O PROBLEMA DA INCOERÊNCIA E A REGULARIZAÇÃO SEMÂNTICA PARA INFERÊNCIA TEXTUAL

Gabriel Garcia de Almeida

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Setembro de 2018

O PROBLEMA DA INCOERÊNCIA E A REGULARIZAÇÃO SEMÂNTICA PARA INFERÊNCIA TEXTUAL

Gabriel Garcia de Almeida

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Bonorino Xexéo, Ph.D.

Prof. Jano Moreira de Souza, Ph.D.

Prof. Eduardo Soares Ogasawara, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2018

Almeida, Gabriel Garcia de

O problema da incoerência e a regularização semântica para inferência textual/Gabriel Garcia de Almeida. – Rio de Janeiro: UFRJ/COPPE, 2018.

XI, 70 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 66 – 70.

1. inferência textual. 2. incoerência. 3. regularização semântica. 4. processamento de linguagem natural. 5. lógica nebulosa. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Sobre o ombro de gigantes

Agradecimentos

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

Primeiramente, gostaria de agradecer especialmente aos meus pais, Sonia e Olimar, que me acompanharam diariamente nessa jornada cheia de altos e baixos, além dos meus familiares, que estavam sempre presentes. Em especial, meus primos Cecília, Demian e Liana, por fazerem recorrentemente a incômoda mas necessária pergunta "E o mestrado?".

Quero agradecer ao meu orientador, professor Geraldo Xexéo, por suas sugestões e direcionamentos nesta pesquisa, e pela paciência comigo, especialmente durante a escolha do tema. Outras duas pessoas que ajudaram diretamente neste trabalho foram os professores Fellipe Duarte e Eduardo Bezerra, com nossas várias conversas e discussões sobre a dissertação. Agradeço também a banca de defesa dos professores Jano Moreira e Eduardo Ogasawara por suas contribuições. Além do pessoal do LINE, meu grupo de pesquisa, por ouvir incontáveis versões do meu projeto.

Importante agradecer também aos meus colegas e amigos do PESC e da UFRJ. Em especial, toda equipe do CAPGov e seus vários braços, onde pude contribuir nos mais variados projetos e expandir meus conhecimentos. Quero agradecer mais diretamente ao Vitor, que foi quase uma constante nos vários projetos que participei, além de me ajudar com a formatação desta dissertação. Ao Jorge e Paulo, que também fizeram parte diretamente de um singelo projeto interno, junto com nossos "usuários/testadores" Ana Carolina, Arthur, Débora, Marcelle, Vanessa e Xiaos, para citar alguns, que contribuíram para a construção de algo com um impacto inesperado. E à Bárbara, que também me ajudou com uma sucinta conversa sobre visualização de dados, aplicada bastante nesta dissertação. Obrigado, pessoal, sobre tudo pelos necessários momentos de descontração e desabafos.

Quero agradecer também aos meus amigos do PPGI pelas ideias trocadas, mesmo mais distantes neste momento. E também aos profissionais da UFRJ, que mantém toda essa grande máquina funcionando mesmo com as diversas incertezas destes tempos, especialmente as secretárias, cobrando assuntos acadêmicos urgentes.

E por fim, quero agradecer também à você, que está lendo esta dissertação agora. Faz parte do esforço valer a pena. Espero que seja útil.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

O PROBLEMA DA INCOERÊNCIA E A REGULARIZAÇÃO SEMÂNTICA PARA INFERÊNCIA TEXTUAL

Gabriel Garcia de Almeida

Setembro/2018

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

O reconhecimento de inferência textual é uma tarefa do processamento de linguagem natural que busca mensurar a capacidade dos algoritmos de comparar um par de sentenças a nível semântico. Ela é útil direta ou indiretamente em diversas aplicações como tradução de máquina, sumarização e respostas automáticas. Esta tarefa recebeu grande atenção com o lançamento do *dataset* SNLI, possibilitando a aplicação de complexas técnicas de *deep learning* que obtiveram diversos resultados expressivos. Alguns trabalhos, porém, começam a questionar tais resultados, observando os vieses explorados pelos algoritmos de aprendizado. Esta dissertação discute mais um possível problema destes métodos: a incoerência entre as respostas. É apresentada uma definição formal, baseada em lógica proposicional, do que é uma resposta coerente. Também é dada uma solução que visa diminuir a incoerência dos modelos, aplicável a qualquer algoritmo de *deep learning*, além de experimentos que avaliam alguns possíveis impactos da incoerência e a eficácia da solução proposta.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

THE INCOHERENCE PROBLEM AND SEMANTIC REGULARIZATION FOR TEXTUAL ENTAILMENT

Gabriel Garcia de Almeida

September/2018

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

Recognition of textual entailment is a task of natural language processing that aims to measure the ability of algorithms to compare a pair of sentences at the semantic level. It is useful directly or indirectly in various applications such as machine translation, summarization and question answering. This task received attention with the release of the SNLI dataset, making possible the application of complex deep learning techniques that obtained several expressive results. Some works, however, begin to question such results, observing the biases explored by the learning algorithms. This dissertation discusses another possible problem of these methods: the incoherence between the answers. A formal definition, based on propositional logic, is presented for what is a coherent response. It is also given a solution that seeks to reduce the incoherence of the models, applicable to any deep learning algorithm, as well as experiments that evaluate some possible impacts of incoherence and the effectiveness of the proposed solution.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Com o uso da tecnologia se tornando cada vez mais ubíquo nas sociedades modernas, as iterações entre humanos e máquinas se tornam mais comuns e diversas. Para facilitar tais iterações, interfaces intuitivas e práticas para os usuários precisam ser desenvolvidas e possivelmente a mais natural de todas as interfaces seja a própria linguagem humana. É comum interagir em redes sociais usando mensagens de texto, buscar informações em motores de busca por meio de consultas textuais ou até fazer requisições a assistentes pessoais com comandos de voz. A comunicação entre pessoas se dá principalmente a partir da linguagem, é um passo natural também usá-la para a comunicação entre humanos e máquinas.

A compreensão da linguagem natural se torna um importante tópico para o avanço da tecnologia. A automação deste processo de forma algorítmica é um difícil problema na ciência da computação, atacado de forma direta ou indireta nas mais diversas aplicações computacionais como assistentes pessoais, tradução e resposta automática (WILLIAMS *et al.*, 2018; DAGAN *et al.*, 2013). A grande dificuldade desse tipo de problema é devida especialmente as ambiguidades quando se mapeiam palavras de uma sentença para seus respectivos significados (DAGAN *et al.*, 2006; MANNING *et al.*, 1999).

Uma das diversas formas que o problema da compreensão da linguagem natural toma é a tarefa de **reconhecimento de inferência textual**. Esta tarefa consiste em, dada uma sentença como premissa, identificar se outra sentença, a hipótese, pode ou não ser inferida a partir da primeira. Por exemplo, a partir da sentença "João está trabalhando no computador em seu escritório." pode-se inferir que uma pessoa está usando o computador, mas não que João está na cozinha. Outra forma de se interpretar esta tarefa é por meio da pergunta: se uma pessoa diz que a premissa é verdadeira, ela também deveria dizer que a hipótese é verdadeira?

Pode se dizer que o reconhecimento de inferência textual é uma generalização do problema de identificação de paráfrases: um par de paráfrases pode ser identificado por um método de inferência textual caso este diga que tanto a premissa implica na hipótese, quanto a hipótese implica na premissa. Na inferência textual, saber qual sentença é a premissa e qual é a hipótese é importante para o algoritmo tomar uma decisão, mas esta informação não faz diferença numa identificação de paráfrases (ANDROUTSOPOULOS e MALAKASIOTIS, 2010).

O reconhecimento de inferência textual é uma tarefa interessante pois requer, em última instância, que os algoritmos compreendam o significado de cada sentença e sejam capazes de fazer deduções a partir delas. Esta tarefa busca medir a capacidade de compreensão semântica de um dado algoritmo (WILLIAMS *et al.*, 2018), mesmo que de forma limitada e, também é útil tanto em aplicações diretas quanto indiretas. Indiretamente, é desejável, por exemplo, que algoritmos de tradução de máquina gerem traduções que podem ser inferidas a partir do texto original. Por outro lado, uma sumarização pode ser feita diretamente usando um método de reconhecimento de inferências textuais para excluir sentenças redundantes, que podem ser inferidas a partir de outras (ANDROUTSOPOULOS e MALAKASIOTIS, 2010). Modelos de inferência textual já foram até usados para se construir representações vetoriais de sentenças arbitrárias e aplica-las num contexto de categorização de textos onde se tem poucos dados disponíveis (CONNEAU *et al.*, 2017).

Com o surgimento do conjunto de dados SNLI (BOWMAN *et al.*, 2015), a tarefa de inferência textual ganhou grande interesse. No SNLI, a tarefa é vista como uma classificação, na qual é dado o par de sentenças e o algoritmo deve dizer sua relação: *inferência*, *neutralidade* ou *contradição*. No SNLI, em vez de se adotar uma categorização binária, optou-se por uma abordagem um pouco mais granular, na qual a "não inferência" foi dividida nas categorias de contradição e neutralidade. No caso da contradição, a premissa se opõe a hipótese, como no exemplo acima: dizer que João está no escritório contradiz que ele está na cozinha. Já a neutralidade pode ser entendida como uma incerteza: se é dito que João está andando de bicicleta com Maria, não é possível nem afirmar e nem contradizer que duas pessoas estão competindo numa corrida de bicicletas.

Possibilitados pelo grande tamanho do SNLI, os trabalhos mais recentes usando este conjunto de dados, como ROCKTÄSCHEL *et al.* (2016) ou CHEN *et al.* (2016), se propuseram a atacar o problema utilizando complexos modelos de aprendizado de máquina e obtiveram altas taxas de acerto durante a avaliação da tarefa. Entretanto, alguns trabalhos começaram levantar dúvidas a respeito da qualidade geral desses sistemas e do próprio conjunto de dados. Por exemplo, o trabalho de GURURANGAN *et al.* (2018) identificou que um classificador que somente usa a sentença da hipótese já tem uma taxa de acerto de mais de 50%, algo indesejado quando se considera que

a tarefa propõe avaliar a relação entre pares de sentenças. Ao remover esses pares que podem ser acertados sem a ajuda da premissa, modelos do estado da arte perderam mais de 10% de acurácia. Outro estudo que mostra mais fragilidades dos modelos é NAIK *et al.* (2018), que cria um conjunto de dados com situações pouco usuais para validar o funcionamento dos modelos. Por exemplo, foram criados pares de sentenças que terminam com tautologias lógicas claras como " e verdade é verdade" como uma forma de "distrair" o modelo, o que ocasionou uma queda de mais de 20% de acurácia de diversos métodos.

Esses trabalhos evidenciam a necessidade de novas formas de validação dos modelos de reconhecimento de inferência textual, ou até dos conjuntos de dados usados, afim de torna-los mais robustos, aplicáveis em mais contextos e contribuir com o avanço da compreensão da linguagem natural. Nesta dissertação é introduzido o conceito da **coerência**, que pode ser usado como uma forma de avaliação dos modelos. É dito que um modelo foi coerente para um dado par de premissa e hipótese, neste contexto das três classes do SNLI, quando suas respostas para este par e seu inverso obedecem a um conjunto de regras simples, derivadas da própria interpretação lógica da inferência textual. Resumidamente, saber a relação da premissa com a hipótese permite limitar as possíveis relações quando a premissa assume o lugar da hipótese e vice-versa, mesmo sem saber qual seria a relação real de fato.

Experimentos realizados sugerem que as respostas coerentes de um modelo tendem a ser mais acuradas que suas respostas incoerentes. Ou seja, a coerência pode funcionar como um critério de qualidade dos modelos de inferência textual, baseada somente nas previsões do próprio modelo e sendo capaz até de ser utilizada para descartar certas predições de baixa qualidade, quando o modelo é incoerente.

1.2 Objetivo e Motivação

Esta dissertação tem por objetivo definir e atacar um possível problema nos sistemas atuais, derivado diretamente da natureza lógica da tarefa: a incoerência. Um modelo será considerado coerente para um par de sentenças se sua resposta cumprir algumas restrições quando avaliado tanto o par original quando o par invertido: a relação da sentença A com a sentença B restringe as possíveis relações de B com A.

Um modelo ser o mais coerente possível é interessante do ponto de vista de uma aplicação real pois os ajudaria a terem uma melhor capacidade de generalização, restringindo seu erro em situações não vistas, mesmo sem saber a resposta exata. Além disso, se evitaria estranheza em avaliadores humanos julgando as respostas do algoritmo, assumindo que os julgadores buscam coerência nos resultados.

Uma forma de se atacar o problema da incoerência lógica também será apresentada para modelos de aprendizado de máquina em geral, como redes neurais

artificiais, através da adição de um termo de regularização, inspirado em aprendizado semissupervisionado e baseado principalmente nos trabalhos de XU *et al.* (2017, 2018).

1.3 Organização

Esta dissertação se divide em sete capítulos. O primeiro é esta introdução, que dá uma pequena contextualização do problema atacado e os objetivos deste trabalho.

O segundo capítulo, a revisão da literatura, dará uma visão mais abrangente do problema. Ele começa falando resumidamente sobre o processamento de linguagem natural com alguns de seus problemas e aplicações, passa por definições e características do aprendizado de máquina e *deep learning*, além de aplicações em linguagem natural. Por fim, este capítulo termina falando da própria inferência textual, com definições, suas aplicações, conjuntos de dados e soluções anteriores, além dos problemas identificados de vieses.

O terceiro capítulo fala sobre técnicas utilizadas mas não mencionadas anteriormente, porém usadas na implementação, como a lógica *fuzzy*, a regularização no contexto de aprendizado de máquina e a *semantic loss*.

No quarto capítulo, a coerência é formalmente definida e a regularização semântica é proposta como uma forma de melhorar a coerência dos modelos via regularização.

O quinto capítulo explica o modelo utilizado para os experimentos e dá alguns detalhes de sua implementação.

O sexto capítulo discorre sobre os experimentos e os dados obtidos a respeito da coerência e do termo de regularização, avaliados nos diversos subconjuntos do *dataset* MultiNLI.

Por fim, o último capítulo conclui com as considerações finais, principais achados e contribuições, além de propostas de trabalhos futuros.

Capítulo 2

Revisão da literatura

2.1 Processamento de linguagem natural

O processamento de linguagem natural pode ser pensado num senso mais amplo como qualquer tipo de manipulação algorítmica da linguagem natural, a linguagem usada para a comunicação humana cotidianamente, como definido por BIRD *et al.* (2009).

O principal desafio do processamento de linguagem natural é que, diferentemente de linguagens artificiais, como a linguagem de programação Python ou a própria linguagem matemática. Dificilmente se chegará a regras de formação claras e bem definidas para a linguagem humana, usando, por exemplo, ferramentas como gramáticas livres de contexto, para se estruturar e atacar o problema de forma algorítmica (BIRD *et al.*, 2009).

São inúmeros os fatores que contribuem para esta dificuldade de estruturação, desde de simples erros ortográficos até neologismos, ambiguidades ou ironias. Entender proceduralmente a comunicação humana é uma atividade complexa e dificilmente objetiva. Bons sistemas de processamento de linguagem natural precisam conseguir decidir bem situações como qual categoria de uma palavra, qual seu significado ou qual estrutura sintática de uma sentença (MANNING *et al.*, 1999).

2.2 Processamento de linguagem natural estatístico

Uma possibilidade de desenvolver algoritmos de análise da linguagem natural é através da programação usual com uso de regras manuais feitas por especialistas. Porém tais abordagens são limitadas, exigindo um número crescente de regras para abranger cada vez mais construções linguísticas e que ainda estão sujeitas a cometer erros devidos a usos de construções pouco comuns ou metáforas (MANNING *et al.*,

1999).

Outra possível abordagem é o uso de modelos estatísticos, que tem a vantagem de se comportar bem diante de novas situações e são mais robustos a erros. Eles tentam aprender os padrões textuais a partir de um corpus (conjunto de documentos) (MANNING *et al.*, 1999). Esta característica permitiria que a demanda por intervenção humana fosse simplificada à construção e avaliação do próprio modelo estatístico, além de possíveis anotações no corpus, quando necessárias para o problema desejado.

2.2.1 Aplicações

Uma categoria importante de aplicação do processamento de linguagem natural, e tipicamente atacada usando métodos estatísticos, é a **classificação de texto** ou documentos. Esta tarefa envolve em, dado um documento, o algoritmo deve dizer qual sua categoria dentre um conjunto finito de alternativas predefinidas.

Um caso clássico de classificação de texto é a **análise de sentimentos**, que envolve dizer qual sentimento predominante (ex.: positivo, negativo, neutro) de um dado texto, como uma postagem numa rede social, o que torna esta tarefa útil para, por exemplo, se estimar a opinião do público a respeito de uma marca ou assunto em especial (PANG *et al.*, 2008).

Outro exemplo de tarefa de classificação de texto menos usual é o chamado **roteamento de incidentes**, que envolve direcionar formulários a respeito de incidentes diversos para a área organizacional adequada (ex.: equipe de infraestrutura, banco de dados, segurança ...) para solucionar o incidente (FERREIRA e XEXÉO, 2017). A motivação do roteamento de incidentes é reduzir custos e o tempo de resposta de um incidente, tipicamente usado no contexto de serviços de tecnologia da informação.

Abordagens baseadas em contagem de palavras usadas em conjunto com classificadores provenientes do contexto de aprendizado de máquina e estatística, como o Naïve-Bayes ou o SVM, são capazes de ter desempenhos satisfatórios em várias tarefas de classificação texto (FERREIRA e XEXÉO, 2017; WANG e MANNING, 2012). Isso pode ser explicado pela capacidade destes algoritmos de ponderar as palavras-chaves, destacando as que mais diferenciam as classes de documentos desejadas a partir do conjunto de textos categorizados. Essa abordagem pode ser suficiente para tarefas que não tem a necessidade um entendimento semântico mais profundo da linguagem na maioria dos casos, como numa análise de sentimentos de sentenças mais polarizada, na qual palavras como “ótimo” ou “péssimo” seriam suficientes para indicar se uma resenha é positiva ou negativa.

Por outro lado, problemas como a **sumarização de textos** podem exigir métodos mais complexos para se tratar a linguagem escrita. A sumarização automática de texto, consiste em gerar um texto condensado a partir de um ou mais documentos

dados. A sumarização pode ser categorizada em duas classes amplas: os métodos extrativos e os abstrativos. Métodos extrativos consistem em selecionar as sentenças mais relevantes dos documentos dados, como por meio de alguma medida de relevância. Já a sumarização abstrativa gera resumos diferente dos textos originais, capturando os conceitos mais importantes e criando diferentes sentenças para os representar, sendo bem mais complexa que a variante extrativa (GAMBHIR e GUPTA, 2017).

O trabalho recente de sumarização abstrativa de RUSH *et al.* (2015) utiliza algumas técnicas similares as usadas em tradução automática, outra tarefa difícil da área, para atacar o problema: são usadas redes neurais artificiais, método proveniente do aprendizado de máquina, para estimar as probabilidades de qual deveria ser a próxima palavra do sumário, dado o texto original e o sumário até o momento. Esta metodologia cria um processo iterativo de construção do resumo, na qual se leva em conta tanto a informação original, quanto a semântica presente nas palavras em sequência que vão formar o resumo. Tal abordagem obtém resultados bem próximos à média humana de referência e supera todos os modelos *baseline* nos conjuntos de dados testados.

2.2.2 Hipótese distribucional

A eficácia dos métodos estatísticos em casos que necessitam de uma análise semântica mais refinada, como a sumarização abstrativa, pode ser entendida teoricamente pela chamada **hipótese distribucional** (HARRIS, 1954). A hipótese diz que palavras que ocorrem nos mesmos contextos tendem a ter significados similares. Desta forma, é possível identificar estatisticamente que as palavras "gato" e "cachorro" (ou "casa" e "quarto") são semanticamente similares, já que aparecem em frases como:

"Um gato está andando pela casa."

"O cachorro está andando no quarto."

Estas relações semânticas latentes entre as palavras podem ser capturadas por métodos como as redes neurais através do uso de grandes volumes de textos e aplicadas para se resolver problemas mais complexos, como proposto em BENGIO *et al.* (2003), trabalho que foi a espinha dorsal de vários outros, como RUSH *et al.* (2015).

2.3 Aprendizado de máquina

O conceito de aprendizado de máquina diz respeito a técnicas que se baseiam na construção de modelos computacionais que extraem padrões a partir de dados brutos, tipicamente utilizados para se atacar problemas difíceis de se usar uma solução lógica clara, como numa classificação de textos, e que frequentemente têm características

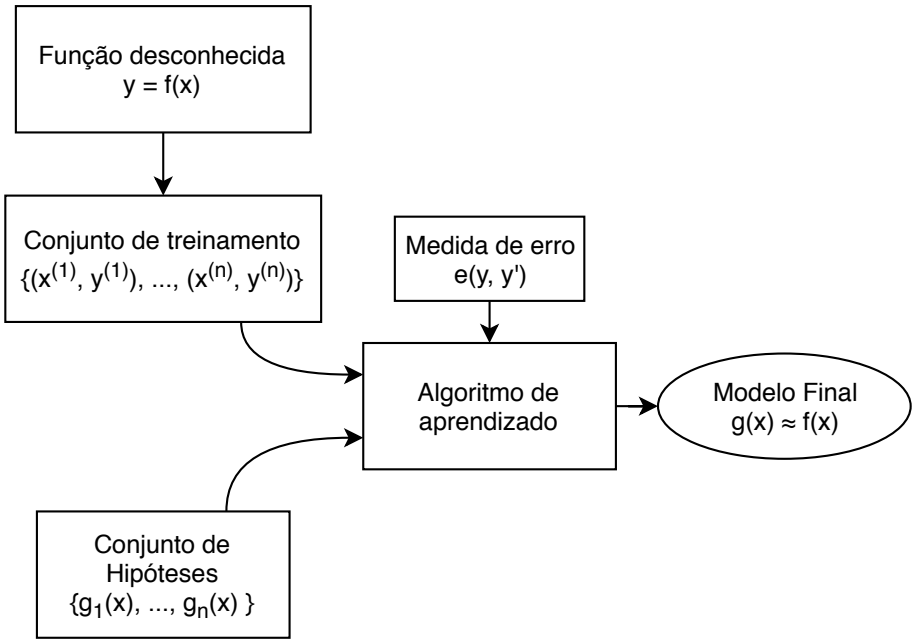

Figura 2.1: Resumo dos componentes de aprendizado de máquina, baseado em ABU-MOSTAFA *et al.* (2012)

subjetivas (GOODFELLOW *et al.*, 2016). Esta é uma área na fronteira entre a ciência da computação e a estatística, e que tem vários elementos em comum com diversas subáreas da própria computação como a mineração de dados e a inteligência computacional.

Para o paradigma do aprendizado supervisionado, métodos de aprendizado de máquina são tipicamente compostos por alguns componentes em comum: um conjunto de treinamento, um conjunto de hipóteses, uma medida de erro e um algoritmo de aprendizado (ABU-MOSTAFA *et al.*, 2012). Resumidamente, assume-se que existe uma função desconhecida que se deseja aproximar, porém só se conhecem finitos pontos dela, provenientes do conjunto de treinamento. O algoritmo de aprendizado deve então usar a medida de erro para escolher a hipótese que melhor representa o conjunto de treinamento, esperando que esta hipótese seja um bom aproximador para a função desconhecida e que generalize bem para casos ainda não vistos, sendo útil para resolver a tarefa proposta. A Figura 2.1 mostra este paradigma de forma resumida.

2.3.1 Conjunto de treinamento

Mais formalmente, os elementos do conjunto de treinamento são definidos como um conjunto de pares na forma $\{(x^{(1)}, y^{(1)}), ..., (x^{(n)}, y^{(n)})\}$, onde $x^{(i)} \in \{x^{(1)}, ..., x^{(n)}\}$ é um vetor do que representaria a i-ésima observação do que se assume ser o domínio de $f(x) = y$, a função de interesse desconhecida, e $y^{(i)} \in \{y^{(1)}, ..., y^{(n)}\}$ é a resposta dada por $f(x^{(i)})$. Caso y seja um valor contínuo ($y^{(i)} \in \mathbb{R}$) vai se configurar um

problema de **regressão**, e caso y seja discreto ($y^{(i)} \in \{1, ..., c\}$), o problema é dito ser uma **classificação** de c classes.

2.3.2 Conjunto de hipóteses

O conjunto de hipóteses é uma família de funções que se supõe que aproximam bem f. Um exemplo é a família de **modelos lineares**, que consiste numa simples combinação linear das entradas, dada matematicamente pela Equação 2.1 para o caso de uma regressão, onde $a_0, ..., a_m$ são os parâmetros do modelo, $\hat{y}$ é a resposta dada e $x_1, ..., x_m$ são os componentes de um vetor x, usado como domínio da função.

$$\hat{y} = a_0 + a_1 \times x_1 + ... + a_m \times x_m \tag{2.1}$$

Para casos de classificação, é desejável que o modelo retorne valores entre zero e um, que são interpretáveis como probabilidades daquele caso pertencer à classe de interesse. Isto pode ser feito normalizando a saída do modelo linear com a função sigmoide, que tem a forma $\sigma(x) = \frac{1}{1+e^{-x}}$. Esta combinação configura em uma **regressão logística**, um dos modelos mais simples usados em classificações binárias. Para casos com n classes, vários classificadores lineares podem ser combinados, o que pode ser descrito matricialmente pela Equação 2.2, onde o vetor b tem n entradas (representando cada uma das n classes), o vetor x tem tamanho m (para os m atributos de entrada) e a matriz A tem dimensão $m \times n$.

$$c = Ax + b \tag{2.2}$$

As probabilidades do classificador linear podem ser normalizadas usando a função ***softmax*** (GOODFELLOW *et al.*, 2016), dada pela Equação 2.3, que descreve a probabilidade respondida pelo modelo para a i-ésima classe, dado o vetor c de probabilidades não normalizadas, onde c_i representa o valor respondido para a i-ésima classe.

$$P(y = i|c) = \frac{e^{c_i}}{\sum_{j=1}^{n} e^{c_j}} \tag{2.3}$$

2.3.3 Medida de erro

A medida de erro mensura o quão bem um modelo explica os dados. Um exemplo clássico de medida de erro é o erro médio quadrático, usado em regressões e dado por $\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$, onde $y^{(i)}$ é a resposta esperada do i-ésimo exemplo do conjunto de treinamento, que tem n observações, e $\hat{y}^{(i)}$ é a i-ésima resposta do modelo.

Para situações de classificação, medidas como a acurácia ou precisão podem não ser boas para os algoritmos de otimização, devido a características como descontinuidades

na função, e por isso **funções substitutas** (*surrogate function*) são preferíveis para representar a função original (GOODFELLOW *et al.*, 2016). A **entropia cruzada** (*cross entropy*), denotada pela função $H(y, \hat{y})$, por exemplo, pode ser uma substituta para a acurácia em situações de classificação entre c classes. A Equação 2.4 define o valor da entropia cruzada individual para a i-ésima observação do conjunto de treinamento (a média deste valor no *dataset* completo seria a medida de erro final), enquanto a Equação 2.5 dá especificamente o caso binário no *dataset* inteiro, com uma notação simplificada, onde $y^{(i)}$ pode assumir os valores 1 ou 0, que representam se a observação referenciada pertence ou não a classe de interesse, enquanto $\hat{y}^{(i)}$ é a probabilidade, dada pelo modelo, da observação pertencer a esta classe.

$$H(y, \hat{y}) = \sum_{j=1}^{c} P(y^{(i)} = classe_j) \times \ln\left(P(\hat{y}^{(i)} = classe_j)\right) \tag{2.4}$$

$$H(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^{n} y^{(i)} \ln \hat{y}^{(i)} + (1 - y^{(i)}) \ln\left(1 - \hat{y}^{(i)}\right) \tag{2.5}$$

2.3.4 Algoritmo de aprendizado

Por fim, o algoritmo de aprendizado é o método que vai selecionar o modelo final dentre o conjunto de hipóteses, usando a função de erro aplicada no *dataset* como uma espécie de medida de qualidade para a solução. Esse processo pode ser entendido como uma **otimização** matemática, onde a medida de erro é a **função objetivo** e os parâmetros do modelo são as variáveis da otimização.

Em certas situações, existem soluções fechadas para o problema de minimização, como no caso da regressão linear, que usa o erro médio quadrático em conjunto com um modelo linear para atacar problemas de regressão. Entretanto, em diversos casos é necessário um procedimento genérico de minimização, como ao treinar uma regressão logística ou uma rede neural usando a entropia cruzada para um problema de classificação.

Nestes casos, pode-se usar métodos como o **gradiente descendente** (*gradient descent*) ou uma de suas variantes, como o gradiente descendente estocástico ou o algoritmo Adam (KINGMA e BA, 2014) para se realizar a otimização dos parâmetros do modelo. O gradiente descendente é uma técnica iterativa de minimização que consiste em realizar pequenas alterações nos parâmetros de forma que a função objetivo seja reduzida localmente a cada passo (ABU-MOSTAFA *et al.*, 2012; GOODFELLOW *et al.*, 2016). Este raciocínio pode ser descrito matematicamente como na Equação 2.6.

$$w_{t+1} = w_t - \eta \nabla_w f(w_t) \tag{2.6}$$

Nesta equação, a variável w_{t+1} representa os novos parâmetros do modelo, enquanto w_t são os parâmetros da iteração anterior. $\nabla_w f(w_t)$ é o gradiente com respeito a w da função objetivo $f(.)$ e avaliado no ponto w_t, que vai indicar a direção que o parâmetro deve ser alterado para causar a maior redução na função objetivo. η é o chamado coeficiente de aprendizado e vai regular o quanto os parâmetros vão ser alterados a cada passo da otimização.

Numa variante mais simples, o algoritmo pode ter seus parâmetros iniciais w_0 inicializados aleatoriamente, o coeficiente de aprendizado η fixado no início do procedimento e ser parado depois de um número fixo de passos. Inúmeras variantes deste procedimento são possíveis, como uso de coeficientes de aprendizado variáveis (como no Adam), diversas condições de parada mais refinadas ou inicialização dos parâmetros a partir de uma distribuição de probabilidades específica.

Esta estratégia de otimização pode ser justificada matematicamente como uma aproximação local da função objetivo usando uma série de Taylor centrada em w_t, porém truncada no segundo termo, como dada na Equação 2.7. Ou seja, a vizinhança de $f(w_t)$ é aproximada por um plano com inclinação igual ao gradiente da função, logo a direção que minimiza esta aproximação é dada pelo próprio vetor gradiente negativo, porém o erro da aproximação aumenta conforme se distancia de w_t, o que torna necessário uma atualização recorrente desta aproximação e cria um algoritmo iterativo.

$$f(w_t + \epsilon) \approx f(w_t) + \epsilon.\nabla_w f(w_t) \tag{2.7}$$

Por completude, a definição da série de Taylor de $f(x)$ centrada em a é dada na Equação 2.8, onde $f^{(n)}(x)$ é a n-ésima derivada da função e $n!$ é o fatorial de n (WEISSTEIN).

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{2.8}$$

O gradiente descente não tem nenhuma garantia convergência para o ótimo global da função objetivo caso a função não seja convexa, porém ótimos locais podem ser suficientemente bons para serem usados em um dado problema (ABU-MOSTAFA *et al.*, 2012; GOODFELLOW *et al.*, 2016).

2.4 *Deep learning*

A qualidade dos algoritmos de aprendizado depende diretamente de como o problema é apresentado para o modelo. Se é desejado atacar uma tarefa usando o arcabouço do aprendizado supervisionado, tipicamente o problema deve ser apresentado de forma

Sentenças	**Representação**						
	cheio	copo	está	meio	o	ou	vazio
"O copo meio cheio."	1	1	0	1	1	0	0
"O copo está meio cheio ou meio vazio?"	1	1	1	2	1	1	1

Tabela 2.1: Exemplo de representação vetorial pelo método do *bag-of-words*

vetorial.

Exemplificando, a forma mais simples de se representar uma informação textual para um problema de classificação de textos é usando a chamada representação em ***bag-of-words***, na qual cada dimensão do vetor resultante irá representar a frequência de uma palavra específica do vocabulário conhecido pelo modelo (BIRD *et al.*, 2009). Desta forma, se é dada uma frase que possui cinco palavras, sendo que uma delas se repete três vezes, então a representação desta frase será um vetor em que todos os elementos serão zero, menos nas três dimensões que representam as palavras existentes, onde a palavra repetida será representada por um valor numérico maior que as demais. Um caso é exemplificado na Tabela 2.1.

A representação usada para um problema de aprendizado vai impactar nas potencialidades e nas limitações de uma solução. O *bag-of-word*, por exemplo, não mantém nenhuma informação a respeito da ordem das palavras. O usuário do algoritmo pode trabalhar nesta entrada de dados para melhorar seu modelo. Isso pode ser feito de várias formas, como agregando outras informações pertinentes ao problema ou facilitando o trabalho do algoritmo de aprendizado, por meio de procedimentos como a normalização das entradas. Porém em domínios como a classificação de imagens, esse manuseio pode não ser trivial.

Deep learning, ou aprendizado profundo, se refere a um conjunto de técnicas e modelos de aprendizado de máquina que buscam minimizar a necessidade dessa influência do usuário na codificação do problema. Em geral, modelos de *deep learning* aprendem representações de conceitos mais complexos a partir de representações de conceitos mais simples (GOODFELLOW *et al.*, 2016). Por exemplo, no caso de se dizer se uma imagem tem algum rosto humano, o modelo poderia entender que uma face é caracterizada pela presença de diversos elementos, dentre eles o nariz, que por sua vez é entendido partir de várias possíveis combinações de arestas que ocorrem nos pixels da imagem.

Essa construção hierárquica de conceitos pode ser pensada na forma de uma composição de funções, como ilustrado na Figura 2.2. Neste caso, para saber a resposta y dada pelo modelo para um caso qualquer x, é necessária a resposta da função f, que aprendeu a responder ao problema usando as funções d e e como subsídio. A função d, por sua vez, precisa da informação obtida pelas funções a e b para dar sua resposta etc. Estas composições podem se tornar bem complexas dependendo

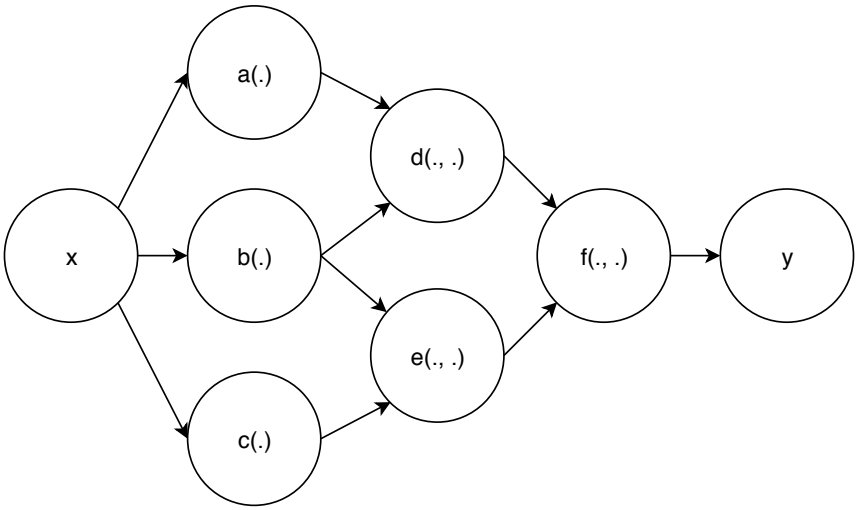

Figura 2.2: Exemplo de grafo computacional de funções

do tamanho do modelo, mas podem ser representadas computacionalmente como **grafos**.

Em sua essência, estes conceitos apresentados se remetem ao trabalho de RUMELHART *et al.* (1985), que fundamentou o aprendizado automático de representações internas em um modelo via o algoritmo de ***backpropagation***. O método é necessário pois mesmo que os dados indiquem a resposta esperada para uma dada entrada, eles podem não indicar o que esperar das funções intermediárias. O *backpropagation* aceita qualquer composição de funções matemáticas como modelo, desde de que sejam diferenciáveis, e consiste basicamente no uso recursivo da regra da cadeia para se calcular o gradiente com respeito a cada um dos parâmetros a serem otimizados. Este gradiente será usado por métodos como o gradiente descendente para ajustar o modelo completo aos dados. Pode se pensar que, dado um grafo computacional, o *backpropagation* volta por suas arestas compondo um novo grafo computacional, a partir da regra da cadeia, para o gradiente de um dado parâmetro.

Modelos de *deep learning* se tornaram muito comuns, especialmente com a criação de grandes conjuntos de dados de alta qualidade, como o ImageNet (DENG *et al.*, 2009), com inicialmente 3.2 milhões de imagens, ou o próprio SNLI (BOWMAN *et al.*, 2015), com mais de 400 mil pares de sentenças. Grandes *datasets* são necessários para se ajustar os parâmetros dos modelos de *deep learning* suficientemente para ultrapassarem outras abordagens. Outro fator importante também foi o aumento do poder computacional disponível em conjunto com o surgimento de bibliotecas computacionais algébricas com capacidade de diferenciação automática e uso transparente de aceleração via GPUs (Graphics Processing Units), como Theano (THEANO DEVELOPMENT TEAM, 2016) e Tensorflow (ABADI *et al.*, 2015). Modelos de *deep learning* são atualmente o estado da arte em diversas tarefas complexas, como categorização de imagens (SABOUR *et al.*, 2017) e tradução de máquina (VASWANI

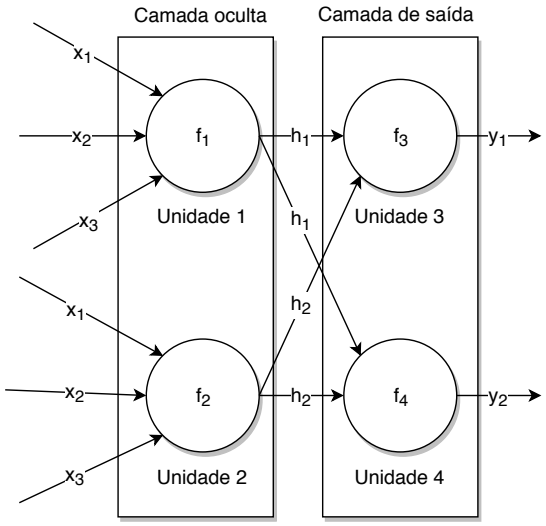

Figura 2.3: Exemplo de uma rede neural simples

et al., 2017).

2.4.1 Redes neurais artificiais

O principal modelo das técnicas de *deep learning* são as chamadas **redes neurais artificiais**, *multilayer perceptron* (MLP) ou *feedforward networks* (GOODFELLOW *et al.*, 2016). O nome vem de uma inspiração biológica: a rede neural seria composta de um grande agregado de neurônios, capazes de receber diversos estímulos, pondera-los e propagar seu próprio estímulo para outros neurônios. Tais neurônios, ou **unidades**, fazem parte uma estrutura hierárquica formada de **camadas**. A Figura 2.3 exemplifica uma rede neural com três entradas, x_1, x_2 e x_3, duas unidades na primeira camada, que é interna ao modelo e por isso chamada de **camada oculta**, e duas unidades na camada de saída, que funcionarão como as respostas do modelo.

Cada unidade de uma rede neural pode ser descrita matematicamente como uma combinação linear de sua entrada seguida de uma não-linearidade, como na Equação 2.9, onde x é um vetor que define a entrada da função, w e b são um vetor e um escalar, respectivamente, de parâmetros ajustáveis durante a otimização e $g(.)$ é uma função não-linear crescente, classicamente chamada de **função de ativação**.

$$f(x) = g(w^T x + b) \tag{2.9}$$

Funções de ativação não-lineares são necessárias quando se deseja utilizar mais de uma camada destas unidades. Isso se deve ao fato de que sem usar uma função destas, existiria uma simplificação matemática equivalente a uma única camada, devido a linearidade destas unidades. Atualmente, a função ReLU (*Rectified Linear Unit*), dada simplesmente por $g(x) = max(x, 0)$, é a não-linearidade preferível para

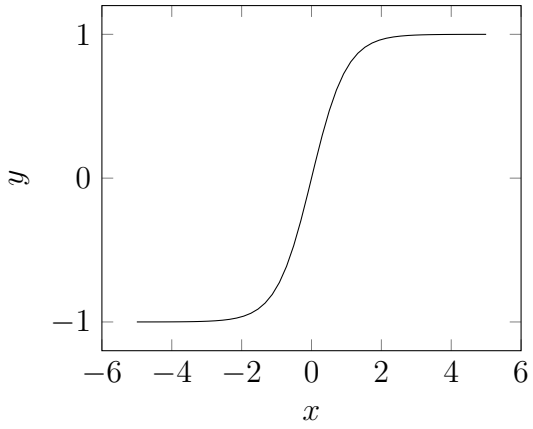

Figura 2.4: Curva da tangente hiperbólica

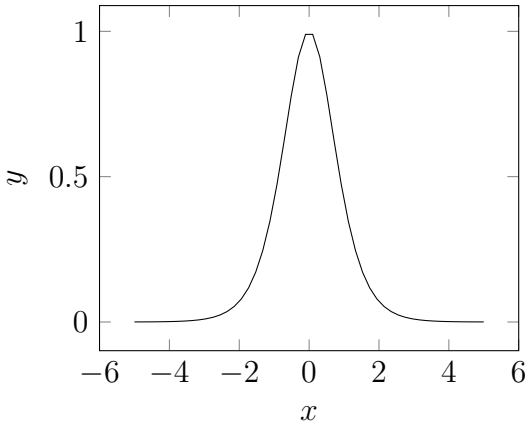

Figura 2.5: Derivada de $tanh(x)$

os modelos modernos (GOODFELLOW *et al.*, 2016).

Uma camada inteira destas unidades pode ser facilmente descrita partindo da Equação 2.9, substituindo w por uma matriz de pesos e b por um vetor. Uma **camada densa** é uma camada destas unidades em todas as unidades utilizam como entrada todas as informações, ou "estímulos", providos pela camada anterior. Composições de pelo menos uma camada densa e um modelo linear são conhecidos aproximadores universais (HORNIK *et al.*, 1989). Mesmo com a possibilidade de se usar funções quaisquer para se definir modelos arbitrários, algumas construções acabam funcionando como blocos lógicos básicos, devido especialmente a sua conhecida efetividade em trabalhar com determinados aspectos de um problema e auxiliar na representação interna desenvolvida pelo modelo. A camada densa é o principal bloco usado em *deep learning* e é o pilar fundamental de toda a área.

2.4.2 *Vanishing gradients*

As funções tangente hiperbólica ($g(x) = \tanh(x)$) e sigmoide ($g(x) = \frac{1}{1+exp(-x)}$) foram classicamente usadas como não-linearidade para redes neurais porém elas tendem a apresentar problemas numéricos durante o processo de treinamento quando o modelo possui muitas camadas. As Figuras 2.4, 2.6 e 2.8 abaixo ajudam a visualizar as três funções de ativação já mencionadas com suas respectivas derivadas.

Funções com gradientes que assumem valores absolutos muito altos ou muito próximos de zero vão propagar gradientes que numericamente explodem ou mínguam multiplicativamente a cada camada do modelo, devido a aplicação recursiva da regra da cadeia no *backpropagation*. Este problema, chamado de ***vanishing gradients*** (ou *exploding gradients*) torna a propagação do erro para os parâmetros em camadas mais distantes da saída ou muito errática ou saturada em zero (CANALLI e SILVA, 2017).

A tangente hiperbólica e a sigmoide ocasionam este problema devido ao fato

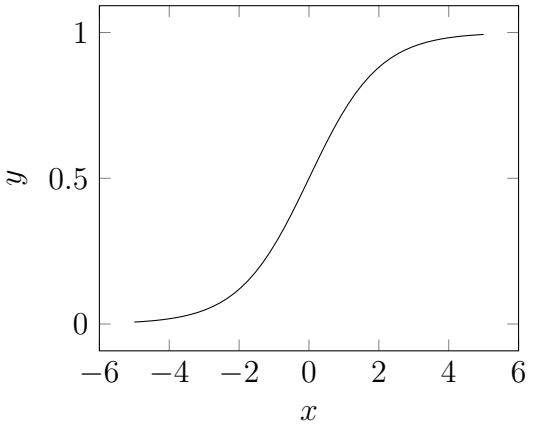

Figura 2.6: Curva da função sigmoide

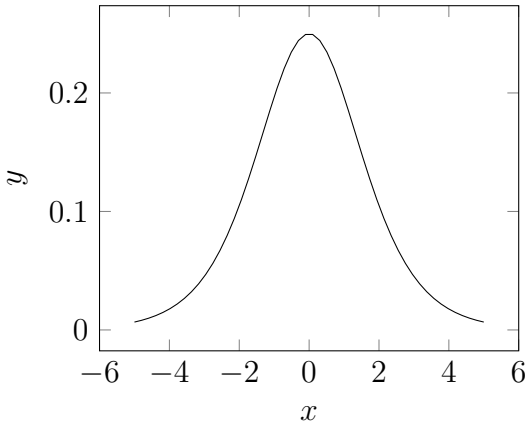

Figura 2.7: Derivada da sigmoide

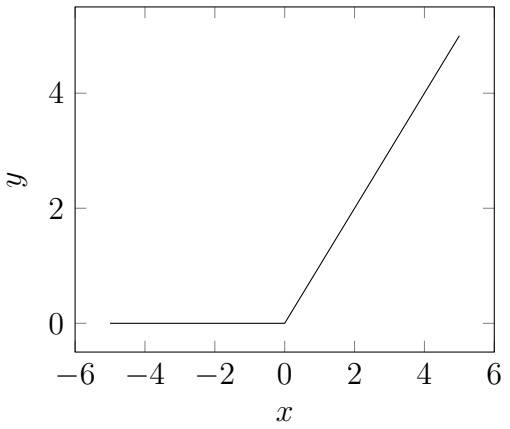

Figura 2.8: Curva da função ReLU

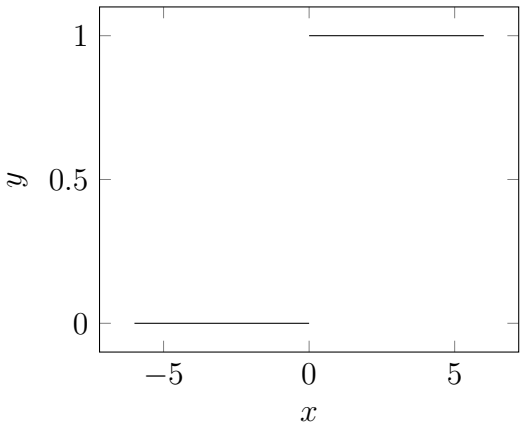

Figura 2.9: Derivada da ReLU

de que suas derivadas tendem a zero conforme seus valores crescem ou diminuem, enquanto a função ReLU (*Rectified Linear Unit*) satura somente em uma das direções.

2.5 Processamento de linguagem natural usando *deep learning*

Se o processamento de linguagem natural é a subárea da ciência da computação que estuda manipulações algorítmicas da linguagem natural, e esta, por sua vez, é uma complexa e, às vezes, subjetiva construção humana, então o uso do aprendizado de máquina é uma ferramenta natural para se atacar suas diversas tarefas, como já mencionado. Porém, para esta aplicação ser possível, é necessário que os diversos componentes da linguagem, como palavras e sentenças, sejam representados idealmente como vetores numéricos.

Novamente, a construção desta representação vetorial vai impactar diretamente a eficácia dos algoritmos utilizados. Pode-se tomar como exemplo o caso de uma tarefa de classificação de texto, na qual é utilizada uma representação em *bag-of-words* junto com um modelo linear para se decidir se o texto em questão faz parte ou não de uma categoria de interesse. Nesta situação, cada palavra do vocabulário conhecido pelo algoritmo possuirá um parâmetro único que pesará o impacto de sua presença na predição final, porém como as dimensões/palavras são totalmente independentes, o modelo provavelmente não vai conseguir generalizar bem em casos com palavras mais raras, mesmo existindo diversas sentenças com outras palavras que sejam sintaticamente ou semanticamente similares, como "escrever" e "escrito" ou "cachorro" e "gato".

Outro problema desta forma de se representar textos é a perda da informação de ordem das palavras, impossibilitando que o algoritmo consiga pesar expressões como "ciência da computação" de forma diferenciada das palavras individuais "ciência" e "computação". Uma forma de minimizar este problema é usando as representações em *n-grams* (BIRD *et al.*, 2009), que podem ser pensados como uma generalização do *bag-of-words* em que as dimensões representam tuplas de n palavras adjacentes em vez das palavras individuais: a frase "O cachorro correu atrás do gato" seria representada em bigramas ($n = 2$) como "O+cachorro cachorro+correu correu+atrás atrás+do do+gato". Um problema dessa estratégia é que conforme n aumenta, o número de dimensões na representação vetorial cresce muito rapidamente, o que fatalmente atrapalha a capacidade de generalização dos modelos: como cada tupla específica aparecerá poucas vezes, o algoritmo terá menos chances de apurar a capacidade preditiva da presença daquele termo para o problema, configurando na chamada **maldição da dimensionalidade**.

2.5.1 Modelos de linguagem

As técnicas do que hoje chamamos de *deep learning* se mostraram interessantes para o processamento de linguagem natural em trabalhos como o de BENGIO *et al.* (2003), na qual se buscava criar **modelos de linguagem** escaláveis usando redes neurais, porém atacando o problema da maldição da dimensionalidade através da chamada **representação distribuída** de palavras.

Modelos de linguagem podem ser explicados resumidamente como a tarefa de se prever qual é a próxima palavra de uma sentença, dado as palavras anteriores. Modelos de linguagem podem ser descritos matematicamente como na Equação 2.10, baseado em BENGIO *et al.* (2003), onde S_n é a variável que representa uma sequência de n palavras e w_i representa a i-ésima palavra da sequência.

$$P(S_n) = \prod_{i=1}^{n} P(w_i|S_{i-1}) \tag{2.10}$$

Este problema é aplicável por si só, como em corretores de texto ou teclados de *smartphones*. Modelos de linguagem também podem ser modificados e usados em diversos problemas que envolvem converter uma sequência em outra, como tradução de máquina ou sumarização abstrativa (RUSH *et al.*, 2015). Esta variante é dada na Equação 2.11, onde D_n representa a sequência de destino com n termos, O_m é a sequência de origem com m termos e w_i é a i-ésima palavra da sequência de destino.

$$P(D_n) = \prod_{i=1}^{n} P(w_i|D_{i-1}; O_m) \tag{2.11}$$

Modelos de linguagem podem ser usados em conjunto com métodos como o algoritmo de Viterbi ou o *beam search* para selecionar a sequência mais provável dentre diversas possibilidades de escolha naquele contexto, e então formar a sentença desejada, como o sumário ou a tradução (RUSH *et al.*, 2015).

2.5.2 Representação distribuída

A representação distribuída de palavras consiste em atribuir vetores em $\mathbb{R}^n$ para cada palavra do vocabulário conhecido pelo modelo, que é tipicamente muito maior que n. No experimento original de BENGIO *et al.* (2003), $n \in \{30, 60, 100\}$ e os vetores de palavras foram ajustados junto com o modelo de linguagem usando um grande corpus de texto. A ideia dessa abordagem é que se as palavras "gato" e "cachorro" tenham papeis sintáticos e/ou semânticos similares, então suas representações vetoriais devem ser similares, em termos de distância entre os vetores, de forma com que mesmo que cachorro apareça poucas vezes no conjunto de dados, ainda é possível que o modelo generalize para casos em que a palavra gato ocorreu, diferentemente da representação

por *bag-of-words*.

A viabilidade da representação distribuída vem novamente da hipótese distribucional (HARRIS, 1954), que sugere que o uso de grandes volumes de textos permitiria ao algoritmo observar vários contextos para as diversas palavras e ajustar os vetores para refletir seus significados. A representação distribuída de palavras, também chamada de ***word embedding***, é usada como um dos componentes iniciais de vários modelos modernos para diversas tarefas em processamento de linguagem natural, especialmente utilizando vetores pré-treinados em grandes volumes de textos obtidos da internet, como o GloVe (PENNINGTON *et al.*, 2014), que é disponibilizado em [1].

2.5.3 Redes recorrentes

Para se incorporar informação sobre a sequência de palavras que formam uma sentença numa representação vetorial, podem ser utilizadas **redes recorrentes**. Estas redes são caracterizadas pela sua retroalimentação, ou seja, sua resposta atual depende tanto de sua entrada corrente quanto do seu estado anterior, o que pode ser sumarizado na Equação 2.12, considerando que tanto a entrada quanto a saída são sequências com um mesmo tamanho, onde $r(.)$ é a função que representa a rede recorrente, x_t é o t-ésimo passo da sequência e h_t é o t-ésimo vetor produzido pela rede recorrente, chamado as vezes de *hidden state* (GOODFELLOW *et al.*, 2016).

$$h_t = r(x_t, h_{t-1}) \tag{2.12}$$

Redes recorrentes são interessante pois, por exemplo, dada as sequências [“copo”, “meio”, “cheio”] e [“copo”, “muito”, “cheio”], a representação vetorial h_3, do elemento “cheio”, produzida por uma rede recorrente na segunda sequência será diferente da representação h_3 na primeira sequência, já que o contexto foi diferente. Desta forma, os vetores criados para uma sequência teriam acumulado informação de todos os passos anteriores e, bastaria agregar, de alguma forma, estes vetores em um único para usá-lo como a representação final da sequência.

Redes recursivas podem ser treinadas usando o método do ***backpropagation through time***, que funciona de forma análoga ao *backpropagation* usual porém usando uma espécie de expansão, ou desdobramento, da rede original na qual se repetem os trechos recorrentes usando os mesmos pesos para cada elemento da sequência, o que possibilitaria uma rede final de profundidade arbitrária. A Figura 2.10 ilustra esta rede desdobrada comparada com a rede original.

[1] `https://nlp.stanford.edu/projects/glove/`

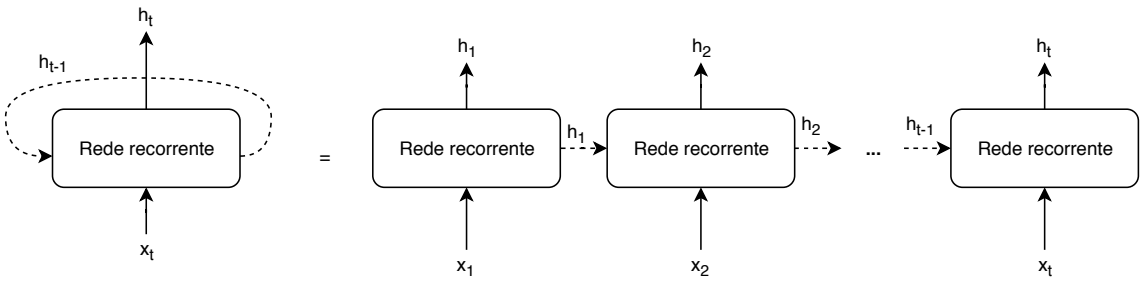

Figura 2.10: Rede recursiva desdobrada, baseada em OLAH, CHRISTOPHER (2015)

2.5.4 LSTM

Essa característica de profundidade arbitrária pode ser problemática devido ao já mencionado problema dos *vanishing gradients*, atrapalhando o aprendizado de situações em que se depende de informações muito anteriores na sequência.

Tendo esta situação em vista, as **unidades LSTM** (*long short-term memory*) foram criadas para facilitar o aprendizado em longas sequências. Introduzidas por HOCHREITER e SCHMIDHUBER (1997), as LSTMs foram modificadas ao longo do tempo e a variante usada atualmente foi descrita originalmente em GRAVES e SCHMIDHUBER (2005), segundo GREFF *et al.* (2017). Uma unidade LSTM pode ser encarada como uma espécie de versão diferenciável dos chips de memória usuais, com três "portas lógicas" que controlam o fluxo de informação pela unidade (GRAVES e SCHMIDHUBER, 2005).

A *output gate* é a primeira destas portas lógicas e seria análoga a porta de leitura nos circuitos lógicos, controlando a informação que vai ser propagada da memória interna para fora da unidade. A *output gate* é usada diretamente no cálculo dos vetores h_t produzido uma camada de unidades LSTM, que é dado na Equação 2.13.

$$h_t = o_t \odot \tanh(C_t) \tag{2.13}$$

O operador $\odot$ denota a multiplicação elemento a elemento de dois vetores de tamanhos iguais. C_t é o estado atual da chamada **célula de memória** da unidade enquanto que o_t é o comando dado pelo **output gate**. Este comando do *output gate* para o passo atual é uma função tanto da entrada corrente x_t quanto do *hidden state* anterior h_{t-1} da unidade, dada pela Equação 2.14.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.14}$$

A notação $[.,.]$ representa uma concatenação de vetores, σ é a função sigmoide $\sigma(x) = \frac{1}{1+exp(-x)}$ aplicada elemento a elemento, W_o e b_o são parâmetros aprendidos. Todas as "portas lógicas" das unidades LSTMs seguem este padrão e podem ser entendidas como uma simples camada densa que dá instruções que vão desde ignorar

completamente um valor na operação atual ou usá-lo definitivamente. Isso é feito através de suas respostas entre zero e um, normalizadas pela função sigmoide.

O valor da célula de memória corrente C_t é calculado decidindo o que manter da memória anterior C_{t-1} e o que aproveitar do novo candidato a memória $\tilde{C}_t$. Essa ponderação é feita por meio do ***forget gate*** f_t e do ***input gate*** i_t, e é resumida na Equação 2.15.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.15}$$

As *input gate* e *forget gate* são análogas às portas lógicas de escrita e *reset* nos circuitos de memória e são dadas pelas Equações 2.16 e 2.17, respectivamente, seguindo a mesma notação anterior.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.16}$$

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.17}$$

Por fim, o candidato a nova célula de memória $\tilde{C}_t$ pode ser entendido como uma função que decide qual informação será extraída da entrada atual x_t, também levando em conta o estado passado h_{t-1}. Isso é representado na forma de uma camada densa e dado pela Equação 2.18. Ambas funções tanh(.) usadas na definição das unidades LSTM poderiam ser substituídas por outra função de ativação.

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.18}$$

2.5.5 Codificador de sentenças

Unidades LSTM e *word embedding* são comumente combinadas para se obter uma representação vetorial de sentenças que agrega tanto a informação da sequência de palavras quanto o próprio significado de cada palavra. Vários detalhes ainda devem ser definidos, mas a combinação destes dois componentes forma um codificador de sentenças básico, que pode ser adaptado para diversas tarefas. Por exemplo, uma classificação de texto pode ser feita simplesmente usando este codificador e, em seguida, uma camada densa e um classificador linear.

São inúmeras as variantes de configuração deste codificador básico de texto, um caso simples é definir se os vetores de palavras são ajustáveis ou fixos durante o treinamento, o que seria útil em situações com *datasets* menores para reduzir o número de parâmetros aprendidos. Outro detalhe interessante é se o LSTM lerá a sentença da esquerda para a direita, da direita para a esquerda ou de ambas as formas (bidirecional). Um terceiro problema seria como agregar os vetores criados

pelo LSTM, já que a sequência pode ter um tamanho variado, mas a representação vetorial deve ter um tamanho fixo. Isto pode ser feito usando operações de soma ou média a cada dimensão, ou simplesmente usando somente o vetor gerado para o último elemento da sequência, que conteria informação agregada de todos os demais passos.

2.6 Inferência textual

Como já explicado, a inferência textual (*natural language inference* ou *textual entailment*) é uma tarefa que busca avaliar isoladamente a capacidade de compreensão semântica dos algoritmos (ANDROUTSOPOULOS e MALAKASIOTIS, 2010). A maioria dos trabalhos se foca na questão do reconhecimento de inferências textuais, que tipicamente toma a forma de um problema de classificação de pares de textos. O tipo de classificação envolvida neste problema variou ao longo de seu desenvolvimento, mas existem dois casos principais: a classificação binária entre inferência e não-inferência (DAGAN *et al.*, 2006) e a classificação de múltiplas classes entre inferência, contradição ou neutralidade (BOWMAN *et al.*, 2015).

Como o próprio nome sugere, o problema do reconhecimento de inferência textual pode ser entendido a partir da inferência lógica: se as sentenças da premissa e da hipótese forem representadas por formulações lógicas, como a lógica de primeira ordem, o problema pode visto como uma implicação lógica. Com a premissa simbolizada por P e a hipótese por H, identificar se ocorre uma inferência textual pode ser analisado a partir da avaliação do valor verdade da proposição $(P \wedge B) \rightarrow H$, onde B representa um banco de conhecimento (ANDROUTSOPOULOS e MALAKASIOTIS, 2010). De forma textual, quer se responder à pergunta: se a sentença P é aceita como verdade, então a sentença H também tem que ser verdadeira?

Observa-se que a relação de inferência textual não é simétrica, ou seja, a proposição $(P \wedge B) \rightarrow H$ ser verdadeira não necessariamente implica que $(H \wedge B) \rightarrow T$ também é verdade. Quando ambos casos ocorrem, pode-se dizer que existe uma relação de paráfrases (ANDROUTSOPOULOS e MALAKASIOTIS, 2010).

2.6.1 Aplicações

Além da função de avaliar a compreensão semântica dos algoritmos, o reconhecimento de inferência textual por si só tem algumas aplicações interessantes, exemplificadas por ANDROUTSOPOULOS e MALAKASIOTIS (2010). Como já dito na introdução, um sistema usado nesta tarefa pode ser aplicado em uma sumarização extrativa, como uma forma de remover as sentenças que podem ser inferidas a partir de outras encontradas no conjunto de documentos.

Outra aplicação seria responder perguntas automaticamente: a pergunta feita pode ser transformada em uma ou mais afirmações falseáveis e o classificador de inferências seria então usado para localizar, em textos de referência, pelo menos uma sentença que pode ser usada para inferir uma das afirmações geradas. Outra aplicação nesta linha é avaliar se respostas dadas em provas podem ser inferidas a partir de um dado gabarito de referência, afim de avaliar as respostas.

Variantes do próprio problema de inferência textual também existem, apesar de menos estabelecidas, como a geração de inferências textuais, explorada em KOLESNYK *et al.* (2016), na qual novas sentenças são produzidas a partir de uma frase original. Estas novas sentenças são tipicamente paráfrases ou simplificações dos originais, algo que pode ser aplicado em casos de sumarização abstrativa.

2.6.2 Conjunto de dados PASCAL

O conjunto de dados PASCAL lançado em DAGAN *et al.* (2006) foi uma das primeiras tentativas de se formalizar uma tarefa clara para o problema de inferência textual. Com apenas 567 exemplos no conjunto de desenvolvimento e 800 no conjunto de teste, o *dataset* tem por objetivo representar sentenças características de diferentes tipos de atividades que poderiam se aproveitar de um modelo genérico para inferências semânticas: tradução de máquina, recuperação da informação, resposta automática, comparação de documentos, compreensão de texto e aquisição de paráfrases. O *dataset* tem por objetivo criar uma mensuração unificada para as diversas aplicações, avaliando vários níveis de raciocínios baseados em informações léxicas, sintáticas, lógicas ou conhecimento de mundo.

Os pares do PASCAL possuem apenas duas possíveis respostas: se a hipótese pode ser inferida a partir do texto dado ou não, o que agruparia os casos em que uma sentença claramente contradiz a outra com casos em que a relação entre as sentenças é dúbia. Seis exemplos do conjunto de teste são exibidos na Tabela 2.2 com seus respectivos tipos de tarefa indicados de forma resumida. Os dados foram obtidos de [2].

Os pares de sentenças para o PASCAL foram coletadas de forma específica para cada uma das seis aplicações visadas pelos autores e o processo de anotação foi feito por dois avaliadores independentemente, na qual os casos de não concordância foram removidos, junto com pares que os autores consideraram controversos. Alguns outros grupos independentes avaliaram partes do *dataset* final, chegando a taxas de concordância acima da faixa dos 90%. O conjunto de dados PASCAL de 2006 foi apenas o primeiro realizado nesses moldes, outras versões maiores e aprimoradas da tarefa de reconhecimento de inferência textual surgiram ao longo dos anos, como em

[2] `http://www.cs.biu.ac.il/~nlp/RTE1/Datasets/annotated_test.zip`

Classificação (Tipo)	**Premissa e Hipótese**
Verdadeiro	Mexico City has a very bad pollution problem because the mountains around the city act as walls and block in dust and smog.
(Comparação)	Poor air circulation out of the mountain-walled Mexico City aggravates pollution.
Falso	Coyote shot after biting girl in Vanier Park.
(Recuperação)	Girl shot in park.
Verdadeiro	Mark Said and Philip Dick, two members of Ontario Bradley First Air's crew, were from Montreal, Canada.
(Extração)	Mark Said works for Ontario Bradley First Air.
Falso	Iowa has lost 26,000 manufacturing jobs since Bush took office.
(Tradução)	Iowa is lost.
Verdadeiro	AOL has more than 33 million paying customers
(Compreensão)	33 million customers pay to use AOL.
Falso	The Ploce mayor Josko Damic spoke of the first Croatian president.
(Resposta)	Josko Damic was the first Croatian president

Tabela 2.2: Exemplos do conjunto de teste do PASCAL

BAR HAIM *et al.* (2006); GIAMPICCOLO *et al.* (2007).

Para atacar o problema proposto no conjunto de dados PASCAL, os sistemas classificadores utilizaram diferentes estratégias para processar as sentenças como contagem de sobreposição de palavras, lógica de primeira ordem ou pareamento sintático. A decisão final poderia ser dada por modelos probabilísticos, técnicas de aprendizado supervisionado, provadores lógicos ou diferentes método de pontuação, como descrito em DAGAN *et al.* (2006).

Técnicas que se baseiam em sobreposição de palavras usam alguma medida de similaridade de texto, como a distância de edição de Levenshtein (LEVENSHTEIN, 1966), para mensurar a diferença entre as duas sentenças, possivelmente após a aplicação de algum passo de pré-processamento mas tipicamente sem se utilizar de representações sintáticas ou semânticas mais elaboradas (ANDROUTSOPOULOS e MALAKASIOTIS, 2010).

A utilização de inferência lógica consiste em mapear as sentenças para uma representação lógica e então utilizar um provador de teoremas em conjunto com um banco de dados de conhecimento comum para tentar provar ou contradizer a proposição de que a representação lógica da premissa implica na representação lógica da hipótese (ANDROUTSOPOULOS e MALAKASIOTIS, 2010).

A estratégia de pareamento sintático consiste em representar a estrutura sintática das sentenças por meio de grafos, tipicamente uma árvore de dependência, que podem então ser comparadas simplesmente contando o número de arestas em comum ou com alguma medida mais elaborada de similaridade entre árvores (ANDROUTSO-

POULOS e MALAKASIOTIS, 2010).

Mais concretamente, um dos primeiros trabalhos a usar o PASCAL foi BOS e MARKERT (2005), que combinou diversas das técnicas mencionadas para resolver o problema. O melhor modelo apresentado pelos autores utiliza um classificador que combina o que chamaram de indicativos semânticos rasos e profundos (*shallow semantic features* e *deep semantic features*). Os indicativos rasos consistem basicamente numa contagem ponderada de palavras em comum, na qual se usou a base de conhecimento WordNet para identificar palavras com significados análogos.

Os indicativos semânticos profundos, por outro lado, utilizam um analisador semântico para se obter uma representação das sentenças em lógica de primeira ordem. As representações lógicas são então usadas com provadores de teoremas para identificar inferências (provando que $A \rightarrow B$) ou inconsistências (provando que $\neg(A \wedge B)$), em conjunto com bases de conhecimento, como o já mencionado WordNet. Essas representações também são usadas com construtores de modelos lógicos, que os autores usaram para tentar encontrar um modelo lógico que contradiz a entrada (na forma $\neg A \rightarrow B$). O classificador final inclui atributos gerados a partir de ambos tipos de indicativos e a qual tipo de tarefa o par de sentenças pertence, obtendo 61.2% de acurácia no PASCAL.

2.6.3 Conjunto de dados SNLI

Tendo em vista problemas passados enfrentados pelo PASCAL e outros conjuntos de dados, como tamanho pequeno dos *datasets*, sentenças geradas por algoritmos ou problemas de anotação relacionados a indeterminação entre eventos e entidades, o conjunto de dados **SNLI** (*Stanford Natural Language Inference*), disponibilizados por BOWMAN *et al.* (2015), tenta superar tais situações. O SNLI é o primeiro conjunto de larga escala produzido por *crowdsorcing*, contendo mais de 500 mil pares de sentenças e categorizado em três classes: inferência, contradição e neutralidade.

As sentenças usadas como premissa para o SNLI provêm de descrições de imagens, na qual a multidão é instruída, sem acesso ao cenário original, a criar três novas sentenças, uma que deve necessariamente descrever aquela mesma situação, outra que pode descrever aquele cenário e uma terceira que necessariamente não descreve. Com essa abordagem, os autores buscaram ganhar exemplos mais diversos para o conjunto de dados e também limitar problemas envolvendo ambiguidade entre os eventos e entidades mencionados, já que o cenário descrito deve ser o mesmo. Seis pares do conjunto de validação do SNLI são exemplificados na Tabela 2.3. O *dataset* completo e os valores de acurácia obtidos por diversos modelos neste *dataset* podem ser encontrados em [3].

[3] `https://nlp.stanford.edu/projects/snli/`

Classificação	Premissa e Hipótese
Inferência	Two women are embracing while holding to go packages.
	Two woman are holding packages.
Contradição	A young female toddler wearing a yellow dress is seated in a swing.
	A toddler is going down a slide.
Neutralidade	A woman in a red hat waiting in line.
	The woman is waiting to order a sandwich.
Inferência	A little girl is sitting on the counter dangling one foot in the sink whilst holding a dish jet washer.
	A human sitting.
Contradição	A woman wearing a white shirt and jeans crosses a rope bridge in a wooded area.
	A woman wearing a black shirt and khakis crosses a rope bridge in a wooded area.
Neutralidade	Protesters in costumes carry signs through a paved city street.
	The protestors wore costumes to relay a serious message.

Tabela 2.3: Exemplos do conjunto de validação do SNLI

O *dataset* exibe uma taxa de concordância de 91,2% entre a etiqueta atribuída pelo autor da sentença e a etiqueta majoritária de cinco anotadores independentes numa amostragem dos dados. Este fato, junto com outras medidas de concordância, sugerem que tal método foi efetivo em criar um conjunto de dados de boa qualidade utilizando *crowdsourcing*.

O SNLI apresentou alguns modelos *baseline* para o conjunto de dados: um classificador baseado em atributos léxicos e três modelos baseados em codificação de sentenças. O primeiro classificador utiliza vários atributos extraídos manualmente, ou seja, a representação vetorial do par de sentenças não foi aprendida autonomamente a partir de dados brutos. Alguns exemplos dos atributos usados são a medida BLEU da hipótese com respeito a premissa, a diferença entre tamanhos das sentenças, o número de palavras em comum e a representação em *bag-of-word* e bigramas da hipótese. O modelo de aprendizado de máquina específico não foi mencionado, mas esta abordagem sozinha obteve 78.2 % de acurácia no conjunto de testes (BOWMAN *et al.*, 2015). Os demais modelos são discutidos na Sessão 2.6.6.

2.6.4 Conjunto de dados ASSIN

No contexto da língua portuguesa, para o problema de reconhecimento de inferência textual, foi disponibilizado o conjunto de dados **ASSIN** (Avaliação de Similaridade Semântica e Inferência Textual), disponibilizado em [4]. Este conjunto possui 10 mil pares de sentenças obtidos de notícias e divididos igualmente entre Português do Brasil

[4] `http://propor2016.di.fc.ul.pt/?page_id=381`

e de Portugal. Ele foi desenvolvido para ser usado tanto para a inferência textual quanto para a comparação de similaridades semânticas. No caso da inferência textual, o conjunto de dados possui três classes diferentes do SNLI: inferência, paráfrase e neutralidade. Segundo a apresentação dos autores em FONSECA *et al.* (2016), contradições quase não ocorreram no processo de coleta. Exemplos do conjunto de validação do ASSIN são exibidos na Tabela 2.4, junto com sua respectiva variante do português.

Classificação (Variante)	**Premissa e Hipótese**
Inferência (Brasil)	A divergência aberta por Toffoli foi seguida pelos ministros Gilmar Mendes, Marco Aurélio Mello e Teori Zavascki.
	Acompanharam a posição defendida por Dias Toffoli os ministros Gilmar Mendes e Marco Aurélio.
Paráfrase (Brasil)	Os demais agentes públicos serão alocados na classe econômica.
	Todo o resto dos funcionários públicos terá que embarcar na classe econômica.
Neutralidade (Brasil)	A Polícia Militar estima o número de pessoas em 700.
	De acordo com o movimento, cerca de 500 pessoas ocuparam o local.
Inferência (Portugal)	De acordo com as marcas, há em Portugal 117 mil veículos equipados com o software fraudulento.
	No total, existem cerca de 117 mil veículos em Portugal que têm o sistema fraudulento incorporado.
Paráfrase (Portugal)	O comício do partido estava a acontecer no Coliseu dos Recreios quando os episódios de violência aconteceram.
	As agressões ocorreram durante o comício do partido no Coliseu dos Recreios.
Neutralidade (Portugal)	Apenas 21 países reconhecem o governo em Taiwan.
	Nem Portugal nem nenhum país europeu reconhecem a soberania de Taiwan.

Tabela 2.4: Exemplos do conjunto de validação do ASSIN

O processo de coleta do ASSIN envolveu o uso do algoritmo de agrupamento LDA para se localizar automaticamente grupos de sentenças potencialmente relacionadas. Os candidatos a pares do *dataset* foram selecionados dos grupos encontrados pelo LDA e então filtrados, removendo pares muito similares, com muita sobreposição de palavras, ou muito longos, além de algumas situações não desejadas como cotações de moedas, resultados de jogos ou até sentenças sem sentido fora de contexto. Os pares

obtidos foram então anotados por quatro pessoas diferentes, com algumas poucas instruções para reduzir a subjetividade do julgamento. Pares com menos do que três votos concordantes foram descartados, o que removeu um pouco mais de 10% das sentenças (FONSECA *et al.*, 2016).

O conjunto de sentenças final obteve uma taxa de concordância de cerca de 80% e possui 7 316 pares neutros, 2 080 pares com inferências e 604 paráfrases. O método *baseline* testado consistiu de uma regressão logística com dois atributos: a proporção de palavras exclusiva da premissa e a proporção exclusiva da hipótese. A regressão logística obteve 82.27 % de acurácia para a inferência textual, superando todos os demais algoritmos competidores, além da classe majoritária, que já obtém 73.47% de acurácia. Segundo os autores, este resultado foi inesperado, já que os participantes tinham boas estratégias e foi realizado um esforço para se remover sentenças com muitas palavras iguais, o que deveria reduzir a efetividade deste tipo de estratégia baseada em contagem de sobreposições de palavras (FONSECA *et al.*, 2016).

2.6.5 Conjunto de dados MultiNLI

O conjunto de dados MultiNLI (*Multi-Genre Natural Language Inference*) de WILLIAMS *et al.* (2018), busca manter as características similares ao SNLI, porém superando sua grande limitação de apenas possuir sentenças baseadas em descrições de cenários. Os autores fizeram isso adicionando diversos estilos textuais, graus de formalismo e assuntos, como transcrições de conversas, guias de viagem e textos de ficção, com o intuito de incorporar novos desafios para a tarefa, como a necessidade de raciocínio temporal e modalidade linguística, relevantes para a compreensão textual em geral. Alguns pares de sentenças do conjunto de validação do MultiNLI são demonstrados na Tabela 2.5 com seu respectivo gênero linguístico, como descrito pelos autores. Os dados podem ser obtidos de [5]

Com instruções para a multidão similares às do SNLI, especialmente no que diz respeito a escrever sobre a mesma situação ou evento da sentença dada, o MultiNLI tem 392.702 pares de sentenças com cinco categorias literárias no conjunto de treinamento. O *dataset* obtém uma taxa de concordância de 92,6% entre a classificação original do autor da sentença e a categoria escolhida pelos anotadores durante a validação, similar ao trabalho anterior. Outro ponto interessante do MultiNLI é a disponibilização de dois conjuntos de validação: um contendo somente os cinco gêneros linguísticos que ocorrem no conjunto de treinamento (chamado de *Matched set*) e outro conjunto de validação com outros cinco gêneros que não aparecem no treinamento (*Mismatched set*). Esta medida tem o intuito de avaliar a capacidade dos modelos de generalizarem para outras situações diversas ainda

[5] `https://www.nyu.edu/projects/bowman/multinli/`

Classificação (Gênero)	**Premissa e Hipótese**
Inferência (*fiction*)	You and your friends are not welcome here, said Severn.
	Severn said the people were not welcome there.
Neutralidade (*government*)	The share of gross national saving used to replace depreciated capital has increased over the past 40 years.
	Gross national saving was highest this year.
Contradição (*telephone*)	i don't know um do you do a lot of camping
	I know exactly.
Inferência (*travel*)	The entire city was surrounded by open countryside with a scattering of small villages.
	The whole countryside is scattered with small villages.
Neutralidade (*fiction*)	Fixing current levels of damage would be impossible.
	The damage could never be fixed by an artisan.
Contradição (*slate*)	What's truly striking, though, is that Jobs has never really let this idea go.
	Jobs never held onto an idea for long.

Tabela 2.5: Exemplos do conjunto de validação *Matched* do MultiNLI

não vistas. WILLIAMS *et al.* (2018) analisa três modelos *baselines* neste *dataset*: dois baseados em codificação de sentenças e um modelo estado da arte no SNLI, discutidos a seguir.

2.6.6 Modelos baseados em codificação

Alguns dos classificadores *baseline* apresentado por BOWMAN *et al.* (2015) e WILLIAMS *et al.* (2018) são baseados em codificação de sentenças. Neles, representações vetoriais da premissa e da hipótese são geradas pelos ditos codificadores, que são então combinadas de alguma forma e usados por num classificador para o veredito final sobre o par de sentenças.

No SNLI, em especial, os vetores gerados pelos codificadores para cada sentença são concatenados e aplicados numa sequência de três camadas densas com 200 dimensões e um classificador linear de três classes. A Figura 2.11 representa esquematicamente este modelo.

A arquitetura usada no MultiNLI é similar a esta, porém se usou apenas uma camada densa e a etapa de combinação de vetores consiste em concatenar quatro vetores: o vetor da premissa, o vetor da hipótese, a diferença entre ambos vetores e seu produto, calculado elemento a elemento dos vetores. Esta etapa foi explorada inicialmente em MOU *et al.* (2015) e se provou uma forma simples de melhorar a performance deste tipo de modelo.

Em BOWMAN *et al.* (2015), os autores testaram três variantes dos codificadores de sentença. Todos eles começam obtendo a representação vetorial das palavras

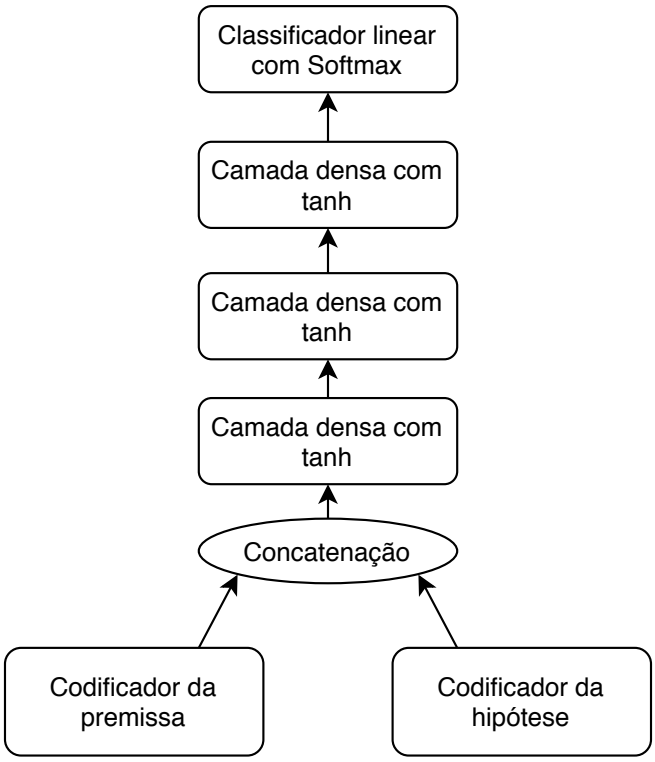

Figura 2.11: Visão esquemática do modelo *baseline* do SNLI.

individuais, na qual os vetores GloVe foram usados como ponto de partida, porém a forma de agregação dos vetores numa representação final da própria sentença varia. A primeira variante é uma simples soma, onde se conserva o número de dimensões do vetor de palavras. A segunda variante é uma rede recorrente clássica enquanto a terceira é baseada em unidades LSTM.

Em WILLIAMS *et al.* (2018), os autores repetem a primeira abordagem, baseada em soma, porém usando sua arquitetura. Este método foi chamado de CBOW (*continuous bag-of-words*) durante o texto, numa alusão à combinação das técnicas de *bag-of-words* e da representação distribuída. Eles também fizeram uso de unidades LSTM, porém na variante bidirecional (GRAVES e SCHMIDHUBER, 2005), na qual a sequência é lida tanto da esquerda para a direita quanto da direita para a esquerda e seus estados ocultos são concatenados.

Uma possível formalização do LSTM bidirecional pode ser vista na Equação 2.19, na qual as unidades são aplicadas numa sequência de tamanho N. O n-ésimo vetor gerado pelo conjunto como um todo é representado por $\overleftrightarrow{LSTM}_n$ no lado esquerdo da equação. A operação de concatenação é descrita como $[.,.]$ e $\overrightarrow{LSTM}_n$ representa o vetor gerado após a leitura do n-ésimo elemento da sequência por unidades lendo da esquerda para a direita, enquanto $\overleftarrow{LSTM}_{N-n}$ é seu análogo que lê a sequência invertida. Os vetores gerados pelo LSTM bidirecional em WILLIAMS *et al.* (2018) são agregados num vetor único por meio de uma soma elemento a elemento para se gerar uma representação final da sentença completa.

$$\overleftrightarrow{LSTM}_n = [\overrightarrow{LSTM}_n, \overleftarrow{LSTM}_{N-n}] \tag{2.19}$$

A abordagem baseada em soma de vetores obteve uma acurácia 75.3% no SNLI, enquanto a rede recorrente clássica e a LSTM obtiveram, respectivamente, 72.2% e 77.6% de acurácia no conjunto de testes. Apesar dos resultados levemente inferiores ao modelo baseado em atributos léxicos, os autores argumentam que o modelo LSTM parece ganhar acurácia mais rapidamente conforme o tamanho do conjunto de treinamento aumenta, além de possuir uma discrepância menor entre as acurácias no conjunto de teste e treino. Tais evidências sugeririam que modelos de codificação de sentenças podem superar os classificadores baseados em atributos manuais, usando variantes com maior capacidade de aprendizado e/ou treinadas com mais dados (BOWMAN *et al.*, 2015).

No caso do MultiNLI, o modelo CBOW apresentou acurácia de 64.8% no conjunto de validação *Matched* e 64.5% no *Mismatched*. Já o LSTM bidirecional obteve 66.9% em ambos subconjuntos de validação. Tais valores sugerem que ambos modelos não se especializaram indevidamente nos gêneros linguísticos do conjunto de treinamento. Quando são comparados estes valores com a acurácia obtida no SNLI, 80.6% para o CBOW e 81.5% para o LSTM bidirecional, observa-se uma queda absoluta de 14.6% no melhor caso. Tal variação sugere que o MultiNLI é um desafio muito maior do que o SNLI para os algoritmos, que ainda estão bem longes da taxa de concordância humana na faixa dos 90%.

2.6.7 Modelos baseados em atenção

Com o surgimento do SNLI, modelos complexos puderam começar a ser usados para atacar o problema da inferência textual. Notoriamente, ainda em 2015, foi publicado o trabalho de ROCKTÄSCHEL *et al.* (2016), que se aproveita de unidades de LSTM em conjunto com um mecanismo de atenção entre as sentenças para modelar mais precisamente as interações entre as palavras de cada sentença.

Mecanismos de atenção criam uma espécie de "pareamento suave" (*soft-alignment*) dos elementos de uma sequência em função dos elementos da outra sequência, o que permitiria destacar as palavras mais importantes de uma sentença em relação a uma dada palavra da outra sentença. A atenção também pode ser pensada como uma agregação ponderada de todos os elementos de uma dada sequência, na qual o próprio modelo decide explicitamente qual informação é mais relevante para uma dada tarefa.

O modelo de ROCKTÄSCHEL *et al.* (2016) primeiramente utiliza um codificador com unidades LSTMs para representar a premissa e um outro para representar a hipótese, porém o primeiro estado oculto do segundo codificador (de hipóteses)

é iniciado com o último estado oculto do primeiro codificador, o que os autores chamaram de codificação condicional. Então, estas codificações vetoriais são usadas por um mecanismo de atenção palavra a palavra para gerar o vetor usado pelo classificador final. Este vetor final é definido na Equação 2.20, com uma notação ligeiramente modificada do original, para melhor compreensão.

$$h_* = \tanh(W_{h^*}[r_N, h_N]) \tag{2.20}$$

O vetor $h_* \in \mathbb{R}^k$ denota a representação final de tamanho k gerada pelo modelo (no experimento original, $k = 100$), que é dado em função de uma camada densa, parametrizada por W_{h^*}, porém sem o termo adicional $+b$, ou seja, uma projeção matricial. Novamente, $[.,.]$ representa uma concatenação de vetores. A representação leva em conta tanto o último estado oculto da hipótese, $h_N \in \mathbb{R}^k$, quanto a última representação "atencional" $r_N \in \mathbb{R}^k$ da hipótese, considerando que ela tenha N palavras.

$$r_t = Y\alpha_t^T + \tanh(W_r r_{t-1}) \tag{2.21}$$

A "representação atencional" do t-ésimo elemento da hipótese $r_t \in \mathbb{R}^k$ é dado tanto em função da representação anterior r_{t-1} quanto das palavras da premissa, representada pela matriz $Y \in \mathbb{R}^{k \times L}$, formada de vetores concatenados provenientes do codificador da premissa para suas L palavras. Esta representação é parametrizada por W_r e também utiliza o vetor de pesos $\alpha_t \in \mathbb{R}^L$, que contém os pesos da atenção para a t-ésima palavra da hipótese, e que são usados como uma ponderação das palavras da premissa Y.

$$\alpha_t = softmax(w_\alpha^T M_t) \tag{2.22}$$

O vetor de pesos "atencionais" α_t é parametrizado pelo vetor w_α e normalizado com a função $softmax$, que força sua soma ser um. Este vetor de pesos é uma simples multiplicação entre o parâmetro aprendido w_α e a matriz $M_t \in \mathbb{R}^{k \times L}$, que contém as representações modificadas da premissa para o t-ésimo passo de processamento da hipótese. M_t é definida na Equação 2.23.

$$M_t = \tanh\left(W_y Y + (W_h[h_t, r_{t-1}]) \otimes e_L\right) \tag{2.23}$$

A matriz M_t é caracterizada pela junção das representações da premissa Y, da t-ésima palavra da hipótese h_t e a representação "atencional" anterior r_{t-1}. Usando a notação dos autores, e_L é um vetor de tamanho L preenchido com valores 1 e $\otimes$ é um operador de produto externo. A operação $(. \otimes e_L)$ representaria efetivamente a criação de uma matriz que repete L vezes o vetor proveniente do esquerdo da

operação.

Por fim, a representação agregada h_* é usada em um classificador linear de três classes. Esta abordagem chega ao patamar de 83,5% de acurácia, ultrapassando em mais de 5% os classificadores usados como *baseline* do SNLI, tanto o baseado somente em unidades LSTM quanto o baseado em diversos descritores léxicos.

Atualmente, um dos modelos estado da arte para o SNLI é o ESIM (*Enhanced Sequential Inference Model*), proposto por CHEN *et al.* (2016). Ele tem uma complexa arquitetura envolvendo o uso de unidades de LSTM, tanto para codificar as sentenças quanto em etapas posteriores, em conjunto com um mecanismo de atenção. O ESIM obteve 88.6% de acurácia no SNLI, próximo à taxa de concordância humana neste *dataset*. Ele foi o último modelo *baseline* avaliado no MultiNLI, obtendo uma acurácia de 72.3% no conjunto de validação *Matched* e 72.1% no *Mismatched*, novamente evidenciando o aumento de dificuldade proposto pelo MultiNLI.

2.6.8 Categorias dos modelos

Uma importante distinção pode ser feita neste ponto, entre os modelos utilizados na tarefa de inferência textual: os baseados em codificação de sentenças e os que incorporam outras estratégias, como mecanismos de atenção. Os modelos baseados em codificação buscam representar internamente suas sentenças em vetores de tamanho fixo, sem qualquer influência da outra sentença do par, e então combinam ambos vetores para se realizar a classificação final, como sugerido em NANGIA *et al.* (2017).

Essa abordagem é interessante pela possibilidade de reutilização desses codificadores em outros problemas de processamento de linguagem natural, especialmente em casos com conjunto de dados menores, como já explorado em CONNEAU *et al.* (2017). Um exemplo desta categoria é o modelo *baseline* do SNLI que utiliza unidades LSTM para codificar as sentenças individualmente.

Apesar de sua possibilidade de adaptação para outros problemas, modelos baseados em codificação de sentenças tendem a ter um desempenho inferior na própria tarefa de inferência textual se comparados com outros modelos que utilizam formas de condicionar a codificação de uma sentença com seu par, ou seja, a representação da hipótese vai depender da premissa e/ou vice-versa. Esta discrepância entre as categorias de modelos e um panorama do estado da arte pode ser visualizada na Figura 2.12, com dados provenientes e adaptados da própria página do SNLI.

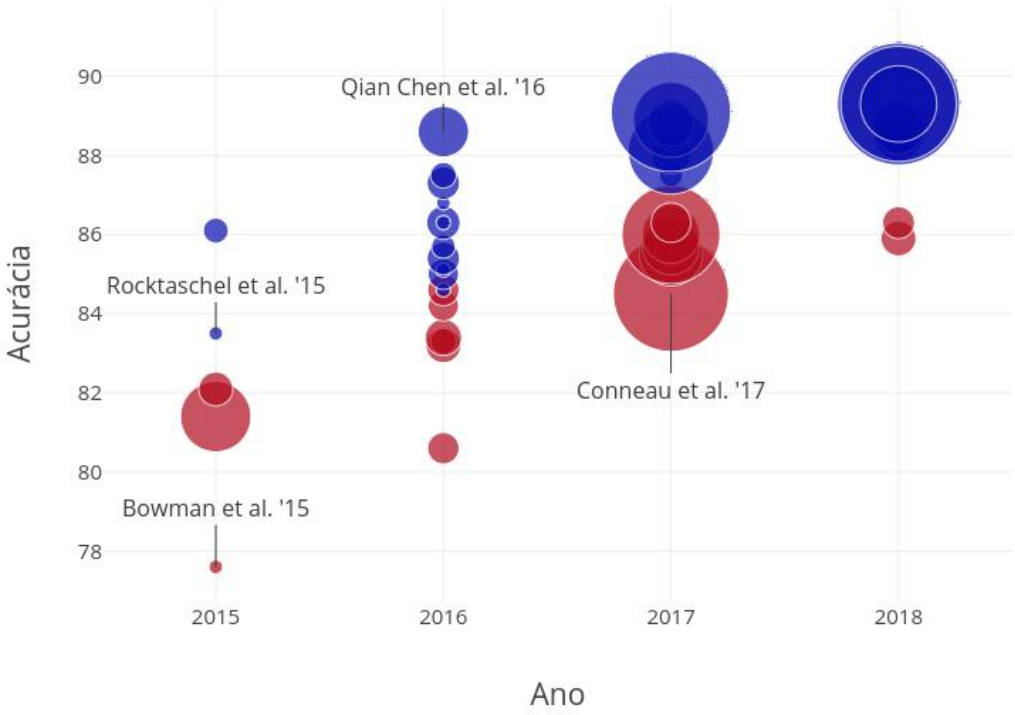

Figura 2.12: Acurácia no *dataset* SNLI dos diversos trabalhos

Na figura, modelos baseados em codificação são apresentados em vermelho e o diâmetro dos círculos é proporcional ao número de parâmetros usados em cada modelo. A diferença entre ambas as categorias fica bem clara quando se analisa a discrepância entre os melhores resultados de cada categoria por ano. Também é possível se observar uma estagnação dos modelos, que já estão bem próximos à taxa de concordância humana neste *dataset*, próxima de 90%, além de um crescimento no número de parâmetros usados. Esta estagnação também permite que modelos baseados em codificação se aproximem, em termos de acurácia, dos demais modelos.

2.6.9 Vieses

Apesar do grande desenvolvimento proporcionado pela disponibilidade do SNLI e do MultiNLI, o estudo de GURURANGAN *et al.* (2018) levanta alguns vieses desses *datasets*: os autores mostram indícios de que os trabalhadores que contribuíram durante a coleta dos dados utilizaram algumas heurísticas para a criação das sentenças usadas como hipótese. Esses vieses, tais como muita utilização de termos negativos (não, nunca, ninguém, ...) nos exemplos de contradição ou termos genéricos (animal, instrumento, ...) nos casos de inferência, são altamente exploradas pelos modelos atuais.

Categoria	**Premissa e Hipótese**
Antônimos	I love the Cinderella story.
	I hate the Cinderella story.
Raciocínio numérico	Tim has 350 pounds of cement in 100, 50, and 25 pound bags.
	Tim has less than 750 pounds of cement in 100, 50, and 25 pound bags.
Sobreposição de palavras	Possibly no other country has had such a turbulent history.
	The country's history has been turbulent and true is true.
Negação	Possibly no other country has had such a turbulent history.
	The country's history has been turbulent and false is not true.
Diferença de tamanho	Possibly no other country has had such a turbulent history and true is true and true is true and true is true and true is true and true is true .
	The country's history has been turbulent.
Erro de ortografia	As he emerged, Boris remarked, glancing up at teh clock: You are early.
	Boris had just arrived at the rendezvous when he appeared.

Tabela 2.6: Exemplos das sentenças usadas como teste de estresse, baseado em NAIK *et al.* (2018)

A análise consistiu em treinar um classificador de prateleira para aprender a tarefa de inferência textual utilizando somente as sentenças da hipótese como entrada. Tal classificador obteve 67% de acerto no conjunto de testes do SNLI, 53,9% para o MultiNLI no conjunto *matched* e 52.4% para o *mismatched*.

Quando três diferentes modelos, dentre eles o ESIM, são avaliados utilizando somente os pares de sentenças que esse classificador acertou, observa-se taxas de acerto na faixa de 92% para o SNLI e na faixa de 86% para ambos os conjuntos de teste do MultiNLI, porém quando os mesmos modelos são avaliados nos pares em que o classificador errou, existe uma perda de mais de 12%, com relação ao conjunto de teste completo, em todos os modelos e *datasets*.

As evidências apresentadas sugerem que de fato os modelos atuais estão aproveitando os vieses inseridos pela multidão. Os autores apontam que ainda se permanece em aberto a questão de como criar tais *datasets*, ao mesmo tempo que se evita esses vieses, já que as heurísticas utilizadas ainda correspondem a fenômenos linguísticos válidos.

Outro trabalho recente que também tenta avaliar os possíveis problemas dos modelos atuais é NAIK *et al.* (2018). Nele, os autores criaram um *dataset* artificial, baseado no MultiNLI, para realizarem "testes de estresse" nos próprios modelos. Os pares deste conjunto de dados buscam avaliar e quantificar algumas possíveis fraquezas baseadas em distrações ou fenômenos linguísticos como antônimos, negação, raciocínio numérico, erros gramaticais e tautologias. Exemplos podem ser encontrados na Tabela 2.6 junto com as categorias dadas pelos autores.

A queda de acurácia nos testes sugeridos, em relação ao conjunto de validação original, é notória para diversos modelos testados. Por exemplo, diversos modelos com mais de 70% de acurácia no MultiNLI caem para cerca de 15% em testes com antônimos, atingem um patamar de 30% para casos que exigem raciocínio numérico e ficam próximas de 50% de acurácia em distrações envolvendo negações.

Um último caso interessante é o já mencionado *dataset* ASSIN (FONSECA *et al.*, 2016), em que uma estratégia simples de contagem de sobreposição de palavras nas sentenças já é capaz de obter resultados acima da taxa de concordância humana, superando até outros métodos mais complexos. O tamanho do conjunto de dados e seu desbalanceamento para as classes de pares neutros e pares de inferências, em menor grau, possivelmente contribuíram para tal fenômeno.

O reconhecimento de inferência textual é um problema de natureza difícil e em aberto, porém importante para o contexto do processamento de linguagem natural. Novos métodos de avaliação e modelos precisam ser desenvolvidos tendo em vista evitar tais problemas e vieses com o objetivo de tornar os algoritmos aplicáveis em situações reais.

Capítulo 3

Técnicas utilizadas

3.1 Lógica *Fuzzy*

A lógica binária, ou lógica booleana, é um ramo da matemática caracterizado pela limitação que suas variáveis assumam apenas um de dois valores possíveis: verdadeiro ou falso. Ela está intimamente ligada com a teoria dos conjuntos, na qual um dado elemento pertence ou não pertence a um conjunto específico. Por outro lado, a lógica difusa, lógica nebulosa ou lógica *fuzzy* (ZADEH, 1965), interpreta suas variáveis como graus de pertencimento a um determinado conjunto. Desta forma, em vez de uma variável representar se um copo pertence ou não ao grupo dos copos cheio, como numa lógica binária, ela passa a representar um grau de pertencimento ao grupo dos copos cheios e um grau de pertencimento ao grupo dos copos vazios.

A lógica *fuzzy* permite uma codificação de incertezas e misturas de forma natural, e são aplicados expressivamente em sistemas de controle, como em MAMDANI (1974), na qual permitem que um operador humano possa definir uma estratégia de controle de forma vaga a partir de regras linguísticas como:

"Se a velocidade é média e a distância até o obstáculo é distante então a mudança de direção é pequena".

Neste exemplo, "velocidade", "distância até o obstáculo" e "mudança de direção" são chamadas **variáveis linguísticas** e as palavras "média", "distante" e "pequena" são os **termos linguísticos**. O interessante desta abordagem é que informações precisas do mundo real, como um sensor dizendo que existe um obstáculo a 100 metros podem ser "fuzzificadas" para termos vagos e analisadas pelas regras dadas, assumindo que "o obstáculo está 0.45 próximo e 0.2 distante". Desta forma, todas as regras que definem o que fazer quando os obstáculos estão "distantes" ou "próximos" seriam avaliadas e, por fim, uma agregação ponderada destas regras pode ser usada para "defuzzificar" a variável linguística "mudança de direção" numa definição numérica precisa como "29 graus".

A lógica *fuzzy* é definida a partir dos chamados conjuntos *fuzzy*. Estes conjuntos difusos são caracterizados pelo uso de uma função de pertinência (*membership function*) que define o grau de pertencimento, entre zero e um, de um dado elemento ao conjunto difuso em questão. Matematicamente, uma função de pertinência pode ser definida em 3.1, onde μ_A é a função para o conjunto A e o domínio da função é representado por X.

$$\mu_A : X \to [0, 1] \tag{3.1}$$

Esta característica permite um elemento pertencer apenas parcialmente ao conjunto e, assim, modelar incertezas ou misturas. Conjuntos *fuzzy* podem ser pensados como uma extensão dos conjuntos matemáticos usuais, por vezes chamados como conjuntos nítidos, onde a função de pertinência deve necessariamente assumir ou o valor zero ou um para qualquer elemento.

As operações básicas dos conjuntos nítidos como união, intercessão e complemento podem ser redefinidas para os conjuntos *fuzzy*, permitindo sua manipulação. Algumas operações *fuzzy* comuns são definidas abaixo como em ZADEH (1965), porém são inúmeras as possíveis definições de cada operador.

$$C = A \cup B \leftrightarrow \mu_C(x) = \max\{\mu_A(x), \mu_B(x)\}, x \in X \tag{3.2}$$

$$C = A \cap B \leftrightarrow \mu_C(x) = \min\{\mu_A(x), \mu_B(x)\}, x \in X \tag{3.3}$$

$$C = \bar{A} \leftrightarrow \mu_C(x) = 1 - \mu_A(x), x \in X \tag{3.4}$$

Observa-se que tais operadores tem a desejável propriedade de se comportarem como as operações de conjuntos nítidos em situações que as funções de pertinência são binárias ($X \in \{0, 1\}$).

Assim como as funções de pertencimento dos conjuntos nítidos podem ser expressos através da lógica booleana clássica, uma lógica *fuzzy* pode ser definida a partir da teoria dos conjuntos *fuzzy*. Desta forma, as operações de disjunção, conjunção e negação da lógica *fuzzy* podem ser definidas pelas suas operações análogas nos conjuntos *fuzzy*: união, intercessão e complemento. Usando novamente os operadores de máximo e mínimo, as operações da lógica *fuzzy* podem ser definidas como:

$$A \vee B = \max\{\mu_A(x), \mu_B(x)\} \tag{3.5}$$

$$A \wedge B = \min\{\mu_A(x), \mu_B(x)\} \tag{3.6}$$

$$\neg A = 1 - \mu_A(x) \tag{3.7}$$

Uma variante destes operadores pode ser obtida definindo-se o operador $\wedge$ da conjunção como uma multiplicação e mantendo-se o operador de negação. A disjunção $\vee$ pode ser derivada usando a lei de De Morgan $\neg(A \wedge B) = \neg A \vee \neg B$. Os operadores da variante multiplicativa da lógica *fuzzy* são resumidos a seguir.

$$A \vee B = \mu_A(x) + \mu_B(x) - \mu_A(x) \times \mu_B(x) \tag{3.8}$$

$$A \wedge B = \mu_A(x) \times \mu_B(x) \tag{3.9}$$

$$\neg A = 1 - \mu_A(x) \tag{3.10}$$

3.2 Regularização

A **regularização** no contexto de aprendizado de máquina são tipicamente restrições ou penalidades usadas tipicamente em conjunto com a função objetivo. O intuito deste método é evitar que o algoritmo de aprendizado chegue em algum conjunto de parâmetros que representam soluções não desejadas para o problema, não generalizando bem para casos fora do conjunto de treinamento, mesmo que o uso da regularização faça com que o modelo erre mais no próprio conjunto de treinamento (GOODFELLOW *et al.*, 2016).

$$\text{objetivo final} = \text{objetivo original} + w \times \text{função de regularização} \tag{3.11}$$

Uma função objetivo regularizada pode ser uma simples soma, como na Equação 3.11, onde w é o peso dessa regularização. Um exemplo clássico do aprendizado supervisionado é a regularização via norma l_2, na qual é usada a soma dos parâmetros do modelo ao quadrado como termo de regularização. Tal termo funciona como uma penalização feita para se evitar que o modelo chegue em soluções que pesam demais alguns poucos fatores. O otimizador, neste caso, será obrigado a encontrar um equilíbrio entre a capacidade de explicar os dados do treinamento, medida possivelmente via entropia cruzada, e essa soma dos próprios valores numéricos dos parâmetros.

Outro exemplo interessante de regularização usada no contexto de *deep learning* é o chamado ***dropout***, proposto em SRIVASTAVA *et al.* (2014). Este método consiste simplesmente em tornar zero a resposta de algumas unidades do modelo (os neurônios,

na analogia de redes neurais) aleatoriamente com uma probabilidade p durante cada passo do treinamento. A ideia é que modelos regularizados com o *dropout* seriam equivalentes a uma espécie de combinação de diversos modelos menores provenientes das 2^n possíveis arquiteturas reduzidas de um modelo com n unidades. A inspiração do método é relacionada a teoria do papel da reprodução sexuada na evolução das espécies, na qual os genes são escolhidos aleatoriamente de ambos pais, o que evitaria que genes resultantes dependessem de um conjunto muito específico de outros, incentivando a permanência de genes que são úteis individualmente ou em colaboração com um conjunto reduzido de outros genes. Os resultados apresentados mostram que o uso *dropout* dá ganhos significativos de acurácia, em relação aos modelos sem seu uso, nas diversas situações testadas, dentre *datasets* de classificação de imagens, reconhecimento de fala e classificação de textos, este último com resultados menos expressivos.

3.3 Lógica matemática e a *semantic loss*

Modelos usados na inteligência artificial para atacar os diversos problemas podem ser categorizados de forma quase dicotômica entre modelos simbólicos e não simbólicos. Modelos simbólicos, baseados em lógica de primeira ordem, expressam seu conhecimento a partir de regras lógicas e tem as características de serem facilmente extensíveis, são interpretáveis e possuem a capacidade de realizar inferências em sequência para obter novos conhecimentos (ROCKTÄSCHEL, 2017). O exemplo abaixo mostra um caso de inferência, na qual a informação de que Alice é neta de Cláudia pode ser obtida a partir do conhecimento existente e das regras disponíveis.

$$Filho(Alice, Bob)$$

$$Filho(Bob, Claudia)$$

$$\forall X \forall Y (Filho(X, Y) \wedge Filho(Y, Z)) \rightarrow Neto(X, Z)$$

Por outro lado, modelos simbólicos não generalizam para novas situações, mesmo que existam similaridades com as situações que ele já trata, e são suscetíveis a erros em sua base de conhecimento, situações em que modelos estatísticos evitam, como já discutido. Porém modelos não-simbólicos, como redes neurais artificias, necessitam de grandes volumes de dados para serem efetivas, não são facilmente interpretáveis ou capazes de incorporar conhecimento existente. Ambos tipos de modelos são complementares, logo obter métodos que aproveitem as melhores características de cada modelo é de grande valor.

Nos trabalhos XU *et al.* (2017, 2018), os autores propõem uma forma de se criar

regularizações a partir de proposições lógicas, visando criar uma ponte entre ambas abordagens da inteligência artificial. Estas proposições são convertidas na chamada *semantic loss*, que pode ser incorporada na função objetivo do método de aprendizado. A *semantic loss* toma a forma apresentada na Equação 3.12.

$$\text{Semantic Loss}(\alpha, p) = -\log\left(\sum_{\forall x \in X}\Big(\prod_{\forall x_i = V} p_i \prod_{\forall x_i = F}(1-p_i)\Big)\right) \tag{3.12}$$

Onde α é uma proposição lógica definida sobre n variáveis binárias, x representa um vetor de uma possível instanciação destas variáveis e X é o conjunto de todas as possíveis instanciações destas variáveis que resultam num caso verdadeiro para a proposição α. O vetor de probabilidades p, que vem do modelo, possui n dimensões que representam cada uma das variáveis usadas por α.

Como exemplo de aplicação da *semantic loss*, considera-se o exemplo da regra "exatamente-um" do próprio trabalho de XU *et al.* (2017): numa situação de classificação com múltiplas classes, somente uma deve prevalecer. Considerando um caso de três classes, a proposição α somente é verdade quando o vetor x assume os valores (V, F, F), (F, V, F) e (F, F, V), onde cada dimensão representa uma classe. Logo a função de regularização deste caso será dada pela Equação 3.13, onde p_i representa a probabilidade, respondida pelo modelo, da observação ser da i-ésima classe.

$$loss = -\log\Big(p_1(1-p_2)(1-p_3) + (1-p_1)p_2(1-p_3) + (1-p_1)(1-p_2)p_3\Big) \tag{3.13}$$

Uma característica deste método é a possibilidade da função de regularização não depender da existência de uma classe anotada, podendo ser usada num contexto semissupervisionado, onde os dados anotados são limitados, porém casos não anotados tendem a ser abundantes. O trabalho de XU *et al.* (2017) apresenta aplicações para o método em algumas tarefas semissupervisionadas, obtendo resultados próximos ou superiores ao estado da arte em condições comparáveis.

Em especial, para o conhecido *dataset* de dígitos MNIST, uma rede densa simples usando a regularização da regra "somente-um" obteve resultados abaixo do melhor modelo em 0.56 % de acurácia (98.94 vs 98.38) para um caso com apenas 100 itens anotados. Esses resultados são bem interessantes tendo em vista que a implementação da *semantic loss* é apenas um termo extra na função objetivo e que ela pode ser utilizada em conjunto com qualquer modelo.

Capítulo 4

Coerência lógica

4.1 Interpretação lógica e a coerência

Como já mencionado, o problema do reconhecimento de inferência textual pode ser entendido a partir da lógica clássica: classificar se um par de sentenças tem uma relação de inferência pode ser descrito como avaliar o valor verdade de $(P \wedge B) \rightarrow H$, onde P é a representação lógica de premissa, H representa a hipótese e B é a base de conhecimento do método. Esta proposição lógica será simplificada para $P \rightarrow H$ sem perda de generalidade para a análise proposta.

Analogamente, classificar se as sentenças são contraditórias, seguindo o esquema de três classes dos *datasets* atuais (WILLIAMS *et al.*, 2018; BOWMAN *et al.*, 2015), poderia se dar por meio da expressão $P \rightarrow \neg H$. Desta forma, a Tabela 4.1 resume as representações lógicas de cada classe do problema e suas respectivas funções indicativas, que assumem um valor booleano verdadeiro quando o par em questão pertence à sua respectiva classe. O caso da neutralidade não pode ser facilmente descrito com uma lógica binária, mas esta indefinição também não será problemática para a análise.

Função indicativa	Representação lógica
Inferência(P, H)	$P \rightarrow H$
Contradição(P, H)	$P \rightarrow \neg H$
Neutralidade(P, H)	N/A

Tabela 4.1: Resumo das representações lógicas

Considerando que o problema da inferência textual não é simétrico (se é dito que P implica em H, pode ser dúbio também dizer que H implica em P), as representações lógicas apresentadas podem ser usadas para se deduzir algumas restrições sobre as possíveis relações do par de sentenças invertido, mesmo que não seja conhecida a relação verdadeira. As restrições são enumeradas abaixo.

1. Inferência(P, H) $\rightarrow \neg$ Contradição(H, P)
   Em lógica proposicional, isso poderia ser descrito como:

$$(P \rightarrow H) \rightarrow \neg(H \rightarrow \neg P) \tag{4.1}$$

   Prova:

$$(P \rightarrow H) \rightarrow \neg(H \rightarrow \neg P)$$

$$(\neg P \vee H) \rightarrow \neg(\neg H \vee \neg P)$$

$$\neg(\neg P \vee H) \vee (H \wedge P)$$

$$(P \wedge \neg H) \vee (H \wedge P)$$

$$P$$

   Como a premissa P é assumida como verdade quando se avalia uma inferência textual, esta restrição é verdadeira. A tabela verdade dada na Tabela 4.2 explicita as condições para se cumprir esta restrição.

Inferência(P, H)	Contradição(H, P)	Satisfaz a restrição?
F	F	V
F	V	V
V	F	V
V	V	F

Tabela 4.2: Tabela verdade da restrição da inferência.

2. Contradição(P, H) $\leftrightarrow$ Contradição(H, P)
   Ou seja, a contradição é uma relação simétrica. Em lógica proposicional:

$$(P \rightarrow \neg H) \leftrightarrow (H \rightarrow \neg P) \tag{4.2}$$

   Prova:

$$(P \rightarrow \neg H) \leftrightarrow (H \rightarrow \neg P)$$

$$(\neg P \vee \neg H) \leftrightarrow (\neg H \vee \neg P)$$

   Considerando que o operador $\vee$ é comutativo, esta proposição é uma tautologia, o que confirma a restrição. Como na restrição anterior, a tabela verdade presente na Tabela 4.3 permite visualizar quando a restrição é cumprida.

Contradição(P, H)	Contradição(H, P)	Satisfaz a restrição?
F	F	V
F	V	F
V	F	F
V	V	V

Tabela 4.3: Tabela verdade da restrição da contradição.

3. Neutralidade(P, H) $\rightarrow \neg$ Contradição(H, P)
   Como este caso não é modelado com lógica binária, uma prova dedutiva, semelhante às demais, não é possível, porém é fácil observar pela simetria da contradição que esta afirmação é válida: se H contradiz P, então P também deveria contradizer H e não ser neutro [1]. A tabela verdade desta restrição é similar ao caso da inferência:

Neutralidade(P, H)	Contradição(H, P)	Satisfaz a restrição?
F	F	V
F	V	V
V	F	V
V	V	F

Tabela 4.4: Tabela verdade da restrição da neutralidade.

Por fim, define-se que um método foi **coerente**, neste contexto da inferência textual, para um dado par de sentenças P e H se suas respostas para os pares (P, H) e (H, P) cumprirem todas as três restrições apresentadas.

Tais regras, mesmo que expressas num formalismo matemático, representam um raciocínio intuitivo: pode-se dizer que uma pessoa foi incoerente se em algum momento ela julga de forma contraditória duas situações análogas. A característica simétrica da contradição é o ponto principal deste raciocínio para o caso da inferência textual: não seria razoável dizer que A contradiz B e não dizer também que B contradiz A. Nota-se também que a coerência é uma característica da própria resposta de um avaliador, humano ou algorítmico, e independe tanto do conhecimento agregado pelo avaliador quanto da resposta assumida como verdadeira. Por consequência, um método pode, ao mesmo tempo, ter altas taxas de erro e ser totalmente coerente se, por exemplo, sempre responder que os pares de sentenças são contraditórios.

[1] O mesmo raciocínio poderia ser usado no caso da restrição da inferência.

4.2 Regularização semântica

As restrições apresentadas poderiam ser forçadas ou, pelo menos, incentivadas para os modelos de reconhecimento de inferência textual, considerando que modelos mais coerentes são qualitativamente desejáveis, usando o argumento de que as restrições são intuitivas e esperadas por possíveis avaliadores humanos. Esta lógica também se estende para um ponto de vista quantitativo, já que modelos coerentes teriam uma taxa de acurácia potencialmente melhor, mesmo que sem acesso as relações esperadas de todos os pares de sentenças, invertidos ou não, apenas assumindo que os anotadores humanos são idealmente coerentes.

Uma forma de incentivar um comportamento mais coerente num modelo de aprendizado de máquina é através de sua função objetivo, por meio de algum critério de regularização. As restrições podem ser pensadas como uma forma de quantificar se as soluções aprendidas por um modelo são interessantes ou não. Porém, como elas são definidas como funções binárias que usam variáveis também binárias, e os modelos usuais de classificação tendem a trabalhar com valores contínuos e responder com distribuições de probabilidades, é necessário definir uma **função substituta** para se construir uma potencial solução.

Uma forma sistemática de construir a função objetivo a partir de regras lógicas é usando a *semantic loss*, apresentada na Sessão 3.3, que será chamada aqui de **regularização semântica**. Como foram apresentadas três restrições para o problema de inferência textual, a Equação 4.3 resume a função objetivo final.

$$\text{objetivo final} = \text{objetivo}(y, \hat{y}) + w_c.S_c(\hat{y}, \hat{y}') + w_i.S_i(\hat{y}, \hat{y}') + w_n.S_n(\hat{y}, \hat{y}') \quad (4.3)$$

Onde y representa a classe real daquela observação, $\hat{y}$ é a predição do modelo, que é uma distribuição de probabilidade para cada classe e $objetivo(.)$ é a função objetivo primária. Os valores w_c, w_i e w_n são os pesos da regularização e S_c, S_i e S_n são de fato as funções da regularização semântica das restrições da contradição, inferência e neutralidade, respectivamente, onde $\hat{y}'$ denota a avaliação do modelo para o par de sentenças invertido.

Importante observar que as restrições propostas não dependem de um par de sentenças com uma classe bem definida, o que importa é a coerência do modelo em não responder certas classes quando as sentenças são invertidas, dada sua resposta no caso original. Desta forma, o modelo pode ser otimizado de forma semissupervisionada, utilizando tanto os pares originais dos *datasets*, quanto os pares inversos, aumentando virtualmente o conjunto de dados disponíveis com observações que não se sabem a classe correta.

A definição das funções S_c, S_i e S_n partem da Equação 3.12, proveniente da

proposta originária de XU *et al.* (2017). Para detalhar o raciocínio de criação destas funções, pode se considerar a restrição da contradição. Esta restrição possui duas variáveis binárias: Contradição(P, H) e Contradição(H, P), e duas possíveis instanciações em que a restrição é cumprida, como dada na Tabela 4.3. Considerando que a função $C(.)$ seleciona a probabilidade de contradição do vetor de probabilidades, denotado por $\hat{y}$ ou $\hat{y}'$ para um par de sentenças arbitrário e seu inverso, a regularização semântica da contradição para este par é dada pela Equação 4.4.

$$S_c(\hat{y}, \hat{y}') = -\log\Big(C(\hat{y})C(\hat{y}') + (1 - C(\hat{y}))(1 - C(\hat{y}'))\Big) \tag{4.4}$$

Após algumas simplificações algébricas, as duas demais restrições apresentadas anteriormente são expressas nas Equações 4.5 e 4.6, onde $C(.)$, $I(.)$ e $N(.)$ são novamente funções seletoras da probabilidade de contradição, inferência e neutralidade, respectivamente.

$$S_i(\hat{y}, \hat{y}') = -\log\Big(1 - I(\hat{y})C(\hat{y}')\Big) \tag{4.5}$$

$$S_n(\hat{y}, \hat{y}') = -\log\Big(1 - N(\hat{y})C(\hat{y}')\Big) \tag{4.6}$$

As Figuras 4.1 e 4.2 mostram as curvas de valores possíveis dessas funções, desconsiderando o logaritmo da fórmula. Importante observar que os pontos de mínimo das superfícies são equivalentes aos pontos em que a restrição é cumprida totalmente enquanto os pontos de máximo são onde a função lógica assume valor falso, o que valida a viabilidade destas funções como funções objetivos: um método de minimização priorizará as soluções que em que as restrições são cumpridas.

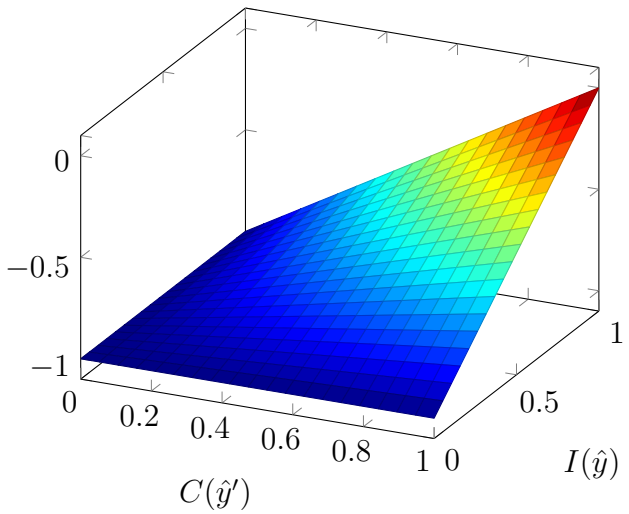

Figura 4.1: Regularização da inferência (ou neutralidade) na definição original.

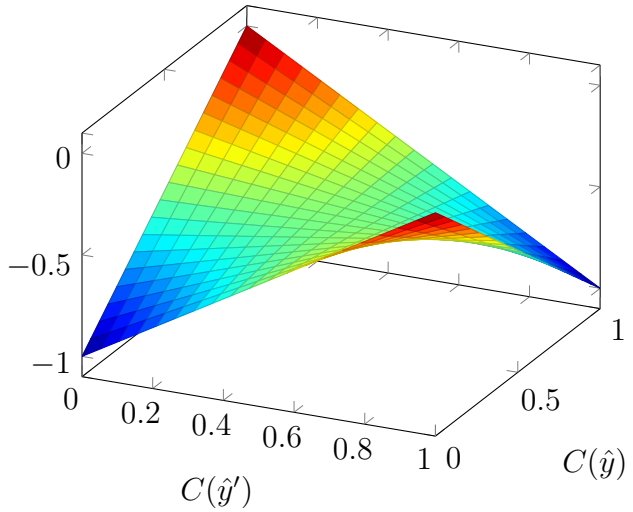

Figura 4.2: Regularização da contradição na definição original.

Uma interpretação interessante para as funções S_c, S_i e S_n é entende-las como uma "fuzzificação" das restrições em lógica binária. As três equações dadas anteriores podem ser interpretadas como uma versão simplificada da lógica difusa baseada em multiplicação, apresentada anteriormente no fim da Sessão 3.1. Uma possível variante destas equações, agora usando os operadores clássicos de mínimo e máximo é dada nas Equações 4.7, 4.8 e 4.9:

$$S_c(\hat{y}, \hat{y}') = -\log\Big(\max\{\min\{C(\hat{y}), C(\hat{y}')\}, \min\{1 - C(\hat{y}), 1 - C(\hat{y}')\}\}\Big) \tag{4.7}$$

$$S_i(\hat{y}, \hat{y}') = -\log\Big(1 - \min\{I(\hat{y}), C(\hat{y}')\}\Big) \tag{4.8}$$

$$S_n(\hat{y}, \hat{y}') = -\log\Big(1 - \min\{N(\hat{y}), C(\hat{y}')\}\Big) \tag{4.9}$$

As Figuras 4.3 e 4.4 novamente mostram as curvas de valores destas funções sem o logaritmo. Os pontos de máximo e mínimo destas curvas têm as mesmas propriedades das curvas das Figuras 4.1 e 4.2 porém, as curvas em si não são suaves, devido as operações de máximo e mínimo.

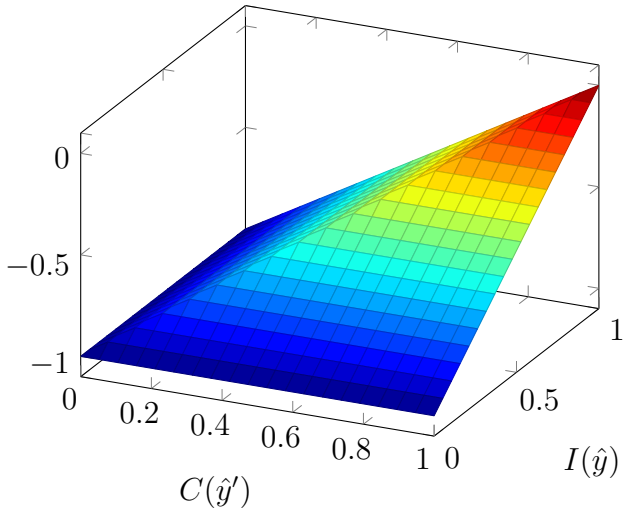

Figura 4.3: Regularização da inferência (ou neutralidade) na versão fuzzy.

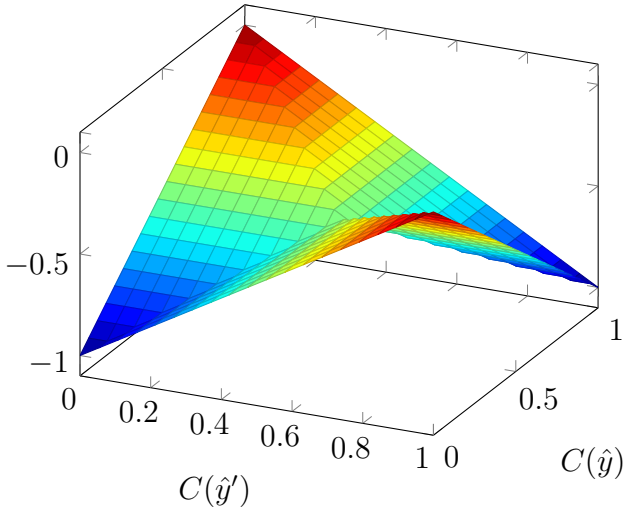

Figura 4.4: Regularização da contradição na versão fuzzy.

Capítulo 5

Implementação

5.1 Modelo *baseline* para a inferência textual

O modelo básico utilizado neste trabalho é baseado diretamente no modelo *baseline* especificado em WILLIAMS *et al.* (2018), escolhido devido à importância teórica de modelos de codificação de sentenças, explicada anteriormente, e sua relativa simplicidade. A arquitetura consiste estruturalmente em três etapas: a codificação das sentenças, a interação entre as representações e o classificador final. O diagrama esquemático da Figura 5.1 dá uma visão geral do modelo usado, onde as setas indicam as dependências de cada operação, retângulos são variáveis fixas, elipses representam operações sem parâmetros treináveis e os blocos arredondados são operações (ou variáveis) com parâmetros treináveis.

A codificação das sentenças, como já explicada, consiste em obter a representação vetorial de cada uma das palavras individuais a partir de uma matriz de *word embeddings*, e transformar esta representação com uma camada de unidades LSTM bidirecionais (GRAVES e SCHMIDHUBER, 2005), visando agregar informação contextual da sequência de palavras da sentença. O resultado da aplicação das LSTMs é uma matriz de dimensões D_{lstm} pelo tamanho da sentença, onde D_{lstm} é a quantidade de unidades utilizada, que então é agregada num vetor único de tamanho D_{lstm} através de um cálculo de média. As unidades LSTMs não têm parâmetro compartilhados entre a premissa e a hipótese, mas a matriz de *word embeddings* é comum entre ambos os codificadores e inicializada com os vetores GloVe (PENNINGTON *et al.*, 2014).

A interação das representações obtidas pelos codificadores é feita segundo especificado por MOU *et al.* (2015), concatenando ambos vetores do passo anterior e mais a diferença e produto elemento a elemento dessas representações, obtendo um vetor final de tamanho $D_{lstm} \times 4$. Por fim, o classificador final é consistido de uma camada densa usando a não-linearidade ReLU e um classificador linear de três

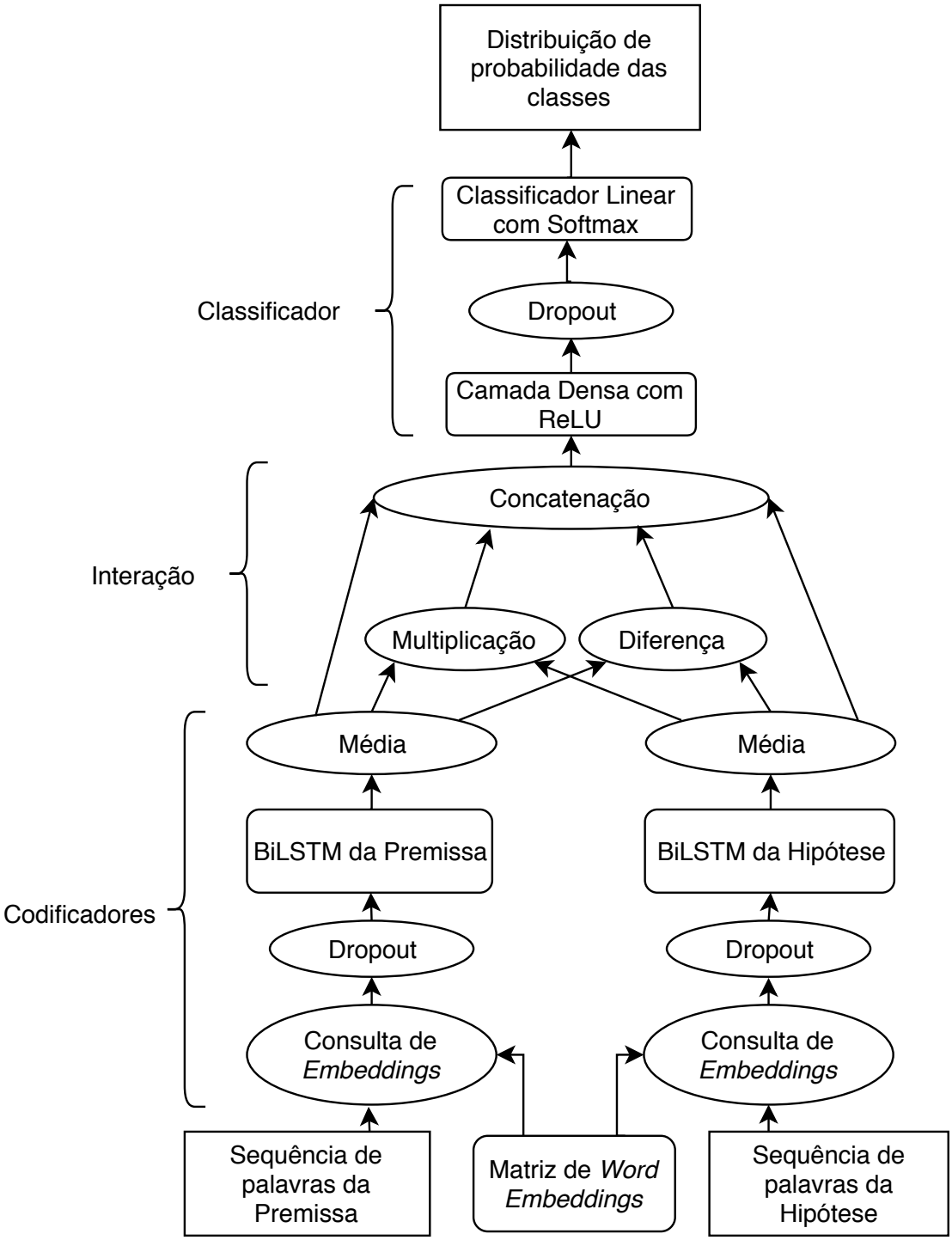

Figura 5.1: Visão esquemática do modelo.

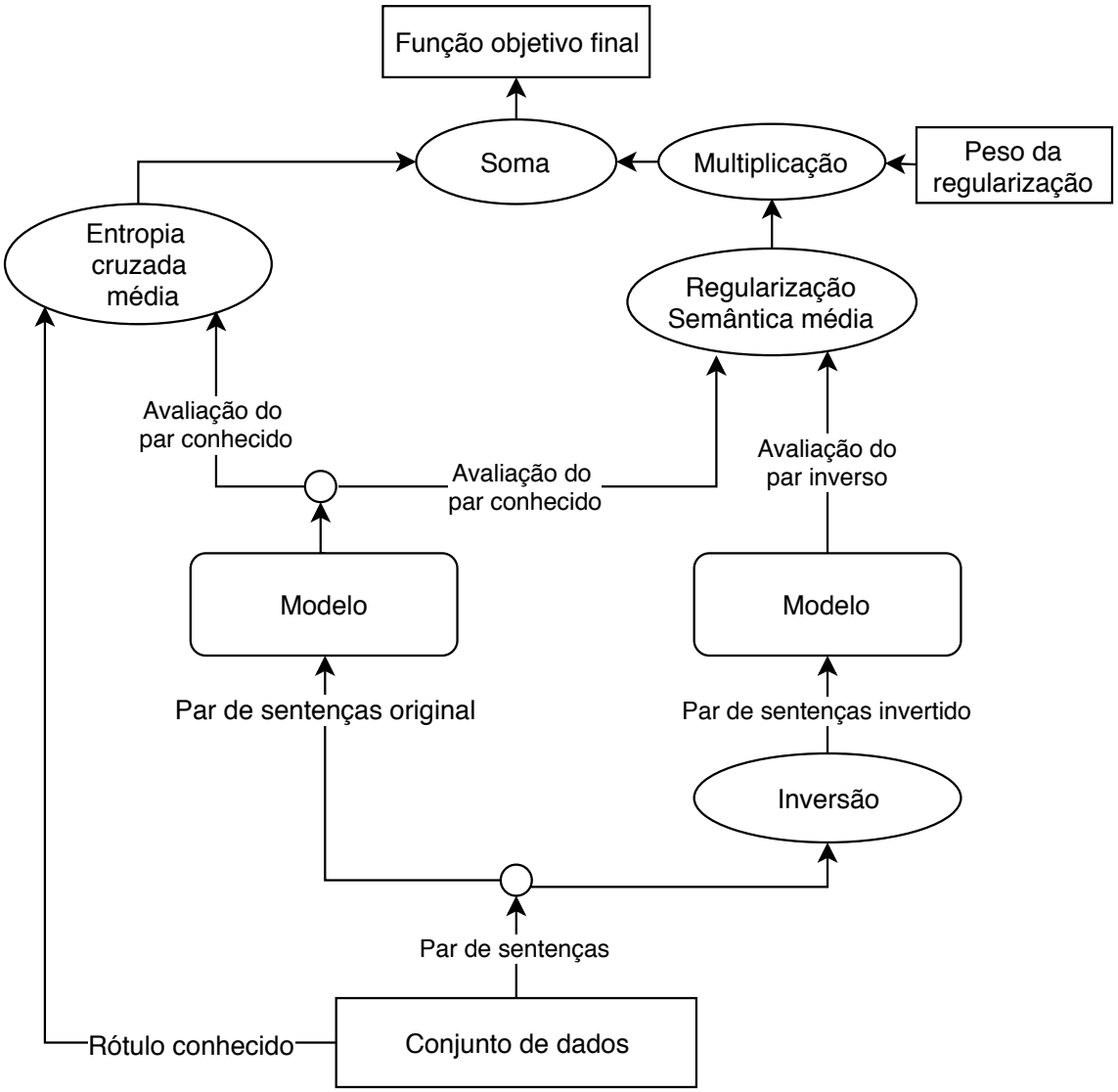

Figura 5.2: Visão geral do cálculo da função objetivo final.

classes, onde as probabilidades finais são normalizadas com a usual função *softmax* (GOODFELLOW *et al.*, 2016).

O modelo final é otimizado usando a combinação da entropia cruzada, que é calculada em função as probabilidades dadas, com uma das variantes da regularização semântica. Uma visão geral deste processo de cálculo da função objetivo final pode ser visualizada de forma esquemática na Figura 5.2.

5.2 Implementação

Os experimentos realizados utilizaram o conjunto de dados MultiNLI, devido a seu tamanho e abrangência superior aos demais *datasets*. Os códigos foram implementados em Python usando a biblioteca Tensorflow, partindo do código originalmente disponi-

bilizado em [1]. Todas as variantes do modelo foram treinadas com os hiperparâmetros baseados em WILLIAMS *et al.* (2018) e resumidos na Tabela 5.1.

Descrição do hiperparâmetro	**Valor**
Probabilidade do *Dropout*	0.1
Coeficiente de aprendizado inicial do Adam	0.0004
Tamanho do *minibatch*	32
Dimensão do *word embedding*	300
Tamanho de saída da camada densa	300
Tamanho da representação de cada codificador	300
Tamanho máximo da sequência dada ao codificador	50

Tabela 5.1: Hiperparâmetros usados na implementação

Foi utilizado o algoritmo Adam (KINGMA e BA, 2014) com *mini batches* de tamanho 32 para a otimização do modelo. Os vetores de palavras foram inicializados usando o GloVe (PENNINGTON *et al.*, 2014) de 300 dimensões baseado no *Common Crawl* com um vocabulário de 2.2 milhões de palavras, disponível em [2]. Palavras inexistentes no vocabulário são representadas com um único vetor para todas, inicializado aleatoriamente. Sentenças com mais do que 50 palavras são truncadas, enquanto sentenças menores são preenchidas com vetores de zeros até serem representadas com 50 vetores.

O treinamento foi parado quando a função objetivo não demonstrava melhoras no modelo depois de 10 avaliações no conjunto de validação *Matched*, avaliações estas que ocorriam periodicamente durante cada iteração no conjunto de treinamento, de forma que cerca de 10% do tempo de treinamento era destinado a validação. Esta escolha ocorreu devido à observação de que o algoritmo original usava a maior parte do tempo de treinamento avaliando o critério de parada, sem praticamente qualquer avanço no conjunto de treinamento.

Foram testadas 3 variantes da regularização semântica (Equação 4.3) com diferentes pesos: a variante multiplicativa (Equações 4.4, 4.5 e 4.6), a variante baseada em lógica *fuzzy* (Equações 4.7, 4.8 e 4.9) e uma variante híbrida, que usa a forma multiplicativa da restrição de contradição e a forma *fuzzy* das duas demais (Equações 4.4, 4.8 e 4.9). Esta escolha para o caso híbrido é justificada a partir de resultados preliminares dos experimentos, descrito na Sessão 6.4.

Por simplicidade, os pesos w_i, w_c e w_n nos experimentos são iguais e escolhidos de forma que a soma seja igual a um valor w, ou seja, o termo da regularização se torna, simplificadamente, em $w \times (S_c + S_i + S_n)/3$. Durante o treinamento, o

[1] `https://github.com/nyu-mll/multiNLI`

[2] `http://nlp.stanford.edu/data/glove.840B.300d.zip`

hiperparâmetro w foi aumentado linearmente até a segunda iteração (*epoch*) no *dataset*, onde atinge seu valor final até o treinamento atingir o critério de parada, visando uma melhor convergência do algoritmo já que a coerência pesa muito pouco no início, se assemelhando a um caso sem a regularização, mas vai gradualmente ganhando importância conforme o algoritmo progride.

Os experimentos foram realizados numa máquina virtual simples hospedada no serviço Google Cloud com oito núcleos e 12 GB de RAM, configuração que ficava próxima de saturar o consumo de processamento e memória. Cinco configurações do modelo foram experimentadas: uma versão *baseline* ($w = 0$), uma versão multiplicativa com $w = 0.25$, uma versão *fuzzy* com $w = 0.25$ e duas versões híbridas com $w = 0.25$ e $w = 0.5$. Cada uma das cinco configurações demorou cerca de um dia para executar. Mais variantes não foram testadas por questões de escopo e tempo. Acrescentar outras variantes pareceu que não agregaria muita informação para a análise pois ao dobrar o valor de w para o caso híbrido não foi notada variações significativas nas estatísticas medidas e reduzir w pareceu ser contraprodutivo observando as pequenas progressões nas funções de regularização semântica que já ocorriam com $w = 0.25$.

5.3 Experimentos

Durante o treinamento, várias estatísticas foram coletadas, tanto relativas ao conjunto de validação, quanto ao próprio treinamento. Ao fim do treinamento, as métricas finais foram avaliadas nos conjuntos de dados disponibilizados nas versões *Matched* e *Mismatched*: validação, teste e teste difícil de GURURANGAN *et al.* (2018). Ambos conjuntos de teste e o conjunto de validação *Mismatched* só foram analisados para a escrita desta dissertação. Importante ressaltar que todos os conjuntos de testes só podem ser avaliados a partir do sistema de competições do Kaggle [3] [4] [5] [6], que apenas notifica a taxa de acurácia para um determinado conjunto de respostas enviadas, impossibilitando análises mais aprofundadas nestes dados.

O impacto da regularização semântica sobre a coerência do modelo foi avaliado inicialmente comparando a taxa de respostas coerentes de modelos regularizados com o *baseline*. Já o impacto da coerência sobre a capacidade de generalização do modelo é avaliado comparando-se o *baseline* com as demais variantes usando a acurácia nos diferentes conjuntos de validação como métrica. Outra análise feita neste contexto foi observar a diferença entre a acurácia segmentada por respostas coerentes e incoerentes nos diferentes modelos, onde se espera que as respostas coerentes sejam

[3] `https://www.kaggle.com/c/multinli-matched-open-evaluation`
[4] `https://www.kaggle.com/c/multinli-mismatched-open-evaluation`
[5] `https://www.kaggle.com/c/multinli-matched-open-hard-evaluation`
[6] `https://www.kaggle.com/c/multinli-mismatched-open-hard-evaluation`

mais acertadas.

Para se analisar as diferenças entre as variantes de regularização semântica, comparou-se a progressão ao longo do treinamento dos valores obtidos nas funções de regularização específicas (S_c, S_i e S_n), visando principalmente identificar-se estagnações no processo de treinamento.

Por fim, foram exploradas algumas possíveis consequências da regularização semântica sobre os modelos, na qual se compararam especificamente o modelo *baseline* contra o modelo híbrido com $w = 0.25$. O primeiro teste foi avaliar o impacto sobre a distribuição das respostas, buscando identificar novos vieses apresentados por modelos regularizados em relação ao modelo *baseline*. Isso foi avaliado usando uma matriz de confusão, na qual se discriminam as frequências de diferentes combinações de resposta esperada e resposta dada pelo modelo. Esta análise também foi expandida se segmentando as respostas entre coerentes e incoerentes.

Na segunda análise, foram observados possíveis vieses nas respostas dadas para os pares de sentenças invertidos. Para tal, foi calculada uma tabela similar a matriz de confusão, porém contendo as frequências de cada resposta dada tanto para o caso original quanto para o caso invertido.

Capítulo 6

Experimentos

6.1 Resultados gerais

Um resumo geral dos resultados pode ser visto nas Tabelas 6.1 e 6.2 e visualizado graficamente nas Figuras 6.1 e 6.2. Valores **grifados** representam os melhores casos de cada métrica enquanto os sublinhados marcam a segunda posição.

Modelo	**Validação**				**Teste**	**Difícil**
	Acerto	**Coerência**	**Acerto (Coerente)**	**Acerto (Incoerente)**	**Acerto**	**Acerto**
Baseline (w=0.0)	**68.24%**	76.01%	69.48%	**64.33%**	67.06%	48.55%
Multiplic. (w=0.25)	67.73%	87.99%	69.60%	54.03%	66.14%	45.90%
Fuzzy (w=0.25)	68.00%	86.73%	**70.41%**	52.23%	66.97%	**49.00%**
Híbrido (w=0.25)	67.96%	88.90%	69.59%	54.91%	**67.31%**	46.83%
Híbrido (w=0.5)	67.81%	**89.71%**	69.39%	54.06%	66.64%	47.60 %

Tabela 6.1: Resumo dos modelos no conjunto de dados *Matched* do MultiNLI

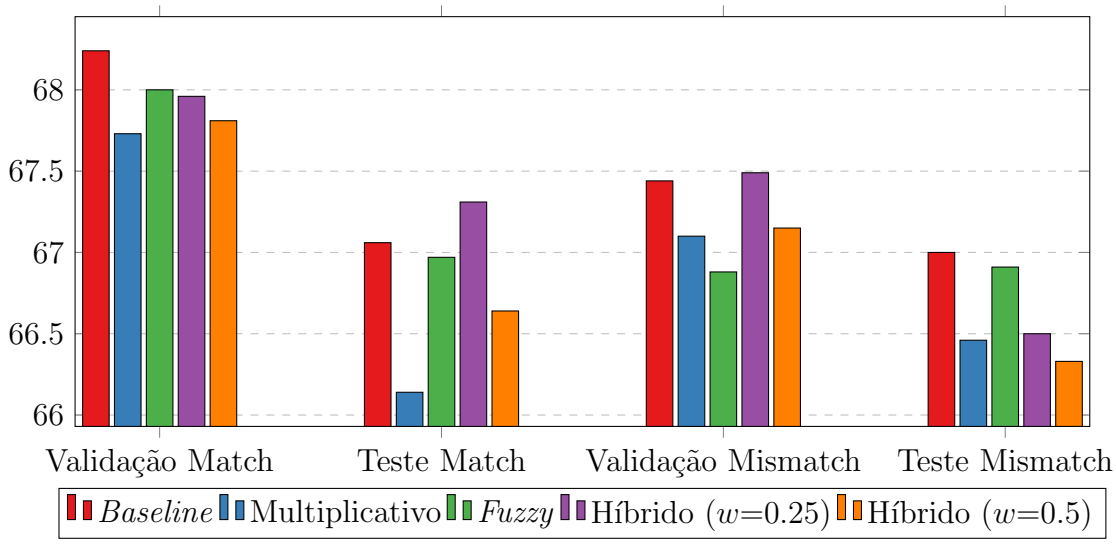

Figura 6.1: Comparativo dos resultados nos conjuntos de dados usuais

Modelo	**Validação**				**Teste**	**Difícil**
	Acerto	**Coerência**	**Acerto (Coerente)**	**Acerto (Incoerente)**	**Acerto**	**Acerto**
Baseline (w=0.0)	67.44%	76.93%	67.99%	**65.61%**	**67.00%**	49.25%
Multiplic. (w=0.25)	67.10%	88.44%	68.80%	54.09%	66.46%	47.11%
Fuzzy (w=0.25)	66.88%	87.19%	**69.14%**	51.55%	66.91%	**49.91%**
Híbrido (w=0.25)	**67.49%**	89.08%	69.02%	55.03%	66.50%	46.16%
Híbrido (w=0.5)	67.15%	**90.10%**	68.92%	50.98%	66.33%	47.22%

Tabela 6.2: Resumo dos modelos no conjunto de dados *Mismatched* do MultiNLI

Nas Tabelas 6.1 e 6.2, cada linha representa uma das variantes analisadas neste trabalho. A coluna "Coerência" representa a proporção de casos em que o modelo foi condizente com as restrições apresentadas anteriormente nas Tabelas 4.2, 4.3 e 4.4, utilizando-se a classe mais provável para se validar a restrição. As colunas "Acerto (Coerente)" e "Acerto (Incoerente)" apresentam os valores de acurácia calculados somente com os casos em que o modelo foi coerente, ou incoerente, com as restrições. A coluna "Difícil" se refere ao conjunto de testes proposto por GURURANGAN *et al.* (2018), que elimina, do conjunto de testes original, os pares que um classificador simples consegue acertar somente usando a sentença da hipótese.

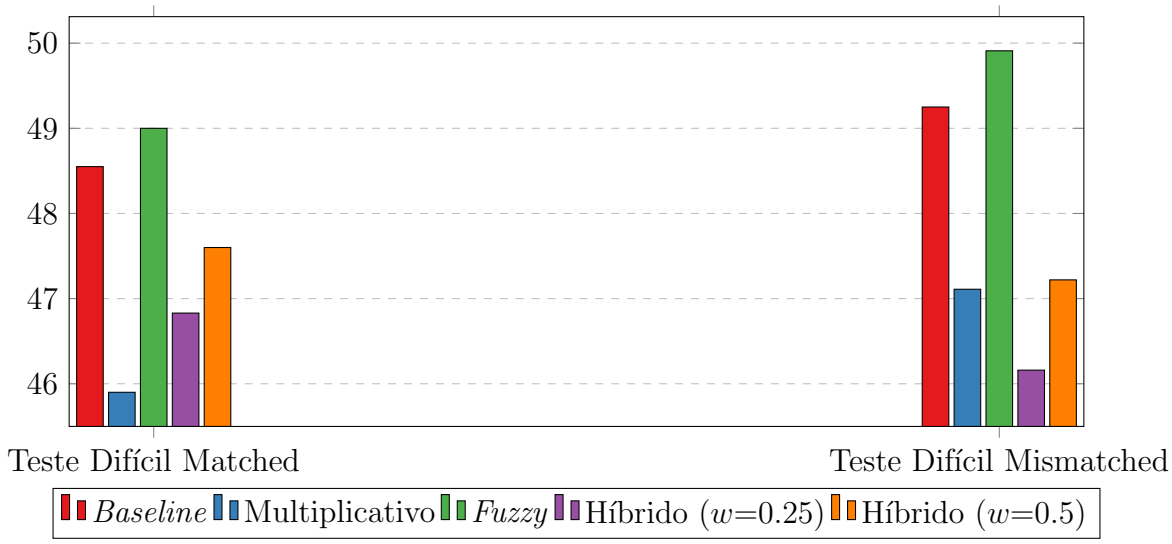

Figura 6.2: Comparativo dos resultados nos subconjuntos difíceis

6.2 Regularização semântica e a coerência lógica

As Tabelas 6.1 e 6.2 dão bons indícios sobre a eficácia da regularização semântica em melhorar a coerência lógica dos modelos baseados em redes neurais: observa-se uma diferença de mais de 10% entre a coerência do modelo *baseline* e todos os demais com alguma variante da regularização semântica. Porém como apenas uma arquitetura de rede neural foi avaliada neste trabalho, ainda é cedo para se concluir seguramente que a regularização semântica é de fato um bom método para se melhorar a coerência em qualquer cenário ou até se de fato qualquer modelo em geral se beneficiaria deste tipo de interferência.

6.3 Coerência e capacidade de generalização

Analisando os dados, observam-se resultados bem diversos em termos de acurácia nos diferentes subconjuntos. O modelo *baseline* tem um desempenho levemente superior no conjunto de validação *Matched* e no conjunto de testes *Mismatched*, avaliando-se as taxas de acerto, porém os modelos regularizados com as variantes *fuzzy* e híbrida (w=0.25) estão piores apenas por cerca de 0.25%, no caso mais discrepante, mas são 10% mais coerentes, em termos absolutos, que o *baseline*. Por outro lado, a variante híbrida é superior por valores similares nos casos dos conjuntos de validação *Mismatched* e teste *Matched*, além de possuir uma alta taxa de coerência, enquanto a variante *fuzzy* obtém o melhor resultado nos subconjuntos difíceis.

Avaliar a coerência neste contexto de generalização também é interessante pois esta medida indicaria um valor máximo para a acurácia geral dos modelos, se eles fossem avaliados em ambas as direções dos pares de sentenças. Este máximo pode

ser estimado usando o valor da coerência multiplicado pela acurácia obtida, o que aponta os modelos híbridos como os que têm a maior capacidade de acurácia, cerca de 60%, enquanto o *baseline* possui quase 52%, porém tal medida não é usual no contexto de aprendizado de máquina e deve ser analisada com ressalvas.

Esperado\Predito	**Inferência**	**Neutro**	**Contradição**	**Total Esperado**
Inferência	25.80% (2532)	6.54% (642)	3.11% (305)	35.45% (3479)
Neutro	7.05% (692)	19.74% (1937)	5.03% (494)	31.82% (3123)
Contradição	5.26% (516)	4.77% (468)	22.71% (2229)	32.74% (3213)
Total Predito	38.10% (3740)	31.04% (3047)	30.85% (3028)	68.24% (6698)

Tabela 6.3: Matriz de confusão do *baseline*

Esperado\Predito	**Inferência**	**Neutro**	**Contradição**	**Total Esperado**
Inferência	26.29% (2580)	6.36% (624)	2.80% (275)	35.45% (3479)
Neutro	7.47% (733)	19.84% (1947)	4.51% (443)	31.82% (3123)
Contradição	5.71% (560)	5.20% (510)	21.83% (2143)	32.74% (3213)
Total Predito	39.46% (3873)	31.39% (3081)	29.15% (2861)	67.96% (6670)

Tabela 6.4: Matriz de confusão do modelo híbrido

As Tabelas 6.3 e 6.4 apresentam as matrizes de confusão do modelo *baseline* e do modelo hibrido (w=0.25) no conjunto de validação *Matched*. Os resultados são muito semelhantes, o que indica que o método de regularização não está nem enviesando o resultado de alguma forma e nem interferindo negativamente no modelo em geral, porém não o ajuda diretamente a generalizar melhor, considerando este método clássico de avaliação. Isso pode se dar pelo fato de que o próprio procedimento de construção do conjunto de dados simplesmente não avalia naturalmente esta característica de coerência nos modelos, pouco interferindo em sua acurácia.

6.4 Variantes da regularização semântica

Para entender o comportamento das diferentes variantes da regularização semântica, foram coletados dados sobre o processo de treinamento destas variantes. As curvas apresentadas nas Figuras 6.3, 6.4 e 6.5 mostram o progresso dos diferentes modelos para os casos específicos das restrições da inferência, neutralidade e contradição. Nas figuras é possível visualizar, para todos as variantes, tanto a média da regularização semântica avaliada no conjunto de treinamento quanto a porcentagem de vezes em que o modelo foi coerente com aquela restrição, avaliada no conjunto *Matched*.

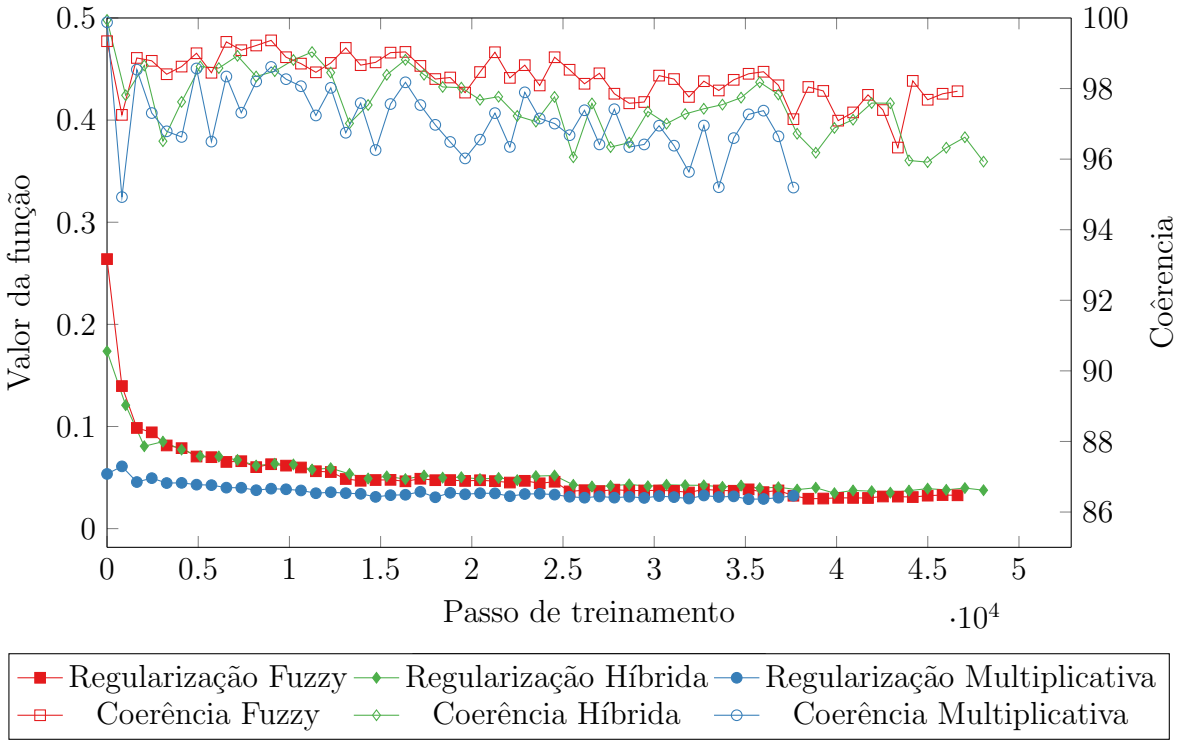

Figura 6.3: Avaliação da restrição da inferência

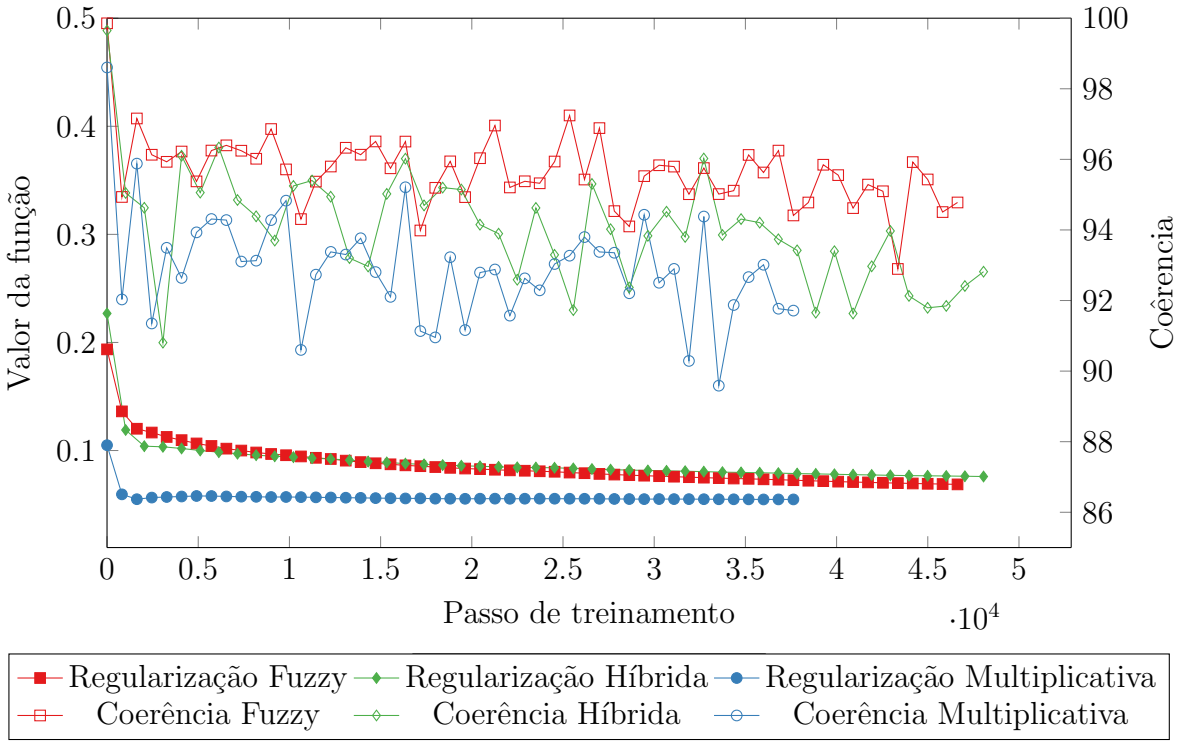

Figura 6.4: Avaliação da restrição da neutralidade

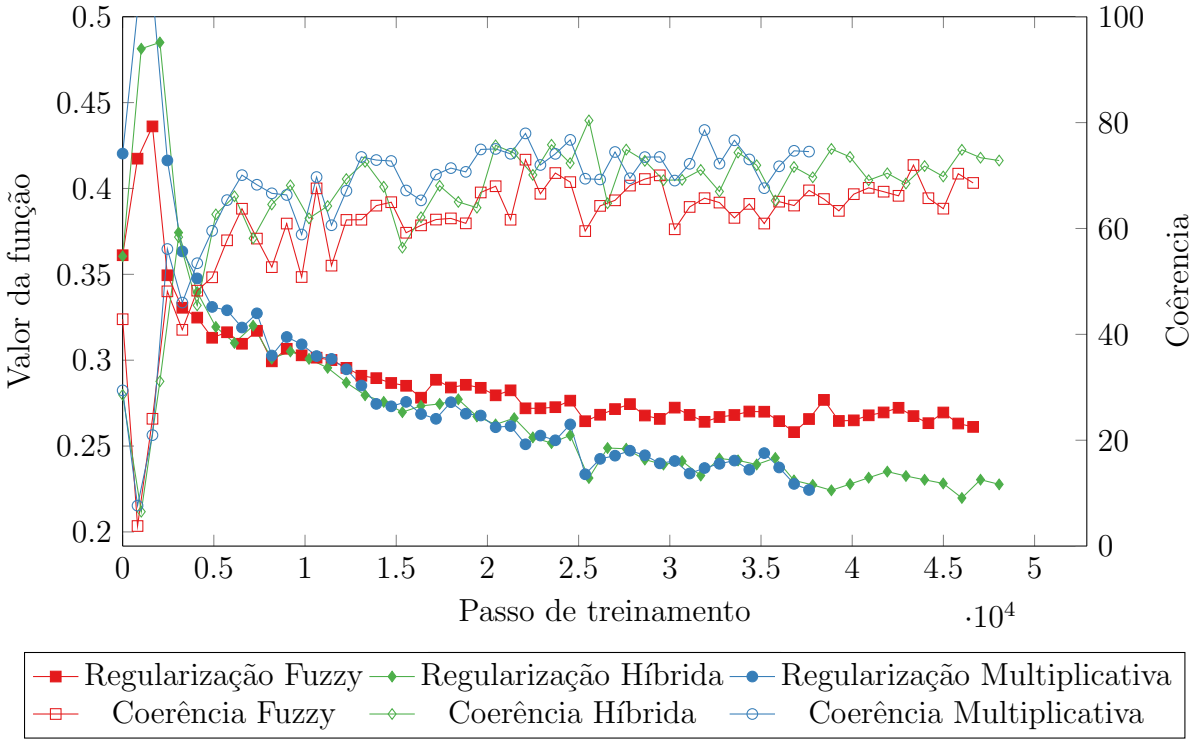

Figura 6.5: Avaliação da restrição da contradição

Observa-se nas Figuras 6.3 e 6.4 que a versão multiplicativa já começa o treinamento com valores baixos para as funções de regularização da inferência e neutralidade, que quase não se alteram durante a otimização, sugerindo que as funções pouco influenciaram o modelo final. A versão *fuzzy*, por sua vez, tem outro comportamento: o valor da regularização decresce mais significativamente. Por outro lado, os valores da restrição de contradição são reduzidos em ambas as variantes, porém a versão multiplicativa parece ser mais eficaz, como pode ser visto na Figura 6.5. Estas verificações também podem ser validadas analisando os valores de coerência apresentados em cada figura: as variantes que de fato apresentaram uma melhora na função de regularização também apresentaram resultados melhores no conjunto de avaliação, como esperado para as funções substitutas, explicadas na Seção 2.3.

A observação deste comportamento sugeriu a possibilidade de criação da versão híbrida, usando a forma multiplicativa para a restrição da contradição e a forma *fuzzy* para as restrições de neutralidade e inferência, visando obter as vantagens de cada variante. As Figuras 6.3, 6.4 e 6.5, além das Tabela 6.1 e 6.2, sugerem que essa estratégia de fato funciona, mantendo um comportamento bem similar as funções de regularização originais e permanecendo entre as duas variantes quando se a analisa a taxa de coerência na validação.

Uma possível explicação para a diferença no comportamento das regularizações da inferência e da neutralidade seria similar ao fenômeno do *vanishing gradients*,

explicado na Seção 2.5. Enquanto a variante *fuzzy* tem uma descida constante em toda região com derivadas definidas (Figura 4.3), a versão multiplicativa tem um declive que vai sendo suavizado quanto mais próximo da origem (Figura 4.3). Como estes valores de regularização já começam próximo da origem, a norma do vetor gradiente para o caso multiplicativo já começa pequena e ainda vai reduzindo mais ainda, o que, consequentemente, vai propagar uma variação quase inexistente para os parâmetros que deveriam ser modificados via *backpropagation*.

6.5 Influências da (in)coerência

Ainda analisando as Tabelas 6.1 e 6.2, pode-se começar a entender a importância da coerência observando os valores de acurácia segmentados pela própria coerência na classificação: todos modelos, incluindo o *baseline*, exibem uma discrepância significativa entre a taxa de acerto dos casos coerentes e dos casos incoerentes, que é bem mais acentuada nos modelos regularizados. Esta observação sugere que saber se o modelo foi coerente para um dado par de sentenças pode ser utilizado como um indicativo de qualidade daquela resposta.

As Tabelas 6.5 e 6.6 apresentam as matrizes de confusão segmentadas somente para os casos coerentes dos modelos *baseline* e híbrido (w=0.25). Novamente, os resultados são proporcionalmente muito similares, porém nota-se uma pequena mudança no viés de classificação dos modelos: o modelo *baseline* é mais enviesado a responder a classe "inferência" do que o modelo hibrido, que compensa esse viés com mais respostas de "contradição", além de possuir mais casos coerentes, como esperado.

Esperado\Predito	**Inferência**	**Neutro**	**Contradição**	**Total Esperado**
Inferência	31.96% (2384)	7.69% (574)	1.23% (92)	40.88% (3050)
Neutro	8.77% (654)	22.04% (1644)	2.08% (155)	32.88% (2453)
Contradição	6.19% (462)	4.56% (340)	15.48% (1155)	26.23% (1957)
Total Predito	46.92% (3500)	34.29% (2558)	18.79% (1402)	69.48% (5183)

Tabela 6.5: Matriz de confusão dos casos coerentes para o *baseline*

Esperado\Predito	**Inferência**	**Neutro**	**Contradição**	**Total Esperado**
Inferência	29.12% (2541)	6.97% (608)	1.28% (112)	37.37% (3261)
Neutro	8.17% (713)	21.06% (1838)	2.80% (244)	32.03% (2795)
Contradição	6.13% (535)	5.07% (442)	19.40% (1693)	30.60% (2670)
Total Predito	43.42% (3789)	33.10% (2888)	23.48% (2049)	69.59% (6072)

Tabela 6.6: Matriz de confusão dos casos coerentes para o modelo híbrido

As Tabelas 6.7 e 6.8 apresentam as matrizes de confusão somente para os casos em que o respectivo modelo foi incoerente. Neste caso, é possível observar que os modelos diferem bastante em suas proporções, porém ambos são nitidamente enviesados para a classe de "contradição", o que pode ser explicado pelo fato de a restrição da contradição ser mais exigente que as demais, tornando estes casos mais propícios de serem incoerentes a priori. Outra possível explicação é um aumento na variância neste caso da análise, já que o modelo regularizado possui apenas 598 casos incoerentes, cerca de 40% da quantidade do *baseline*.

Esperado\Predito	**Inferência**	**Neutro**	**Contradição**	**Total Esperado**
Inferência	6.28% (148)	2.89% (68)	9.04% (213)	18.22% (429)
Neutro	1.61% (38)	12.44% (293)	14.39% (339)	28.45% (670)
Contradição	2.29% (54)	5.44% (128)	45.61% (1074)	53.33% (1256)
Total Predito	10.19% (240)	20.76% (489)	69.04% (1626)	64.33% (1515)

Tabela 6.7: Matriz de confusão dos casos incoerentes para o *baseline*

Esperado\Predito	**Inferência**	**Neutro**	**Contradição**	**Total Esperado**
Inferência	3.58% (39)	1.47% (16)	14.97% (163)	20.02% (218)
Neutro	1.84% (20)	10.01% (109)	18.27% (199)	30.12% (328)
Contradição	2.30% (25)	6.24% (68)	41.32% (450)	49.86% (543)
Total Predito	7.71% (84)	17.72% (193)	74.56% (812)	54.91% (598)

Tabela 6.8: Matriz de confusão dos casos incoerentes para o modelo híbrido

As Tabelas 6.9 e 6.10 mostram as coocorrências entre as respostas dadas pelo modelo para o par original e invertido do conjunto de validação *Matched*. A marcação $^{\times}$ denota casos que são considerados incoerentes e as proporções estão calculadas em função de cada linha. Nota-se que o modelo *baseline* tende a manter a resposta anterior ou responder com a classe "'neutra" quando o par é invertido, inclusive para a classe de "contradição", o que configura numa incoerência. Já o modelo regularizado tende a responder com a classe "neutra" nos casos de inferência e neutralidade, porém mantém sua resposta para as contradições. Outro ponto interessante é que o modelo regularizado diz existir 30% menos paráfrases que o modelo *baseline*, o que é indicado quando o modelo diz em ambas situações que o par se trata de uma inferência, algo que não aparece em nenhuma das restrições da regularização porém que é esperado quando se leva em conta que o processo de construção do *dataset* não pediu aos trabalhadores para que escrevessem necessariamente paráfrases.

Original**Inverso	**Inferência**	**Neutro**	**Contradição**	**Coerentes**
Inferência	45.13% (1688)	48.45% (1812)	6.42% (240)$^{\times}$	93.58% (3500)
Neutro	33.84% (1031)	50.11% (1527)	16.05% (489)$^{\times}$	83.95% (2558)
Contradição	8.22% (249)$^{\times}$	45.48% (1377)$^{\times}$	46.30% (1402)	46.30% (1402)
			Total	76.01% (7460)

Tabela 6.9: Matriz de inversão para o *baseline*

Original**Inverso	**Inferência**	**Neutro**	**Contradição**	**Coerentes**
Inferência	31.06% (1203)	66.77% (2586)	2.17% (84)$^{\times}$	97.83% (3789)
Neutro	22.88% (705)	70.85% (2183)	6.26% (193)$^{\times}$	93.74% (2888)
Contradição	5.24% (150)$^{\times}$	23.14% (662)$^{\times}$	71.62% (2049)	71.62% (2049)
			Total	88.90% (8726)

Tabela 6.10: Matriz de inversão para o modelo híbrido

Capítulo 7

Conclusão

O reconhecimento de inferência textual é uma importante e complexa tarefa intermediária para a criação de algoritmos capazes de compreender a linguagem humana. Por isso, é necessária a contínua revisão de seus métodos de avaliação, como conjuntos de dados e métricas.

Esta dissertação apresenta o problema da incoerência, inerente da própria natureza lógica desta tarefa, e define formalmente restrições que os modelos devem obedecer em suas respostas para serem considerados coerentes, baseando-se na configuração de três classes usada pelos dois conjuntos de dados mais abrangentes até o momento: o SNLI e o MultiNLI. Também é proposta uma solução que visa aumentar a coerência dos métodos de aprendizado de máquina apenas adicionando-se um termo de regularização na função objetivo do modelo.

Foram realizados experimentos que testaram a solução proposta usando um modelo de codificação de sentenças para a tarefa, demonstrando ganhos expressivos na coerência do modelo sem perda significativa de acurácia.

7.1 Contribuições

As contribuições e achados interessantes desta dissertação são resumidos abaixo:

1. Definição formal, baseada em lógica proposicional, do problema da incoerência no contexto do reconhecimento de inferência textual;

2. Definição da regularização semântica, que é uma aplicação da *semantic loss* (XU *et al.*, 2017), como forma de aumentar a coerência de forma genérica nos modelos de aprendizado de máquina por meio de um termo de regularização;

3. Definição de três variantes da regularização semântica: a variante multiplicativa, baseada inteiramente em XU *et al.* (2017), a variante *fuzzy* e a variante híbrida,

que é uma combinação das duas anteriores, proposta após análise preliminar do treinamento dos modelos anteriores;

4. Os experimentos realizados sugerem que a regularização semântica é de fato eficaz em aumentar a coerência dos modelos, com ganhos absolutos de mais de 10%;

5. Também é sugerido que a regularização em geral não atrapalha na capacidade de generalização dos modelos, porém não melhora sua taxa de acurácia diretamente. É levantada a hipótese de que isso pode ser um artefato da própria construção do *dataset* testado;

6. A variante híbrida parece ser o modelo mais coerente, quando comparada em condições similares aos demais;

7. A variante *fuzzy* se mostra competitiva, em termos acurácia, na maioria dos casos e foi a melhor nos dois conjuntos de testes difíceis;

8. Observou-se que as respostas incoerentes de todos os modelos regularizados tendem a ter uma acurácia muito menor que a acurácia geral, informação que pode ser um útil indicador de confiança do modelo. Isso também ocorreu no modelo *baseline*, porém em menor escala;

9. As respostas do modelo híbrido apontam para menos casos de paráfrases, quando comparado ao *baseline*, algo esperado qualitativamente, dado a forma de construção do *dataset* MultiNLI;

7.2 Trabalhos futuros

Esta dissertação possui alguns pontos fracos que poderiam ser melhor explorados e são deixados como sugestões de trabalhos futuros, em ordem de importância:

1. Experimentos com modelos do estado da arte, avaliando se as observações feitas a partir do *baseline* se estendem aos demais;

2. Avaliação do impacto da coerência na qualidade do codificador usando transferência de aprendizado, similar ao trabalho de CONNEAU *et al.* (2017);

3. Desenvolvimento de arquiteturas ou métodos que são, por construção, sempre coerentes;

4. Avaliar outras variantes da regularização semântica, possivelmente se baseando em outros operadores da lógica nebulosa;

Referências Bibliográficas

ABU-MOSTAFA, Y. S., MAGDON-ISMAIL, M., LIN, H.-T. *Learning from data*, v. 4. AMLBook New York, NY, USA:, 2012.

OLAH, CHRISTOPHER. "Understanding LSTM Networks". 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acessado em 20-07-2018.

NAIK, A., RAVICHANDER, A., SADEH, N., et al. "Stress Test Evaluation for Natural Language Inference". In: *The 27th International Conference on Computational Linguistics (COLING)*, Santa Fe, New Mexico, USA, August 2018.

WILLIAMS, A., NANGIA, N., BOWMAN, S. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122. Association for Computational Linguistics, 2018. Disponível em: <http://aclweb.org/anthology/N18-1101>.

DAGAN, I., ROTH, D., SAMMONS, M., et al. "Recognizing textual entailment: Models and applications", *Synthesis Lectures on Human Language Technologies*, v. 6, n. 4, pp. 1–220, 2013.

DAGAN, I., GLICKMAN, O., MAGNINI, B. "The PASCAL Recognising Textual Entailment Challenge". In: *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pp. 177–190, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN: 3-540-33427-0, 978-3-540-33427-9. doi: 10.1007/11736790_9. Disponível em: <http://dx.doi.org/10.1007/11736790_9>.

MANNING, C. D., MANNING, C. D., SCHÜTZE, H. *Foundations of statistical natural language processing*. MIT press, 1999.

ANDROUTSOPOULOS, I., MALAKASIOTIS, P. "A survey of paraphrasing and textual entailment methods", *Journal of Artificial Intelligence Research*, v. 38, pp. 135–187, 2010.

CONNEAU, A., KIELA, D., SCHWENK, H., et al. "Supervised learning of universal sentence representations from natural language inference data", *arXiv preprint arXiv:1705.02364*, 2017.

BOWMAN, S. R., ANGELI, G., POTTS, C., et al. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

ROCKTÄSCHEL, T., GREFENSTETTE, E., HERMANN, K. M., et al. "Reasoning about Entailment with Neural Attention". In: *International Conference on Learning Representations (ICLR)*, 2016.

CHEN, Q., ZHU, X., LING, Z., et al. "Enhancing and combining sequential and tree LSTM for natural language inference", *arXiv preprint arXiv:1609.06038*, 2016.

GURURANGAN, S., SWAYAMDIPTA, S., LEVY, O., et al. "Annotation Artifacts in Natural Language Inference Data", *arXiv preprint arXiv:1803.02324*, 2018.

XU, J., ZHANG, Z., FRIEDMAN, T., et al. "A Semantic Loss Function for Deep Learning Under Weak Supervision". In: *NIPS 2017 Workshop on Learning with Limited Labeled Data: Weak Supervision and Beyond*, dez. 2017. Disponível em: <`http://web.cs.ucla.edu/~guyvdb/papers/XuLLD17.pdf`>.

XU, J., ZHANG, Z., FRIEDMAN, T., et al. "A Semantic Loss Function for Deep Learning with Symbolic Knowledge". In: Dy, J., Krause, A. (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, v. 80, *Proceedings of Machine Learning Research*, pp. 5498–5507, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. Disponível em: <`http://proceedings.mlr.press/v80/xu18h.html`>.

BIRD, S., KLEIN, E., LOPER, E. *Natural language processing with Python: analyzing text with the natural language toolkit.* "O'Reilly Media, Inc.", 2009.

PANG, B., LEE, L., OTHERS. "Opinion mining and sentiment analysis", *Foundations and Trends® in Information Retrieval*, v. 2, n. 1–2, pp. 1–135, 2008.

FERREIRA, M. C., XEXÉO, G. B. *Incident routing: text classification, feature selection, imbalanced datasets, and concept drift in incident ticket management.* Tese de Mestrado, Universidade Federal do Rio de Janeiro, 2017.

WANG, S., MANNING, C. D. "Baselines and bigrams: Simple, good sentiment and topic classification". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 90–94. Association for Computational Linguistics, 2012.

GAMBHIR, M., GUPTA, V. "Recent automatic text summarization techniques: a survey", *Artificial Intelligence Review*, v. 47, n. 1, pp. 1–66, 2017.

RUSH, A. M., CHOPRA, S., WESTON, J. "A neural attention model for abstractive sentence summarization", *arXiv preprint arXiv:1509.00685*, 2015.

HARRIS, Z. S. "Distributional structure", *Word*, v. 10, n. 2-3, pp. 146–162, 1954.

BENGIO, Y., DUCHARME, R., VINCENT, P., et al. "A neural probabilistic language model", *Journal of machine learning research*, v. 3, n. Feb, pp. 1137–1155, 2003.

GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

KINGMA, D. P., BA, J. "Adam: A Method for Stochastic Optimization", *CoRR*, v. abs/1412.6980, 2014.

WEISSTEIN, E. W. "Taylor Series. From MathWorld—A Wolfram Web Resource". Disponível em: <`http://mathworld.wolfram.com/TaylorSeries.html`>. Acessado em 03-08-2018.

RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J. *Learning internal representations by error propagation.* Relatório técnico, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

DENG, J., DONG, W., SOCHER, R., et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*, 2009.

THEANO DEVELOPMENT TEAM. "Theano: A Python framework for fast computation of mathematical expressions", *arXiv e-prints*, v. abs/1605.02688, maio 2016. Disponível em: <http://arxiv.org/abs/1605.02688>.

ABADI, M., AGARWAL, A., BARHAM, P., et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". 2015. Disponível em: <https://www.tensorflow.org/>. Software available from tensorflow.org.

SABOUR, S., FROSST, N., HINTON, G. E. "Dynamic routing between capsules". In: *Advances in Neural Information Processing Systems*, pp. 3856–3866, 2017.

VASWANI, A., SHAZEER, N., PARMAR, N., et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

HORNIK, K., STINCHCOMBE, M., WHITE, H. "Multilayer feedforward networks are universal approximators", *Neural networks*, v. 2, n. 5, pp. 359–366, 1989.

CANALLI, Y. D. M., SILVA, G. Z. D. *Funções de ativação hiperbólicas em redes neurais*. Tese de Mestrado, Universidade Federal do Rio de Janeiro, 2017.

PENNINGTON, J., SOCHER, R., MANNING, C. D. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. Disponível em: <http://www.aclweb.org/anthology/D14-1162>.

HOCHREITER, S., SCHMIDHUBER, J. "Long short-term memory", *Neural computation*, v. 9, n. 8, pp. 1735–1780, 1997.

GRAVES, A., SCHMIDHUBER, J. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures", *Neural Networks*, v. 18, n. 5-6, pp. 602–610, 2005.

GREFF, K., SRIVASTAVA, R. K., KOUTNÍK, J., et al. "LSTM: A search space odyssey", *IEEE transactions on neural networks and learning systems*, v. 28, n. 10, pp. 2222–2232, 2017.

KOLESNYK, V., ROCKTÄSCHEL, T., RIEDEL, S. "Generating natural language inference chains", *arXiv preprint arXiv:1606.01404*, 2016.

BAR HAIM, R., DAGAN, I., DOLAN, B., et al. "The second pascal recognising textual entailment challenge", 2006.

GIAMPICCOLO, D., MAGNINI, B., DAGAN, I., et al. "The third pascal recognizing textual entailment challenge". In: *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9. Association for Computational Linguistics, 2007.

LEVENSHTEIN, V. I. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*, v. 10, pp. 707–710, 1966.

BOS, J., MARKERT, K. "Recognising Textual Entailment with Logical Inference". In: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pp. 628–635, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220654. Disponível em: <https://doi.org/10.3115/1220575.1220654>.

FONSECA, E., ALUÍSIO, S. M., DOS SANTOS, L., et al. "Overview of the ASSIN shared task and corpus". 2016. Disponível em: <http://propor2016.di.fc.ul.pt/wp-content/uploads/2015/10/overview-assin.pdf>. Acessado em 24-07-2018.

MOU, L., MEN, R., LI, G., et al. "Natural language inference by tree-based convolution and heuristic matching", *arXiv preprint arXiv:1512.08422*, 2015.

NANGIA, N., WILLIAMS, A., LAZARIDOU, A., et al. "The repeval 2017 shared task: Multi-genre natural language inference with sentence representations", *arXiv preprint arXiv:1707.08172*, 2017.

ZADEH, L. "Fuzzy sets", *Information and Control*, v. 8, n. 3, pp. 338 – 353, 1965. ISSN: 0019-9958. doi: https://doi.org/10.1016/S0019-9958(65)90241-X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001999586590241X>.

MAMDANI, E. H. "Application of fuzzy algorithms for control of simple dynamic plant". In: *Proceedings of the institution of electrical engineers*, v. 121, pp. 1585–1588. IET, 1974.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: a simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, v. 15, n. 1, pp. 1929–1958, 2014.

ROCKTÄSCHEL, T. *Combining Representation Learning with Logic for Language Processing*. Tese de Doutorado, University College London, Gower Street, London WC1E 6BT, United Kingdom, 2017.