem	1	0,59	1,36	0,54	1,10	0,53	1,23
	2	0,59	1,31	0,53	1,14	0,54	1,23
	3	0,58	1,37	0,53	1,18	0,53	1,27
Blue Man Group	1	0,65	0,44	0,63	0,73	0,63	0,59
	2	0,64	0,45	0,64	0,72	0,63	0,59
ASAPP	1	0,65	0,44	0,68	0,70	0,65	0,57
	2	0,65	0,44	0,67	0,71	0,64	0,58
	3	0,65	0,44	0,68	0,73	0,65	0,58
LEC-UNIFOR	1	0,62	0,47	0,64	0,72	0,62	0,59
	2	0,56	2,83	0,59	2,49	0,57	2,66
	3	0,61	1,29	0,63	1,04	0,61	1,17
L2F/INESC-ID	1			0,73	0,61		
	2			0,63	0,70		
	3			0,63	0,70		
Baseline (média)	–	0,00	0,76	0,00	1,19	-0,08	0,97
Baseline (sobreposição)	–	0,63	0,46	0,64	0,75	0,62	0,60

Tabela 14 – Resultados oficiais de todas as execuções para a tarefa de similaridade semântica

Para a similaridade semântica, como mostrado na Tabela 14, o primeiro *baseline* obteve 0 na correlação de Pearson pelo fato de não haver variação em suas respostas, e a medida ser baseada na correlação de duas variáveis. Ao se combinar as respostas para as duas metades do corpus, é obtido um valor negativo, indicando uma performance pior que dar a mesma resposta sempre.

Equipe	Exec.	PB		PE		Geral	
		Acurácia	F1	Acurácia	F1	Acurácia	F1
Reciclagem	1	77,65%	0,29	73,10%	0,43	75,38%	0,4
	2	79,05%	0,39	72,10%	0,38	75,58%	0,38
	3	78,30%	0,33	70,80%	0,32	74,55%	0,32
Blue Man Group	2	81,65%	0,52	77,60%	0,61	79,62%	0,58
ASAPP	1	81,20%	0,50	77,75%	0,57	79,47%	0,54
	2	81,65%	0,47	78,90%	0,58	80,27%	0,54
	3	77,10%	0,50	74,35%	0,59	75,72%	0,55
L2F/INESC-ID	1			83,85%	0,7		
	2			78,50%	0,58		
	3			78,50%	0,58		
Baseline (maioria)	–	77,65%	0,29	69,30%	0,27	73,47%	0,28
Baseline (sobreposição)	–	82,80%	0,64	81,75%	0,7	82,27%	0,67

Tabela 15 – Resultados oficiais de todas as execuções para a tarefa de inferência textual

No entanto, considerando o MSE, esse baseline teve resultados melhores que algumas execuções dos participantes. Isso significa que tais execuções computaram valores muito distantes da similaridade real dos pares. Já o segundo baseline teve resultados competitivos, chegando a superar diversas execuções.

Quanto à inferência, com resultados na Tabela 15, o primeiro baseline é também facilmente superado, mas o segundo se saiu bastante bem. Na variante brasileira, chegou a superar todos os três participantes e, na europeia, apenas uma execução da L2F/INESC-ID se saiu melhor.

O último resultado foi bastante inesperado. Apesar de toda a modelagem do problema feita pelas equipes participantes, um baseline com apenas dois atributos simples, sem acesso a nenhum recurso externo e usando apenas modelos lineares foi capaz de superar quase todos os sistemas empregados na tarefa. Ao mesmo tempo, esse resultado indica que a presença de inferência no ASSIN é fortemente relacionada com a sobreposição lexical, ainda que tenhamos nos esforçado em incluir tanto pares com inferência que tivessem palavras distintas quanto pares sem relação e palavras compartilhadas.

5.3 Publicações após a Avaliação Conjunta

Desde a publicação do conjunto de dados, tem-se conhecimento de outros três trabalhos publicados sobre o ASSIN, descritos a seguir.

Silva *et al.* (2017) trabalharam apenas com STS, e tentaram combinar uma representação de atributos baseados em conhecimento linguístico com outros baseados em

estatísticas de cada par de sentença. Seu sistema pré-processa os pares, removendo pontuação, números e normalizando algumas palavras (não ficou claro o processo de normalização usado). Quando dois sinônimos são encontrados entre as duas sentenças, um dele é trocado para que a mesma palavra apareça nas duas.

Os autores usaram também representações vetoriais de palavras na forma de embeddings e TF-IDF, que são somados para se chegar a uma representação da sentença. A partir daí, é calculado o cosseno entre cada par $\langle T, H \rangle$. Outros atributos utilizados são a proporção de palavras em comum e palavras diferentes de cada sentença. O modelo obtém bons resultados, acima do baseline do ASSIN: 0,64 de ρ de Pearson e MSE de 0,44.

Feitosa e Pinheiro (2017) trabalharam com o RIT, e buscaram combinar atributos usados por Fialho *et al.* (2016) (que obteve os melhores resultados para a tarefa em português europeu) com novos atributos relacionados à similaridade semântica. Foram usados um total de oito atributos que refletiam a proximidade entre duas palavras na WordNet (FELLBAUM, 1998) — estes foram calculados para cada par de palavras w_1, w_2 presente em $\langle T, H \rangle$, e em seguida sua média foi tomada como um atributo do par. Além disso, foram calculados os atributos tanto sobre as sentenças originais quanto sobre transcrições fonéticas simplificadas.

No entanto, mesmo experimentando variações nas combinações de atributos e com técnicas de amostragem para lidar com o desbalanceamento do corpus, Feitosa e Pinheiro (2017) não conseguiram melhorias significativas dos resultados. Sua melhor configuração chegou a 0,71 de medida F_1 , usando os dados da variante europeia (PE).

Pinheiro *et al.* (2017) usaram uma abordagem de computar a similaridade entre T e H a partir de quatro métricas. A primeira é o cosseno entre os vetores TF-IDF de cada sentença. A segunda calcula a similaridade entre os vetores de embeddings (obtidos com word2vec) das palavras de T com as de H , e em seguida obtém a média dos valores mais altos para cada palavra. A terceira computa a proporção de palavras em comum das duas sentenças. A quarta medida é a proporção do tamanho de uma sentença em relação à outra. Todas estas são computadas após um processo de *stemming*.

Usando estes atributos em conjunto com um regressor linear, obtiveram os melhores resultados reportados para a tarefa de similaridade, com ρ de Pearson de 0,70. Os mesmos atributos foram usados para classificação, obtendo os melhores resultados com uma rede MLP simples (sem camadas ocultas), com 83,05% de acurácia.

5.4 Considerações Finais

Foi descrita a proposta da Avaliação de Similaridade Semântica e Inferência Textual, como foi criado seu corpus anotado, quais foram as equipes participantes da avaliação

conjunta e os resultados que obtiveram. Apresentaram-se, ainda, dois sistemas baseline bastante simples, mas dos quais um superou a maioria dos participantes na tarefa de inferência textual. A seguir, apresentam-se algumas conclusões que dizem respeito à criação do corpus e aos sistemas desenvolvidos para a tarefa.

Nova Divisão do Corpus

Após a avaliação conjunta, achou-se conveniente padronizar uma nova divisão do corpus ASSIN, contando com um conjunto de validação à parte dos de treino e teste. Para tal, foram selecionados 500 exemplos do conjunto de treino de cada variante da língua, totalizando 1000 exemplos. Os conjuntos de teste foram mantidos inalterados. O ASSIN com esta nova divisão está disponível em <<http://nilc.icmc.usp.br/assin/>>.

Criação do Corpus

O método usado para a compilação do corpus, apesar de funcional, apresenta alguns empecilhos. O primeiro é o gargalo da etapa de limpeza antes da anotação em si. Durante essa etapa, os critérios para se eliminar ou editar pares são bastante delicados, como nossa experiência mostrou. É uma parte da anotação que deve ficar a cargo de pessoas que tenham conhecimento sobre a tarefa e seus objetivos, e dificilmente poderia ser delegada para uma plataforma de *crowdsourcing*.

Outra dificuldade diz respeito à subjetividade da tarefa. Em alguns casos, os anotadores gastaram bastante tempo tentando se decidir quanto aos julgamentos que deveriam dar para certos pares. Esse tipo de problema retoma o anterior: certas alterações no conteúdo das sentenças torna as decisões mais fáceis, e portanto, a anotação mais confiável e produtiva.

Abordagens

Os participantes do ASSIN exploraram diferentes tipos de estratégia para as duas tarefas propostas. É particularmente interessante notar que dentre os melhores resultados obtidos estão duas abordagens muito simples: na similaridade semântica, a comparação da combinação de vetores de palavras, como feito pelo Solo Queue (HARTMANN, 2016); e para inferência, a comparação da proporção de palavras exclusivas de cada sentença, que foi um dos baselines propostos.

Todavia, a equipe L2F/INESC-ID (FIALHO *et al.*, 2016) obteve os melhores resultados do ASSIN na variante europeia (a única em que competiu), empregando um sistema baseado em um rico conjunto de atributos. Esse resultado indica que superar métodos simples como os listados acima requer uma modelagem extensiva do problema.

Outra linha de pesquisa bastante bem sucedida na literatura recente são redes neurais, como mostrado no Capítulo 4. O Blue Man Group (BARBOSA *et al.*, 2016) foi o único grupo a explorá-las, mas as descartou após obter resultados preliminares negativos. Uma possível explicação para esse fato é que o conjunto de dados do ASSIN é menor e com sentenças mais complexas do que as que se encontram para conjuntos semelhantes em inglês, nos quais os modelos neurais obtêm os melhores resultados.

Por fim, notamos que nenhum dos participantes modelou as sentenças em alguma estrutura sintática ou semântica; em vez disso, todos exploraram apenas o nível lexical. Pelo menos para a inferência textual, há evidências na literatura de que a compreensão da estrutura das sentenças tem um papel importante (DAGAN *et al.*, 2013), e a ausência desse tipo de análise pode explicar o desempenho dos sistemas abaixo do baseline.

Futuras Edições

Novas edições do ASSIN teriam o potencial de estimular e melhorar a pesquisa nas duas tarefas propostas para a língua portuguesa. No entanto, acredita-se que seria interessante trabalhar com outros tipos de pares de sentença, especialmente na tarefa de inferência.

Uma possibilidade seria o uso de pares de sentenças escritos especificamente com o objetivo de terem ou não uma relação de implicação, como foi feito no SICK e SNLI. Nesse caso, a subjetividade da anotação é reduzida drasticamente, com o preço de não se trabalhar com um cenário realista. De fato, a motivação principal da criação destes dois corpora foi fornecer um ambiente para sistemas de PLN aprenderem o funcionamento de certos mecanismos da linguagem humana.

Outro direcionamento seria usar apenas fatos simples, na forma de sentenças com uma única oração, como o segundo componente de cada par. Essa foi a estratégia adotada na criação dos corpora dos RTE Challenges, e mantêm o realismo da tarefa na medida em que a primeira sentença pode ser extraída de um jornal ou outra fonte real. Por outro lado, esse cenário não requer que os sistemas processem e comparem duas sentenças inteiras, mas apenas busque por confirmação de um fato.

Por fim, uma estratégia que facilitasse a anotação do corpus seria também interessante por permitir a criação de um novo recurso em maior escala, tornando mais viável a exploração de métodos neurais que necessitam de grandes bases de treinamento.

MÉTODOS PROPOSTOS PARA RIT

Neste capítulo, são apresentados os modelos propostos para RIT: o TEDIN e o Infernal. O primeiro é um modelo de rede neural, baseado no conceito de TED (Seção 6.1), enquanto o segundo está no paradigma clássico de aprendizado de máquina, definindo atributos manualmente para descrever os dados (Seção 6.2). Por fim, são trazidas as considerações finais a respeito de suas performances em conjuntos de dados de RIT (Seção 6.3).

6.1 TEDIN

O TEDIN (*Tree Edit Distance Network*) é um modelo neural para TED, fundamentado em possibilitar uma representação flexível para operações de edição. O modelo gera uma representação vetorial $\mathbf{v} \in \mathbb{R}^d$ para cada operação, permitindo que sejam usadas como dados de entrada para um modelo neural de RIT.

A motivação para seu desenvolvimento foi a de flexibilizar o funcionamento de algoritmos de TED, fazendo com que custos otimizados de operações fossem aprendidos automaticamente. Neste sentido, o TEDIN é uma melhoria bastante direta do modelo de Zanolini e Colombo (2016), que trabalha com custos fixos. Além disso, como se observou no Capítulo 5, os trabalhos de RIT publicados com dados do ASSIN não exploraram as estruturas sintáticas das sentenças, e aqui buscou-se uma nova forma de fazê-lo.

A seguir, é descrito o pré-processamento dos pares tratados pelo TEDIN (Seção 6.1.1), seu módulo de cálculo de TED (Seção 6.1.2), o módulo de classificação para RIT (Seção 6.1.3) e a avaliação nos conjuntos ASSIN, SICK e SNLI, sendo comparado com dois outros métodos da literatura: a rede neural modular (PARIKH *et al.*, 2016) e uma implementação do AdArte (ZANOLI; COLOMBO, 2016) (Seção 6.1.4). Por fim, na Seção 6.1.5 é mostrado um exemplo de cálculo de TED com custos de operações calculados

pelo TEDIN.

6.1.1 Pré-processamento

Para o cálculo da TED, é naturalmente necessário que se tenha estruturas arbóreas. Para tal, são geradas as árvores de dependência para cada sentença dos pares processados pelo TEDIN.

Usou-se o parser do pacote de código aberto CoreNLP¹ (MANNING *et al.*, 2014), treinado com dados em português provenientes do projeto Universal Dependencies² (UD; mais especificamente, da versão 2.0 do conjunto Portuguese-BR), também disponível abertamente³.

6.1.2 Cálculo de TED

A representação de uma inserção, remoção ou substituição é obtida pela aplicação de uma função $F_i(\cdot)$, $F_r(\cdot)$ ou $F_s(\cdot)$, respectivamente, sobre os nós envolvidos na operação. Cada uma destas funções é implementada por uma camada de rede neural MLP com d unidades. No caso da inserção e remoção, a entrada das funções é a representação vetorial do nó inserido ou removido; já na substituição, concatenam-se as representações dos dois nós com sua diferença: $w = [w_1; w_2; w_1 - w_2]$.

Os nós, por sua vez, são representados pela concatenação do vetor de embedding da palavra correspondente com alguma informação extra. Nos experimentos realizados, avaliou-se o uso de um vetor de embedding do rótulo da relação de dependência na árvore sintática. Embeddings para rótulos de dependência são inicializadas aleatoriamente e ajustadas ao longo do treinamento do modelo.

A Figura 11 ilustra a arquitetura do TEDIN para o cálculo do custo de uma operação envolvendo um único token, como é o caso da inserção ou remoção. Para substituições, a camada de embedding deveria conter a representação de um segundo token e sua diferença para o primeiro. Cada retângulo da figura representa um vetor, e o círculo final representa um único valor. Os trios de setas entre a camada de embedding e a representação, bem como entre esta e o custo, representam a aplicação de camadas densas.

6.1.2.1 Treinamento

Antes do treinamento, é interessante que o TEDIN produza custos próximos a 1 para todas as entradas, funcionando similarmente a uma parametrização simples de

¹ Mais informações em <<https://stanfordnlp.github.io/CoreNLP/>>

² Mais informações em <<http://universaldependencies.org/>>

³ O treinamento e configuração do CoreNLP com este corpus é descrito em detalhes em <<https://goo.gl/NY5PNX>>

Figura 11 – Ilustração da arquitetura do TEDIN para o cálculo do custo de uma operação. A *Tag* pode se referir a qualquer informação extra sobre a palavra, como seu rótulo na árvore de dependências.

um algoritmo de TED. Ao ser treinado, passa a produzir custos mais razoáveis para cada operação. Para tal, o viés b associado ao cálculo do custo é inicializado como 1, em vez de 0, que é prática mais comum com redes neurais. Retomando a Equação 4.2 e a adaptando para o caso de uma única unidade, temos:

$$y = f(\mathbf{w}x + b) \quad (6.1)$$

Conforme implementado, para calcular efetivamente a TED entre duas sentenças, calculam-se inicialmente os custos de inserção, remoção e substituição de todas as combinações de palavras. Em seguida, é executado o algoritmo de Zhang-Shasha tomando estes valores. O cálculo a priori dos custos de operação resulta em ganho significativo de performance, dado que o algoritmo precisa verificar alguns custos mais de uma vez.

Além disso, as implementações de bibliotecas de álgebra linear para redes neurais realizam processamento vetorial, fazendo com que o tempo computacional para se calcular os custos de operação tenha complexidade menor que linear quanto ao número de operações. Com efeito, durante o treinamento, são calculados de uma única vez os custos de operações para todo um batch de pares de sentenças, que tem tamanho tipicamente entre 8 e 64 pares.

Um empecilho para o treinamento do TEDIN é o fato de não se ter uma ideia clara de quais devem ser os custos de cada possível operação de edição entre duas sentenças, ou mesmo sua distância de edição total. Dessa forma, não há sentido em se produzir um conjunto de dados contendo custos de TED para treinar um modelo.

Por outro lado, é factível determinar pares de sentenças mais próximos entre si do que outros. Tomando um conjunto de dados com esta distinção, pode-se treinar um modelo para distinguir pares mais similares de menos similares, e deixá-lo ajustar os custos de operações individuais como parte não diretamente supervisionada de seu treinamento.

Conjuntos de dados de RIT podem ser usados como fonte de tais dados: podem-se considerar pares com relação de paráfrase ou implicação como *positivos*, enquanto que pares sem relação servem de exemplos *negativos*. De modo geral, deseja-se que a distância calculada entre as sentenças de um par negativo seja maior do que a de um par positivo. Em outras palavras, espera-se que as sentenças de pares negativos sejam mais diferentes entre si do que as dos pares positivos.

O treinamento do TEDIN para o cálculo da TED é ilustrado na Figura 12 e se dá da seguinte forma:

1. Dado um conjunto de pares positivos $\mathcal{D}^+$ e um de pares negativos $\mathcal{D}^-$, é selecionado um par positivo p^+ e um negativo p^- .
2. Os valores de TED de cada um, $TED(p^+)$ e $TED(p^-)$, são calculados aplicando-se o algoritmo de Zhang-Shasha com os custos de operações determinados pelo TEDIN.
3. Por fim, calcula-se o valor da função de perda que se deseja minimizar:

$$L(p^+, p^-) = \max(1 + TED(p^+) - TED(p^-), 0) \quad (6.2)$$

Esta função impõe uma margem de valor 1 com a qual a TED de um par negativo deve ser maior do que a de um par positivo. Quando esta margem é superada, a função $L(\cdot)$ tem valor nulo.

4. Uma vez calculado o valor de perda, o TEDIN ajusta seus pesos por meio do *back-propagation*.

Durante o treinamento, a amostragem de pares de $\mathcal{D}^+$ e $\mathcal{D}^-$ é feita de modo independente, o que permite maior variação nos dados. Para a avaliação da performance do modelo, é usado um conjunto fixo de dados de validação.

No experimentos com o modelo, observou-se que havia uma tendência de os pesos para todas as operações tenderem a zero. Para evitar isto, um termo de regularização pode ser incluído no cálculo da função de perda. Sendo O_i , O_r e O_s respectivamente os conjuntos de operações de inserções, remoções e substituições, e C_i , C_r e C_s os custos totais das operações de cada um dos conjuntos, o termo de regularização é definido da seguinte forma:

Figura 12 – Treinamento do TEDIN para comparar pares positivos e negativos

$$R_i = \max(|O_i| - C_i, 0) \quad (6.3)$$

$$R_r = \max(|O_r| - C_r, 0) \quad (6.4)$$

$$R_s = \max(|O_s| - C_s, 0) \quad (6.5)$$

$$R = \lambda (R_i + R_r + R_s) \quad (6.6)$$

onde $\lambda \geq 0$ é um hiperparâmetro que define a importância da regularização. O termo R aumenta conforme o custo médio das operações cai abaixo de 1. A função de perda final, incorporando R , é descrita como:

$$L(p^+, p^-) = \max(1 + TED(p^+) - TED(p^-), 0) + R \quad (6.7)$$

6.1.3 Classificação para RIT

Uma vez que o TEDIN tenha sido treinado para estimar custos de operações de edição e gerar representações vetoriais para as mesmas, pode ser usado para a classificação de pares quanto à presença de implicação textual.

Seu funcionamento para classificação é ilustrado na Figura 13. Para classificar um par, inicialmente é executado o algoritmo Zhang-Shasha com os custos de operação calculados pelo TEDIN (esta etapa não é mostrada na figura). Neste momento, o custo total de TED não é considerado, apenas o conjunto O de operações envolvidas em gerar o menor custo possível. Os vetores $\mathbf{x}_i$ das operações, com $1 \leq i \leq |O|$, são os dados de entrada para o módulo de classificação, e são concatenados em uma matriz $\mathbf{M}$:

$$\mathbf{M}_{[i,*]} = \mathbf{x}_i \quad \mathbf{M} \in \mathbb{R}^{|O| \times d} \quad (6.8)$$

A matriz $\mathbf{M}$ passa por uma camada densa, gerando outra matriz $\mathbf{H}$, da qual são extraídos os valores máximos de cada dimensão. Esta operação de máximo reduz a dimensão relativa à quantidade de operações, resultando em um vetor $\mathbf{h}_1$:

$$\mathbf{H} = \text{relu}(\mathbf{M}\mathbf{W}_1 + \mathbf{b}_1) \quad \mathbf{H} \in \mathbb{R}^{|O| \times d} \quad (6.9)$$

$$\mathbf{h}_1[i] = \max(\mathbf{H}_{[i,*]}) \quad \mathbf{h}_1 \in \mathbb{R}^d \quad (6.10)$$

$\mathbf{W}_1 \in \mathbb{R}^{d \times d}$ e $\mathbf{b}_1 \in \mathbb{R}^d$ são parâmetros treináveis do modelo. O vetor $\mathbf{h}_1$ pode ser visto como uma representação vetorial de tamanho fixo para todo o conjunto de operações que transformam a primeira sentença na segunda. Este passa ainda por uma camada densa oculta e, por fim, pela camada softmax final que corresponde às possíveis classes do problema.

$$\mathbf{h}_2 = \text{relu}(\mathbf{h}_1\mathbf{W}_2 + \mathbf{b}_2) \quad \mathbf{h}_2 \in \mathbb{R}^d \quad (6.11)$$

$$\mathbf{y} = \text{softmax}(\mathbf{h}_2\mathbf{W}_3 + \mathbf{b}_3) \quad \mathbf{y} \in \mathbb{R}^{|C|} \quad (6.12)$$

$\mathbf{W}_2 \in \mathbb{R}^{d \times d}$, $\mathbf{W}_3 \in \mathbb{R}^{d \times |C|}$, $\mathbf{b}_2 \in \mathbb{R}^d$ e $\mathbf{b}_3 \in \mathbb{R}^{|C|}$ são parâmetros treináveis, e C é o conjunto de classes que o modelo é treinado para discriminar.

O treinamento do TEDIN, portanto, dá-se em duas etapas: uma em que pares positivos e negativos são comparados, objetivando-se otimizar os pesos neurais para a representação de operações; e o segundo em que a classificação para RIT (ou outra tarefa) é feita em função das operações de TED usadas. Na segunda parte, os pesos referentes ao cálculo do custo de uma operação não são afetados. As duas etapas são chamadas, respectivamente, de *ranking* (pois os pares positivos devem ser ranqueados com distância menor que os negativos) e *classificação*.

Figura 13 – Arquitetura do TEDIN para classificação

6.1.4 Avaliação

Aqui são descritos os experimentos realizados com o TEDIN no ASSIN, SICK e SNLI. Para efeito de comparação, foram avaliados outros dois modelos de RIT: uma implementação da arquitetura modular de Parikh *et al.* (2016), aqui referenciada por DFF (*decomposable feed forward*); e uma implementação do AdArte (ZANOLI; COLOMBO, 2016). O primeiro modelo foi escolhido por ser outra arquitetura neural, enquanto o segundo se baseia no mesmo fundamento do TEDIN, mas sem sua flexibilidade.

Todos os modelos foram implementados em Python, e os modelos neurais com auxílio da biblioteca Tensorflow (ABADI *et al.*, 2015). Para o TEDIN e AdArte, foi usada a implementação do algoritmo de Zhang-Shasha disponível em <<https://github.com/timtadh/zhang-shasha>>. O AdArte usou algoritmos de aprendizado de máquina disponíveis na biblioteca scikit-learn (PEDREGOSA *et al.*, 2011).

A implementação do AdArte, similarmente à original, considera que dois nós de uma árvore são equivalentes quando (i) têm o mesmo rótulo de dependência e (ii) o mesmo lema, ou são sinônimos. Para a verificação de sinonímia, foi usada a OpenWordNet-PT (PAIVA; RADEMAKER; MELO, 2012), o recurso lexical mais amplo disponível para o português.

A implementação do TEDIN encontra-se disponível em <<https://github.com/erickrf/tedin>>; a do DFF, em <<https://github.com/erickrf/multiffn-nli>>; e a do AdArte, em <<https://github.com/erickrf/infernal>>.

6.1.4.1 ASSIN

Para o treinamento no ASSIN, foram usados os seguintes hiperparâmetros. Todas as camadas ocultas têm 100 unidades (incluindo as representações vetoriais das transformações), e os rótulos de dependência são mapeados para embeddings de 10 dimensões — no total, são pouco mais de 186 mil parâmetros para serem aprendidos. O aumento no número de unidades não trouxe nenhum ganho de performance. Na etapa de ranking, foi

usada uma taxa de aprendizado de 0,001, 10% de dropout, e a constante λ do termo de regularização R foi de 0,01. O modelo foi treinado com batches de 32 pares, e sua performance no conjunto de validação foi avaliada a cada 50 batches, salvando um novo modelo sempre que o valor da perda diminuísse. Os mesmos hiperparâmetros foram usados na etapa de classificação, e o modelo foi otimizado pelo algoritmo Adam (KINGMA; BA, 2014).

Com o DFF, foram usadas camadas ocultas também de 100 unidades, com taxa de aprendizado de 0,05, sem dropout. Foram usados batches de 8 unidades, com validação a cada 50 batches. Esta instanciación do modelo contém cerca de 151 mil parâmetros, e foi otimizada pelo algoritmo Adagrad (DUCHI; HAZAN; SINGER, 2011). Em ambas as abordagens neurais, os valores de embeddings das palavras foram mantidos constantes.

Para o AdArte, foram verificados no conjunto de treinamento 947 diferentes tipos de operação — recorde-se que o modelo representa cada transformação pelo seu tipo (inserção, remoção e substituição) e o rótulo de dependência envolvido (ou dois rótulos, no caso de substituição). Desta forma, considerando o conjunto de 38 rótulos de dependência do corpus utilizado, o número máximo de operações possíveis seria $38 + 38 + 38 \times 38 = 1.520$.

Um classificador é treinado com vetores que representem estas 947 operações e, quando avaliado com dados em que hajam operações ausentes do conjunto de treinamento, estas são ignoradas. Foi avaliado um modelo de regressão logística, por ser um classificador relativamente simples de se configurar e com baixo custo computacional. Não se pretendia, de qualquer forma, uma análise extensa de implementações do AdArte.

Em todos os modelos, foi experimentado também definir uma importância maior para os exemplos de treinamento de classes das quais menos exemplos haja. Para tal, um fator de balanceamento de cada classe é calculado como o inverso de sua frequência no conjunto de dados, e o ajuste de pesos do modelo é proporcional a este fator.

Foram combinados os dados das duas variantes do português no conjunto de treinamento e no de validação. Esta estratégia foi empregada com sucesso por Fialho *et al.* (2016), resultando em suas melhores performances.

A Tabela 16 mostra os resultados obtidos. A performance de todos os métodos ficou bastante abaixo do baseline de sobreposição de palavras. No caso do DFF, que apresenta bons resultados no SNLI⁴, podem-se identificar duas causas principais: (i) a pequena quantidade de dados; recorde-se que o conjunto de treino do ASSIN conta com apenas cinco mil exemplos, contra 550 mil do SNLI, e (ii) as sentenças do ASSIN são mais complexas e com vocabulário mais amplo.

⁴ A implementação do DFF usada no ASSIN foi também avaliada no SNLI, e obteve resultados coerentes com os de Parikh *et al.* (2016).

Modelo	Validação		PT-BR		PT-PT		Geral	
	Acurácia	F ₁	Acurácia	F ₁	Acurácia	F ₁	Acurácia	F ₁
TEDIN	74,40%	0,56	72,70%	0,46	71,60%	0,53	72,15%	0,5
TEDIN, balanceado	72,20%	0,28	77,05%	0,35	69,35%	0,28	72,92%	0,33
DFF	76,10%	0,61	77,60%	0,57	76,50%	0,59	77,05%	0,58
DFF, balanceado	63,20%	0,52	64,70%	0,49	66,15%	0,54	65,42%	0,51
AdArte	73,20%	0,44	77,65%	0,47	74,10%	0,49	75,88%	0,48
AdArte, balanceado	70,20%	0,52	70,20%	0,49	72,55%	0,55	71,38%	0,52
L2F/INESC-ID	—	—	85,85%	0,66	84,90%	0,71	—	—
Baseline	81,40%	0,69	82,80%	0,64	81,75%	0,7	82,27%	0,67

Tabela 16 – Performance do TEDIN, DFF e AdArte no ASSIN. A parte inferior da tabela traz os resultados do estado-da-arte e do baseline de sobreposição.

Já para o TEDIN, que obteve resultados ainda inferiores, os mesmos pontos são válidos, mas há também de se questionar a adequação da TED para a tarefa de RIT. Com efeito, o desempenho do AdArte foi também fraco, o que contraria a hipótese de que a TED contenha informação suficiente para a tarefa. O bom resultado obtido pelo AdArte no SICK (reportado por [Zanoli e Colombo \(2016\)](#)) pode ser atribuído à simplicidade do corpus.

Em geral, o desempenho do AdArte foi superior ao do TEDIN. O conjunto de validação, onde o TEDIN alcançou melhores valores, foi justamente usado como referencial para a seleção do melhor modelo, e portanto era de se esperar melhor performance no mesmo.

Por fim, observa-se que o uso de balanceamento do peso de classes resultou em piora de performance para o TEDIN e DFF, e aumento do F₁ em detrimento da acurácia para o AdArte. O último resultado pode ser explicado pelo fato de a técnica favorecer a classificação de exemplos de classes minoritárias. Já o primeiro é menos claro — possivelmente, aumentar o peso dos exemplos de implicação e paráfrase pode ter feito o modelo se sobreadaptar (*overfit*) com esses.

6.1.4.2 SNLI

O corpus SNLI, por conter mais dados e sentenças mais simples, traz um cenário mais favorável para avaliar o TEDIN.

O modelo treinado no SNLI tem 300 unidades em suas camadas ocultas e usa embeddings de 20 dimensões para rótulos de dependência. A etapa de ranking usou taxa de aprendizado de 0,001, dropout de 40%, regularização λ de 0,01 e batches de 32 unidades, avaliando a performance no conjunto de validação a cada 50 batches. Na classificação, foram usados os mesmos hiperparâmetros, exceto pelo tamanho de batch de 64 unidades. A etapa de ranking foi executada por 34 mil batches e a de classificação por 20 mil, a partir de onde não houve mais melhoras de performance na validação.

Modelo	Treino	Validação	Teste
TEDIN	73,14%	76,68%	75,39%
KIM	94,1%	—	88,6%
DFF	89,5%	—	86,3%
Baseline não-lexicalizado	49,4%	—	50,4%
Baseline lexicalizado	99,7%	—	78,2%

Tabela 17 – Performance do TEDIN no SNLI

Modelo	Treino	Validação	Teste
TEDIN	85,31%	82,83%	80,17%
AdArte	—	—	82,4%
Mediana SemEval 2014	—	—	77,06%
Baseline	—	—	56,2%

Tabela 18 – Performance do TEDIN e AdArte no SICK. Os valores mostrados são a acurácia.

Vistos os resultados negativos do balanceamento de classes para os modelos neurais no ASSIN, a técnica não foi utilizada em modelos treinados para o SNLI. A Tabela 17 traz a performance do TEDIN no conjunto, além de valores do modelo KIM, estado-da-arte no conjunto sem o uso de ensembles (CHEN *et al.*, 2017b), do DFF e dos baselines criados pelos autores do SNLI.

O resultado do TEDIN é inferior a todos os modelos, exceto pelo baseline não-lexicalizado. Este resultado contraria fortemente a ideia de que a TED seja adequada para o RIT, especialmente com a implementação usada no TEDIN.

6.1.4.3 SICK

O modelo instanciado tem apenas 50 unidades nas camadas ocultas e as embeddings de rótulos de dependência têm 10 unidades. Na etapa, de ranking, foi usada taxa de aprendizado de 0,001, dropout de 40%, regularização λ de 0,01 e batches de 32 exemplos, com o modelo avaliado a cada 50 batches. O modelo convergiu em apenas 450 batches. Para a etapa de classificação, foram usados os mesmos hiperparâmetros, com o melhor resultado na validação após 1200 batches.

A Tabela 18 mostra os resultados obtidos pelo TEDIN no SICK e pela implementação original do AdArte (ZANOLI; COLOMBO, 2016), que se desejava melhorar, além do baseline de sobreposição de palavras divulgado pelos criadores do corpus e a mediana dos resultados dos participantes na avaliação conjunta. Os dois últimos têm valores divulgados para o conjunto de teste.

Nota-se que o TEDIN teve performance inferior ao AdArte, contrariando a hipótese

Figura 14 – Árvore de dependência de um par do conjunto de validação do ASSIN

de que sua flexibilidade e arquitetura mais poderosa aumentariam a performance. Ainda assim, uma abordagem baseada em TED funciona bem no corpus, visto que ambos têm performance superior à mediana dos participantes da avaliação conjunta.

Uma vantagem do AdArte em relação ao TEDIN é o fato de consultar bases de conhecimento lexical, como a WordNet, para identificar palavras sinônimas ou que implicam em outras (como *child* → *boy*). No TEDIN, optou-se por não usar conhecimento externo, mas embasar todo o entendimento de relações lexicais nas representações vetoriais das palavras, o que é praxe em trabalhos de deep learning e parte de sua motivação prática. No entanto, no caso específico da TED, as consultas a bases externas podem fazer maior diferença, visto os resultados superiores do AdArte no SICK e ASSIN.

6.1.5 Análise de Transformações

Para se entender melhor as causas de erros do TEDIN, é interessante visualizar as operações de edição envolvidas no cálculo da TED com custos determinados pelo modelo. A seguir, é analisado um par do conjunto de validação do ASSIN e um do SNLI, cujas árvores de dependência são mostradas, respectivamente, nas Figuras 14 e 15, e com as operações de edição listadas nas Tabelas 19 e 20.

O mais relevante a se observar nas operações é que quase todas têm custo 0. Isto não faz sentido linguisticamente — operações de custo nulo deveriam acontecer apenas substituindo sinônimos, ou em algumas remoções que não comprometessem a presença de implicação. No entanto, palavras sem relação alguma entre si também produzem tais

Op	Nós	Custo	Resultado
S	Robert → Mary	0	Dirigido por Mary Stevenson, o longa de 1964 foi baseado no primeiro dos oito livros infantis de P.L. Travers.
S	Stevenson → Poppins	0	Dirigido por Mary Poppins , o longa de 1964 foi baseado no primeiro dos oito livros infantis de P.L. Travers.
R	por	0	Dirigido Ø Mary Poppins, o longa de 1964 foi baseado no primeiro dos oito livros infantis de P.L. Travers.
S	, → foi	0	Dirigido Mary Poppins foi o longa de 1964 foi baseado no primeiro dos oito livros infantis de P.L. Travers.
R	Dirigido	0	Ø Mary Poppins foi o longa de 1964 foi baseado no primeiro dos oito livros infantis de P.L. Travers.
R	o	0	Mary Poppins foi Ø longa de 1964 foi baseado no primeiro dos oito livros infantis de P.L. Travers.
S	1964 → primeiramente	0	Mary Poppins foi longa de primeiramente foi baseado no primeiro dos oito livros infantis de P.L. Travers.
R	de	0	Mary Poppins foi longa Ø primeiramente foi baseado no primeiro dos oito livros infantis de P.L. Travers.
R	longa	0	Mary Poppins foi Ø primeiramente foi baseado no primeiro dos oito livros infantis de P.L. Travers.
R	foi	0	Mary Poppins foi primeiramente Ø baseado no primeiro dos oito livros infantis de P.L. Travers.
S	P.L. → P.	0	Mary Poppins foi primeiramente baseado no primeiro dos oito livros infantis de P. Travers.
I	L.	0.41	Mary Poppins foi primeiramente baseado no primeiro dos oito livros infantis de P. L. Travers.
I	Travers	0.29	Mary Poppins foi primeiramente baseado no primeiro dos oito livros infantis de P. L. Travers Travers .
I	por	0.29	Mary Poppins foi primeiramente baseado no primeiro dos oito livros infantis de por P. L. Travers Travers.
S	Travers → escritos	0	Mary Poppins foi primeiramente baseado no primeiro dos oito livros infantis de escritos por P. L. Travers.
R	de	0	Mary Poppins foi primeiramente baseado no primeiro dos oito livros infantis Ø escritos por P. L. Travers.
I	no	0	Mary Poppins foi primeiramente baseado no no primeiro dos oito livros infantis escritos por P. L. Travers.
R	no	0	Mary Poppins foi primeiramente baseado Ø no primeiro dos oito livros infantis escritos por P. L. Travers.
S	baseado → inspirado	0.57	Mary Poppins foi primeiramente inspirado no primeiro dos oito livros infantis escritos por P. L. Travers.

Tabela 19 – Sequência de edições encontrada pelo TEDIN para transformar a primeira sentença da Figura 14 na segunda

Figura 15 – Árvores de dependência de um par do conjunto de validação do SNLI

Op	Nós	Custo	Resultado
I	Two	0	Two Two women are embracing while holding to go packages.
I	woman	0	Two Two woman women are embracing while holding to go packages.
I	are	0	Two Two woman women are are embracing while holding to go packages.
R	Two	0	Two ∅ woman women are are embracing while holding to go packages.
R	women	0	Two woman ∅ are are embracing while holding to go packages.
R	are	0	Two woman ∅ are embracing while holding to go packages.
R	while	0	Two woman are embracing ∅ holding to go packages.
R	holding	0	Two woman are embracing ∅ to go packages.
R	to	0	Two woman are embracing ∅ go packages.
R	go	0	Two woman are embracing ∅ packages.
S	embracing → holding	4.8	Two woman are holding packages.

Tabela 20 – Transformações do TEDIN para uma sentença do SNLI

custos. Ironicamente, a única operação de substituição na Tabela 19 com custo acima de zero, na última linha, envolve sinônimos; e a única operação com custo acima de zero na Tabela 20 envolve conceitos razoavelmente semelhantes.

Quando inserções e remoções tem custo 0, o algoritmo para o cálculo de TED pode optar por sequências pouco intuitivas: em vez de operações de *match*, que combinam a mesma palavra nas duas sentenças (omitidas das tabelas), uma palavra já existente na primeira sentença pode ser inserida, gerando uma repetição, para depois a palavra repetida ser removida ou substituída.

Foi mencionado na Seção 6.1.2 que já se observara a tendência de custos 0 para operações no TEDIN nos primeiros experimentos, e que por tal motivo se definira a

constante de regularização λ . Ainda assim, esta tendência continuou. Treinar o modelo com valores maiores de λ fez com que o mesmo divergisse, atingindo valores de perda acima de 1.

Analisando todos os pares do conjunto de validação do ASSIN, verificou-se que têm custo nulo 9,6% das operações de inserção, 97,2% das de remoção e 49,1% das de substituição de palavras de lemas diferentes. O fato de as remoções terem quase sempre custo nulo se justifica pela natureza do RIT: quando uma sentença H tem palavras a menos do que T , de modo geral ainda é viável que se mantenha uma relação de implicação, o que é mais difícil no caso de inserções, que em sua maioria têm um custo mais alto. Já as substituições estão em um meio-termo.

Com o conjunto de validação do SNLI, observou-se um efeito diferente: o custo nulo aparece em 98,8% das remoções, 63,1% das inserções e 10,7% das substituições. Em comparação ao ASSIN, as cifras de inserção e substituição são praticamente invertidas. Aqui, não se pôde encontrar uma justificativa para este comportamento do modelo.

Cabe também ressaltar o valor da função de perda obtido na etapa de ranking em ambos os conjuntos (valor que se deseja minimizar). No ASSIN, o valor foi de 0,67, contra 0,28 no SNLI. Em comparação, um modelo totalmente aleatório teria perda de 1, valor da margem definida na Equação 6.2. Em outras palavras, a performance da etapa de ranking no SNLI foi significativamente melhor que no ASSIN, de modo que o TEDIN treinado no SNLI encontra custos de TED menores para sentenças com implicação do que para sentenças neutras, apesar de os custos de operações individuais mostrados aqui não terem sentido linguístico.

Uma possível conclusão é de que o TEDIN aprendeu certas idiossincrasias dos conjuntos de dados durante seu treinamento que não necessariamente refletem diferenças relevantes para o RIT. De fato, em muitos pares do SNLI (inclusive o ilustrado na Figura 15) que têm relação de implicação, H é uma versão reduzida de T , enquanto pares neutros contém mais informação irrelevante. Desta forma, apenas o uso de várias operações de remoção (que acontece nos pares positivos) em contraste com eventuais inserções e substituições (nos pares negativos) é suficiente para discriminar bem as duas categorias, sem que o modelo leve em conta a semântica das palavras envolvidas.

6.2 Infernal

O Infernal (*INFERence in NATural Language*) é um modelo baseado na engenharia de atributos para realizar o RIT. Alguns dos atributos empregados são encontrados em trabalhos da literatura, enquanto outros são, até onde saibamos, inéditos. A seguir, são descritos o processo de pré-processamento dos pares (Seção 6.2.1), os atributos que lhes são extraídos (Seção 6.2.2), resultados de avaliação (Seção 6.2.3), uma análise da rele-

vância dos atributos propostos (Seção 6.2.4) e uma análise dos erros do modelo Infernal (Seção 6.2.5).

6.2.1 *Pré-processamento*

O pré-processamento dos pares usados pelo Infernal inclui anotação de árvores sintáticas, lematização de palavras, detecção de entidades nomeadas e busca por alinhamentos lexicais. Para o primeiro, é usada a mesma configuração do TEDIN, descrita na Seção 6.1.1.

A lematização é a conversão de palavras em seu lema, ou forma base, que é a que consta em dicionários. No caso de verbos, significa convertê-los para o infinitivo; no caso de substantivos, para o singular, removendo sufixos de aumentativo ou diminutivo e, se houver variantes feminina e masculina (como em *menino* e *menina*), para o masculino. Adjetivos seguem o mesmo processo que substantivos, mas sempre são convertidos para o masculino.

Para implementá-la, foi aproveitado o dicionário DELAF⁵, que contém lemas para mais de 9 milhões de formas flexionadas em português do Brasil. O que torna o recurso específico para esta variante do português é a sua grafia pré-acordo ortográfico, já que as regras de flexões das palavras em todas as variantes da língua são as mesmas. O recurso não inclui, por exemplo, palavras como *facto* ou *acção*, usadas na grafia europeia, ou *voo*, usado em ambas pós o acordo ortográfico. Para contornar esta limitação, sempre que uma palavra não é encontrada no dicionário, é verificado se contém uma das sequências de letras listadas na Tabela 21, e feita a substituição⁶.

Substituir	Por	Exemplo
ct	t	facto → fato
pt	t	óptimo → ótimo
cç	ç	acção → ação
pç	ç	adopção → adoção
mn	n	amnístia → anistia
oo	ôo	voo → vôo
ee	êe	veem → vêem

Tabela 21 – Sequências de letras substituídas ao se consultar o dicionário DELAF

A grande quantidade de formas flexionadas no dicionário se deve, em grande parte, às combinações de formas verbais flexionadas com pronomes clíticos. No entanto, o pro-

⁵ Disponível em <<http://www.nilc.icmc.usp.br/nilc/projects/unitex-pb/web/dicionarios.html>>

⁶ Os mantenedores do DELAF foram informados sobre as formas pré-acordo ortográfico, visto que o recurso é mantido atualizado. Ao momento de escrita da tese, após a realização dos experimentos, uma nova versão do DELAF foi produzida.

cesso de pré-processamento aqui utilizado buscou a compatibilidade com o Universal Dependencies, onde tais verbos e pronomes são separados, e grande parte das entradas do DELAF pôde ser ignorada. Na prática, é consultado um dicionário com 890 mil entradas.

Etiquetas morfossintáticas, ou *PoS tags*, produzidas pelo CoreNLP auxiliam a desambiguar palavras; por exemplo, o lema do substantivo *casas* é *casa*, mas do verbo homógrafo é *casar*. O código usado para a lematização foi disponibilizado abertamente em <<https://github.com/erickrf/unitex-lemmatizer>>.

Para buscar alinhamentos, as palavras da premissa são comparadas com as da hipótese, e quando ambas são iguais ou têm o mesmo lema, é considerado um alinhamento. Além disso, foram consultados a OpenWordNet-PT e o PPDB. No caso da primeira, duas palavras são consideradas alinhadas caso compartilhem algum synset. O código escrito para leitura da OpenWordNet-PT foi também disponibilizado em <<https://git.io/vxTdK>>.

O PPDB poderia ser usado de forma semelhante; no entanto, é um recurso que conta com algum ruído inerente à sua origem automatizada. Sua vantagem, e que compensa o ruído, é conter regras de paráfrase⁷ compostas de várias palavras, o que não é possível pela OpenWordNet-PT. Tais regras foram usadas para criar alinhamentos de expressões entre as duas sentenças. Para ler os dados do recurso, foi escrito código que cuida de evitar a inclusão de regras irrelevantes, como as que apresentam as mesmas palavras com mudanças em gênero ou número. O código foi disponibilizado em <<https://github.com/erickrf/ppdb>>. Durante o desenvolvimento do mesmo, foi encontrado um erro na codificação de caracteres dos arquivos em português do PPDB, que impossibilitaram a leitura de algumas regras. O problema foi comunicado aos autores.

Por fim, para a detecção de entidades nomeadas, foi usado o spaCy⁸, uma suíte de ferramentas para PLN que contém um módulo pré-treinado de reconhecimento de entidades nomeadas (REN). O spaCy reconhece entidades que constem de uma ou mais palavras, incluindo siglas.

Após a identificação das entidades das duas sentenças, são buscados alinhamentos entre elas. Um alinhamento acontece em dois casos: i) trivialmente, quando todas as palavras são iguais em uma entidade de *T* e em uma de *H* ou ii) quando as iniciais maiúsculas de uma entidade multipalavra em uma sentença, combinadas, são iguais a uma entidade monopalavra da outra, neste caso entendida como uma sigla.

⁷ Recorda-se novamente que o uso do termo *paráfrase* no contexto do PPDB não é o mesmo que do ASSIN, referindo-se apenas a palavras ou expressões equivalentes.

⁸ Disponível em <<http://spacy.io>>

6.2.2 Atributos

Nesta seção são listados os atributos extraídos pelo Infernal para representar cada par. Alguns atributos lançam mão de um conjunto de *stopwords*; nestes casos, foi usada a lista fornecida pela biblioteca NLTK⁹, acrescida de sinais de pontuação e das palavras *é*, *ser* e *estar*.

Outro ponto relevante a se observar é que alguns atributos consideram o sujeito de determinados verbos. Na anotação do Universal Dependencies, quando há um verbo com auxiliar (por exemplo, em *vão fazer* ou *começaram a estudar*), o token com relação de sujeito (*nsubj*) é conectado ao verbo principal da locução, e não ao auxiliar, com o qual deve concordar. Desta forma, tanto *os jogadores vão fazer* como *os jogadores farão* permitem a extração da relação de *jogadores* como sujeito do verbo *fazer*.

Durante análises dos dados para definição dos atributos, verificou-se que os alinhamentos produzidos pelo PPDB eram de pouca utilidade. Em sua maioria, são alinhamentos espúrios envolvendo verbos de ligação, artigos e preposições, que não foram filtrados pela etapa de leitura do recurso. Para reduzir a quantidade destes, foram considerados apenas os alinhamentos que incluíssem mais de uma palavra de *T* ou uma única palavra com classe gramatical dita de conteúdo (substantivo, verbo, adjetivo ou advérbio). Ainda assim, os alinhamentos espúrios predominam.

Foram experimentados dois tamanhos do PPDB: L e XXL, respectivamente o quarto e segundo maiores conjuntos de regras (dos quais existem seis). Os alinhamentos extraídos usando-se as duas versões do recurso são listados na Tabela 22.

Pode-se observar na tabela que a versão L cobre poucos alinhamentos dentre oito pares, todos espúrios. Já a versão XXL inclui alinhamentos para todos os pares, com alguns de fato informativos, como *atacar* → *ataques* ou *reunião* → *se reúne*. No entanto, a maioria dos seus alinhamentos ainda é espúria, o que introduz muito ruído para a representação. Em alguns casos extremos, há alinhamentos contraditórios, como *é* alinhado com *não é*.

Por outro lado, os alinhamentos de uma única palavra para outra podem ser mapeados de forma mais confiável pela OpenWordNet-PT. Tendo isto em vista, decidiu-se por não criar atributos baseados nos alinhamentos PPDB. A análise destas duas versões do recurso se mostrou suficiente para se descartar também o uso de outras: uma versão maior teria ainda mais ruído, enquanto versões menores não têm cobertura suficientes.

Em seguida, são listados os atributos que compõem a representação dos pares. Há 14 atributos conceituais, alguns dos quais codificados por mais de um valor numérico, totalizando 28 valores.

Para ilustrar sua extração, é considerado o par de sentenças mostrado na Figura 16,

⁹ Mais informações em <<https://www.nltk.org/>>

T	H	PPDB L	PPDB XXL
A gente faz o aporte financeiro, é como se a empresa fosse parceira do Monte Cristo.	Fernando Moraes afirma que não tem vínculo com o Monte Cristo além da parceira.	faz → tem é → {não, tem}	gente → que faz → {não, não tem} faz o → tem é → {não tem, além, que não, que, o, afirma} como se → {o, que, com} se a → {o, que} de como → a como → a
Em 2013, a história de como Walt Disney convenceu P.L.	P.L.Travers era completamente contra a adaptação de Walt Disney.	—	
Para os ambientalistas, as metas anunciadas pela presidente Dilma foram um avanço.	Dilma aproveitou seu discurso ontem, na Conferência de Desenvolvimento Sustentável pós-2015 para anunciar desde já essas metas.	—	Para os → {essas, para, seu} anunciadas → anunciar foram → já avanço → {de Desenvolvimento, desde já, Desenvolvimento}
De acordo com a PM, por volta das 10h30 havia 2 mil militantes no local.	O protesto encerrou por volta de 12h15 (horário local).	com a → o	De acordo com → por acordo com → por com a → por por volta → {volta de, volta} no local → O
O caminho de ajuste via aumento de carga tributária é muito mal visto pela sociedade e pelo Congresso.	O aumento da carga tributária também não é visto com bons olhos pelo congresso.	é → não é → não é é muito → é	caminho → O é → {também, também não, O} muito → {O, também, não, é, com} mal → não visto → {pelo, com} tem → {se, com}
Randolfe tem uma reunião com Marina Silva na tarde desta segunda (28), em Brasília.	Nesta segunda-feira, 28, Randolfe Rodrigues se reúne com Marina Silva.	—	tem uma → se reunião → se reúne
A principal novidade são os retornos do volante Thiago Maia e do lateral-direito Victor Ferraz, na equipe titular.	Além disso, o time terá os retornos de Victor Ferraz e Thiago Maia.	—	são os → {o, os} e → {o, Além}
Até então, França e Reino Unido limitavam-se a atacar alvos do 'Estado Islâmico' no Iraque.	A França já havia realizado ataques contra alvos do 'Estado Islâmico', mas apenas no Iraque.	—	então → {já, mas} Reino → A se a → a atacar → ataques

Tabela 22 – Alinhamentos encontrados segundo o PPDB. A metade superior mostra pares sem relação de implicação, e a inferior com a relação. Todos os alinhamentos da versão L também estão presentes na XXL, e portanto foram omitidos da quarta coluna. As diferentes expressões entre colchetes listam diferentes possibilidades de alinhamento.

Figura 16 – Árvore de dependências das sentenças usadas para ilustrar a extração de atributos do Infernal.

com os valores dos 28 atributos extraídos pelo Infernal na Tabela 23.

1. **BLEU** O BLEU (*BiLingual Evaluation Understudy*) é uma medida típica da área de tradução automática, já usada como atributo por alguns trabalhos de RIT. Grosso modo, calcula a média de sobreposição de n -gramas com $n = 1$ até 4 de um texto sobre outro. Este atributo contém dois valores: o quanto T contém n -gramas de H e vice-versa.
2. **Sobreposição de dependências** São contabilizadas as tuplas de dependência sobrepostas em T e H . Uma tupla de dependência é representada pelo seu rótulo, o pai e o filho; duas tuplas são consideradas sobrepostas quando têm rótulos iguais ou equivalentes e pai e filho iguais ou sinônimos. O único caso de rótulos equivalentes considerado no Infernal foram as relações *nsubjpass*, que indica o sujeito verbal na voz passiva, e *dobj*, que indica objeto direto. Esta equivalência permite abstrair a variação de voz ativa e passiva. No exemplo, uma tupla sobreposta é (*adpobj*, *por*, *volta*). Dois valores são usados como atributos: a proporção de tuplas de H que aparece em T e vice-versa.
3. **Nominalização** Este atributo verifica a presença de um verbo em uma sentença e de uma nominalização, ou seja, um substantivo derivado dele, como objeto direto na outra (indicado pela relação de dependência *dobj*). Um exemplo de nominalização é *correr* e *corrida*. A verificação de objeto direto torna este atributo mais específico; apenas a presença de nominalização já é considerada um alinhamento lexical e

influencia o valor de outros atributos. Dois valores são extraídos para este atributo, considerando a presença do substantivo nominalizado em T ou em H .

4. **Proporção de tamanho** Proporção entre a quantidade de tokens de T e H , excluindo *stopwords*.
5. **Argumentos verbais** Este atributo codifica em dois valores binários se há a correspondência de pelo menos um verbo com sujeito e objeto direto nas duas sentenças. Primeiro, verifica-se se há dois verbos e seus respectivos sujeitos alinhados. Em caso positivo, verificam-se os objetos diretos. Se forem diferentes, não há correspondência e o atributo tem o valor de $(0,0)$; se forem os mesmos há, e o valor é $(1,1)$. Caso haja objeto direto em apenas uma das sentenças, é considerada uma correspondência parcial, e o valor do atributo é $(1,0)$ se o objeto estiver somente em T , ou $(0,1)$ caso contrário. Este atributo foi inspirado no trabalho de [Sharma, Sharma e Biswas \(2015\)](#).
6. **Negação** Este atributo verifica se um verbo alinhado entre as duas sentenças aparece negado em uma delas. Um verbo é considerado negado quando tem um dependente sintático com a relação de negação, *neg* na anotação do UD.
7. **Quantidades** Indica se quantidades iguais ou diferentes são usadas nas duas sentenças para se referir a um alguma palavra alinhada. Para computar o atributo, são checadas todas as palavras alinhadas que tenham algum dependente sintático de quantidade, indicado pela relação *num*, e verificam-se tanto números em algarismos como escritos por extenso. Este atributo é útil no corpus ASSIN, que tem exigências mais restritas quanto a quantificação para considerar uma relação como implicação ou paráfrase. O atributo é codificado em dois valores: o primeiro é 1 caso sejam encontrados valores iguais, e 0 caso contrário; o segundo é 1 caso sejam encontrados valores diferentes, e 0 caso contrário. Note que ambos os valores podem ser 1 ou 0, caso mais de um quantificador haja (com um alinhado e outro não) ou nenhum, respectivamente.
8. **Cosseno das sentenças** São calculados vetores para as sentenças como a média dos vetores de embeddings de todas as suas palavras. Este atributo contém o cosseno dos vetores. A similaridade de embeddings têm forte poder preditivo da similaridade de duas sentenças, como mostrado por [Hartmann \(2016\)](#) no ASSIN.
9. **TED simples** Este atributo contém o valor da TED entre as duas sentenças, considerando todos os custos de inserção, remoção e substituição como 1. Dois nós são considerados iguais apenas quando têm o mesmo lema e relação de dependência. São usados três valores: a TED total, a mesma normalizada pelo tamanho de T (isto é, dividido pelo número de tokens) e pelo tamanho de H .

BLEUT→H 0.29	BLEU H→T 0.37	TED Cosseno 15.96	TED Cosseno / T 0.94
Cosseno TED / H 1.33	Sobrep. Dep. T 0.07	Sobrep. Dep. H 0.11	Nominal. em H 0.00
Nominal. em T 0.00	Proporção Tam. 1.50	Estr. Verbal T→H 0.00	Estr. Verbal H→T 0.00
Negação 0.00	Quant. Igual 0.00	Quant. Diferente 0.00	Cosseno 0.89
TED Simples 18.00	TED / T 1.06	TED / H 1.50	Sobrep Soft T→H 0.56
Sobrep. Soft H→T 0.61	Sobreposição T 0.22	Sobreposição H 0.33	Sobrep. Sinônimo T 0.22
Sobrep. Sinônimo H 0.33	Ent. Não Alinh. T 0.00	Ent. Não Alinh. H 0.00	Ent. Alinhada 0.00

Tabela 23 – Valores dos atributos do Infernal para o par da Figura 16

- 10. TED com distância de cosseno** Outra versão do atributo acima, usando a distância do cosseno entre as embeddings das duas palavras no caso de substituição (a distância do cosseno é igual a $1 - \cos(w1, w2)$). Apesar de ser uma extensão bastante simples da TED, não se conhece outro trabalho que a tenha usado.
- 11. Sobreposição de palavras** Contém a proporção de lemas em comum das duas sentenças em relação ao total de palavras de T e de H . Stopwords são desconsideradas.
- 12. Sobreposição de sinônimos** Outra versão do atributo acima, mas considerando quaisquer palavras alinhadas, não apenas as que tenham o mesmo lema.
- 13. Sobreposição *soft*** Terceira versão do atributo acima; esta considera não apenas a sobreposição de palavras, mas também sua similaridade, avaliado pelo cosseno de embeddings. Para cada palavra de uma sentença, exceto stopwords, é tomado o valor de maior similaridade com as da outra sentença. Em seguida, é calculada a média das similaridades. Dois valores são computados: um considerando as palavras de T e outro com as de H .
- 14. Entidades nomeadas** Este atributo codifica a presença de entidades nomeadas alinhadas ou desalinhadas nas duas sentenças em três valores: o primeiro indica a presença de uma entidade em T sem equivalente em H , o segundo indica o contrário, e o terceiro indica a presença de uma entidade alinhada no par. Todas as combinações destes valores são possíveis, dependendo da quantidade de entidades presentes no par.

6.2.3 Avaliação

O Infernal foi avaliado somente no ASSIN. Sua implementação também foi em Python, com os algoritmos de aprendizado de máquina do scikit-learn. Assim como na avaliação do TEDIN, foram combinados os dados das variantes brasileira e europeia do ASSIN nos subconjuntos de treino e validação. A implementação do Infernal está disponível em <https://github.com/erickrf/infernal>.

Antes de treinar os classificadores, foi feito o seguinte pré-processamento dos dados. Dada uma matriz $\mathbf{X} \in \mathbb{R}^{n \times d}$, em que n é o número de exemplos de treinamento e d é o número de atributos por exemplo, cada coluna $\mathbf{X}_{*,j}$ foi normalizada de modo a ter média 0 e variância 1. Os valores normalizados de cada vetor $\mathbf{x}$ são obtidos em função da sua média e desvio padrão:

$$\text{norm}(\mathbf{x}) = \frac{\mathbf{x} - \text{mean}(\mathbf{x})}{\text{std}(\mathbf{x})} \quad (6.13)$$

Os valores de média e desvio padrão dos dados de treino são guardados para serem usados *a posteriori* para a normalização de novos dados, quando da execução de um modelo treinado. Caso se usassem a média e desvio padrão dos dados que se desejam classificar, o resultado seria não-determinístico: diferentes partições de um conjunto de teste podem ter valores diferentes de média e desvio padrão, levando a normalizações diferentes e possivelmente resultados diferentes da classificação.

Avaliaram-se diferentes algoritmos de aprendizado de máquina: regressão logística, SVM, Naive Bayes, *random forest* e *gradient boost*. Para os hiperparâmetros não mencionados, foram usados valores padrão definidos no scikit-learn versão 0.18. Algumas particularidades que os concernem são interessantes destacar:

Naive Bayes Um dos algoritmos mais simples de aprendizado de máquina, que ignora possíveis interações entre os atributos. O Naive Bayes foi usado nos experimentos como um baseline para a performance com os atributos definidos no Infernal.

Regressão Logística Um algoritmo razoavelmente simples que calcula apenas uma separação linear dos dados, mas que pode obter muito bons resultados quando os mesmos têm alta dimensionalidade. A regressão logística (RL) tem pouca sensibilidade a hiperparâmetros, que controlam seus critérios de término de treinamento e regularização de pesos.

SVM Um algoritmo que também calcula uma separação linear dos dados, mas que busca maximizar a margem de distância entre os mesmos para aumentar a confiabilidade de suas predições, diminuindo o sobreajustamento. Além disso, comumente é usado em conjunto com funções kernel, que projetam os dados em distribuições mais fáceis de serem separadas linearmente. O SVM é bastante sensível a ajustes em seus

hiperparâmetros; para selecionar uma configuração, foram avaliadas todas as combinações do parâmetro de penalidade c (variando entre 0,01, 0,1, 1, 10 e 100), as funções kernel RBF (*Radial Based Function*), polinomial e linear, e o coeficiente de kernel γ (usado apenas com os dois primeiros kernels, e variando entre 0.001, 0.01, 0.1, 1 e 10) em validação cruzada no conjunto de treinamento. Ao final, a combinação com melhores resultados foi o kernel RBF com $c = 10$ e $\gamma = 0,01$.

Random Forest Um *ensemble* de árvores de decisão. É um algoritmo com alto poder preditivo e com poucos hiperparâmetros. Nos experimentos, foram selecionados os seguintes, também baseado na performance em validação cruzada: 500 árvores treinadas (também foi avaliado com 1000, com aumento do custo computacional sem ganho de performance), cada uma selecionando aleatoriamente seis atributos do modelo (valor aproximado da raiz quadrada do seu número total, recomendado para o algoritmo), e podendo se expandir ao máximo (foram avaliadas expansões máximas de dois a sete atributos).

Gradient Boost É um tipo diferente de *ensemble*, em que novos classificadores são sucessivamente treinados para corrigir os erros cometidos por outros. Normalmente, são treinadas árvores de decisão. Ao contrário do random forest, em que todas as árvores têm o mesmo peso na decisão final, no gradient boost o peso de cada nova árvore treinada é diminuído por uma constante chamada encolhimento ou taxa de aprendizado. Nos experimentos, foram avaliadas combinações de taxas de aprendizado (de 0,01 e 0,1), profundidade máxima das árvores (dois a seis) e número de árvores (500 e 1000). A configuração com melhor resultado teve taxa de aprendizado de 0,01, profundidade máxima de três e 500 árvores.

Com os classificadores de RL e SVM, é possível também definir pesos para as instâncias de classes raras, similarmente a como foi feito com os modelos neurais. Foi experimentado treinar instâncias que o fizessem.

Os resultados obtidos por todos os modelos, em termos de acurácia e de medida F_1 , são apresentados na Tabela 24. A Tabela também traz os resultados do estado-da-arte no ASSIN, e do baseline de sobreposição de palavras.

Pode-se ver que o Infernal supera o atual estado-da-arte no ASSIN com diferentes algoritmos de classificação. Os resultados indicam que os atributos definidos no Infernal modelam bem o problema, ainda mais ao se comparar seu número (28) com os 96 utilizados por Fialho *et al.* (2016).

Quanto aos diferentes algoritmos, exceto pelo SVM com balanceamento de classes e Naive Bayes, todos superam a medida F_1 do estado-da-arte. Considerando-se que diferentes algoritmos se destacaram em diferentes métricas e em diferentes subconjuntos de

Modelo	Validação		PT-BR		PT-PT		Geral	
	Acurácia	F ₁	Acurácia	F ₁	Acurácia	F ₁	Acurácia	F ₁
Naive Bayes	80,30%	0,68	79,05%	0,62	80,05%	0,68	79,55%	0,65
RL	85,50%	0,72	87,30%	0,71	85,75%	0,72	86,52%	0,72
RL, balanceado	85,20%	0,74	85,00%	0,69	84,60%	0,74	84,80%	0,72
Random Forest	85,20%	0,72	86,20%	0,67	86,20%	0,74	86,20%	0,71
Gradient Boost	85,80%	0,73	86,35%	0,67	86,10%	0,74	86,22%	0,71
SVM	85,60%	0,73	86,90%	0,70	85,75%	0,73	86,33%	0,72
SVM, balanceado	80,20%	0,69	79,20%	0,64	80,95%	0,71	80,08%	0,68
L2F/INESC-ID	—	—	85,85%	0,66	84,90%	0,71	—	—
Baseline	81,40%	0,69	82,80%	0,64	81,75%	0,7	82,27%	0,67

Tabela 24 – Performance do Infernal no ASSIN. A parte superior da tabela contém os resultados do Infernal com diferentes algoritmos de aprendizado de máquina. A parte inferior traz os resultados até então do estado-da-arte no ASSIN e do baseline de sobreposição de palavras.

teste, não é possível apontar qual seja objetivamente o melhor em termos de performance. No entanto, merecem destaque os modelos de regressão logística, por terem custo computacional baixo comparado aos outros (exceto Naive Bayes) e não serem muito sensíveis a ajuste de hiperparâmetros.

6.2.4 Relevância de Atributos

Uma análise importante para um modelo baseado em engenharia de atributos é entender quais deles foram mais relevantes para sua performance. No entanto, não há uma forma de análise de atributos que seja objetivamente a mais precisa para encontrar esta resposta.

Isto se deve à complexidade de problemas de classificação com vários atributos. Por exemplo, dependendo do problema, é possível que um classificador observe apenas um atributo x_1 e obtenha a mesma performance de outro classificador que observe os atributos x_2 e x_3 . Embora se possa determinar quais atributos foram importantes para quais classificadores, o mesmo não ocorre para a capacidade representativa dos atributos para o problema em geral.

Ainda assim, há meios de se estimar a importância dos atributos. Em particular, os algoritmos random forest e gradient boost podem calcular valores de importância baseados em quão bem a observação dos atributos permitiu separar as classes do problema durante seu treinamento. Por se tratarem de algoritmos que usam ensembles de classificadores mais simples, têm ainda o atrativo conceitual de computarem uma média de importância em várias iterações.

Para se estimar a importância dos atributos do Infernal, foram usados tanto os valores de importâncias do random forest (RF) como do gradient boost (GB). Além disso, foram treinadas 10 instâncias de cada modelo, inicializadas com sementes aleatórias

#	Atributo	#GB	#RF	%GB	%RF
1	Sobreposição Soft H→T	1	1	12.68%	14.06%
2	Sobreposição / Tamanho H	2	2	11.30%	13.74%
3	Sobreposição Sinônimos / Tamanho H	8	3	4.85%	8.73%
4	Sobreposição Soft T→H	3	7	7.56%	4.93%
5	Cosseno	5	5	6.64%	5.36%
6	Sobreposição / Tamanho T	4	9	6.85%	4.89%
7	Proporção Tamanho	7	6	6.01%	5.13%
8	TED Cosseno / Tamanho H	6	11	6.63%	3.92%
9	TED / Tamanho T	11	8	3.50%	4.90%
10	Sobreposição Dependências H	14	4	2.52%	5.51%
11	TED Cosseno / Tamanho T	12	10	3.06%	4.13%
12	Sobreposição Sinônimos / Tamanho T	10	13	3.75%	3.14%
13	Sobreposição Dependências / Tamanho T	13	15	2.81%	3.06%
14	BLEU H→T	15	16	2.32%	3.02%
15	TED / Tamanho H	21	12	1.94%	3.23%
16	BLEU T→H	17	14	2.08%	3.07%
17	TED Cosseno	19	17	1.97%	2.95%
18	Quantidade Diferente	9	19	3.97%	0.84%
19	TED Simples	16	18	2.25%	1.93%
20	Quantidade Igual	18	20	2.07%	0.71%
21	Entidade Não Alinhada H	20	25	1.96%	0.38%
22	Entidade Não Alinhada T	22	24	1.16%	0.44%
23	Estrutura Verbal T→H	27	21	0.25%	0.52%
24	Estrutura Verbal H→T	26	22	0.25%	0.52%
25	Nominalização em T	23	28	0.63%	0.11%
26	Verbo Negado	24	26	0.51%	0.14%
27	Entidade Alinhada	28	23	0.09%	0.49%
28	Nominalização em H	25	27	0.39%	0.13%

Tabela 25 – Importância dos atributos do Infernal. A terceira e a quarta colunas mostram, respectivamente, a posição de importância do atributo para o GB e RF (valores menores significam maior importância). A quinta e a sexta mostram a porcentagem de importância do atributo para os algoritmos. Os atributos estão ordenados pelo valor médio das duas últimas colunas.

diversas, ao fim das quais as médias dos valores de cada um foram tomadas. Desta forma, efeitos de aleatoriedade são mitigados ainda mais.

A Tabela 25 mostra os valores de importância de cada atributo conforme computados pelo GB e RF. Os atributos estão ordenados pela importância média que têm para os dois algoritmos.

Há algumas diferenças na ordenação para os dois algoritmos, mas a maioria dos atributos tem importância relativa bastante próxima para ambos. Em especial, atributos relacionados à sobreposição de palavras têm grande importância — o que já era evidenciado pelo bom desempenho do baseline do ASSIN.

O cosseno entre as duas sentenças, ainda que calculado sobre vetores obtidos de forma bastante simplificada (o vetor de cada sentença é apenas a média das embeddings de suas palavras), também tem grande poder preditivo. Este atributo agrega a similaridade das palavras usadas em cada sentença, podendo ser visto como complementar aos que medem a sobreposição.

Os atributos que envolvem TED apresentam contribuição mediana para ambos os algoritmos. Esta observação é coerente com a performance observada do TEDIN e AdArte: a TED pode de fato ser útil para RIT, mesmo no ASSIN, mas com importância limitada e bem abaixo do alinhamento de palavras das sentenças.

Dentre os atributos que calculam a proporção de alguma métrica (como sobreposição) e o tamanho de cada sentença, as versões que consideram o tamanho de H tendem a ter maior importância. Isto se deve ao fato de que a proporção em relação a T é relevante apenas para se determinar paráfrases, que acontecem em menor quantidade, enquanto a relação com o tamanho de H é importante tanto para paráfrases quanto implicação.

Em sétimo lugar, está a proporção de tamanho entre T e H . Apesar de a princípio a implicação ser indiferente ao tamanho de cada sentença envolvida, normalmente sentenças com mais palavras (e, em geral, mais informação) implicam sentenças com menos palavras.

Entre os atributos menos relevantes, estão os que dizem respeito a entidades nomeadas, negações e nominalizações. Isto se explica por sua relativamente baixa abrangência — muitos pares, das três classes, não apresentam nenhum destes fenômenos. Ainda assim, esperava-se que sua relevância para o problema fosse maior.

Experimentou-se ainda retrainar modelos do Infernal sem alguns dos atributos de menor importância, mas não se verificaram melhoras de performance. Em vez disso, houve queda de até 0,02 de F_1 . Isto reforça a tese de que os atributos definidos modelam bem o problema de RIT, ao menos nos moldes do corpus ASSIN.

6.2.5 Análise de Erros

Para entender as dificuldades de classificação do Infernal, foram analisados manualmente 65 pares erroneamente classificados no conjunto de validação, listando-se os fenômenos que causaram erros. A Tabela 26 lista as contagens, e a seguir as categorias são explicadas e exemplificadas.

Muita sobreposição de palavras é a principal causa para o modelo apontar como implicação casos neutros, ou como paráfrase casos neutros ou de implicação mono-direcional.

Exemplo (rótulo correto neutro, resposta do Infernal implicação):

Fenômeno	Ocorrências
Muita sobreposição de palavras	23
Reformulação	21
Sinônimos contextuais	19
Quantidades	5
Entidade nomeada qualificada	4

Tabela 26 – Dificuldades para o Infernal no ASSIN

- a. A presidente Dilma Rousseff empossa, nesta segunda-feira (5), os novos ministros, em cerimônia no Palácio do Planalto.
- b. Dez ministros tomaram posse nesta segunda-feira (5) numa cerimônia no Palácio do Planalto.

No exemplo há a menção de dez ministros e o fato narrado está no passado.

Reformulação ocorre quando o mesmo conteúdo é expresso de forma diferente, sem usar as mesmas palavras ou mesmos sinônimos, dificultando a sua captura pelos atributos do Infernal. Esta categoria também inclui casos em que informação fica implícita.

Exemplo (rótulo correto implicação, resposta do Infernal neutro):

- a. Os trabalhadores protestam contra a regulamentação da terceirização, a retirada de direitos trabalhistas e o ajuste fiscal.
- b. Os trabalhadores protestam contra o projeto de lei que regulamenta a terceirização no país.

No exemplo, *projeto de lei* pode ser subentendido de *regulamentação*. A própria palavra *regulamentação* é uma nominalização do verbo *regulamenta*; mas como se viu na Seção 6.2.4, o modelo não aprendeu a dar muito peso para nominalizações. Além disso, *no país* não tem correspondente na primeira sentença, mas fica implícito pelo contexto.

Sinônimos contextuais são casos em que duas palavras são efetivamente usadas como sinônimos, mas em que a relação de sinonímia só existe em contextos muito específicos, de modo a não figurar em um recurso como a WordNet. Esta categoria pode incluir metáforas, mas não apenas.

Exemplo (rótulo correto paráfrase, resposta do Infernal neutro):

- a. Os demais agentes públicos serão alocados na classe econômica.

- b. Todo o resto dos funcionários públicos terá que embarcar na classe econômica.

ser alocado e *embarcar* podem ser considerados sinônimos aqui, mas apenas pelo contexto. Além disso, há a dificuldade de se tratar a equivalência entre *os demais* e *todo o resto*.

Quantidades dizem respeito a casos em que é preciso entender alguma relação envolvendo quantidades, como identificar valores maiores ou menores, ou identificar como equivalentes valores não exatamente iguais, mas que sejam explicitamente ditos aproximados.

Exemplo (rótulo correto implicação, resposta do Infernal neutro):

- a. De acordo com a polícia, 56 agentes e 12 manifestantes ficaram feridos.
- b. Pelo menos 46 policiais e sete manifestantes ficaram feridos.

Para o entendimento da relação de implicação, é necessário entender que a expressão *pelo menos* admite que as quantidades referidas em *T* sejam maiores que as de *H*.

Entidade nomeada qualificada ocorre quando uma determinada entidade nomeada é referida em uma das sentenças por uma descrição mais ampla de sua natureza, que fica subentendida na outra.

Exemplo (rótulo correto implicação, resposta do Infernal neutro):

- a. Tite, no segundo tempo, trocou Ralf por Mendoza.
- b. O atacante Mendoza entrou no lugar do volante Ralf.

O termo *atacante* é uma descrição a mais de *Mendoza*. Note-se que há ainda uma reformulação de ideias neste par, com *trocou* implicando em *entrou no lugar*.

Os problemas apontados se mostram bastante delicados para receberem soluções abrangentes. A quantificação é possivelmente o mais viável de se tratar, observando-se certas expressões que indiquem quantidades aproximadas e relações, mas ainda assim poderia ter diversos casos particulares. Recursos de PLN que listassem expressões equivalente, como o PPDB, seriam de alguma ajuda para lidar com reformulações.

Um corpus maior de RIT possibilitaria o treinamento de modelos neurais com melhores resultados, possivelmente capazes de modelar certas sutilezas de dados como os sinônimos contextuais. No entanto, por ora pode-se apenas especular a respeito, já que os modelos neurais mais bem sucedidos na literatura são treinados no SNLI, um corpus

relativamente simples. Com a popularização da tarefa de RIT e surgimento recente de alguns novos corpora (vide Seção 2.5.5), há uma tendência de que esta situação mude.

6.3 Considerações Finais

Neste capítulo foram apresentados dois modelos propostos para RIT: o TEDIN e o Infernal. O primeiro contou com a motivação teórica de tratar operações de edição de TED de forma mais flexível, aprendendo automaticamente seus custos, e fornecendo representações mais ricas para um classificador de RIT. Neste sentido, o TEDIN apresentou um progresso em relação às soluções usadas por outros sistemas de RIT baseados em transformações textuais (vide Tabela 7). Estes usam custos de edição fixos como o AdArte, ou aprendem custos diferentes para poucas categorias. Já na representação dos pares para o classificador, usam ou um simples limiar da TED ou vetores esparsos que codificam todas as possíveis operações — que pode chegar a centenas de dimensões.

Apesar de sua motivação teórica, o TEDIN teve resultados negativos quando testado em benchmarks da literatura, não superando nem mesmo o AdArte, modelo do qual se pretendia uma melhoria direta. A análise do TED calculado pelo modelo indica que aprender custos coerentes para operações de edição é bastante difícil, ao menos em um cenário de RIT, visto que o modelo parece se sobreadaptar a certas idiossincrasias dos cenários de treino.

Já o Infernal, modelo de extração de atributos, ainda que com pouca inovação teórica, obteve excelentes resultados no ASSIN, estabelecendo um novo estado-da-arte para o RIT em português. Foram analisadas as contribuições dos seus atributos, e mostrou-se que aqueles que indicam sobreposição de palavras ou similaridade segundo modelos de embeddings são os mais úteis para a tarefa. Atributos baseados em TED também contribuem, ainda que de forma mais limitada.

Verificou-se que a maior dificuldade do modelo foi em detectar implicações quando o conteúdo de uma sentença é rephraseado sem usar as mesmas palavras ou sinônimos, ou com informações implícitas. Esta era uma dificuldade que já se previa, vistas as análises apresentadas na Seção 2.3. Buscou-se usar o PPDB para contorná-la, já que o recurso conta com paráfrases de diversas expressões. No entanto, o mesmo se mostrou infrutífero, pois a maioria de seu conteúdo é de paráfrases ruidosas, e por isso não foi aproveitado para o Infernal.

Os resultados positivos do Infernal, bem como as causas de seus erros, indicam que já se tem uma modelagem bastante eficaz para os fenômenos mais superficiais relacionados ao RIT: palavras e conceitos em comum, casos básicos de quantificação e variações sintáticas.

Para se lidar com fenômenos mais complexos, uma solução deve envolver a criação de recursos lexicais mais abrangentes e precisos que o PPDB, e/ou a criação de corpora maiores para RIT em português. A capacidade de modelos neurais de aprender sem a modelagem explícita de atributos não pôde ser bem explorada no ASSIN, e permanece a questão de se seriam capazes de tratar melhor os fenômenos linguísticos que foram obstáculos para o Infernal.

CONCLUSÕES

Este capítulo retoma os objetivos desta pesquisa, apresentando uma visão geral sobre o trabalho desenvolvido e também as contribuições, limitações e os trabalhos futuros.

7.1 Revisão da Literatura

Como parte essencial da pesquisa, a revisão da literatura se faz necessária para se conhecer como diferentes pesquisadores têm tratado o problema em questão, e pontos fortes e fracos de diferentes abordagens. Nesta tese, foram apresentados e discutidos tanto os corpora compilados para a tarefa de RIT como as abordagens computacionais desenvolvidas para tratá-la.

Para melhor contextualização, foram também apresentados recursos de PLN importantes para as abordagens aqui chamadas clássicas, e conceitos de redes neurais comuns nas abordagens de deep learning. Desta forma, a revisão da literatura que aparece nos Capítulos 2, 3 e 4 fornece um retrato bastante amplo da atual pesquisa em RIT, e pode ser útil para pesquisadores que queiram conhecer a área.

7.2 Criação de Recursos para RIT em Português

Um dos objetivos de pesquisa era a criação de recursos para o RIT em português, em especial um corpus anotado para a tarefa. Este objetivo foi cumprido com sucesso com a compilação do corpus ASSIN, divulgado em uma avaliação conjunta e que atraiu o interesse de diversos pesquisadores da área. Além do recurso em si, criou-se uma base comum para avaliação de abordagens de RIT.

7.3 Abordagens Computacionais para RIT

Outro objetivo da pesquisa, explorar estruturas sintáticas para o RIT, também foi alcançado. Desenvolveu-se o modelo TEDIN, baseado em distância de edição de árvores sintáticas; em menor grau, também o modelo Infernal se aproveitou da análise sintática para modelar o problema. Ambos os modelos foram avaliados no corpus ASSIN, e o TEDIN adicionalmente o foi no SICK e no SNLI, outros recursos de referência da literatura sobre RIT em inglês.

7.3.1 TEDIN

Relacionada ao TEDIN, havia a hipótese de que sua maior flexibilidade e poder representacional em relação a outros modelos da literatura baseados em TED levariam a melhores resultados no RIT. No entanto, esta hipótese não se confirmou — em comparação com um modelo simples da literatura que se visava melhorar diretamente, o AdArte, o TEDIN teve resultados próximos ou inferiores em diferentes conjuntos de dados.

A análise do funcionamento do TEDIN auxilia a explicar sua baixa performance para RIT. Os custos de operações de edição aprendidos muitas vezes não são coerentes com seu impacto para a tarefa; no entanto, se adequaram razoavelmente bem ao objetivo de discriminar pares positivos (com relação de implicação) de negativos (sem a relação), especialmente no corpus SNLI. Além disso, a análise do modelo Infernal revelou que a TED tem capacidade mediana de predizer a presença de implicação.

7.3.2 Infernal

A abordagem do Infernal foi mais bem sucedida que o TEDIN. Este modelo se valeu da definição de diversos atributos para resolver o problema de RIT, alguns dos quais considerando relações sintáticas das sentenças envolvidas. Diferentes algoritmos de classificação treinados com os atributos do Infernal alcançaram resultados de estado-da-arte no ASSIN.

Foi também feita uma análise quantitativa da contribuição de cada atributo definido no Infernal, verificando-se, como esperado, a maior importância de atributos relacionados a sobreposição de palavras e sinônimos; importância mediana da sintaxe, e razoavelmente baixa de atributos relacionados a negação, nominalização e entidades nomeadas.

7.3.3 Outros Modelos Neurais

Modelos neurais para RIT conhecidos na literatura também foram explorados no ASSIN. Em particular, foram reportados os resultados da arquitetura modular de (PA-

RIKH *et al.*, 2016), que atingiu resultados abaixo do baseline. Isto indica que o corpus apresenta dificuldades ausentes do SNLI, onde o mesmo modelo alcança resultados próximos do estado da arte.

7.4 Trabalhos Futuros

7.4.1 *Corpora para RIT*

A pesquisa em RIT ainda tem algumas lacunas a serem superadas. Para a língua portuguesa, foco da pesquisa desta tese, um dos primeiros pontos é a disponibilidade de dados em maiores quantidades. A compilação do ASSIN foi um importante primeiro passo, mas ainda limitado para o treinamento de modelos mais poderosos por seu tamanho e desbalanceamento de classes.

Outro ponto a ser considerado é a complexidade das sentenças envolvidas e da dificuldade de se realizar a tarefa. No ASSIN, cujos pares vêm de textos jornalísticos, figuram frequentemente sentenças com mais de uma oração, entidades nomeadas, informação implícita, entre outros fenômenos. Em comparação, o SNLI, que foi o principal recurso para treinamento de sistemas de RIT em inglês nos últimos dois anos, contém sentenças mais simples.

Enquanto o SNLI foi desenvolvido com o propósito de possibilitar o treinamento de modelos computacionais capazes de elaborar representações para estruturas linguísticas simples, sem incluir a dificuldade de certos fenômenos, o ASSIN visou, mais pragmaticamente, avaliar o RIT em textos reais. Com a recente publicação de novos corpora para RIT em inglês com textos mais complexos, será possível analisar como as atuais técnicas do estado-da-arte funcionam nos mesmos. Com efeito, há de se observar se o treinamento em conjunto em corpora mais simples (como o SNLI) e mais complexos (como o SciTail) leva a melhores resultados em pares do segundo tipo. Em caso positivo, seria um estímulo para o desenvolvimento de um corpus de pares de sentenças mais simples que o ASSIN para português.

7.4.2 *Dificuldades para RIT*

Quanto a técnicas de RIT baseadas em métodos clássicos, como é o caso do Infernal, concluiu-se que as principais dificuldades na tarefa dizem respeito a informações implícitas, rephraseamentos, descrições de entidades nomeadas e quantidades. Não parece haver uma solução simples para qualquer destes problemas; no entanto, sua investigação é importante para a área.

Como sugerido ao final do Capítulo 6, recursos com grandes volumes de dados seriam fundamentais, seja na forma de mais pares de RIT ou como coleções de expres-

sões equivalentes. Em conjunto com a disponibilidade de recursos, módulos para o reconhecimento mais preciso de entidades nomeadas descritas de formas diferentes ou de quantidades aproximadas também seriam úteis para sistemas de RIT.

Outra fonte de conhecimento para auxiliar o treinamento de sistemas de RIT seria a elaboração de algum tipo de anotação extra sobre um corpus já anotado para a tarefa que explicasse o motivo dos casos de inferência. Não se tem aqui uma ideia exata de como seria tal anotação, mas a indicação de que determinadas expressões são equivalentes ou que determinado trecho é subentendido seria útil para a tarefa.

7.4.3 TEDIN

O TEDIN poderia ser treinado de formas diferentes das exploradas neste trabalho. Uma versão mais simples seria desconsiderar o cálculo da distância de edição executado pelo seu primeiro módulo. Em vez disso, a TED poderia ser calculada de forma mais simples (e menos flexível) tornando todos os custos de operação iguais a 1, ou à distância do cosseno das palavras envolvidas, no caso de substituição. Esta alteração vai de encontro à motivação do TEDIN, que era de possibilitar um cálculo melhor da TED, mas poderia render melhor resultados em RIT ou outras aplicações.

Outra possibilidade é alterar apenas a função de perda, mas mantendo a arquitetura de duas etapas. É possível trocar a margem 1 usada na Equação 6.2 por um valor mais alto; mais genericamente, tratá-la como um hiperparâmetro do algoritmo em vez de uma constante.

7.5 Produção Técnica

Durante a pesquisa conduzida neste doutorado, foi desenvolvido código-fonte para diversas atividades relacionadas ao RIT. Recapitulam-se aqui as contribuições divulgadas:

- Infernal e reimplementação do AdArte (ZANOLI; COLOMBO, 2016) em Python. Disponível em <<https://github.com/erickrf/infernal>>
- TEDIN. Disponível em <<https://github.com/erickrf/tedin>>
- Baseline e scripts de avaliação para o ASSIN. Disponível em <<https://github.com/erickrf/assin>>
- Implementação das arquiteturas neurais de Parikh *et al.* (2016) e Chen *et al.* (2017a). Disponível em <<https://github.com/erickrf/multiffn-nli>>
- Implementação de um *autoencoder* para textos. Disponibilizado em <<https://github.com/erickrf/autoencoder>>

- Funções para facilitar o acesso ao PPDB. Disponível em <<https://github.com/erickrf/ppdb>>
- Funções para facilitar o acesso à OpenWordNet-PT. Disponível em <<https://git.io/vxTdK>>
- Lematizador baseado no dicionário DELAF. Disponível em <<https://github.com/erickrf/unitex-lemmatizer>>
- Buscador de pares candidatos para a geração de um corpus de RIT. Disponível em <<https://github.com/erickrf/rte-bootstrapper>>
- Sistema web para anotação de pares para RIT e similaridade semântica, usado para o desenvolvimento do ASSIN. Disponível em <<https://bitbucket.org/erickrf/rte-annotator>>
- Desenvolvimento de nova funcionalidade (listar todas as operações de edição envolvidas no cálculo de TED) em uma implementação do algoritmo Zhang-Shasha. Contribuído para o código em <<https://github.com/timtadh/zhang-shasha>>

7.6 Produção Acadêmica

No período do doutorado, foi publicado um total de nove artigos em revistas e conferências. Alguns deles não são diretamente ligados ao tema de pesquisa da tese, mas à área de PLN em geral.

- Criscuolo, M., Fonseca, E.R., Aluísio, S.M., Sperança-Criscuolo, A.C. *MilkQA: a Dataset of Consumer Questions for the Task of Answer Selection*. In: Proceedings of the 2017 Brazilian Conference on Intelligent Systems. Piscataway: IEEE, 2017. v. 1. p. 354–359.
- Hartmann, N.S., Fonseca, E.R., Shulby, C., Treviso, M., Rodrigues, J.S., Aluísio, S.M. *Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks*. In: Proceedings of the XI Brazilian Symposium in Information and Human Language Technology. Porto Alegre: Sociedade Brasileira de Computação, 2017. v. 1. p. 122–131.
- Fonseca, E.R., Magnolini, S., Feltracco, A., Qwaider, M.R.H., Magnini, B. *Tweaking Word Embeddings for FAQ Ranking*. In: Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, 2016.

- Fonseca, E., Aluísio, S.M. *Improving POS Tagging Across Portuguese Variants with Word Embeddings*. In: 12th International Conference on the Computational Processing of Portuguese, 2016, Tomar. Proceedings of the International Conference on the Computational Processing of Portuguese, 2016. v. 1. p. 227–232.
- Mendonça, G., Avanço, L., Duran, M., Fonseca, E.R., Nunes, M. G. V., Aluísio, S.M. *Evaluating Phonetic Spellers for User-generated Content in Brazilian Portuguese*. In: 12th International Conference on the Computational Processing of Portuguese, 2016, Tomar. Proceedings of the 12th International Conference on the Computational Processing of Portuguese, 2016. v. 1. p. 361–373.
- Fonseca, E.R., Santos, L.B., Criscuolo, M., Aluísio, S.M. *Visão Geral da Avaliação de Similaridade Semântica e Inferência Textual*. Linguamática (Braga), v. 8, p. 3–13, 2016.
- Fonseca, E.R., Rosa, J.L.G., Aluísio, S.M. *Evaluating word embeddings and a revised corpus for part-of-speech tagging in Portuguese*. Journal of The Brazilian Computer Society (Online), v. 21:2, 2015.
- Fonseca, E.R., Aluísio, S.M. *Semi-Automatic Construction of a Textual Entailment Dataset: Selecting Candidates with Vector Space Models*. In: X Brazilian Symposium in Information and Human Language Technology, 2015, Natal. Proceedings of the X Brazilian Symposium in Information and Human Language Technology, 2015. v. 1. p. 201–210.
- Fonseca, E.R., Aluísio, S.M. *A Deep Architecture for Non-Projective Dependency Parsing*. In: Workshop on Vector Space Modeling for NLP, co-located with NAACL 2015, 2015, Denver. Proceedings of the NAACL-HLT 2015, Workshop on Vector Space Modeling for NLP 2015, 2015. v. 1. p. 56–61.

REFERÊNCIAS

ABACHA, A. B.; DINA, D.-F. Recognizing Question Entailment for Medical Question Answering. In: **AMIA Annual Symposium Proceedings**. [S.l.: s.n.], 2016. p. 310–318. Citado na página 23.

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIÉGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <<https://www.tensorflow.org/>>. Citado nas páginas 81 e 121.

AGIRRE, E.; BANEJA, C.; CARDIE, C.; CER, D.; DIAB, M.; GONZALEZ-AGIRRE, A.; GUO, W.; LOPEZ-GAZPIO, I. n.; MARITXALAR, M.; MIHALCEA, R.; RIGAU, G.; URIA, L.; WIEBE, J. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In: **Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)**. [S.l.: s.n.], 2015. p. 252–263. Citado na página 103.

AL-RFOU, R.; ALAIN, G.; ALMAHAIRI, A.; ANGERMUELLER, C.; BAHDANAU, D.; BALLAS, N.; BASTIEN, F.; BAYER, J.; BELIKOV, A.; BELOPOLSKY, A.; BENGIO, Y.; BERGERON, A.; BERGSTRA, J.; BISSON, V.; Blecher Snyder, J.; BOUCHARD, N.; BOULANGER-LEWANDOWSKI, N.; BOUTHILLIER, X.; BRÉBISSE, A. de; BREULEUX, O.; CARRIER, P.-L.; CHO, K.; CHOROWSKI, J.; CHRISTIANO, P.; COOIJMANS, T.; CÔTÉ, M.-A.; CÔTÉ, M.; COURVILLE, A.; DAUPHIN, Y. N.; DELALLEAU, O.; DEMOUTH, J.; DESJARDINS, G.; DIELEMAN, S.; DINH, L.; DUCOFFE, M.; DUMOULIN, V.; Ebrahimi Kahou, S.; ERHAN, D.; FAN, Z.; FIRAT, O.; GERMAIN, M.; GLOT, X.; GOODFELLOW, I.; GRAHAM, M.; GULCEHRE, C.; HAMEL, P.; HARLOUCHET, I.; HENG, J.-P.; HIDASI, B.; HONARI, S.; JAIN, A.; JEAN, S.; JIA, K.; KOROBOK, M.; KULKARNI, V.; LAMB, A.; LAMBLIN, P.; LARSEN, E.; LAURENT, C.; LEE, S.; LEFRANCOIS, S.; LEMIEUX, S.; LÉONARD, N.; LIN, Z.; LIVEZEY, J. A.; LORENZ, C.; LOWIN, J.; MA, Q.; MANZAGOL, P.-A.; MASTROPIETRO, O.; MCGIBBON, R. T.; MEMISEVIC, R.; MERRIËNBOER, B. van; MICHALSKI, V.; MIRZA, M.; ORLANDI, A.; PAL, C.; PASCANU, R.; PEZESHKI, M.; RAFFEL, C.; RENSHAW, D.; ROCKLIN, M.; ROMERO, A.; ROTH, M.; SADOWSKI, P.; SALVATIER, J.; SAVARD, F.; SCHLÜTER, J.; SCHULMAN, J.; SCHWARTZ, G.; SERBAN, I. V.; SERDYUK, D.; SHABANIAN, S.; SIMON, E.; SPIECKERMANN, S.; SUBRAMANYAM, S. R.; SYGNOWSKI, J.; TANGUAY, J.; TULDER, G. van; TURIAN, J.; URBAN, S.; VINCENT, P.; VISIN, F.; VRIES, H. de; WARDE-FARLEY, D.; WEBB, D. J.; WILLSON, M.; XU, K.; XUE, L.; YAO, L.; ZHANG, S.; ZHANG, Y. Theano: A

- Python framework for fast computation of mathematical expressions. **arXiv e-prints**, abs/1605.02688, maio 2016. Disponível em: <<http://arxiv.org/abs/1605.02688>>. Citado na página 81.
- ALABBAS, M.; RAMSAY, A. Optimising Tree Edit Distance with Subtrees for Textual Entailment. In: **Proceedings of Recent Advances in Natural Language Processing**. [S.l.: s.n.], 2013. p. 9–17. Citado na página 73.
- ALVES, A. O.; GONÇALO OLIVEIRA, H.; RODRIGUES, R. ASAPP e Reciclagem no ASSIN: Alinhamento Semântico Automático de Palavras aplicado ao Português. **Linguamática**, 2016. Citado na página 65.
- ANDROUTSOPOULOS, I.; MALAKASIOTIS, P. A Survey of Paraphrasing and Textual Entailment Methods. **Journal of Artificial Intelligence Research**, v. 38, n. 1, 2010. Citado nas páginas 31 e 51.
- BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural Machine Translation by Jointly Learning to Align and Translate. **CoRR**, abs/1409.0473, 2014. Disponível em: <<http://arxiv.org/abs/1409.0473>>. Citado na página 85.
- BAR-HAIM, R.; BERANT, J.; DAGAN, I.; GREENTAL, I.; MIRKIN, S.; SHNARCH, E.; SZPEKTOR, I. Efficient smantic deduction and approximate matching over compact parse forests. In: **Proceedings of the Text Analysis Conference 2009**. [S.l.: s.n.], 2009. Citado nas páginas 35 e 69.
- BARBOSA, L.; CAVALIN, P.; MARTINS, B.; GUIMARÃES, V.; KORMAKSSON, M. Blue Man Group no ASSIN: Usando Representações Distribuídas para Similaridade Semântica e Inferência Textual. **Linguamática**, 2016. Citado nas páginas 66 e 114.
- BARRÓN-CEDENO, A.; VILA, M.; MART, M. A.; ; ROSSO, P. Plagiarism meets paraphrasing: Insights for the next generation in automatic plagiarism detection. **Computational Linguistics**, v. 4, p. 917–947, 2013. Citado na página 32.
- BAUDAT, G.; ANOUAR, F. Kernel-based methods and function approximation. In: **Proceedings of the International Joint Conference on Neural Networks**. [S.l.: s.n.], 2001. Citado na página 53.
- BEAM, A. L. **You can probably use deep learning even if your data isn't that big**. 2017. Disponível em: <http://beamandrew.github.io/deeplearning/2017/06/04/deep_learning_works.html>. Citado na página 81.
- BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on Neural Networks**, v. 5, p. 157–166, 1994. Citado na página 84.
- BENTIVOGLI, L.; BERNARDI, R.; MARELLI, M.; MENINI, S.; BARONI, M.; ZAMPARELLI, R. SICK Through the SemEval Glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. **Journal of Language Resources and Evaluation**, v. 50, p. 95–124, 2016. Citado na página 43.
- BENTIVOGLI, L.; CLARK, P.; DAGAN, I.; DANG, H.; GIAMPICCOLO, D. The Seventh PASCAL Recognizing Textual Entailment Challenge. In: **Proceedings of the Text Analysis Conference 2011**. [S.l.: s.n.], 2011. Citado nas páginas 25, 30 e 41.

- BENTIVOGLI, L.; DAGAN, I.; DANG, H. T.; GIAMPICCOLO, D.; MAGNINI, B. The Fifth PASCAL Recognizing Textual Entailment Challenge. In: **Proceedings of TAC 2009**. [S.l.: s.n.], 2009. Citado na página 40.
- BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. **Journal of Machine Learning Research**, v. 3, p. 993–1022, 2003. ISSN 1532-4435. Disponível em: <<http://dl.acm.org/citation.cfm?id=944919.944937>>. Citado na página 104.
- BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. Enriching word vectors with subword information. **ArXiv e-prints**, 2016. Citado nas páginas 78 e 80.
- BOS, J.; ZANZOTTO, F. M.; PENNACCHIOTTI, M. Textual Entailment at EVALITA 2009. In: **Proceedings of EVALITA 2009**. [S.l.: s.n.], 2009. Citado na página 26.
- BOWMAN, S. R.; ANGELI, G.; POTTS, C.; MANNING, C. D. A large annotated corpus for learning natural language inference. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. [S.l.: s.n.], 2015. Citado nas páginas 23, 24, 25, 31, 44, 87 e 104.
- CHEN, Q.; ZHU, X.; LING, Z.-H.; WEI, S.; JIANG, H.; INKPEN, D. Enhanced LSTM for Natural Language Inference. In: **Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Association for Computational Linguistics, 2017. p. 1657–1668. Disponível em: <<http://aclanthology.coli.uni-saarland.de/pdf/P/P17/P17-1152.pdf>>. Citado nas páginas 45, 95, 96, 97 e 148.
- CHEN, Q.; ZHU, X.; LING, Z.-H.; INKPEN, D.; WEI, S. Natural Language Inference with External Knowledge. **ArXiv e-prints**, nov. 2017. Citado nas páginas 98 e 124.
- CHEN, Q.; ZHU, X.; LING, Z.-H.; WEI, S.; JIANG, H.; INKPEN, D. **Recurrent Neural Network-Based Sentence Encoder with Gated Attention for Natural Language Inference**. Association for Computational Linguistics, 2017. 36–40 p. Disponível em: <<http://aclweb.org/anthology/W17-5307>>. Citado na página 91.
- CHIERCHIA, G.; MCCONNELL-GINET, S. **Meaning and Grammar: An Introduction to Semantics**. [S.l.]: MIT Press, 2000. Citado na página 30.
- CHKLOVSKI, T.; PANTEL, P. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In: **Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)**. [S.l.: s.n.], 2004. p. 343–360. Citado na página 56.
- CHO, K.; MERRIENBOER, B. van; GÜLÇEHRE, Ç.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. **CoRR**, abs/1406.1078, 2014. Disponível em: <<http://arxiv.org/abs/1406.1078>>. Citado na página 81.
- CLARK, P.; MURRAY, W. R.; THOMPSON, J.; HARRISON, P.; HOBBS, J.; FELLBAUM, C. On the Role of Lexical and World Knowledge in RTE3. In: **Proceedings of the Workshop on Textual Entailment and Paraphrasing**. [S.l.: s.n.], 2007. p. 54–59. Citado nas páginas 32, 33 e 34.

COLLOBERT, R. Deep learning for efficient discriminative parsing. In: **AISTATS**. [S.l.: s.n.], 2011. Citado na página [81](#).

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**. [S.l.: s.n.], 2008. p. 160–167. Citado na página [80](#).

DAGAN, I.; DOLAN, B.; MAGNINI, B.; ROTH, D. Recognizing textual entailment: Rational, evaluation and approaches. **Natural Language Engineering**, v. 15, n. 4, p. i–xvii, 2009. ISSN 1351-3249. Citado nas páginas [30](#), [36](#) e [38](#).

DAGAN, I.; GLICKMAN, O.; MAGNINI, B. The PASCAL Recognising Textual Entailment Challenge. In: CANDELA, J. Quiñonero; DAGAN, I.; MAGNINI, B.; BUC, F. d'Alché (Ed.). **Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment**. [S.l.: Springer Berlin Heidelberg, 2006, (Lecture Notes in Computer Science, v. 3944). p. 177–190. Citado nas páginas [23](#), [24](#), [29](#), [104](#), [106](#) e [107](#).

DAGAN, I.; ROTH, D.; SAMMONS, M.; ZANZOTTO, F. M. **Recognizing Textual Entailment: Models and Applications**. [S.l.: Morgan & Claypool, 2013. (Synthesis Lectures on Human Language Technologies). Citado nas páginas [23](#), [24](#), [25](#), [29](#), [30](#), [32](#), [35](#), [51](#), [60](#) e [114](#).

DEMAINE, E. D.; MOZES, S.; ROSSMAN, B.; WEIMANN, O. an optimal decomposition algorithm for tree edit distance. Citado na página [57](#).

DOLAN, B.; QUIRK, C.; BROCKETT, C. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In: **Proceedings of the 20th International Conference on Computational Linguistics**. [S.l.: s.n.], 2004. p. 350–356. Citado na página [104](#).

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. **Journal of Machine Learning Research**, v. 12, p. 2121–2159, 2011. Citado na página [122](#).

FEITOSA, D. B.; PINHEIRO, V. C. Análise de Medidas de Similaridade Semântica na Tarefa de Reconhecimento de Implicação Textual. In: **Proceedings of Symposium in Information and Human Language Technology**. [S.l.: s.n.], 2017. Citado na página [112](#).

FELLBAUM, C. **WordNet: An Electronic Lexical Database**. [S.l.: The MIT Press, 1998. Citado nas páginas [53](#) e [112](#).

FIALHO, P.; MARQUES, R.; MARTINS, B.; COHEUR, L.; QUARESMA, P. INESC-ID no ASSIN: Measuring semantic similarity and recognizing textual entailment. **Linguamática**, 2016. Citado nas páginas [65](#), [112](#), [113](#), [122](#) e [137](#).

FONSECA, E. R.; ALUÍSIO, S. M. **A Deep Architecture for Non-Projective Dependency Parsing**. 2015. Aceito para publicação no Workshop on Vector Space Modeling for NLP/NAACL. Citado na página [104](#).

- FONSECA, E. R.; SANTOS, L. B. dos; CRISCUOLO, M.; ALUÍSIO, S. M. Visão Geral da Avaliação de Similaridade Semântica e Inferência Textual. **Linguamática**, v. 8, n. 2, p. 3–13, 2016. Citado na página 26.
- GANITKEVITCH, J.; CALLISON-BURSCH, C. The Multilingual Paraphrase Database. In: **Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)**. [S.l.: s.n.], 2014. p. 4276–4283. Citado na página 54.
- GIAMPICCOLO, D.; MAGNINI, B.; DAGAN, I.; DOLAN, B. The Third PASCAL Recognizing Textual Entailment Challenge. In: **Proceedings of the Workshop on Textual Entailment and Paraphrasing**. [S.l.: s.n.], 2007. p. 1–9. Citado nas páginas 106 e 107.
- GOLDBERG, Y. A Primer on Neural Network Models for Natural Language Processing. v. 57, p. 345–420, 2016. Citado nas páginas 25, 77 e 82.
- GONÇALO OLIVEIRA, H. CONTO.PT: Groundwork for the Automatic Creation of a Fuzzy Portuguese Wordnet. In: **Proceedings of 12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)**. [S.l.: s.n.], 2016. Citado na página 54.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <<http://www.deeplearningbook.org>>. Citado nas páginas 78, 82 e 83.
- GUPTA, A.; KAUR, M.; SINGH, A.; GOEL, A.; MIRKIN, S. Text summarization through entailment-based minimum vertex cover. In: **Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)**. [S.l.: s.n.], 2014. p. 75–80. Citado nas páginas 23 e 31.
- HABASH, N.; DORR, B. A Categorical Variation Database for English. In: **Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1**. [S.l.: s.n.], 2003. p. 17–23. Citado na página 55.
- HARABAGIU, S.; HICKL, A. Methods for using textual entailment in open-domain question answering. In: **Proceedings of the 21st International Conference on Computational Linguistics and 44th Meeting of the ACL**. [S.l.: s.n.], 2006. Citado na página 23.
- HARMELING, S. Inferring textual entailment with a probabilistically sound calculus. **Natural Language Engineering**, v. 15, n. 4, p. 459–477, 2009. ISSN 1351-3249. Citado nas páginas 68, 69 e 72.
- HARTMANN, N. S. Solo Queue no ASSIN: Mix of a Traditional and an Emerging Approaches. **Linguamática**, 2016. Citado nas páginas 113 e 134.
- HEILMAN, M.; SMITH, N. A. Tree Edit Models for Recognizing Textual Entailments, Paraphrases, and Answers to Questions. In: **Proceedings of HLT '10 Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics**. [S.l.: s.n.], 2010. p. 1011–1019. Citado nas páginas 58 e 71.
- HICKL, A.; BENSLEY, J. A Discourse Commitment-Based Framework for Recognizing Textual Entailment. In: **Proceedings of the Workshop on Textual Entailment and Paraphrasing**. [S.l.: s.n.], 2007. p. 171–176. Citado na página 41.

- HICKL, A.; BENSLEY, J.; WILLIAMS, J.; ROBERTS, K.; RINK, B.; SHI, Y. Recognizing Textual Entailment with LCC's GROUNDHOG System. 2006. Citado na página 41.
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Disponível em: <<http://dx.doi.org/10.1162/neco.1997.9.8.1735>>. Citado nas páginas 42 e 84.
- IFTENE, A. UAIC Participation at RTE4. In: **Proceedings of the Text Analysis Conference 2008**. [S.l.: s.n.], 2008. Citado na página 62.
- IFTENE, A.; MORUZ, M.-A. UAIC Participation at RTE5. In: **Proceedings of the Text Analysis Conference 2009**. [S.l.: s.n.], 2009. Citado nas páginas 62 e 63.
- JIA, H.; HUANG, X.; MA, T.; WAN, X.; XIAO, J. PKUTM participation at TAC 2010 RTE and summarization track. In: **Proceedings of the Text Analysis Conference 2010**. [S.l.: s.n.], 2010. Citado na página 63.
- KENTER, T.; RIJKE, M. de. Short text similarity with word embeddings. In: ASSOCIATION FOR COMPUTING MACHINERY. **Proceedings of the 24th ACM International on Conference on Information and Knowledge Management**. [S.l.], 2015. p. 1411–1420. Citado na página 66.
- KHOT, T.; SABHARWAL, A.; CLARK, P. SciTail: A textual entailment dataset from science question answering. In: **AAAI**. [S.l.: s.n.], 2018. Citado nas páginas 25, 31 e 46.
- KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. **ArXiv e-prints**, dez. 2014. Citado na página 122.
- KIROS, R.; ZHU, Y.; SALAKHUTDINOV, R. R.; ZEMEL, R.; URTASUN, R.; TORRALBA, A.; FIDLER, S. Skip-Thought Vectors. In: CORTES, C.; LAWRENCE, N. D.; LEE, D. D.; SUGIYAMA, M.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 28**. Curran Associates, Inc., 2015. p. 3294–3302. Disponível em: <<http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf>>. Citado na página 45.
- KLEIN, P. N. Computing the edit-distance between unrooted ordered trees. In: BILARDI, G.; ITALIANO, G. F.; PIETRACAPRINA, A.; PUCCI, G. (Ed.). **Algorithms — ESA' 98**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 91–102. Citado na página 57.
- KOCH, I. V. **A Coesão Textual**. 22. ed. [S.l.]: Contexto, 2013. Citado na página 34.
- KOUYLEKOV, M.; NEGRI, M. An Open-Source Package for Recognizing Textual Entailment. In: **Proceedings of the ACL 2010 System Demonstrations**. [S.l.: s.n.], 2010. p. 42–47. Citado na página 61.
- LAI, A.; HOCKENMAIER, J. Illinois-LH: A Denotational and Distributional Approach to Semantics. In: **Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)**. [S.l.: s.n.], 2014. p. 329–334. Citado na página 43.
- LAI, Y. B. A.; HOCKENMAIER, J. Natural Language Inference from Multiple Premises. In: **Proceedings of The 8th International Joint Conference on Natural Language Processing**. [S.l.: s.n.], 2017. p. 100–109. Citado na página 47.

- LE, Q.; MIKOLOV, T. Distributed Representations of Sentences and Documents. In: **Proceedings of the 31st International Conference on Machine Learning (ICML-14)**. [S.l.: s.n.], 2014. p. 1188–1196. Citado na página 42.
- LECUN, Y.; BENGIO, Y.; HINTON, G. Deep Learning. **Nature**, v. 521, p. 436–444, 2015. Citado na página 78.
- LIN, D.; PANTEL, P. Discovery of inference rules for question-answering. **Natural Language Engineering**, v. 7, n. 04, 2001. ISSN 1351-3249. Citado nas páginas 55 e 69.
- LOBUE, P.; YATES, A. Types of common-sense knowledge needed for recognizing textual entailment. In: **Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics**. [S.l.: s.n.], 2011. p. 329–334. Citado na página 34.
- MANNING, C. D.; SURDEANU, M.; BAUER, J.; FINKEL, J.; BETHARD, S. J.; MCCLOSKEY, D. The Stanford CoreNLP natural language processing toolkit. In: **Association for Computational Linguistics (ACL) System Demonstrations**. [s.n.], 2014. p. 55–60. Disponível em: <<http://www.aclweb.org/anthology/P/P14/P14-5010>>. Citado na página 116.
- MARELLI, M.; BENTIVOGLI, L.; BARONI, M.; BERNARDI, R.; MENINI, S.; ZAMPARELLI, R. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In: **Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)**. [S.l.: s.n.], 2014. p. 1–8. Citado nas páginas 43 e 104.
- MARELLI, M.; MENINI, S.; BARONI, M.; BENTIVOGLI, L.; BERNARDI, R.; ZAMPARELLI, R. A SICK cure for the evaluation of compositional distributional semantic models. In: **Proceedings of the Ninth International Conference on Language Resources and Evaluation**. [S.l.: s.n.], 2014. p. 216–223. Citado nas páginas 24, 25 e 41.
- MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **CoRR**, abs/1301.3781, 2013. Disponível em: <<http://arxiv.org/abs/1301.3781>>. Citado na página 80.
- MIRKIN, S.; DAGAN, I.; PADÓ, S. Assessing the role of discourse references in entailment inference. In: **Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics**. [S.l.: s.n.], 2010. Citado na página 35.
- MUNKHDALAI, T.; YU, H. Neural semantic encoders. **CoRR**, abs/1607.04315, 2016. Disponível em: <<http://arxiv.org/abs/1607.04315>>. Citado nas páginas 42 e 88.
- NEGRI, M.; MARCHETTI, A.; MEHDAD, Y.; BENTIVOGLI, L.; GIAMPICCOLO, D. Semeval-2012 Task 8: Cross-lingual Textual Entailment for Content Synchronization. In: **First Joint Conference on Lexical and Computational Semantics (*SEM)**. [S.l.: s.n.], 2012. p. 399–407. Citado nas páginas 26 e 48.
- NVIDIA. **GPU-Based Deep Learning Inference: A Performance and Power Analysis**. [S.l.], 2015. Disponível em: <https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf>. Citado na página 81.

- PAIVA, V. de; RADEMAKER, A.; MELO, G. de. OpenWordNet-PT: An Open Brazilian WordNet for Reasoning. In: **Proceedings of the 24th International Conference on Computational Linguistics**. [s.n.], 2012. See at <http://www.coling2012-iitb.org> (Demonstration Paper). Published also as Techreport <http://hdl.handle.net/10438/10274>. Disponível em: <http://hdl.handle.net/10438/10274>. Citado nas páginas 54 e 121.
- PARIKH, A. P.; TÄCKSTRÖM, O.; DAS, D.; USZKOREIT, J. A Decomposable Attention Model for Natural Language Inference. In: **Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing**. [S.l.: s.n.], 2016. p. 2249–2255. Citado nas páginas 45, 86, 93, 95, 96, 97, 115, 121, 122, 147 e 148.
- PAWLIK, M.; AUGSTEN, N. RTED: A Robust Algorithm for the Tree Edit Distance. In: **Proceedings of the VLDB Endowment**. [S.l.: s.n.], 2011. p. 334–345. Citado na página 57.
- PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citado na página 121.
- PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: **Empirical Methods in Natural Language Processing (EMNLP)**. [s.n.], 2014. p. 1532–1543. Disponível em: <http://www.aclweb.org/anthology/D14-1162>. Citado na página 80.
- PINHEIRO, A.; FERREIRA, R.; DIONÍSIO, M.; ROLIM, V.; TENÓRIO, J. Statistical and Semantic Features to Measure Sentence Similarity in Portuguese. In: **Proceedings of the Brazilian Conference on Intelligent Systems**. [S.l.: s.n.], 2017. p. 342–347. Citado na página 112.
- ROCKTÄSCHEL, T.; GREFFENSTETTE, E.; HERMANN, K. M.; KOČISKÝ, T.; BLUNSOM, P. Reasoning about entailment with neural attention. **arXiv preprint**, 2015. Disponível em: <http://arxiv.org/abs/1509.06664>. Citado nas páginas 44, 45, 86 e 92.
- SHARMA, N.; SHARMA, R.; BISWAS, K. K. Recognizing Textual Entailment using Dependency Analysis and Machine Learning. In: **Proceedings of NAACL-HLT 2015 Student Research Workshop**. [S.l.: s.n.], 2015. p. 147–153. Citado na página 134.
- SHEN, T.; ZHOU, T.; LONG, G.; JIANG, J.; PAN, S.; ZHANG, C. DiSAN: Directional Self-Attention Network for RNN/CNN-free Language Understanding. **ArXiv e-prints**, 2017. Citado nas páginas 45, 86 e 90.
- SHINYAMA, Y.; SEKINE, S. Paraphrase acquisition for information extraction. In: **Proceedings of the Second International Workshop on Paraphrasing**. [S.l.: s.n.], 2003. Citado na página 32.
- SILVA, A. de B.; RIGO, S. J.; ALVES, I. M.; BARBOSA, J. L. V. Avaliando a similaridade semântica entre frases curtas através de uma abordagem híbrida. In: **Proceedings of Symposium in Information and Human Language Technology**. [S.l.: s.n.], 2017. p. 93–102. Citado na página 111.

- SOCHER, R.; CHEN, D.; MANNING, C. D.; NG, A. Y. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In: **Advances in Neural Information Processing Systems 26**. [S.l.: s.n.], 2013. Citado nas páginas 42 e 81.
- SOCHER, R.; HUVAL, B.; MANNING, C. D.; NG, A. Y. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In: **Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. [S.l.: s.n.], 2012. Citado nas páginas 42 e 81.
- SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal of Machine Learning Research**, v. 15, p. 1929–1958, 2014. Citado na página 87.
- STERN, A.; DAGAN, I. A Confidence Model for Syntactically-Motivated Entailment Proofs. In: **Proceedings of Recent Advances in Natural Language Processing**. [S.l.: s.n.], 2011. p. 455–462. Citado na página 69.
- _____. BIUTEE: A Modular Open-Source System for Recognizing Textual Entailment. In: **Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics**. [S.l.: s.n.], 2012. p. 73–78. Citado nas páginas 69 e 72.
- SUTSKEVER, I.; MARTENS, J.; HINTON, G. Generating Text with Recurrent Neural Networks. In: **Proceedings of the 28th International Conference on Machine Learning**. [S.l.: s.n.], 2011. Citado na página 78.
- TAI, K. S.; SOCHER, R.; MANNING, C. D. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In: **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing**. [S.l.: s.n.], 2015. p. 1556–1566. Citado na página 95.
- TANG, D.; WEI, F.; YANG, N.; ZHOU, M.; LIU, T.; QIN, B. Learning sentiment-specific word embedding for twitter sentiment classification. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics**. [S.l.: s.n.], 2014. p. 1555–1565. Citado na página 81.
- TATU, M.; ILES, B.; SLAVICK, J.; NOVISCHI, A.; MOLDOVAN, D. COGEX at the second recognizing textual entailment challenge. In: **Proceedings of the Second PASCAL Challenges Workshop on Recognizing Textual Entailment**. [S.l.: s.n.], 2006. Citado na página 66.
- TSUCHIDA, M.; ISHIKAWA, K. IKOMA at TAC2011: A Method for Recognizing Textual Entailment using Lexical-level and Sentence Structure-level features. In: **Proceedings of the Text Analysis Conference 2011**. [S.l.: s.n.], 2011. Citado na página 62.
- TURIAN, J.; RATINOV, L.; BENGIO, Y. Word representations : A simple and general method for semi-supervised learning. In: **Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics**. [S.l.: s.n.], 2010. p. 384–394. Citado na página 65.
- TURNEY, P. D.; PANTEL, P. From frequency to meaning: Vector space models of semantics. **Journal of Artificial Intelligence Research**, v. 37, p. 141–188, 2010. Citado na página 61.

- VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. **CoRR**, abs/1706.03762, 2017. Disponível em: <<http://arxiv.org/abs/1706.03762>>. Citado nas páginas 85 e 86.
- WANG, R.; ZHANG, Y.; NEUMANN, G. A joint syntactic-semantic representation for recognizing textual relatedness. In: **Proceedings of the Text Analysis Conference 2009**. [S.l.: s.n.], 2009. Citado na página 64.
- WANG, S.; JIANG, J. Learning Natural Language Inference with LSTM. **arXiv pre-print**, 2015. Citado na página 44.
- WANG, Z.; HAMZA, W.; FLORIAN, R. Bilateral multi-perspective matching for natural language sentences. **CoRR**, abs/1702.03814, 2017. Disponível em: <<http://arxiv.org/abs/1702.03814>>. Citado nas páginas 15 e 99.
- WILLIAMS, A.; NANGIA, N.; BOWMAN, S. R. A broad-coverage challenge corpus for sentence understanding through inference. **ArXiv e-prints**, abs/1704.05426, 2017. Disponível em: <<http://arxiv.org/abs/1704.05426>>. Citado nas páginas 25, 32 e 45.
- YU, T. M. H. Neural Semantic Encoders. In: **Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics**. [S.l.: s.n.], 2017. Citado na página 45.
- ZANOLI, R.; COLOMBO, S. A transformation-driven approach for recognizing textual entailment. **Natural Language Engineering**, v. 23, n. 4, p. 507–534, 2016. Citado nas páginas 72, 115, 121, 123, 124 e 148.
- ZELLER, B.; PADÓ, S. A Search Task Dataset for German Textual Entailment . In: **Proceedings of the 10th International Conference on Computational Semantics (IWCS)**. Potsdam: [s.n.], 2013. p. 288–299. Disponível em: <<http://www.cl.uni-heidelberg.de/~zeller/publications/iwcs2013.pdf>>. Citado na página 26.
- ZHANG, F. D.; YANG, J. Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring. In: **Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)**. [S.l.: s.n.], 2017. p. 153–162. Citado na página 85.
- ZHANG, K.; SHASHA, D. Simple fast algorithms for the editing distance between trees and related problems. **SIAM Journal of Computing**, v. 18, p. 1245–1262, 1989. Citado na página 57.
- ZHANG, S.; RUDINGER, R.; DUH, K.; Van Durme, B. Ordinal common-sense inference. **Transactions of the Association for Computational Linguistics**, v. 5, p. 379–395, 2017. ISSN 2307-387X. Disponível em: <<https://transacl.org/ojs/index.php/tacl/article/view/1082>>. Citado na página 48.
- ZHAO, S.; LAN, X.; LIU, T.; LI, S. Application-driven statistical paraphrase generation. In: **Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP**. [S.l.: s.n.], 2009. Citado na página 31.