PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Milton Santos, s/n - Ondina
Salvador, BA, Brasil, 40170-110

https://pgcomp.ufba.br
pgcomp@ufba.br

A compreensão temporal em linguagem natural desempenha um papel fundamental na eficácia da comunicação, permitindo a compreensão da sequência e ordem dos eventos. Este estudo tem como objetivo desenvolver um método computacional para a identificação de tipos de relações temporais entre evento e expressão temporal em textos em português. A abordagem adotada baseia-se em regras e incorpora elementos linguísticos, incluindo informações lexicais, morfossintáticas e contextuais, tempos verbais de Reichenbach, sinais temporais e conhecimento prévio sobre o mundo, além das anotações TimeML presentes do *corpus* TimeBankPT. O método consistiu na criação de um conjunto abrangente de *features* relevantes, que foram utilizadas na construção de conjuntos de regras. Foram explorados algoritmos de aprendizagem de regras, como `CBA`, `CN2`, `IDS` e `RIPPER`, além de regras manuais. Os conjuntos de regras foram aplicados individualmente, bem como em combinação, aos pares compostos por evento e expressão temporal, utilizando duas estratégias de aplicação: a primeira regra acionada e um sistema de votação. Destaca-se que este é o primeiro trabalho que conhecemos a empregar técnicas de aprendizagem de regras para solucionar essa tarefa específica. Os resultados estatísticos mostraram a eficácia da abordagem baseada em regras, destacando-se o conjunto de regras gerado pelo algoritmo `RIPPER`, que obteve o melhor desempenho. Esse conjunto de regras superou o método de referência, alcançando uma acurácia de 69,2% e um *F1-score* de 66,1%. Houve um aumento significativo de 2,3 pontos percentuais em acurácia e 3,6 pontos percentuais em *F1-score* nos dados de teste. A aplicação dos conjuntos de regras pelo sistema de votação foi mais eficaz em dados desconhecidos. A diferença significativa entre os conjuntos de regras e o *baseline* utilizado destaca a importância das *features* adicionais empregadas pelas regras na identificação das relações temporais. Essas *features* forneceram informações complementares e permitiram uma análise mais precisa dos dados. Além disso, os conjuntos de regras demonstraram capacidade de generalização, capturando padrões e regularidades nos dados que podem ser aplicados a novas instâncias, possibilitando previsões precisas. Isso evidencia a utilidade e eficácia dos conjuntos de regras como uma abordagem robusta para lidar com a complexidade das relações temporais em textos. Essa pesquisa contribui para o avanço do processamento de linguagem natural, proporcionando uma compreensão aprimorada e explicável das relações temporais. Também possui aplicações práticas em áreas como descrição de cenas, compreensão de histórias, resumo de documentos, representação da estrutura temporal de prontuários médicos e análise de notícias. A continuidade desse trabalho pode desvendar novas possibilidades para a compreensão temporal em textos.

Palavras-chave: Relações Temporais; *Event-Time*; TimeBankPT; Extração de Informação; Baseada em Regras; Aprendizagem de Regras; Classificação Associativa.

MSC | 161 | 2023

Identificação de Tipos de Relações Temporais *Event-Time* em Português: Uma Abordagem Baseada em Regras com Classificação Associativa

Dárcio Santos Rocha

UFBA

Identificação de Tipos de Relações Temporais *Event-Time* em Português: Uma Abordagem Baseada em Regras com Classificação Associativa

Dárcio Santos Rocha

Dissertação de Mestrado

Universidade Federal da Bahia

Programa de Pós-Graduação em Ciência da Computação

Agosto | 2023

Universidade Federal da Bahia
Instituto de Computação

Programa de Pós-Graduação em Ciência da Computação

IDENTIFICAÇÃO DE TIPOS DE RELAÇÕES TEMPORAIS *EVENT-TIME* EM PORTUGUÊS: UMA ABORDAGEM BASEADA EM REGRAS COM CLASSIFICAÇÃO ASSOCIATIVA

Dárcio Santos Rocha

DISSERTAÇÃO DE MESTRADO

Salvador
14 de agosto de 2023

DÁRCIO SANTOS ROCHA

IDENTIFICAÇÃO DE TIPOS DE RELAÇÕES TEMPORAIS *EVENT-TIME* EM PORTUGUÊS: UMA ABORDAGEM BASEADA EM REGRAS COM CLASSIFICAÇÃO ASSOCIATIVA

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Marlo Vieira dos Santos e Souza
Co-orientadora: Daniela Barreiro Claro

Salvador
14 de agosto de 2023

Ficha catalográfica elaborada pela Biblioteca Universitária de Ciências e Tecnologias
Prof. Omar Catunda, SIBI – UFBA.

R672 Rocha, Dárcio Santos.
Identificação de Tipos de Relações Temporais *Event-Time* em Português: Uma Abordagem Baseada em Regras com Classificação Associativa / Dárcio Santos Rocha – Salvador, 2023.
141 f.

Orientador: Prof. Dr. Marlo Vieira dos Santos e Souza.
Co-orientadora: Prof[a]. Dr[a]. Daniela Barreiro Claro.
Dissertação (Mestrado) – Universidade Federal da Bahia, Instituto de Computação, 2023.

1. Computação. 2. Linguagem Natural. 3. Extração de Informações. I. Souza, Marlo Vieira dos Santos e. II. Claro, Daniela Barreiro. III. Universidade Federal da Bahia. IV. Título.

CDU 004

TERMO DE APROVAÇÃO

DÁRCIO SANTOS ROCHA

IDENTIFICAÇÃO DE TIPOS DE RELAÇÕES TEMPORAIS *EVENT-TIME* EM PORTUGUÊS: UMA ABORDAGEM BASEADA EM REGRAS COM CLASSIFICAÇÃO ASSOCIATIVA

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 14 de agosto de 2023

Documento assinado digitalmente
gov.br
MARLO VIEIRA DOS SANTOS E SOUZA
Data: 23/10/2023 20:06:25-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Marlo Vieira dos Santos e Souza
Universidade Federal da Bahia

Documento assinado digitalmente
gov.br
RERISSON CAVALCANTE DE ARAUJO
Data: 18/10/2023 22:28:43-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rerisson Cavalcante de Araújo
Universidade Federal da Bahia

Documento assinado digitalmente
gov.br
ROBESPIERRE DANTAS DA ROCHA PITA
Data: 22/10/2023 17:50:35-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Robespierre Dantas da Rocha Pita
Universidade Federal da Bahia

Dedico este sonho realizado a minha amada esposa, que cuidou de nossos filhos incansavelmente durante todo o longo processo de preparo para este momento.

AGRADECIMENTOS

Neste momento de conclusão do meu mestrado, gostaria de expressar meus mais sinceros agradecimentos às pessoas e instituições que contribuíram para o sucesso desta dissertação.

Em primeiro lugar, sou profundamente grato a Deus, cuja força inabalável me acompanhou ao longo deste caminho, tornando-me resiliente e determinado em alcançar esse sonho. Sua presença foi fundamental para superar os desafios e obstáculos que surgiram durante todo o processo.

Gostaria de estender meus agradecimentos ao meu orientador, Marlo Souza, pelo seu suporte, disponibilidade e orientações precisas e eficazes. Sua expertise e comprometimento foram essenciais para o meu desenvolvimento como pesquisador, e sou imensamente grato por todo o conhecimento compartilhado.

À minha coorientadora, Daniela Claro, expresso minha gratidão pela paciência, apoio contínuo e opiniões construtivas.

À minha amada esposa e meus filhos, dedico uma imensa gratidão. Foram eles que me inspiraram e motivaram ao longo desta jornada. Seu amor, compreensão e apoio foram fundamentais para que eu pudesse me dedicar a este projeto, mesmo diante de inúmeros desafios e sacrifícios pessoais.

Agradeço também aos meus pais, pelo acolhimento e pelo suporte oferecido em todos os momentos desta trajetória.

À Universidade Estadual do Sudoeste da Bahia (UESB), expresso minha gratidão pela política interna de valorização de pessoal, da qual fui beneficiado com a liberação para cursar este mestrado. Agradeço pela confiança em investir no meu crescimento profissional e estou comprometido em retribuir, aplicando os conhecimentos adquiridos em minha atuação na instituição.

Por fim, gostaria de expressar minha gratidão a todos aqueles que, de alguma forma, contribuíram para a conclusão desta etapa importante da minha vida.

A todos vocês, o meu mais profundo reconhecimento e gratidão.

O importante consiste em estar pronto para, a qualquer momento, sacrificar o que somos pelo que podemos vir a ser.

—CHARLES DUBOIS

RESUMO

A compreensão temporal em linguagem natural desempenha um papel fundamental na eficácia da comunicação, permitindo a compreensão da sequência e ordem dos eventos. Este estudo tem como objetivo desenvolver um método computacional para a identificação de tipos de relações temporais entre evento e expressão temporal em textos em português. A abordagem adotada baseia-se em regras e incorpora elementos linguísticos, incluindo informações lexicais, morfossintáticas e contextuais, tempos verbais de Reichenbach, sinais temporais e conhecimento prévio sobre o mundo, além das anotações TimeML presentes do *corpus* TimeBankPT. O método consistiu na criação de um conjunto abrangente de *features* relevantes, que foram utilizadas na construção de conjuntos de regras. Foram explorados algoritmos de aprendizagem de regras, como `CBA`, `CN2`, `IDS` e `RIPPER`, além de regras manuais. Os conjuntos de regras foram aplicados individualmente, bem como em combinação, aos pares compostos por evento e expressão temporal, utilizando duas estratégias de aplicação: a primeira regra acionada e um sistema de votação. Destaca-se que este é o primeiro trabalho que conhecemos a empregar técnicas de aprendizagem de regras para solucionar essa tarefa específica. Os resultados estatísticos mostraram a eficácia da abordagem baseada em regras, destacando-se o conjunto de regras gerado pelo algoritmo `RIPPER`, que obteve o melhor desempenho. Esse conjunto de regras superou o método de referência, alcançando uma acurácia de 69,2% e um *F1-score* de 66,1%. Houve um aumento significativo de 2,3 pontos percentuais em acurácia e 3,6 pontos percentuais em *F1-score* nos dados de teste. A aplicação dos conjuntos de regras pelo sistema de votação foi mais eficaz em dados desconhecidos. A diferença significativa entre os conjuntos de regras e o *baseline* utilizado destaca a importância das *features* adicionais empregadas pelas regras na identificação das relações temporais. Essas *features* forneceram informações complementares e permitiram uma análise mais precisa dos dados. Além disso, os conjuntos de regras demonstraram capacidade de generalização, capturando padrões e regularidades nos dados que podem ser aplicados a novas instâncias, possibilitando previsões precisas. Isso evidencia a utilidade e eficácia dos conjuntos de regras como uma abordagem robusta para lidar com a complexidade das relações temporais em textos. Essa pesquisa contribui para o avanço do processamento de linguagem natural, proporcionando uma compreensão aprimorada e explicável das relações temporais. Também possui aplicações práticas em áreas como descrição de cenas, compreensão de histórias, resumo de documentos, representação da estrutura temporal de prontuários médicos e análise de notícias. A continuidade desse trabalho pode desvendar novas possibilidades para a compreensão temporal em textos.

Palavras-chave: Relações Temporais; *Event-Time*; TimeBankPT; Extração de Informação; Baseada em Regras; Aprendizagem de Regras; Classificação Associativa.

ABSTRACT

Temporal understanding in natural language plays a fundamental role in communication effectiveness, enabling the comprehension of sequence of event and their order. This study aims to develop a computational method for identifying types of temporal relations between event and temporal expression in Portuguese texts. The adopted approach is rule-based and incorporates linguistic elements, including lexical, morphosyntactic, and contextual information, Reichenbach's tenses, temporal signals, and prior world knowledge, in addition to TimeML annotations from the TimeBankPT corpus. The method consisted in creating a comprehensive set of relevant features used to construct rule sets. We explore rule learning algorithms such as `CBA`, `CN2`, `IDS`, `RIPPER`, and manual rules. The rule sets were applied individually, as well as in combination, to pairs composed of an event and a temporal expression, using two application strategies: the first triggered rule and a voting system. It is worth noting that this is the first work we are aware of to employ rule-learning techniques to solve this specific task. The statistical results showed the effectiveness of the rule-based approach, with the rule set generated by the `RIPPER` algorithm standing out and achieving the best performance. This rule set outperformed the baseline method, achieving an accuracy of 69.2% and an F1-score of 66.1%. There was a significant increase of 2.3 percentage points in accuracy and 3.6 percentage points in F1-score on the test data. The application of rules by the voting system was more effective on unseen data. The significant difference between the rule sets and the baseline used highlights the importance of the additional features employed by the rules in identifying temporal relations. These features provided complementary information and allowed for a more precise analysis of the data. Furthermore, the rule sets demonstrated generalization ability, capturing patterns and regularities in the data that can be applied to new instances, enabling accurate predictions. This underscores the utility and effectiveness of rule sets as a robust approach to dealing with the complexity of temporal relations in texts. This research contributes to the advancement of natural language processing, providing an enhanced and explainable understanding of temporal relations. It also has practical applications in areas such as scene description, story comprehension, document summarization, representation of temporal structure in medical records, and news analysis. The continuation of this work can unveil new possibilities for temporal understanding in texts.

Keywords: Temporal Relations; Event-Time; TimeBankPT; Information Extraction; Rule-based; Rule Learning; Associative Classification.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACE	Automatic Content Extraction
ANOVA	Analysis of Variance
AUC	Area under the ROC curve
CARs	Class Association Rules
CBA	Classification based on Associations
DCT	Data de Criação do Documento
DLS	Deterministic Local Search
E	Event time
EI	Extração da Informação
IDS	Interpretable Decision Sets
I-REP	Incremental Reduced Error Pruning
OvA	One-vs-All
PLN	Processamento de Linguagem Natural
POS	Part-of-speech
R	Reference time
RFECV	Recursive Feature Elimination with Cross-Validation
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
S	Speech time
SLS	Smooth Local Search
SRL	Semantic Role Labeler

Capítulo 1

INTRODUÇÃO

A compreensão do tempo em texto escrito em linguagem natural é de grande importância para uma comunicação eficaz e para transmitir planos e histórias. É essencial entender as informações temporais contidas no discurso, uma vez que muitas atividades humanas estão intrinsecamente ligadas ao tempo. Compreender quando e em que ordem as coisas aconteceram é fundamental para entender eventos descritos em linguagem natural. Assim, a identificação dos diferentes tipos de relações temporais é crucial para entender eventos descritos em um texto, revelando a cronologia dos eventos e explicitando a ordem temporal em que ocorrem.

De acordo com Pustejovsky et al. (2010), para uma interpretação completa de uma declaração em linguagem natural, é necessário compreender as informações temporais transmitidas no texto, o que inclui todos os eventos, expressões temporais e relações temporais presentes. Enquanto os eventos e expressões temporais em um texto têm representações lexicalizadas, as relações temporais que existem entre eles são abstratas. Existem diversas aplicações práticas que se beneficiam do entendimento das relações temporais, como a descrição de cenas, a compreensão de histórias, a descrição de rotas, o resumo de documentos, a representação da estrutura temporal de prontuários médicos e a análise de notícias.

Conforme citado por Verhagen et al. (2010), o objetivo final do processamento temporal é realizar a identificação automática de todos os eventos, expressões temporais e relações temporais presentes em um texto. A identificação das relações temporais pode ser subdividida em identificar o tipo da relação entre: (i) um evento e uma expressão temporal na mesma frase, (ii) um evento e a Data de Criação do Documento (DCT), (iii) dois eventos principais em sentenças consecutivas e (iv) dois eventos, onde um evento domina sintaticamente o outro.

Consideremos o seguinte exemplo extraído do *corpus* TimeBankPT (COSTA; BRANCO, 2012), ilustrado na Figura 1.1. Nessa sentença, na etapa de **extração de expressões temporais** todas as palavras sublinhadas precisam ser reconhecidas, assim, "*este ano*" seria reconhecido como uma expressão temporal do tipo data. Na etapa de **extração de eventos** todas as palavras em negrito precisam ser reconhecidas. Assim, "*Teremos*"

seria reconhecido como um evento da classe Estado e "*diz*" como um evento da classe Reportar. Por fim, na **identificação das relações temporais**, todos os eventos devem ser relacionados a cada expressão temporal, identificando o tipo da relação. Nesse exemplo, o tipo da relação entre o par ("*Teremos*", "*este ano*") é de sobreposição, assim como entre o par ("*diz*", "*este ano*").

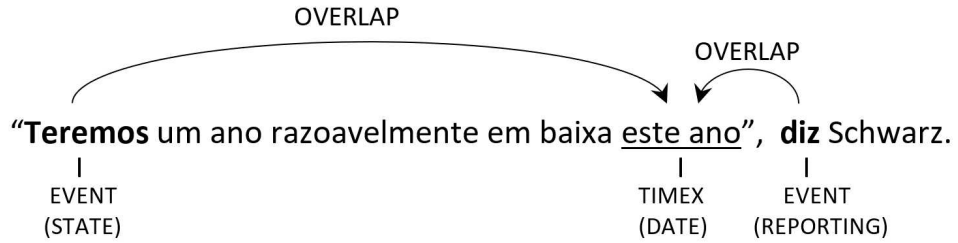


Figura 1.1: Sentença ilustrando o processamento temporal identificando uma expressão temporal "*este ano*", dois eventos "*Teremos*" e "*diz*", além de duas relações temporais do tipo sobreposição entre os eventos e a expressão temporal ("*Teremos*", "*este ano*") e ("*diz*", "*este ano*").

Fonte: Elaborado pelo autor.

A tarefa de identificar os tipos de relações temporais ainda é um problema difícil. Conforme Derczynski (2017), apesar das muitas abordagens sofisticadas e mesmo após muitos anos de esforço, os níveis de desempenho atual atingem cerca de 70% de acurácia. Em complemento a este fato, Setzer, Gaizauskas e Hepple (2005) constatam que essa tarefa é difícil também para humanos, fato corroborado pelo baixo acordo inter-anotadores para a anotação de relações temporais no *corpus* TimeBank (PUSTEJOVSKY et al., 2003).

O TimeBank 1.2 utiliza um conjunto mais completo de tipos de relações temporais, ao todo são 14 tipos baseados no trabalho de Allen (1983). A anotação dessas relações foi dividida em duas etapas, a primeira identifica quais pares de entidades deveriam ser relacionados e a segunda identifica o tipo da relação dos pares identificados. Segundo Derczynski (2017), o acordo inter-anotador para identificar os pares das relações, no geral, foi muito baixa, aproximadamente 55%. Já para identificar o tipo da relação foi de 77% - não tão baixo, mas abaixo do recomendado que é de 90%, segundo Hovy et al. (2006). Comparando com a versão desse *corpus* para o TempEval-1 (VERHAGEN et al., 2007), no qual o conjunto de tipos de relações foi mais simples, apenas 3 tipos principais, o acordo inter-anotador para identificar o tipo da relação entre evento e expressão temporal (*event-time*) foi de 72% e para relações entre dois eventos (*event-event*) foi de 65%.

Dessa forma, o estado atual do desempenho geral, que é de cerca de 70% de acurácia, fica abaixo do acordo inter-anotador de 77% para a identificação dos tipos de relações temporais. Essa persistente dificuldade na identificação dos tipos de relações temporais indica a necessidade de recorrer a fontes de informação diversificadas. No Capítulo 3, apresentaremos várias combinações e variações de informações lexicais, morfossintáticas, conhecimento prévio sobre o mundo e anotações temporais do *corpus* que foram úteis na resolução do problema abordado neste trabalho, conforme detalhado no Capítulo 4.

Este trabalho tem como foco o avanço do estado da arte na área da identificação automática dos tipos de relações temporais entre um evento e uma expressão temporal na mesma frase. Essa tarefa é de suma relevância e apresenta desafios significativos, uma vez que demanda a compreensão das sutilezas temporais na linguagem humana. Diante desse contexto, a presente pesquisa oferece importantes contribuições para o campo do processamento de linguagem natural.

Ao propor um conjunto abrangente de *features* que incorporam informações linguísticas relevantes e ao disponibilizar conjuntos de regras de fácil interpretação, esta pesquisa busca não apenas desenvolver um modelo eficaz na identificação de relações temporais na língua portuguesa, mas também proporcionar uma compreensão clara e transparente dos critérios adotados para tal identificação. Tal abordagem possibilita que especialistas em linguística possam analisar e discutam as decisões do sistema, contribuindo de maneira substancial para o avanço da interpretabilidade no que concerne à identificação dos diferentes tipos de relações temporais.

A interpretabilidade é um tema de grande relevância e interesse na comunidade científica de Inteligência Artificial (IA). A capacidade de compreender e explicar as decisões tomadas por modelos de IA é fundamental, não somente para garantir a transparência e a confiabilidade desses sistemas, mas também para viabilizar a análise crítica por parte de especialistas com conhecimento linguístico relevante. Neste contexto, este trabalho adota uma abordagem de considerável significância, ao optar por uma metodologia baseada em regras, as quais, por natureza, são altamente interpretáveis.

Adicionalmente, outra contribuição relevante é a divulgação dos principais resultados em conferências científicas de grande relevância. Essa divulgação proporciona uma oportunidade única de compartilhar os avanços e descobertas alcançados neste estudo com a comunidade científica, promovendo a disseminação do conhecimento e estimulando discussões acadêmicas pertinentes no âmbito do processamento de linguagem natural. Com isso, busca-se impulsionar o desenvolvimento da área e fomentar novas investigações e inovações que possam beneficiar a compreensão e o processamento automático da linguagem natural em diversos domínios e aplicações práticas.

1.1 PROBLEMA DE PESQUISA

Segundo Derczynski (2017), a simples identificação de expressões temporais e eventos, que são as entidades fundamentais do raciocínio temporal, dentro do discurso da linguagem natural não é suficiente para compreender sua estrutura temporal. Para estabelecer relações temporais, esses eventos devem estar relacionados a outros eventos ou expressões temporais.

Nosso problema de pesquisa é como desenvolver métodos computacionais para identificar automaticamente o tipo de relação temporal entre evento e expressão temporal (*event-time*) em sentenças em língua portuguesa, utilizando uma abordagem baseada em regras. Para descobrir regras, utilizamos técnicas de aprendizado de regras descritas na Seção 2.6. Em nossos experimentos iniciais, propomos um conjunto de regras para o português, utilizando informações lexicais, morfossintáticas e posicionais, além de informações anotadas no *corpus*, baseando-nos no trabalho de D'Souza (2015) para a língua

inglesa. Até onde temos conhecimento, este trabalho é o primeiro a explorar técnicas de aprendizado de regras para resolver o problema proposto.

Utilizamos dados provenientes do *corpus* TimeBankPT (COSTA; BRANCO, 2012), que contém anotações de um conjunto simplificado de três rótulos principais para os tipos de relações temporais: "BEFORE" (Antes), "AFTER" (Depois) e "OVERLAP" (Simultâneo). Além desses rótulos, também é utilizado o valor "VAGUE" (Vago) quando não é possível estabelecer uma relação temporal específica. Adicionalmente, são utilizados dois valores para relações disjuntivas menos específicas: "BEFORE-OR-OVERLAP" (Antes ou Sobreposição) e "OVERLAP-OR-AFTER" (Sobreposição ou Depois), que são empregados em casos ambíguos. O *corpus* TimeBankPT é uma tradução para o português do *corpus* TimeBank (PUSTEJOVSKY et al., 2003).

1.2 HIPÓTESE

A hipótese examinada neste trabalho pode ser formalizada pela seguinte proposição:

A combinação de conjuntos de regras induzidos por algoritmos distintos, conforme discutidos na Seção 2.6.2, resulta em desempenho superior em comparação com os resultados individuais de cada conjunto de regras ao identificar tipos de relações temporais na língua portuguesa.

1.3 JUSTIFICATIVA

Determinar relações temporais é de suma importância para compreender a disposição temporal dos eventos descritos no discurso. Ao identificar automaticamente as relações temporais entre eventos e expressões temporais, técnicas poderosas se tornam disponíveis para aprimorar a interação entre humanos e computadores, bem como o processamento automático de informações armazenadas em linguagem natural. De acordo com Derczynski (2017), sistemas poderiam obter um desempenho melhor ao responder perguntas formuladas em linguagem natural, facilitar a criação de resumos mais precisos de textos, automatizar o trabalho em aplicações forenses, como a construção de cronogramas de depoimentos de testemunhas em casos de crimes, e utilizar a extração de informações temporais na comunicação cotidiana para organizar eventos e criar compromissos automaticamente em agendas eletrônicas a partir de comunicações pessoais.

A adoção de uma abordagem baseada em regras é motivada principalmente por dois fatores. Primeiro, um método baseado em regras, conforme afirma D'Souza (2015), pode melhor aproveitar os *insights* humanos para combinar os recursos linguísticos disponíveis e formular regras de decisão mais eficientes que não poderiam ser capturadas por métodos de aprendizado de máquina. O segundo fator é que os sistemas de extração de informações baseados em regras, como mencionado por Chiticariu, Li e Reiss (2013), são valorizados por sua facilidade de adoção, maior compreensibilidade e interpretação, bem como pela facilidade de manutenção e correção de erros, embora exijam um trabalho manual tedioso.

Quanto à escolha de utilizar técnicas de aprendizado de regras, isso tornará menos complexa a tarefa de descobrir relações no conjunto de *features* que possam ser relevantes para a criação de regras capazes de identificar eficientemente tipos de relações tempo-

rais. A quantidade de informações na base de dados tornaria altamente dispendioso e ineficiente realizar esse trabalho de forma totalmente manual, especialmente devido à explosão exponencial gerada pelas combinações de todos as *features* disponíveis.

1.4 OBJETIVOS

O presente estudo tem como objetivo o desenvolvimento de um método computacional para a identificação automática de tipos de relações temporais entre pares de entidades evento/expressão temporal em textos escritos em língua portuguesa, adotando uma abordagem baseada em regras.

Para alcançar tal objetivo, foram estabelecidos os seguintes objetivos específicos:

- Estudar o estado da arte na identificação de tipos de relações temporais;
- Investigar a aplicação de técnicas de aprendizado de regras para identificar regras com base em diferentes informações linguísticas;
- Propor conjuntos de regras para identificar o tipo de relação temporal entre um evento e uma expressão temporal.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

O restante do texto deste trabalho está organizado da seguinte forma:

- O Capítulo 2 apresenta os tópicos essenciais desse projeto: os conceitos de eventos, de expressões temporais e de relações temporais presentes na literatura, o *corpus*, o esquema de anotação TimeML e a estratégia de resolução do problema por meio de técnicas de aprendizagem de regras.
- O Capítulo 3 apresenta alguns trabalhos relacionados, permitindo ter uma visão geral do estado-da-arte em contextos específicos relacionados ao escopo, bem como as principais informações linguísticas utilizadas para resolver o problema proposto, além de trabalho específico da língua portuguesa.
- O Capítulo 4 apresenta o método baseado em regras para identificar tipos de relações temporais entre eventos e expressões temporais que ocorrem na mesma frase. Ele descreve as informações linguísticas utilizadas e a abordagem adotada para sua composição, bem como os procedimentos para gerar conjuntos de regras que identificam diferentes tipos de relações temporais. O capítulo também explora as formas de aplicação dos conjuntos de regras nos pares *event-time* para identificar relações temporais específicas. Além disso, é apresentada uma estratégia de aumento de dados que calcula o fechamento temporal de todos os pares *event-time* identificados, juntamente com as relações correspondentes.
- O Capítulo 5 apresenta a configuração experimental adotada neste estudo, que utilizou o *corpus* TimeBankPT. São descritas as métricas de avaliação utilizadas para medir o desempenho do método em comparação com o *baseline* estabelecido. Além

disso, o capítulo fornece uma descrição detalhada dos experimentos conduzidos, incluindo os procedimentos e abordagens adotadas.

- O Capítulo 6 expõe os resultados dos experimentos conduzidos nos conjuntos de dados de treinamento e teste com o objetivo de identificar tipos de relação temporal *event-time* dentro da mesma sentença. São destacados os principais elementos de cada conjunto de regras. São apresentados os resultados da seleção otimizada dos hiperparâmetros para cada conjunto de regras, assim como os resultados obtidos ao aplicá-los nos dados do *corpus*.

- O Capítulo 7 apresenta as conclusões deste trabalho, abordando as contribuições alcançadas e as principais descobertas. Além disso, serão destacadas as limitações encontradas durante a realização do estudo e as oportunidades para trabalhos futuros. Este capítulo encerra o trabalho, fornecendo uma visão abrangente dos resultados alcançados e estabelecendo bases para a continuidade da pesquisa na área de identificação de tipos de relações temporais em textos escritos em língua portuguesa.

Capítulo 2

REFERENCIAL TEÓRICO

A identificação de tipos de relações temporais é uma tarefa de suma importância no campo da Extração da Informação (EI). A EI consiste em converter informações não estruturadas presentes em textos para formatos de dados estruturados, conforme mencionado por Jurafsky e Martin (2000). Segundo Verhagen et al. (2007), a tarefa de identificar tipos de relações temporais envolve a identificação automática de todas as referências temporais presentes em um texto, incluindo eventos, expressões e relações temporais.

O desafio TempEval foi desenvolvido como um *framework* para avaliar sistemas que realizam anotações automáticas de relações temporais em textos. Esse desafio foi criado no contexto do *workshop* internacional de avaliações semânticas, SemEval-2007 (VERHAGEN et al., 2009). O objetivo principal desse desafio era identificar tipos de relações temporais e contribuir para o avanço da pesquisa na área de processamento de informações temporais.

No restante deste capítulo, serão apresentados alguns conceitos fundamentais para a pesquisa em questão. Na Seção 2.1, discutiremos o conceito de evento, suas diferentes classes e apresentaremos exemplos. Na Seção 2.2, abordaremos o conceito de expressões temporais, bem como seus principais atributos. Na Seção 2.3, nos deteremos sobre a tarefa de identificação de tipos de relações temporais, incluindo os principais conjuntos de tipos e as diferentes formas pelas quais a linguagem pode expressar essas relações. Na Seção 2.4, apresentaremos o *corpus* utilizado neste trabalho, o TimeBankPT, e discutiremos as principais decisões tomadas durante o processo de tradução. Na Seção 2.5, será apresentado o esquema de anotação de entidades temporais utilizado, conhecido como TimeML. Por fim, na Seção 2.6, abordaremos as técnicas de aprendizagem de regras.

2.1 EVENTOS

Os eventos compõem as relações temporais e sua classificação pode ser realizada por meio do termo eventualidade, utilizado na literatura para se referir tanto a eventos (não-estativos) quanto a estados (estativos), conforme introduzido por Bach (1986). Um estado é caracterizado por uma eventualidade na qual não ocorre mudança relevante durante o

período de tempo em que é verdadeiro, como conhecer alguém ou ser feliz. Por outro lado, um evento é uma eventualidade que envolve uma mudança de estado, como aprender uma língua ou construir uma casa. Eles podem ser definidos como *"algo que acontece e pode frequentemente ser descrito como uma mudança de estado."* (ACE, 2005, p. 5, tradução nossa). Por outro lado, Guizzardi et al. (2013, p. 333, tradução nossa) os definem como *"transformações de uma parte da realidade para outra, isto é, eles podem mudar a realidade mudando o estado das coisas de uma situação para outra"*. Neste trabalho, o termo evento é utilizado para se referir àquilo que, de outra forma, estaria incluído no termo eventualidade, englobando, portanto, também os estados.

No que tange à especificação da linguagem TimeML, Pustejovsky et al. (2004, p. 4, tradução nossa) afirmam que *"os eventos podem ser pontuais ou durarem um período de tempo"*. Ademais, salientam que *"eles são geralmente expressos por meio de verbos, nominalizações, adjetivos, orações predicativas ou preposicionais"*. A título de exemplo, evento expresso por **verbo** pode ser observado em (1), enquanto evento **nominalizado** pode ser encontrado em (2), em que "*explosão*" é um substantivo que descreve um evento. Já um evento expresso por **adjetivo** pode ser identificado em (3), enquanto um evento descrito por meio de **predicativo** pode ser observado em (4). Estes exemplos foram adaptados de Derczynski (2017), que também acrescenta que eventos podem ser expressos por meio de **estativos**, como em (3) e (4).

(1) *O ônibus* ***parou*** *repentinamente.*

(2) *A* ***explosão*** *da usina nuclear foi um desastre.*

(3) *A tempestade está* ***ativa****.*

(4) *Elizabeth é* ***rainha****.*

De acordo com a pesquisa realizada por Marsic (2011), a maioria dos linguistas associa os eventos com o tempo verbal, o qual é considerado central para a sentença. Por essa razão, a análise dos eventos geralmente se concentra nas propriedades do verbo. Neste contexto, investigamos as classes gramaticais das anotações do *corpus* TimeBankPT. Observamos que, em 64,5% dos casos, os eventos são expressos por verbos, em 28,2% dos casos por substantivos, em 2,7% por adjetivos, em 0,8% por preposições e em 0,2% por advérbios. A categoria *OTHER* representa 3,6% dos casos e consiste principalmente em expressões numéricas.

A anotação completa de um evento requer a execução de duas tarefas: (i) identificação do evento, que consiste na determinação das palavras ou expressões que podem ser marcadas como eventos, e (ii) classificação do evento, que envolve a atribuição de uma classe específica de acordo com um esquema apresentado a seguir, como descrito em Derczynski (2017), Marsic (2011), Sauri et al. (2006):

REPORTING Esses eventos são de algum ator repassando informações sobre outros eventos ou estados, de modo que sua função é associar a fonte de informação ao evento relatado.

Por exemplo, "João **disse** que comprou um pouco de vinho."

Exemplos de verbos: dizer, relatar, contar, explicar e declarar.

PERCEPTION São eventos que descrevem a observação de algum outro evento.

Por exemplo: "Maria **viu** João carregando apenas cerveja."

Exemplos de verbos: ver, observar, vislumbrar, contemplar, ver, ouvir, escutar, exagerar e descobrir.

ASPECTUAL São aquelas expressões que descrevem certas partes da vida de um evento, como seu início, reinício, encerramento, auge, continuação e assim por diante.

Por exemplo: "Os cientistas estavam **começando** a mostrar sinais de exaustão."

Exemplos de verbos: iniciar, começar, arrancar, reiniciar, recomeçar, reativar, encerrar, parar, terminar, interromper, desistir, culminar, atingir, continuar, manter, prosseguir, avançar e perseverar.

I ACTION Envolvem algum ator com um objetivo específico (talvez não declarado) em mente, que realiza ações distintas seguindo essa intenção.

Por exemplo: "A Microsoft **tentou** monopolizar o acesso à internet."

Exemplos de verbos: tentar, experimentar, investigar, examinar, adiar e atrasar.

I STATE Inclui estados que se referem a cenários alternativos ou possíveis (delimitados por colchetes) que podem ser introduzidos por cláusulas subordinadas e nominalizações.

Por exemplo: "A Rússia agora **sente** que [os EUA devem adiar pelo menos até que o secretário-geral da ONU, Kofi Annan, visite Bagdá]".

Exemplos de verbos: acreditar, pensar, suspeitar, imaginar, duvidar, sentir, querer, desejar e esperar.

STATE Os Estados descrevem circunstâncias em que algo se mantém verdadeiros.

Por exemplo: "Eles **viveram** na Holanda por 2 anos."

Exemplos de verbos: viver, morar, mediar e buscar.

OCCURRENCE Inclui todos os muitos outros tipos de eventos descrevendo algo que acontece ou ocorre no mundo.

Por exemplo: "John **perdeu** a carteira de identidade."

Exemplos de verbos: pousar, chegar, perder e perceber.

Conforme destacado por Sauri et al. (2006), é importante salientar que verbos podem apresentar ambiguidades em relação às classes de eventos, de modo que não é possível inferir que toda ocorrência de um verbo denote um evento da mesma classe.

Ademais, segundo Derczynski (2017), os eventos não necessitam ser reais e observáveis para serem anotados em um texto. Eventos fictícios, mesmo que inseridos em um contexto imaginário, devem ser considerados na anotação temporal de um documento. O autor ainda acrescenta que descrições de eventos futuros ou relacionados ao contexto condicional de um "se" também devem ser considerados como eventos e, portanto, anotados devidamente.

2.2 EXPRESSÕES TEMPORAIS

As expressões temporais consistem em frases de linguagem natural que se referem diretamente ao tempo, fornecendo informações acerca de quando um evento ocorreu, quanto tempo algo durou ou com que frequência algo aconteceu, conforme relatado por Marsic (2011). Tais expressões podem denotar datas, horas, períodos de tempo, durações ou conjuntos de tempos recorrentes. Por sua vez, Derczynski (2017) define uma expressão temporal como qualquer expressão que denote um momento, intervalo ou outra região temporal sem depender de um evento. Por exemplo, "24 de agosto de 1997", "duas semanas" e "todos os domingos" são todos exemplos de expressões temporais.

Segundo Marsic (2011), existem diversas formas gramaticais para as expressões temporais, dentre as quais destacam-se as frases substantivas que podem ocorrer individualmente ou precedidas por uma preposição, assumindo a forma de frases preposicionais. Além disso, a autora enfatiza que outra maneira frequente de identificar as expressões temporais é por meio de certos advérbios, adjetivos, frases adverbiais ou adjetivas. É comum que uma expressão temporal seja sinalizada por uma ou mais palavras de tempo, também conhecidas como "gatilhos lexicais", tais como:

- **substantivos:** século, ano, mês, dia, fim de semana, minuto, futuro, passado;
- **nomes próprios:** Natal, Pascoa, Dia do Trabalho;
- **adjetivos:** passado, atual, futuro, próximo, medieval, mensal;
- **advérbios:** atualmente, então, semanalmente, hoje, ontem, amanhã, hoje à noite;
- **padrões de tempo específico:** 9:00, 26/12/2002, anos 80;
- **números:** 4 (como em João chegou no dia 4.).

A autora ainda pontua que as expressões temporais desempenham um papel crucial na transmissão de informações relacionadas ao tempo, tais como: posição no tempo, duração temporal e frequência no tempo.

No que diz respeito à **posição no tempo**, as expressões temporais especificam quando algo acontece e, geralmente, servindo como resposta a uma possível pergunta "*Quando?*". Sempre que uma posição no tempo é expressa usando frases preposicionais, frequentemente inclui determinantes, como nos exemplos (5a) e (5b). A posição no tempo também

pode ser expressa por meio de advérbios, os mais frequentes são: agora, então, hoje, atrás, ontem, como em (6).

(5) a. *Maria encontrou-se com ele* ***naquela tarde****.*

b. *O casamento foi* ***na quinta-feira****.*

(6) *João saiu para uma caminhada* ***ontem****.*

Além disso, a posição no tempo pode ser indicada com precisão pela utilização de pontos de tempo, como no exemplo (7), ou vagamente especificada por expressões que delimitam períodos de tempo ou intervalos, como no exemplo (8).

(7) *Nós encontramos a carta ao* ***meio-dia****.*

(8) *Encontramos a carta* ***durante o verão****.*

Outro significado carregado pelas expressões temporais é a **duração temporal**, caso em que elas representam respostas apropriadas para a pergunta "*Quanto tempo?*". As durações oferecem maior liberdade para usar frases substantivas, como no exemplo (9). Expressões de tempo que denotam duração são tipicamente formadas pela junção de um quantificador (por exemplo, vários, três, muitos) com uma unidade de tempo (por exemplo, ano, semana, hora).

(9) *Eles viveram* ***vários anos*** *na Itália.*

De acordo com Marsic (2011), as expressões temporais podem transmitir informações sobre a **frequência no tempo**, descrevendo com que frequência algo ocorre. Essas expressões, em relação a um período de tempo especificado ou implícito, geralmente representam respostas à pergunta "*Com que frequência?*". Para expressar frequência, sintagmas nominais geralmente seguem a construção de "todos os" ou "a cada", seguidos por uma palavra referente ao tempo, como "hora", "dia", "segunda-feira" ou "meses", como exemplificado em (10). Outra maneira de expressar frequência é por meio de adjetivos ou advérbios derivados de unidades de tempo, como "de hora em hora", "mensal" ou "anualmente", como em (11).

(10) *Maria escreve um artigo ou uma resenha* ***todos os meses****.*

(11) *Um boletim informativo* ***mensal*** *é enviado por e-mail a todos os clientes.*

No que tange à tarefa de anotar automaticamente expressões temporais, Derczynski (2017) a decompôs em várias etapas. Nesse sentido, as principais etapas consistem em identificar quais palavras e frases em um documento são expressões temporais e, posteriormente, atribuir-lhes diversos valores de atributos. Uma vez identificada, uma expressão

temporal pode ser convertida em uma data ou intervalo totalmente especificado, o que é conhecido como normalização da expressão temporal. Em outras palavras, a etapa em questão se caracteriza pela atribuição de uma escala temporal absoluta, ou uma aproximada, às expressões previamente identificadas, mediante a utilização de uma notação pré-estabelecida, especificada em Sauri et al. (2006).

Por último, vale destacar que eventos e expressões temporais são considerados os pilares fundamentais da anotação do discurso temporal. Na próxima seção, discutiremos as relações temporais existentes entre eles.

2.3 RELAÇÕES TEMPORAIS

A tarefa de determinar quais pares de eventos ou expressões temporais devem ser conectados é conhecida como "identificação de relações temporais". Já a tarefa de determinar o tipo da relação que se estabelece entre um par específico dessas entidades é chamada de "identificação de tipos de relações temporais". O presente trabalho está relacionado com a tarefa de identificar os tipos de relações temporais.

Conforme destacado por UzZaman et al. (2012), uma relação temporal conecta eventos ou expressões temporais e indica a ordem em que ocorreram, ou se ocorreram simultaneamente. Essas relações podem ser direcionais (quando a ordem entre as entidades é importante) ou não-direcionais (quando apenas a relação de simultaneidade é relevante). De acordo com D'Souza (2015), a tarefa de extração de relações temporais envolve identificar se um par de eventos ou de um evento e uma expressão temporal está em uma relação temporal e, em seguida, classificar o par com um conjunto predefinido de tipos de relações temporais.

A relação temporal entre entidades temporais é abstrata, enquanto eventos e expressões temporais possuem representações lexicalizadas, conforme observado por Derczynski (2017). A ordenação temporal entre eventos e expressões temporais nem sempre é explicitada, o que dificulta a identificação do tipo de relação temporal existente. Portanto, a identificação de tipos de relação temporal continua sendo um desafio, apesar da existência de abordagens sofisticadas. Essa persistência das dificuldades sugere que múltiplas fontes de informação devem ser consideradas para compreender a ordenação temporal dos eventos descritos em um texto.

A expressão dos diferentes tipos de relação temporal na linguagem pode não coincidir com as classes disponíveis em um esquema de anotação. No entanto, é fundamental determinar um conjunto de relações temporais para a realização da extração automática dessas relações. De acordo com a tese de Derczynski (2017), essa definição tem como objetivo auxiliar na inferência e fornecer uma estrutura consistente para a anotação manual. Além disso, o autor argumenta que as álgebras e lógicas temporais possibilitam deduzir relações entre eventos com base em sua conexão com outras expressões temporais e eventos, usando um conjunto de regras específicas que dependem do conjunto de tipos de relação temporal definidos.

Há três conjuntos de tipos de relação temporal amplamente utilizados para a anotação temporal: o conjunto original de Allen (ALLEN, 1983), as relações de intervalo TimeML (SAURI et al., 2006) e o conjunto simplificado TempEval-1 e TempEval-2 (VERHAGEN

et al., 2007; VERHAGEN et al., 2009). A Tabela 2.1 apresenta os três conjuntos de tipos de relação temporal, suas correspondências, relações inversas e a representação gráfica do intervalo correspondente.

Tabela 2.1: Conjuntos de relações temporais do TempEval-1 e 2, de Allen e da TimeML.

Relações ↓			**INTERVALO**	**↑ Relações Inversas**		
TempEval	**Allen**	**TimeML**		**TimeML**	**Allen**	**TempEval**
Before	Before	Before	A|----| B|----|	After	After	After
-	Meets	I-Before	A|-----| B|------|	I-After	Met-by	-
-	-	Includes	A|----------| B|----|	Is-Included	-	-
-	Starts	Begins	A|-----| B|----------|	Begun-by	Started-by	-
-	Finishes	Ends	A|-----| B|----------|	Ended-by	Finishied-by	-
-	During	During	A|----| B|----------|	During-Inv	Contains	-
-	Equal	Simultaneous	A|----------| B|----------|	-	-	-
-	-	Identity	A|==========|B	-	-	-
-	Overlaps	-	A|-------| B|--------|	-	Overlapped-by	-
Overlap	-	-	Qualquer parte de A e B co-ocorrem	-	-	Overlap
Before-or-Overlap	-	-	Disjunção entre Before e Overlap	-	-	-
Overlap-or-After	-	-	Disjunção entre Overlap e After	-	-	-
Vague	-	-	Completamente subespecificada	-	-	-

Fonte: Elaborada pelo autor.

Com o intuito de otimizar a leitura da Tabela 2.1, apresentamos a seguir um exemplo ilustrativo: na primeira linha, à esquerda, está presente a relação temporal Before, indicando que o evento A ocorre antes do evento B, conforme a coluna "intervalo". Já à direita, encontra-se a relação temporal inversa, After, que deve ser lida a partir do evento B, ou seja, que o evento B ocorre depois do evento A.

O conjunto de relações temporais do TempEval-1 e TempEval-2 é simplificado, consistindo de apenas três tipos de relações principais, bem como outros dois tipos formados pela disjunção dos principais e um tipo vago. O número reduzido de tipos de relacionamento torna a visualização das relações mais fácil e simplifica a implementação e análise. Este conjunto é adotado neste trabalho. No entanto, um conjunto tão pequeno pode limitar a representação das relações temporais expressas em línguas naturais. Por outro lado, o conjunto original de Allen define relações suficientes para abranger todas as possíveis relações entre um par de entidades temporais. Finalmente, segundo Derczynski (2017), as relações de intervalo TimeML devem ser interpretadas de forma um pouco menos restrita do que o conjunto de Allen.

Após apresentar os conjuntos de relações temporais presentes na linguagem natural, descreveremos algumas formas pelas quais ela expressa essas relações. Conforme observado por Marsic (2011), um problema comum das línguas naturais é que, frequentemente, elas não expressam diretamente o intervalo de tempo que um evento ocupa na linha do tempo, isto é, seus pontos de início e fim. As relações temporais são geralmente expressas de forma parcial na linguagem natural, por meio de conceitos e contribuições relevantes que serão apresentados a seguir.

Tempo Verbal: É um mecanismo gramatical específico incorporado à linguagem que permite localizar informações em relação ao tempo, seja ele presente, passado ou futuro. Ele está relacionado à capacidade dos verbos de se flexionarem para transmitir informações sobre a localização de um evento no tempo. Por exemplo, a sentença (12a), a forma verbal "dançou" é gerada através da flexão do verbo "dançar" no passado, indicando que o evento ocorreu em um momento anterior ao da fala. Já no exemplo (12b), a forma verbal "dançará" é gerada através da flexão do verbo "dançar" no futuro, sendo utilizada para localizar o evento como ocorrendo em um momento futuro em relação ao tempo da fala.

(12) a. *Maria* ***dançou*** *na festa.*

b. *Maria* ***dançará*** *na festa.*

Aspecto Verbal: É uma categoria gramatical que indica como uma determinada situação é percebida pelo falante em relação ao tempo. Essa categoria está relacionada à visão que o falante tem da ação, podendo ser concebida como completa (perfectiva) ou contínua (imperfectiva ou progressiva). O aspecto verbal também expressa nuances do significado pretendido, já que ele pode indicar a finalização de uma ação (perfectiva) ou a continuidade ou progressão da mesma (imperfectiva ou progressiva).

Por exemplo, em (13a) a forma verbal "comeu" é perfectiva, indicando que a ação de comer foi concluída. Em (13b), a forma verbal "comia" é imperfectiva, indicando que a ação de comer não tem um fim específico e pode ser repetida no futuro. Já em (13c), a forma verbal "estava comendo" é progressiva, indicando que a ação de comer estava em andamento em um determinado momento.

(13) a. *João **comeu** a maçã.*

b. *João **comia** maçãs todos os dias.*

c. *João **estava comendo** uma maçã quando o telefone tocou.*

Reichenbach: Uma das contribuições mais significativas para a compreensão profunda de como as relações temporais são codificadas no texto é o estudo realizado por Reichenbach em seu trabalho intitulado "Elements of Symbolic Logic" (REICHENBACH, 1947). Nesse trabalho, o autor sustenta que expressões marcadas por tempos e aspectos introduzem referências a três momentos de tempo distintos: o momento da fala S, o momento do evento E e o momento de referência R.

O momento da fala S é definido como o instante em que a enunciação é proferida. O momento do evento E, por sua vez, é o tempo em que o evento descrito ocorreu. Por fim, o momento de referência R é o momento a partir do qual o orador está visualizando o evento em uma linha de tempo. Na Seção 3.1.2, será abordado com mais detalhes o *framework* proposto por Reichenbach.

Advérbios de Tempo: São elementos gramaticais que expressam relações temporais entre o tempo denotado e o evento verbal. Eles podem indicar o início, a duração, a repetição, a frequência e a finalização de um evento. Por exemplo, em (14a), o advérbio "hoje" indica que o evento verbal (comer) ocorreu no dia presente em relação ao momento da fala. Em (14b), o advérbio "sempre" indica que o evento verbal (chegar) ocorre com frequência ou regularidade.

(14) a. *Eu comi pizza **hoje**.*

b. *Ele **sempre** chega atrasado.*

Na Seção 3.1.1, serão apresentados os sinais temporais que serão utilizados neste trabalho, incluindo tanto conjunções quanto advérbios temporais.

Nível Sintático: As relações temporais podem ser inferidas examinando certas relações de dependência. Por exemplo, em (15), as **expressões temporais** incluídas em frases nominais para qualificar o substantivo núcleo do sintagma nominal indicam que há uma relação temporal de sobreposição entre o evento denotado pelo núcleo do sintagma nominal e o tempo indicado pela **expressão temporal**. Na Seção A.1.6, apresentamos as informações sintáticas que utilizaremos.

(15) *Eles não sabem o resultado da eleição de **domingo***

Nível Semântico: Um papel importante é desempenhado pelo conhecimento prévio sobre o mundo. Sem esse conhecimento, muitas vezes é impossível saber que um evento representa uma parte integrante de outro evento, ou que um evento causa outro evento. Por exemplo, em (16), a ordem temporal dos eventos inferidos é que o evento de **cometer o crime** ocorreu antes do evento de **ser acusado pelo advogado**. Essa inferência só é possível graças ao conhecimento prévio sobre o mundo de que eventos de acusação seguem os eventos que alguém é acusado.

(16) *O advogado* ***acusou*** *o réu de ter* ***cometido*** *o crime.*

Na Seção 3.2.3, apresentamos uma informação de conhecimento prévio sobre o mundo que utilizaremos nessa pesquisa.

Correferência do Evento: Na linguagem, é comum a referência a eventos previamente mencionados no texto. Esse fenômeno é chamado de correferência de evento, em que eventos são mencionados mais de uma vez em um texto. Nesse caso, podemos inferir que as relações temporais que se aplicam a uma instância do evento são válidas para todas as outras instâncias referidas. Por exemplo, na sentença (17), o evento de "ir ao mercado" é mencionado novamente na segunda frase por meio da correferência "Essa visita", a qual retoma a ação da primeira frase. A partir desse ato de correferenciar, é possível inferir que a relação temporal entre o evento "visita" e a expressão temporal "ontem" é idêntica à relação temporal entre o evento "foi" e a expressão temporal "ontem", ou seja, de sobreposição.

(17) *Maria* ***foi*** *ao mercado ontem. Essa* ***visita*** *teve como propósito a aquisição dos alimentos para o jantar.*

Inferência: É uma importante fonte de relações temporais, pois permite estabelecer relações entre eventos com base em outras relações já conhecidas, utilizando regras simples como a regra de transitividade. Por exemplo, se sabemos que o evento A ocorreu antes do evento B e que o evento B ocorreu antes do evento C, então podemos inferir que o evento A ocorreu antes do evento C.

Na linguagem natural, essa inferência pode ser feita de maneira implícita, como em (18), em que sabemos que o evento de "acordar" ocorreu antes do evento de "tomar banho", e que este ocorreu antes do evento de "sair de casa", permitindo-nos inferir que o evento de "acordar" ocorreu antes do evento de "sair de casa". Essa é uma inferência transitiva com base na relação temporal de antes.

(18) *João* ***acordou****,* ***tomou*** *banho e* ***saiu*** *de casa.*

Na Seção 4.4, serão apresentadas algumas inferências mais detalhadas que serão utilizadas neste trabalho.

Essa foi uma breve visão sobre as relações temporais em textos escritos em linguagem natural, destacando a complexidade da tarefa de identificação de tais relações e de seus respectivos tipos. Conforme apontado por Marsic (2011), embora as relações temporais explícitas presentes no texto, tais como aquelas expressas por meio de tempo verbal, aspecto verbal e advérbios temporais, possam ser identificadas automaticamente, as relações temporais semanticamente implícitas representam desafios reais aos sistemas automáticos devido à necessidade de possuir conhecimento prévio sobre o mundo para sua identificação.

2.4 TIMEBANKPT

Para a tarefa de identificação de tipos de relações temporais na língua portuguesa, há à disposição o *corpus* TimeBankPT. Este *corpus* é composto por artigos contendo anotações não apenas de expressões temporais, mas também de eventos e relações temporais, sendo o primeiro *corpus* disponível para esse idioma.

O TimeBankPT é composto por 182 documentos, sendo cada um deles uma notícia de televisão, telejornais ou jornais. Dentre esses documentos, 134 são edições do *Wall Street Journal* (WSJ) de 1989, o que representa 73,6% do *corpus*. No que diz respeito ao seu conteúdo, predominam os jargões e os dados relacionados ao mercado de ações, tornando o *corpus* bastante específico do domínio.

Apesar de a mídia WSJ corresponder à maior parte dos documentos, os textos presentes no *corpus* abrangem uma ampla variedade de mídias relacionadas ao domínio de notícias e originam-se de diversas fontes, incluindo o *Associated Press Writer* (APW), *Public Radio International* (PRI), *Cable News Network* (CNN), *Voice of America News* (VOA), *American Broadcasting Company* (ABC) e o *The New York Times* (NYT). A quantidade de documentos no *corpus* atribuída a cada fonte, juntamente com a proporção e os principais temas abordados, são apresentados na Tabela 2.2.

Todos os documentos que compõem o TimeBankPT foram traduzidos do TimeBank (PUSTEJOVSKY et al., 2003) por Costa e Branco (2012) e seu esquema de anotação segue a linguagem de marcação TimeML (PUSTEJOVSKY et al., 2004), a qual discutiremos na próxima seção.

Ao longo do processo de tradução, os autores foram confrontados com a necessidade de efetuar algumas escolhas acerca das particularidades do idioma. Uma questão levantada diz respeito à anotação de expressões temporais que se apresentam como frases preposicionais. Seguindo as diretrizes estabelecidas no padrão TimeML, determinou-se que a preposição não deve ser incluída dentro das *tags* `TIMEX3`. Por conseguinte, na língua inglesa, a preposição foi posicionada externamente, ao passo que o determinante foi acomodado internamente.

No caso do idioma português, surge a questão de como tratar as contrações de preposições com determinantes dentro das *tags* de marcação de expressões temporais. No *corpus* TimeBankPT, os autores optaram por deixar a preposição e o determinante fora das *tags*, diferindo da lógica adotada para o inglês. Por exemplo, para a frase preposicional em inglês "*from the night*", a expressão temporal anotada foi "*the night*", enquanto em português, para a frase "*da noite*", a expressão temporal anotada foi apenas "*noite*",

Tabela 2.2: Distribuição da quantidade de documentos e percentual no *corpus* TimeBankPT por fontes de notícias e principais temas abordados

Fonte	**Quant Docs**	**% *Corpus***	**Principais Temas Abordados**
Wall Street Journal (WSJ)	134	73,6	Jargões e dados do mercado de ações e eventos econômicos
Associated Press Writer (APW)	18	9,9	Geopolítica, conflitos regionais, diplomacia e eventos internacionais
Public Radio International (PRI)	8	4,4	Eventos políticos, sociais, de segurança e judiciais
Cable News Network (CNN)	5	2,7	Geopolítica, relações internacionais, economia e questões militares
Voice of America News (VOA)	5	2,7	Questões políticas, sociais e de saúde em várias partes do mundo
American Broadcasting Company (ABC)	4	2,2	Política, economia, finanças, aviação e exploração espacial
The New York Times (NYT)	4	2,2	Segurança pública, economia, política e questões sociais
Outras	4	2,2	Questões militares, eventos internacionais, condições climáticas, política e visita papal em Cuba
Total Documentos	**182**	**100**	

Fonte: Elaborada pelo autor.

deixando a contração da preposição com o determinante fora da *tag* `TIMEX3`. Já em contraste com essa decisão, os dados do *corpus* HAREM[1] incluíram as preposições e as contrações dentro dos elementos de marcação de expressões temporais.

Uma decisão adicional importante tomada por Costa e Branco (2012) no processo de tradução do *corpus*, foi referente ao atributo `ASPECT`, o qual codifica o aspecto gramatical. A escolha foi de restringir o valor do atributo ao PROGRESSIVE, enquanto que os valores PERFECTIVE e PERFECTIVE_PROGRESSIVE foram codificados como NONE. Essa decisão se baseou no fato de que expressões em português que são formadas de

[1]O HAREM é uma avaliação conjunta na área do reconhecimento de entidades mencionadas em português organizado pela Linguateca. É uma iniciativa que pretende avaliar o sucesso na identificação e consequente classificação automática dos nomes próprios na língua portuguesa. Disponível em <https://www.linguateca.pt/HAREM>.

maneira similar ao *perfect* em inglês (combinadas com um particípio passado) não têm necessariamente o mesmo valor semântico, e, portanto, não foram anotadas como tendo aspecto perfeito.

A decisão de Costa e Branco (2012) baseou-se nos estudos de Portner (2003), que argumenta que o aspecto gramatical *perfective* em inglês é utilizado para denotar uma relação temporal entre um evento e um ponto específico no tempo em que esse evento é avaliado. Por exemplo, a frase *"I have eaten breakfast"* em inglês com o aspecto *perfective* indica que o evento de comer o café da manhã ocorreu antes do momento presente em que a frase é dita. No entanto, em português, as expressões formadas de maneira semelhante ao inglês *perfect* (por exemplo, "Eu tenho comido") não carregam necessariamente o mesmo valor semântico de relação temporal com um ponto específico no tempo. Essas expressões podem, por exemplo, ser usadas para indicar uma ação habitual ou repetida, ao invés de uma ação que ocorreu em um ponto específico no passado e tem efeito no presente. Portanto, embora as duas línguas possam utilizar construções similares, a interpretação semântica do aspecto *perfective* em inglês é diferente daquela das expressões em português formadas de maneira similar.

2.5 ANOTAÇÕES TIMEML

Considerando as definições das entidades relevantes para a extração de informações temporais, tais como eventos, expressões temporais e relações temporais, bem como a necessidade de processá-las automaticamente, torna-se essencial abordar algum método formal para descrevê-las. Com esse objetivo, apresentaremos a TimeML (PUSTEJOVSKY et al., 2004), uma linguagem de marcação de metadados no estilo XML que permite a anotação de eventos, expressões temporais e relações temporais em textos de linguagem natural. A TimeML é atualmente reconhecida como um padrão ISO (PUSTEJOVSKY et al., 2010).

A TimeML propõe a anotação de eventos presentes em textos com a *tag* `<EVENT>`, expressões temporais com a *tag* `<TIMEX3>`, e as relações temporais existentes entre estas entidades com a *tag* `<TLINK>`. Conforme relatado por Pustejovsky et al. (2010), a linguagem foi projetada para permitir identificar um evento e ancorá-lo a um tempo específico, ordenar eventos em relação aos outros, lidar com expressões temporais que são subespecificadas em seu contexto (como "na semana passada" e "duas semanas antes"), e raciocinar sobre a duração de um evento. Ela fornece uma capacidade expressiva adicional para capturar e representar as complexidades dessas relações temporais. O foco principal da TimeML é nas relações temporais, e cada uma das entidades temporais possui atributos que contêm uma ampla gama de valores definidos. A seguir, serão apresentados mais detalhes a respeito desses atributos.

2.5.1 Anotações de Eventos

Na TimeML, palavras que denotam eventos são incluídas dentro da *tag* `<EVENT>`. Atributos anexados a essa *tag* são usados para registrar mais informações relevantes, sendo o principal deles o elemento *class*. Os valores desse atributo são: *Reporting, Perception, Aspecual, I_Action, I_State, State* e *Occurrence*, conforme apresentado na Seção 2.1.

2.5.2 Anotações de Expressões Temporais

Na TimeML, as expressões temporais são anotadas dentro de *tags* `<TIMEX3>`, com atributos importantes como *value*, *type* e *functionInDocument*. O atributo *value* codifica uma representação normalizada da expressão e exprime sua semântica temporal, o que facilita o processamento automático. O atributo *type* indica a categoria da expressão temporal, que pode ser `DATE`, `TIME`, `DURATION` ou `SET`. Além disso, o atributo *functionInDocument* informa se a expressão temporal representa a Data de Criação do Documento (DCT).

Para fornecer uma descrição mais precisa do atributo *type*, Chang e Manning (2012) conceituam que esse atributo determina a granularidade temporal da expressão. Por exemplo, uma expressão do tipo `DATE` representa uma data específica ou tempos relativos, como a "próxima segunda-feira" e "anos 90", enquanto uma expressão do tipo `TIME` representa um horário específico. Já uma expressão do tipo `DURATION` representa um intervalo de tempo, como "duas horas", "três semanas", "mês" ou "ano", e uma expressão do tipo `SET` representa um conjunto de datas ou horários periódicos que ocorrem com alguma frequência, como "todos os terceiros domingos".

2.5.3 Anotações de Relações Temporais

As anotações TimeML utilizam *tags* `<TLINK>` para codificar as relações temporais. Os atributos associados a essa *tag* são utilizados para registrar informações adicionais relevantes sobre a relação temporal, enquanto que o atributo *relType* indica o tipo de relação existente. Na Seção 2.3, são apresentados três conjuntos principais de tipos de relação temporal. Para o conjunto de tipos de relação adotado pelo TimeBankPT, foi utilizado o conjunto simplificado proposto pelos desafios TempEval.

Nesse sentido, o atributo *relType* possui como possíveis valores principais: BEFORE, AFTER e OVERLAP, além do valor VAGUE que é utilizado quando não se pode estabelecer uma relação temporal específica, e dois valores adicionais para relações disjuntivas menos específicas, BEFORE-OR-OVERLAP e OVERLAP-OR-AFTER, que são empregados em casos ambíguos. Como exemplo, no trecho retirado do *corpus* (19), há uma relação temporal entre o evento "disse" e a expressão temporal "agora" na qual os anotadores não foram capazes de especificar se o evento ocorreu antes ou durante o tempo representado pela expressão "agora", e, por isso, atribuíram o tipo BEFORE-OR-OVERLAP. É importante destacar que os últimos três valores mencionados são pouco comuns. Além disso, a efetividade das relações disjuntivas foi posta em questionamento por Verhagen et al. (2009), visto que elas não foram capazes de reduzir a ambiguidade de maneira satisfatória.

(19) *Nhek Bunchhay* ***disse*** *que* ***agora*** *acreditava que Howes tinha sido morto (...)*

Outro atributo relevante associado à `<TLINK>` é o *<eventID>*, que se refere a um evento que representa o primeiro argumento da relação temporal. Já o segundo argumento é estabelecido pelo atributo *<relatedToTime>*, no caso de expressões temporais, ou pelo atributo *<relatedToEvent>*, caso se trate de outro evento.

Por fim, é importante destacar que o atributo *task* indica o nome de uma das três tarefas do desafio TempEval-1 às quais essa relação temporal pertence. As relações da

tarefa A são estabelecidas entre um evento e uma expressão temporal que ocorrem na mesma frase. Na tarefa B, as relações temporais são estabelecidas entre eventos e a DCT. Já na tarefa C, as relações representam a conexão entre os eventos principais de duas frases adjacentes. As tarefas A, B e C podem ser consideradas como tarefas de classificação, onde um valor específico é atribuído ao atributo *relType*.

2.5.4 Exemplo de Anotação TimeML

Com o intuito de ilustrar os conceitos apresentados anteriormente, a seguir é exposto um exemplo do *corpus* na sentença (20):

(20) Os bancos aceitaram a operação em agosto de 1988.

```
<TIMEX3 tid="t23" type="DATE" value="1989-10-27" temporalFunction="false"
functionInDocument="CREATION_TIME">10/27/89</TIMEX3>

<s>Os bancos <EVENT eid="e18" class="I_ACTION" stem="aceitar" aspect="NONE"
tense="PPI" polarity="POS" pos="VERB">aceitaram</EVENT> a <EVENT eid="e19"
class="OCCURRENCE" stem="operação" aspect="NONE" tense="NONE" polarity="POS"
pos="NOUN">operação</EVENT> em <TIMEX3 tid="t27" type="DATE" value="1988-08"
temporalFunction="false" functionInDocument="NONE">agosto de 1988</TIMEX3>.</s>

<TLINK lid="l6" relType="OVERLAP" eventID="e18" relatedToTime="t27" task="A"/>
<TLINK lid="l13" relType="BEFORE" eventID="e18" relatedToTime="t23" task="B"/>
```

Figura 2.1: Fragmento de exemplo retirado do TimeBankPT, utilizando marcação TimeML. O texto bruto é: "*Os bancos aceitaram a operação em agosto de 1988.*"
Fonte: Elaborada pelo autor.

Na Figura 2.1 são apresentadas as anotações de um exemplo do *corpus* TimeBankPT na sentença (20). Pode-se observar a anotação de dois eventos, `e18`: "*aceitaram*" e `e19`: "*operação*"; duas expressões temporais, `t23`: "*10/27/89*" e `t27`: "*agosto de 1988*". Nota-se que o atributo "*functionInDocument*" de `t23` recebe o valor "`CREATION_TIME`" para denotar a data de criação do documento.

Além disso, são apresentadas duas relações temporais anotadas. A primeira relação, `l6`, pertencente ao grupo de tarefas A, conforme especificado no atributo *task*, portanto ocorre entre um evento e uma expressão temporal. Ela é uma relação do tipo "OVERLAP" indicando que o evento "*aceitaram*" sobrepõe à expressão de tempo "*agosto de 1988*". A segunda relação, `l13`, pertencente ao grupo de tarefas B, portanto ocorre entre um evento e a DCT. Ela é do tipo "BEFORE", conforme valor do atributo *relType*, o que indica que o evento `e18` ocorreu antes da data de criação do documento.

Com efeito, neste contexto é possível observar que uma relação temporal é estabelecida em um par ordenado. Caso o tipo da relação temporal associada ao par ordenado `(aceitaram, 10/27/89)` seja BEFORE, e considerando que a precedência temporal é uma relação transitiva, então conclui-se que o tipo da relação temporal correspondente ao par `(10/27/89, aceitaram)` será AFTER.

2.6 APRENDIZAGEM DE REGRAS

Nessa pesquisa iremos investigar a aplicação de técnicas de aprendizagem de regras para identificação de tipos de relações temporais entre evento e expressão temporal (*event-time*). Baseado nos estudos de Palanisamy (2006), observamos que aprendizagem de regras é mais frequentemente usado no contexto de aprendizagem de regras de classificação (do inglês, *classification rule learning*) e de aprendizagem de regras de associação (do inglês, *association rule learning*). Enquanto a aprendizagem de regras de classificação é uma abordagem de indução preditiva (ou aprendizagem supervisionada), destinada a construir um conjunto de regras a serem usadas para classificação ou previsão, a aprendizagem de regras de associação é uma forma de indução descritiva (ou aprendizagem não supervisionada), visando a descoberta de regras individuais que definem padrões interessantes nos dados.

Entre as aplicações que podem se beneficiar do uso de aprendizagem de regras, encontram-se aquelas destinadas à análise de transações de compras e estratégias de marketing personalizadas (AGRAWAL; SRIKANT et al., 1994), sistemas de recomendação e detecção de intrusão na rede (PALANISAMY, 2006) e aplicações médicas desenvolvidas para descobrir relações entre doenças, pacientes e remédios (CUNHA-FILHO; ROCHA-JUNIOR, 2022). Modelos baseados em regras também podem ajudar no planejamento de serviços públicos (educação, saúde, transporte), bem como em negócios públicos (instalação de novas fábricas, *shopping centers* ou bancos) utilizando-se de dados de censos (RAJAK; GUPTA, 2008).

Nas próximas subseções apresentaremos com mais detalhes as técnicas de aprendizagem de regras de associação e de classificação.

2.6.1 Aprendizagem de Regras de Associação

Segundo Kumbhare e Chobe (2014), as regras de associação são instruções condicionais que ajudam a descobrir relações entre dados em repositórios de informações. As regras de associação são usadas para encontrar as relações entre objetos que são frequentemente usados juntos.

O problema da aprendizagem de regras de associação é um subcampo da mineração de dados e foi popularizado particularmente devido ao artigo de Agrawal, Imieliński e Swami (1993). Consiste basicamente em extrair padrões ou conjuntos frequentes nos dados.

Rai, Chakraborty e Chakraborty (2023) explicam que uma regra de associação é uma implicação na forma $A \rightarrow B$, em que A e B podem ser conjuntos compostos por um ou mais itens. A é o antecedente e B é o consequente. O antecedente é um conjunto de itens encontrado no conjunto de dados e um consequente é um item observado em combinação com o antecedente.

Para exemplificar o uso de *features* que geram regras e suas principais métricas, apresentamos um exemplo hipotético. A Tabela 2.3 expõe uma pequena base de dados com quatro *features* (colunas do grupo "`A`") e oito transações (linhas). As *features* compõem o antecedente da regra e são representadas pelo grupo de colunas "`A`", que inclui a classe gramatical (`E_pos`) e a classe do evento (`E_class`), o tipo da expressão temporal (`T_type`)

Tabela 2.3: *Dataset* hipotético contendo 8 transações e 4 *features*. As colunas iniciadas por "`E`" indicam um atributo do evento e "`T`" da expressão temporal. O grupo de colunas "`A`" representa o antecedente e "`B`" o consequente (o tipo da relação temporal).

	`A`				`B`
`TID`	`E_pos`	`E_class`	`T_type`	`E_before_T`	`relType`
1	verb	occurrence	date	True	overlap
2	verb	-	date	False	after
3	verb	reporting	date	True	after
4	verb	reporting	time	False	overlap
5	noun	reporting	date	True	after
6	noun	reporting	time	True	after
7	noun	occurrence	date	False	overlap
8	noun	occurrence	time	True	overlap

Fonte: Elaborada pelo autor.

e a informação se o evento antecede textualmente a expressão temporal (`E_before_T`). O consequente é representado pela coluna "`B`" que contém o tipo de relação temporal que se deseja classificar (`relType`). Cada transação (linha) representa as informações de um par composto pelas entidades evento/expressão temporal e seu tipo de relação temporal. Neste exemplo, a classe gramatical do evento pode assumir os valores `verb`, que indica que o evento é um verbo, e `noun`, que indica que ele é um substantivo; a classe do evento pode ser do tipo `occurrence` ou `reporting`; o tipo de expressão temporal assume os valores `date` ou `time`; a posição do evento em relação à expressão temporal (`E_before_T`) assume o valor `True` se preceder e `False` se suceder textualmente a expressão temporal. Por fim, o tipo da relação temporal pode assumir os valores `after` ou `overlap`. `TID` representa apenas o ID da transação.

Para exemplificar as principais métricas utilizadas para medir a qualidade de regras, suponha que a seguinte regra foi gerada a partir desse *dataset*:

$$\texttt{\{E\_pos=verb, T\_type=date\}} \rightarrow \texttt{\{relType=after\}} \tag{2.6.1.1}$$

Isso significa que, se a classe gramatical do evento for um verbo e o tipo da expressão temporal for uma data, então o tipo de relação temporal entre o evento e a expressão temporal de interesse é After.

Para selecionar regras de qualidade a partir do conjunto de todas as regras possíveis e validar sua eficácia, são utilizadas restrições em diversas medidas de significância e interesse. As restrições mais comumente utilizadas são os limites mínimos de suporte (*support*) e confiança (*confidence*), inicialmente introduzidos por Agrawal, Imieliński e Swami (1993). No entanto, outras medidas podem ser utilizadas para este propósito, como o grau de interesse (*lift*) e a convicção (*conviction*), introduzidos por Brin et al. (1997). A seguir, detalharemos o significado de cada uma dessas medidas para uma regra

A $\rightarrow$ B.

Support: O Suporte representa a porcentagem de transações da base de dados que contêm os itens de A e B, indicando a relevância da regra. Essa medida informa com que frequência o conjunto de itens aparece no conjunto de dados.

$$Suporte(A \rightarrow B) = \frac{frequência(A, B)}{N^o\ total\ transações} \tag{2.6.1.2}$$

Considerando o *dataset* apresentado na Tabela 2.3, a regra apresentada em (2.6.1.1) possui um suporte de 2/8 = 0,25, uma vez que ela ocorre em 25% de todas as transações.

Confidence: A medida Confiança representa a força ou validade de uma regra de associação. Ou seja, refere-se à correlação entre os dois conjuntos de itens que constituem uma regra, o antecedente e o consequente. Significa dizer que, dentre as transações que possuem os itens de A, a porcentagem de transações que possuem também os itens de B. Por exemplo, se a confiança de uma regra de associação for 60%, então 60% das transações que contêm os itens de A também contêm os itens de B juntos. A Confiança pode ser definida de duas formas:

$$Confiança(A \rightarrow B) = \frac{frequência(A, B)}{frequência(A)} \quad ou \quad \frac{Suporte(A \rightarrow B)}{Suporte(A)} \tag{2.6.1.3}$$

Considerando o conjunto de dados apresentado na Tabela 2.3, é possível calcular que a regra indicada pela expressão (2.6.1.1) possui uma confiança de 2/3, o que significa que em 67% das ocorrências em que há um evento que é verbo e uma expressão temporal do tipo data, a relação temporal entre eles é do tipo AFTER.

Lift: A medida *Lift* é utilizada para definir o grau de interesse de uma regra de associação e pode ser definida de duas formas:

$$Lift(A \rightarrow B) = \frac{Confiança(A \rightarrow B)}{Suporte(B)} \quad ou \quad \frac{Suporte(A \rightarrow B)}{Suporte(A) * Suporte(B)} \tag{2.6.1.4}$$

O resultado pode ser interpretado da seguinte forma:

- Se $\boldsymbol{Lift = 1}$, então a probabilidade de ocorrência do antecedente (A) e do consequente (B) não é dependente uma da outra. Nesse caso, a regra de associação não apresenta uma relação significativa.
- Se $\boldsymbol{Lift > 1}$, então o valor de *Lift* determina o grau de dependência entre o antecedente (A) e o consequente (B). Indica que a ocorrência conjunta dos itens de A e B não é acidental, mas sim devido à presença de uma relação significativa entre eles.

Essas regras de associação são potencialmente úteis para prever o consequente em conjuntos de dados futuros. Por exemplo, se o *lift* de uma regra for igual a 2, isso indica que, ocorrendo os itens de A, há uma chance 2 vezes maior de ocorrerem os itens de B.

- Se $\boldsymbol{Lift < 1}$, então um item tem um efeito negativo sobre o outro na regra de associação. Isso significa que a presença de um item tem um efeito inibidor sobre a presença do outro item e vice-versa. Nesse caso, os itens são considerados substitutos um do outro na regra.

Considerando o *dataset* apresentado na Tabela 2.3, a regra apresentada em (2.6.1.1) possui um grau de interesse de 0,67/0,5 = 1,33. Isso significa que, quando os itens `{E_pos=verb, T_type=date}` ocorrem, há uma probabilidade 1,33 vezes maior de ocorrer o item `{relType=after}`, indicando que o tipo da relação temporal é AFTER.

Conviction: A medida de Convicção é de fato uma medida de implicação porque é direcional, é máxima para implicações perfeitas e leva em conta adequadamente a probabilidade dos itens de A e a probabilidade dos itens de B. Os autores dessa medida, Brin et al. (1997), tinham como objetivo avaliar uma regra de associação como uma verdadeira implicação. Ou seja, ela respeita o sentido da implicação: $Convicção(A \rightarrow B) \neq Convicção(B \rightarrow A)$. É definida da seguinte forma:

$$Convicção(A \rightarrow B) = \frac{(1 - Suporte(B))}{(1 - Confiança(A \rightarrow B))} \tag{2.6.1.5}$$

Podemos interpretá-la como sendo a razão da frequência que a regra faz uma previsão incorreta se A e B fossem independentes, dividido pela frequência observada de previsões incorretas.

Por exemplo, na base de dados representada na Tabela 2.3, a regra descrita em (2.6.1.1) tem uma convicção de (1 - 0,5) / (1 - 0,67) = 1,5. Isso significa que essa regra seria considerada incorreta 1,5 vezes mais frequentemente se a associação entre os itens A e B fosse aleatória. Em outras palavras, a probabilidade do evento ser verbo e a expressão temporal ser do tipo data sem que o tipo da relação temporal seja AFTER é 1,5 vezes menor do que o esperado aleatoriamente.

Os valores dessa medida podem ser interpretados da seguinte forma, se:

- $\boldsymbol{Convicção = 1}$: Indica independência completa entre o antecedente (A) e o consequente (B) da regra.
- $\boldsymbol{Convicção > 1}$: Quanto maior o valor de convicção, maior é a dependência de B em relação a A. Regras em que o antecedente nunca ocorre sem o consequente terão um valor de convicção igual a infinito.

Nos experimentos realizados pelos autores da medida, identificaram que as regras mais interessantes apresentaram um valor de convicção entre 1,01 e 5. Foi percebido que muitas das regras com valor de convicção acima de 5 representavam informações óbvias ou ilusórias.

2.6.1.1 Tipos de Restrições Estamos interessados em gerar todas as regras que satisfaçam certas restrições. Segundo Agrawal, Imieliński e Swami (1993), essas restrições podem ser feitas de duas formas diferentes:

1. **Restrições sintáticas:** Envolvem restrições em itens que podem aparecer em uma regra. Por exemplo, podemos estar interessados apenas em regras que tenham um item específico aparecendo no consequente, ou regras que tenham um item específico aparecendo no antecedente. É possível ainda uma combinação de ambas, podemos solicitar todas as regras que tenham itens de alguns itens pré-definidos para que aparecem no consequente, e itens de algum outro conjunto de itens que aparecem no antecedente.

2. **Restrições de suporte:** Estamos interessados apenas em regras com suporte acima de algum limite mínimo e que possuem significância estatística. Se o suporte não for grande o suficiente, isso significa que a regra não vale a pena ser considerada ou que é simplesmente menos preferida podendo ser considerada mais tarde.

Para obter regras de alta qualidade, é necessário estabelecer um valor mínimo para certas medidas, descartando aquelas que não atingirem esse critério. A utilização de várias medidas de filtragem de regras proporciona diferentes perspectivas sobre as associações presentes na base de dados, permitindo a extração de regras mais interessantes.

Adicionalmente, considerando nosso interesse exclusivo em regras cujo resultado contenha somente o tipo de relação temporal que estamos buscando prever, é necessário aplicar restrições sintáticas. Com a imposição dessa restrição, o problema de associação em um sentido amplo se transforma em um problema de aprendizado de regras de classificação ou classificação associativa, conforme será explicado adiante.

2.6.2 Aprendizagem de Regras de Classificação

As regras de classificação representam o conhecimento na forma de instruções condicionais que atribuem uma classe a exemplos não rotulados. Assim como nas regras de associação, as regras de classificação também possuem um antecedente e um consequente, sendo este último restringido sintaticamente ao atributo da classe. Seus algoritmos geralmente são gulosos e usam o princípio de "separar e conquistar" (do inglês, *separate-and-conquer*, cunhado por Pagallo e Haussler (1990)). Eles são relativamente mais rápidos na extração de regras importantes, quando comparados com algoritmos exaustivos de aprendizagem de regras como os métodos de descoberta de subgrupos (do inglês, *subgroup discovery* (KLÖSGEN, 1996)). Exemplos de algoritmos de aprendizagem de regras de classificação incluem `AQ` (MICHALSKI, 1969), `ID3` (QUINLAN, 1983), `PRISM` (CENDROWSKA, 1987), `CN2` (CLARK; NIBLETT, 1989), `FOIL` (QUINLAN, 1990), `C4.5` (QUINLAN, 1993), `NNGE` (MARTIN, 1995), `RIPPER` (COHEN, 1995), `CBA` (LIU; HSU; MA, 1998), `FURIA` (HÜHN; HÜLLERMEIER, 2009) e `IDS` (LAKKARAJU; BACH; LESKOVEC, 2016).

O objetivo da aprendizagem de regras de classificação é criar modelos interpretáveis, que consistem em conjuntos de regras descrevendo as características de uma classe com

base nas propriedades presentes nos exemplos de treinamento. Os classificadores baseados em regras possuem algumas propriedades importantes, e uma delas é a não exaustividade, o que significa que os conjuntos de regras gerados podem deixar algumas instâncias de dados sem serem contempladas por nenhuma regra específica. De acordo com Fürnkranz e Kliegr (2015), esses casos geralmente são tratados prevendo a classe majoritária.

Outra propriedade importante é a sobreposição. As regras geradas não são mutuamente exclusivas, ou seja, muitas regras podem cobrir a mesma instância de dados. Uma questão sobre a sobreposição é como a classe seria decidida no caso de regras diferentes com consequências diferentes cobrirem a mesma instância de dados. Existem pelo menos duas soluções para esse problema (i) as regras podem ser ordenadas por algum critério e a classe correspondente à regra acionada de prioridade mais alta é tomada como a classe final ou (ii) podemos atribuir votos para cada classe, a mais frequente seria atribuída. As duas formas de resolver conflitos em sobreposição de regras são aplicados nesse trabalho e estão detalhados na Seção 4.3.

As regras de classificação associativa são consideradas uma abordagem eficaz para a representação de informações devido à sua capacidade de serem facilmente lidas e compreendidas. No âmbito deste trabalho, foram empregados alguns algoritmos de classificação associativa com o intuito de construir conjuntos de regras capazes de identificar os tipos de relações temporais *event-time*. Dentre os algoritmos utilizados, destacam-se o `CBA` (LIU; HSU; MA, 1998), o `CN2` (CLARK; NIBLETT, 1989), o `IDS` (LAKKARAJU; BACH; LESKOVEC, 2016) e o `RIPPER` (COHEN, 1995).

Esses algoritmos são especialmente adequados para lidar de maneira satisfatória com conjuntos de dados que possuam ruídos, como desequilíbrios entre as classes e dados ausentes. Além disso, eles se concentram em garantir a interpretabilidade das regras e demonstram um desempenho sólido ao serem aplicados a conjuntos de dados desconhecidos. Adicionalmente, são capazes de aprender regras com alta acurácia e são eficazes na prevenção do *overfitting*. O *overfitting* ocorre quando um modelo se ajusta demais aos dados de treinamento, de modo que ele se torna excessivamente específico e não consegue generalizar bem para dados desconhecidos. A seguir, apresentamos um breve resumo sobre cada algoritmo.

2.6.2.1 `CBA` O algoritmo de Classificação Baseada em Associações (`CBA` - do inglês, *Classification based on Associations*) é um método que integra técnicas de mineração de regras de associação com mineração de regras de classificação visando classificar os dados. Liu, Hsu e Ma (1998) denominaram essa técnica de classificação associativa (do inglês, *Associative Classification*).

A integração é feita concentrando-se em um subconjunto especial de regras de associação cujo lado direito é restrito ao atributo da classe de classificação. Liu, Hsu e Ma (1998) referem-se a esse subconjunto de regras como as Regras de Associação de Classe (CARs - do inglês, *Class Association Rules*). Um algoritmo de mineração de regras de associação baseado no algoritmo *Apriori* (AGRAWAL; SRIKANT et al., 1994) é adaptado para minerar todas as CARs que satisfazem as restrições mínimas de suporte e confiança. Dessa forma, o algoritmo de mineração de regras de associação minera apenas as CARs, de modo a reduzir o número de regras geradas, evitando assim uma explosão

combinatória.

O algoritmo proposto por Liu, Hsu e Ma (1998) consiste em duas etapas distintas. A primeira etapa, denominada `CBA-RG`, engloba um gerador de regras de associação. Esse gerador se baseia no algoritmo *Apriori* de Agrawal, Srikant et al. (1994). A segunda etapa, fundamental no `CBA`, é conhecida como `CBA-CB`. Ela se concentra na remoção de regras redundantes identificadas e na construção de um classificador a partir da lista de regras podadas.

A etapa `CBA-RG` gera todas as regras (*ruleitems*) frequentes, através de múltiplas iterações sobre os dados. As regras frequentes são aquelas que atendem a um suporte mínimo predefinido (*minsup*). Na primeira iteração, o suporte de cada *ruleitem* é computado e avaliado quanto à frequência. Em iterações subsequentes, parte-se do conjunto inicial de regras consideradas frequentes na iteração anterior. Esse conjunto é utilizado para gerar novas regras potencialmente frequentes, conhecidas como *ruleitems* candidatos.

Os suportes reais para esses candidatos são calculados durante as iterações sobre os dados. Ao final de cada iteração, é possível determinar quais das regras candidatas são verdadeiramente frequentes. Com base nesse conjunto de regras frequentes, as Regras de Associação de Classe (CARs) são geradas.

Considere *k-ruleitem* como uma regra cujo conjunto de condições (*condset*) compreende k itens. Seja F_k o conjunto dos *k-ruleitems* mais frequentes, e C_k o conjunto dos *k-ruleitems* candidatos. A etapa `CBA-RG` é detalhada no Algoritmo 1.

Algoritmo 1: CBA-RG - Etapa de Geração de Regras de Associação

```
1  F_1 = {large 1-ruleitems};
2  CAR_1 = genRules(F_1);
3  prCAR_1 = pruneRules(CAR_1);
4  for (k = 2; F_{k-1} ≠ ∅; k++) do
5      C_k = candidateGen(F_{k-1});
6      for each data case d ∈ D do
7          C_d = ruleSubset(C_k, d);
8          for each candidate c ∈ C_d do
9              c.condsupCount++;
10             if d.class = c.class then
11                 c.rulesupCount++ ;
12     F_k = {c ∈ C_k | c.rulesupCount ≥ minsup};
13     CAR_k = genRules(F_k);
14     prCAR_k = pruneRules(CAR_k);
15 CARs = ⋃_k CAR_k;
16 prCARs = ⋃_k prCAR_k;
```

As linhas de código numeradas de 1 a 3 descrevem a fase inicial do algoritmo e ocorre a primeira iteração sobre os dados. Nessa etapa, ocorre a contagem das ocorrências de regras e classes, visando determinar a frequência de *1-ruleitems* (linha 1). A partir

do conjunto de *1-ruleitems*, a função *genRules* é empregada para gerar um conjunto de Regras de Associação de Classe (CARs) designado como CAR_1 (linha 2). É possível realizar uma operação de poda em CAR_1 (linha 3), que pode ser opcional, e os resultados são armazenados em $prCAR_1$. Tal procedimento de poda é adicionalmente aplicado nas passagens subsequentes para o conjunto CAR_k (linha 14). A função *pruneRules* emprega o método de poda baseado na taxa de erro pessimista do algoritmo `C4.5` (QUINLAN, 1993). Essa função promove a poda de regras da seguinte forma: quando a taxa de erro pessimista de uma regra r excede a taxa de erro pessimista de uma regra r' (obtida pela eliminação de uma condição de r), a regra r é sujeita à poda. Essa ação de poda tem o efeito de considerável redução no volume de regras geradas.

Para cada iteração subsequente, por exemplo, a iteração k, o algoritmo executa quatro operações fundamentais. Primeiro, os elementos de regra frequentes F_{k-1} identificados na $(k-1)^{\text{a}}$ iteração são utilizados para gerar o conjunto de regras candidatas C_k por meio da função *candidateGen* (linha 5). Logo após, a função percorre a base de dados, atualizando várias contagens de suporte para os candidatos presentes em C_k (linhas 6-11). Uma vez que os novos *ruleitems* frequentes foram identificados, formando o conjunto F_k (linha 12), o algoritmo emprega a função *genRules* para gerar as regras CAR_k (linha 13). Finalmente, a operação de poda das regras é aplicada (linha 14) a esse conjunto.

A função *ruleSubset* (linha 7) emprega o conjunto de regras candidatas C_k e uma instância de dados d para localizar todas as regras em C_k cujas condições são suportados por d. As operações das linhas 8 a 11 assemelham-se às empregadas no algoritmo *Apriori*. No entanto, difere-se no fato de que a contagem de suporte dos conjuntos de condições (linha 9) e a contagem de suporte das regras (linha 11) são incrementadas separadamente. No algoritmo *Apriori*, apenas uma contagem é atualizada. Essa variação permite o cálculo da confiança das regras, sendo relevante também para a poda das regras.

O conjunto final de Regras de Associação de Classe encontra-se em $CARs$ (linha 15). As regras remanescentes após a operação de poda estão em $prCARs$ (linha 16).

Na etapa denominada `CBA-CB`, é efetuada a criação de um classificador utilizando as Regras de Associação de Classe ($CARs$) geradas na etapa anterior. Esta etapa é notável por sua maior complexidade temporal e pela exigência de recursos de memória mais elevados. A descrição desta etapa pode ser encontrada no Algoritmo 2.

No passo inicial, todas as regras são organizadas em ordem com base nas métricas de confiança e suporte (linha 1). Isso assegura que as regras com alta confiança e suporte sejam selecionadas prioritariamente. Posteriormente, as regras são percorridas de acordo com essa ordem. Para cada regra, verifica-se o número de instâncias que satisfazem a regra. Caso uma instância satisfaça tanto o antecedente da regra quanto a classe do consequente, a regra é marcada e inserida em *temp* (linhas 2-6).

Após o término da iteração pelas instâncias, verifica-se se a regra está marcada (linha 7). Se estiver, ela é adicionada ao final do classificador (linha 8), e todas as instâncias presentes em *temp* são eliminadas do conjunto de dados (linha 9). A classe padrão, que corresponde à classe majoritária das instâncias restantes, é selecionada (linha 10). Em seguida, avalia-se o número de erros cometidos por um classificador C, o qual consiste em todas as regras acima da regra atual, a regra atual e uma regra padrão (linha 11). Esse número de erros é associado à regra atual.

Algoritmo 2: CBA-CB - Etapa de Construção de Classificador (M1)

```
1  R = sort(R);
2  for each rule r ∈ R in sequence do
3      temp = ∅;
4      for each case d ∈ D do
5          if d satisfies the conditions of r then
6              store d.id in temp and mark r if it correctly classifies d;
7      if r is marked then
8          insert r at the end of C;
9          delete all the cases with the ids in temp from D;
10         selecting a default class for the current C;
11         compute the total number of errors of C;
12 Find the first rule p in C with the lowest total number of errors and drop all the
     rules after p in C;
13 Add the default class associated with p to the end of C and return C;
```

Após a conclusão da iteração por todas as regras, identifica-se a regra r com o menor número de erros associados. As regras localizadas abaixo da regra r são descartadas (linha 12). Posteriormente, anexa-se uma regra com um antecedente vazio, que prevê a classe padrão associada à regra r, como a última regra no classificador final C (linha 13).

Esse algoritmo satisfaz duas condições fundamentais: primeiramente, cada caso de treinamento é coberto pela regra de maior precedência entre aquelas capazes de cobrir o caso, devido à ordenação executada na linha 1. Em segundo lugar, cada regra em C classifica corretamente, no mínimo, um caso de treinamento restante no momento de sua seleção, o que é alcançado nas linhas 5 e 6.

De acordo com Filip e Kliegr (2018), constatou-se que o algoritmo `CBA` é frequentemente adotado como um algoritmo de referência em diversos artigos recentes no campo da Associação de Regras de Classificação, apesar de ser relativamente antigo. Além disso, Kliegr e Kuchar (2019) mencionam que o algoritmo `CBA` continua sendo um dos melhores algoritmos de classificação baseados em regras, em termos de equilíbrio entre compreensibilidade do modelo, poder preditivo e escalabilidade, embora existam abordagens mais recentes disponíveis.

Em um estudo mais recente conduzido por Kumi, Lim e Lee (2021) para detecção de URLs maliciosas utilizando *features* de conteúdo da página da web e de URLs, os resultados demonstraram que o algoritmo `CBA` obteve excelente desempenho em um conjunto de dados do mundo real, em relação à precisão, *recall* e taxa de falsos positivos, apresentando desempenho comparável a algoritmos de classificação de referência.

Por outro lado, um estudo realizado por Hasanpour, Meibodi e Navi (2019) concluiu que a utilização de um algoritmo ganancioso pelo `CBA` para seleção das melhores regras de associação de classe (CARs) durante a etapa de construção do classificador não é a abordagem mais adequada. Isso se deve ao fato de que, segundo os autores, o uso de

abordagens gananciosas resulta na seleção de um conjunto de regras que não representa o melhor subconjunto disponível.

Neste trabalho, observou-se experimentalmente que o algoritmo `CBA` enfrentou dificuldades ao lidar com conjuntos de dados de alta dimensionalidade. A presença de um grande número de *features* conduziu a um aumento exponencial no espaço de busca de regras, tornando a mineração de regras computacionalmente mais intensiva. À medida que a quantidade de *features* ou o tamanho do conjunto de dados aumenta, o tempo de execução e a demanda de memória também aumentam significativamente, limitando, assim, a escalabilidade do algoritmo.

Nesta pesquisa, utilizou-se a implementação realizada por Filip e Kliegr (2018)[2].

2.6.2.2 `CN2` O algoritmo `CN2`, nomeado com as iniciais de seus desenvolvedores Clark e Niblett (1989), funciona encontrando uma regra que abrange algumas instâncias de aprendizagem, removendo essas instâncias e repetindo isso até que todas as instâncias sejam cobertas. As regras são pontuadas por heurísticas, como a impureza da distribuição de classes de instâncias cobertas.

O algoritmo `CN2`, conforme descrito por Clark e Niblett (1989), é um algoritmo de indução projetado com o propósito de realizar a indução eficiente de regras compreensíveis em domínios nos quais problemas como má descrição da linguagem ou ruído nos dados possam estar presentes. Essa característica é de extrema importância, uma vez que as aplicações em domínios do mundo real exigem métodos capazes de lidar adequadamente com dados que apresentem ruídos, como o desequilíbrio entre as classes, o qual é um problema encontrado no *corpus* utilizado nesta pesquisa.

Para evitar *overfitting*, Clark e Boswell (1991) aperfeiçoaram o algoritmo `CN2` adicionando uma busca de feixe (do inglês, *beam search*) guiada pelas estimativas de Laplace e o teste de significância da razão de verossimilhança, conforme descrito por Fürnkranz e Kliegr (2015). A estimativa de Laplace calcula a fração de exemplos positivos em todos os exemplos cobertos, sendo que cada classe é inicializada com um exemplo virtual, a fim de penalizar as regras com baixa cobertura. O `CN2` pode operar em dois modos: um para aprender conjuntos de regras, modelando cada classe de forma independente, e outro para aprender listas de decisão.

Este algoritmo emprega uma heurística para interromper a busca durante a construção da regra, com base em uma estimativa do ruído presente nos dados. Isso resulta em regras que podem não classificar corretamente todos os exemplos de treinamento, mas apresentam bom desempenho em dados novos.

O `CN2` incorpora ideias dos algoritmos `AQ` de Michalski (1969) e `ID3` de Quinlan (1983) fazendo uso de uma variante do `ID3` denominada `ASSISTANT` de Kononenko, Bratko e Roskar (1984).

Para classificar novas instâncias usando as regras induzidas, o `CN2` tenta cada regra em ordem até encontrar uma cujas condições sejam satisfeitas pela instância que está sendo classificada. A previsão de classe dessa regra é então atribuída como a classe da instância. Se nenhuma regra for satisfeita, a regra padrão final atribui a classe mais comum à nova

[2]disponível em <https://github.com/jirifilip/pyARC>

instância. Um resumo do `CN2` é apresentado em Algoritmo 3.

Algoritmo 3: CN2

```
   Input: examples, classes
   Output: rulelist
1  rulelist = [];
2  repeat
3  |  bestcond = FindBestCondition(examples);
4  |  if bestcond is not null then
5  |  |  class = the most common class of examples covered by bestcond;
6  |  |  addRule('if bestcond then predict class') to end of rulelist;
7  |  |  remove(from examples) all examples covered by bestcond;
8  until bestcond is null;
9  return rulelist;
```

O algoritmo recebe como entrada um conjunto de exemplos e suas classes correspondentes. A saída do algoritmo é uma lista de regras de classificação ("*rulelist*"). O algoritmo entra em um *loop* repetitivo (linha 2) que continua até que não seja mais possível encontrar mais condições satisfatórias.

Em cada iteração, o algoritmo chama a função "`FindBestCondition`" (linha 3) para encontrar a melhor condição (*bestcond*) dentre todas as condições possíveis. A melhor condição é aquela que abrange um grande número de exemplos de uma única classe (a classe mais comum) e poucos exemplos de outras classes. Esta condição deve ser preditiva e confiável, de acordo com critérios de avaliação específicos do `CN2`.

Se uma melhor condição for encontrada, ou seja, *bestcond* não é nulo (linha 4), o algoritmo determina a classe mais comum entre os exemplos cobertos por essa condição (linha 5). Em seguida, adiciona uma regra à "*rulelist*" (linha 6) no formato "*if bestcond then predict class*", indicando que se a melhor condição for satisfeita, a classe prevista é a classe mais comum determinada anteriormente. O algoritmo remove os exemplos do conjunto de treinamento que são cobertos pela melhor condição encontrada (linha 7). Isso garante que esses exemplos não sejam considerados novamente nas iterações subsequentes.

Este processo é repetido até que não seja possível encontrar mais condições satisfatórias (*bestcond* se torna nulo, linha 8). Finalmente, o algoritmo retorna a lista de regras geradas (*rulelist*) como seu resultado (linha 9).

O sistema de busca por melhores condições para as regras, representado pela função `FindBestCondition`, realiza uma busca onde começa de forma geral e, à medida que avança, poda opções menos promissoras para chegar a condições específicas desejadas. Esta função é detalhada em Algoritmo 4.

Esta função tem como objetivo encontrar a melhor condição, ou seja, um conjunto de atributos, que pode ser usado na geração de regras de classificação. Essa condição deve ser informativa, preditiva e estatisticamente significativa para melhor descrever as classes dos exemplos de treinamento.

Algoritmo 4: FindBestCondition

```
   Input: examples
   Output: bestcond
 1 mgc = True ;                                  // the most general condition
 2 star = mgc ;                                // initially contain only the mgc
 3 bestcond = null;
 4 while star is not empty do
 5     newstar = {};
 6     for each condition cond in star do
 7         for each possible attribute test not already tested on in cond do
 8             cond′ = cond and test ;   // a specialisation of cond, formed
                by adding test as an extra conjunct to cond
 9             if cond′ is better then bestcond and cond′ is statistically significant
                then
10                 bestcond = cond′;
11             add(cond′ to newstar);
               // maxstar is a user-defined constant
12             if size of newstar > maxstar then
13                 remove(the worst condition in newstar);
14     star = newstar;
15 return bestcond;
```

O processo começa com a inicialização de variáveis importantes, incluindo a condição mais geral (*mgc*, linha 1), o conjunto de melhores condições (*star*, linha 2) e a melhor condição encontrada até o momento (*bestcond*, linha 3).

O algoritmo entra em um *loop* (linha 4) que continua enquanto o conjunto *star* não estiver vazio. Dentro deste *loop*, ele cria um novo conjunto vazio (*newstar*, linha 5) para armazenar possíveis especializações das condições existentes.

Para cada condição (*cond*) no conjunto *star* (linha 6), o algoritmo realiza um *loop* sobre todos os possíveis testes de atributos que ainda não foram considerados na condição atual (linha 7). Uma especialização (*cond′*, linha 8) da condição atual é formada adicionando um novo termo conjuntivo (*test*) a *cond*.

Em seguida, o algoritmo avalia se essa nova condição (*cond′*) é melhor do que a melhor condição encontrada até o momento (*bestcond*) e se é estatisticamente significativa (linha 9). Se isso for verdade, a melhor condição é atualizada para ser igual a *cond′* (linha 10).

Além disso, o conjunto *newstar* é atualizado adicionando a nova condição (*cond′*) (linha 11). Para evitar que o conjunto cresça indefinidamente, é realizada uma poda, removendo a condição de pior desempenho (linha 13) caso o tamanho de *newstar* exceda um limite predefinido (*maxstar*, linha 12).

Esse processo de avaliação, especialização e poda é repetido até que não seja mais possível encontrar condições satisfatórias no conjunto *star*, o que significa que o algo-

ritmo busca constantemente melhorar a condição atual até que não haja mais melhorias significativas (linha 4).

Em resumo, a função `FindBestCondition` (Algoritmo 4) busca de maneira iterativa a melhor condição para criar regras de classificação, avaliando e refinando condições em busca daquela que é mais informativa e estatisticamente significativa para descrever as classes dos exemplos de treinamento.

Em termos gerais, pode-se afirmar que o algoritmo `CN2` demonstra eficiência na tarefa de extrair informações a partir de conjuntos de dados caracterizados por ruído, enquanto também demonstra habilidade em evitar o problema de *overfitting* por meio da aplicação de técnicas estatísticas, conforme indicado na análise realizada por Boll e Clair (1995) em seu estudo sobre o `CN2`. Conclui-se, com base nos estudos realizados por Clark e Boswell (1991), que o `CN2` é uma ferramenta útil para a construção indutiva de sistemas baseados em conhecimento.

2.6.2.3 `IDS` O *framework* denominado de Conjunto de Decisões Interpretável (`IDS` - do inglês, *Interpretable Decision Sets*), proposto por Lakkaraju, Bach e Leskovec (2016), aprende regras com alta acurácia e não sobrepostas que cobrem todas as *features* prestando atenção também em classes minoritárias. Regras sobrepostas ocorrem quando uma instância de dados é classificada por mais de uma regra.

A aprendizagem do conjunto de decisões é conduzida por meio de uma função objetivo que otimiza simultaneamente a interpretabilidade e o desempenho das previsões por meio de pesos que podem ser atribuídos a cada um dos sete objetivos parciais, sendo cinco referentes à interpretabilidade e dois ao desempenho.

Em relação aos objetivos relacionados à interpretabilidade, são favorecidos os conjuntos de decisão (i) com um número menor de regras, (ii) com menos condições (tripla formada por `atributo, operador, valor`) em suas regras, (iii) com regras da mesma classe que não se sobrepõem, (iv) com regras de classes diferentes que não se sobrepõem, (v) que tenha pelo menos uma regra que preveja cada classe. Em relação aos objetivos relacionados ao desempenho, são favorecidos os conjuntos de decisão (vi) com conjuntos menores de coberturas incorretas (incentivando a precisão) e (vii) que cubram corretamente as instâncias de dados com pelo menos uma regra (incentivando o *recall*).

Os conjuntos de decisões desse algoritmo se contrapõem à representação para funções booleanas denominada listas de decisões de Rivest (2001). Nas listas de decisões, as regras precisam estar ordenadas e dependem implicitamente do fato de que todas as regras acima não serem verdadeiras, uma vez que são aninhadas e possuem hierarquia. Por outro lado, os conjuntos de decisão (`IDS`) aplicam as regras de forma independente, tornando-as mais compreensíveis para os seres humanos e não possuindo uma ordem inerente. Em outras palavras, são regras individuais que, quando combinadas, formam coletivamente um classificador (FÜRNKRANZ; KLIEGR, 2015). Além disso, o algoritmo `IDS` resolve deficiências em abordagens anteriores como a abordagem de Liu, Hsu e Ma (1998) discutida acima, ao evitar sobreposição de regras em uma mesma instância de dados, por exemplo.

Para avaliar a interpretabilidade das regras geradas por este modelo, Lakkaraju, Bach e Leskovec (2016) desenvolveram métricas com base nos seguintes critérios: (i) as regras

no conjunto descrevem espaços de recursos não sobrepostos, facilitando aos usuários identificar quais valores de atributo estão relacionados a rótulos de classe específicos; (ii) a maioria dos dados é abrangida por pelo menos uma regra do conjunto; (iii) o conjunto contém um número reduzido de regras, e cada uma delas é concisa; e (iv) as regras no conjunto descrevem a maioria das classes presentes nos dados.

O framework para aprendizagem de conjuntos de decisões, o `IDS`, recebe como entrada um conjunto de treinamento $\mathcal{D}$, um conjunto de conjunções de condições $\mathcal{S}$ e um conjunto de possíveis classes $\mathcal{C}$. Seu objetivo é encontrar um conjunto de decisões $\mathcal{R}$ que faça previsões precisas, equilibrando interpretabilidade e desempenho. Para facilitar a compreensão, a Tabela 2.4 oferece um resumo da notação utilizada.

Tabela 2.4: Resumo da Notação

Notação	**Termo**	**Definição**
$\mathcal{D}$	Dataset	Conjunto de dados de entrada $\{(\mathrm{x}_1, y_1), \ldots, (\mathrm{x}_\mathrm{N}, y_N)\}$
x	Atributos	Valores de atributos observados em uma instancia
y	Classe	Classe de uma instância
$\mathcal{C}$	Conjunto de classes	Conjunto de classes em $\mathcal{D}$
p	Condição	Tupla (atributo, operador, valor)
s	Itemset	Conjunções de condições
$\mathcal{S}$	Itemsets	Conjunto de conjunções de condições
r	Regra	Par formado pelas conjunções de condições e a classe (s, y)
$\mathcal{R}$	Conjunto de decisão	Conjunto de regras $\{(s_1, y_1), \ldots, (s_k, y_k)\}$

Fonte: Adaptada de Lakkaraju, Bach e Leskovec (2016).

O algoritmo `IDS` é dividido em duas etapas principais. Primeiro, utiliza a mineração de conjuntos de itens frequentes com o algoritmo Apriori de Agrawal, Imieliński e Swami (1993) para extrair o conjunto de conjunções de condições $\mathcal{S}$. Em seguida, seleciona um subconjunto de $\mathcal{S} \times \mathcal{C}$, que são pares formados pelas conjunções de condições e suas classes correspondentes, para definir o conjunto de regras que compõem um conjunto de decisões interpretáveis $\mathcal{R}$.

Nesta etapa de seleção de regras, o algoritmo `IDS` maximiza uma função objetivo não negativa, não normal, não monótona e submodular. Essa função atribui pontuações aos conjuntos de decisões com base em sua interpretabilidade e desempenho. O algoritmo mais conhecido para maximizar uma função com essas características é a Busca Local

Suave (SLS - do inglês, *Smooth Local Search*) introduzida por Feige, Mirrokni e Vondrák (2011), apresentado no Algoritmo 5.

Algoritmo 5: *Smooth Local Search* (SLS)

Input: Objetive f, domain $X = \mathcal{S} \times \mathcal{C}$, parameters δ e δ'
Output: Decision set
1 $A = \emptyset$;
2 $\texttt{OPT} = f(\Phi_X(X, 0))$;
3 **for** *each element* $x \in X$ **do**
4 Estimate $\mathbb{E}\left[f(\Phi_X(A, \delta) \cup x)\right] - \mathbb{E}\left[f(\Phi_X(A, \delta) \backslash x)\right]$ within an error of $\frac{1}{|X|^2} \cdot \texttt{OPT}$;
5 Call this estimate $\tilde{\omega}_{A,\delta}(x)$;
6 **for** *each element* $x \in X \backslash A$ *such that* $\tilde{\omega}_{A,\delta}(x) > \frac{2}{|X|^2} \cdot \texttt{OPT}$ **do**
7 $A = A \cup x$;
8 Goto Line 3;
9 **for** *each element* $x \in A$ *such that* $\tilde{\omega}_{A,\delta}(x) < \frac{-2}{|X|^2} \cdot \texttt{OPT}$ **do**
10 $A = A \backslash x$;
11 Goto Line 3;
12 **return** $\Phi_X(A, \delta')$;

O algoritmo SLS (*Smooth Local Search*) é utilizado para encontrar um conjunto de decisões $\mathcal{R}$ por meio da amostragem de elementos em $X = \mathcal{S} \times \mathcal{C}$, levando em consideração um conjunto $A \subseteq X$. As regras no conjunto A são selecionadas com base nos objetivos relacionados à interpretabilidade e ao desempenho, resultando em um conjunto de decisões $\mathcal{R}$ que representa uma solução local ótima suavizada.

O procedimento de amostragem, que é a parte central do algoritmo SLS, é definido como $\Phi_X(A, \delta)$. Esse procedimento amostra um subconjunto de X com viés δ com base em um conjunto $A \subseteq X$ da seguinte maneira: cada elemento $x \in X$ é amostrado de forma independente com probabilidade $p = (1 + \delta)/2$ se $x \in A$, e com probabilidade $p = (1 - \delta)/2$ se $x \notin A$.

O Algoritmo 5 começa com A sendo um conjunto vazio (linha 1) e calcula uma estimativa do valor ótimo da função objetivo f ao calcular `OPT` como $f(\Phi_X(X, 0))$ (linha 2), onde $\Phi_X(X, 0)$ representa um conjunto de elementos amostrados aleatoriamente de X, com cada elemento em X sendo escolhido com probabilidade 1/2.

Em seguida, o algoritmo estima o efeito de cada elemento $x \in X$ (linhas 3 a 5) calculando a diferença no valor da função objetivo quando x é adicionado a $\Phi_X(A, \delta)$ e quando x é removido de $\Phi_X(A, \delta)$. Essas estimativas são calculadas com base em amostragem repetida até que o erro padrão para a diferença nos escores seja menor do que $1/|X|^2$ vezes o valor ótimo `OPT`.

Depois de calcular essas estimativas, o algoritmo procura elementos $x \in X$ para adicionar a A verificando se a estimativa para algum $x \in X$ é maior do que $2/|X|^2$ vezes o valor ótimo `OPT` (linha 6). Se um elemento adequado for encontrado, ele é adicionado a A, e o processo é reiniciado a partir da linha 3 (linhas 7 e 8). Se nenhum elemento

adequado for encontrado para adicionar, o algoritmo procura elementos $x \in A$ para remover, verificando se a estimativa para algum $x \in A$ é menor do que $-2/|X|^2$ vezes o valor ótimo `OPT` (linha 9). Se um elemento adequado for encontrado para remover, ele é removido de A, e o processo é reiniciado a partir da linha 3 (linhas 10 e 11).

Quando não é possível adicionar nem remover mais elementos de A, o algoritmo SLS retorna $\Phi_X(A, \delta')$, que é um subconjunto aleatório de X amostrado com viés δ' em relação a A (linha 12).

O algoritmo SLS tem uma complexidade temporal polinomial e oferece uma solução de qualidade garantida. Se o algoritmo for executado com duas configurações de parâmetros diferentes: $(\delta = 1/3, \delta' = 1/3)$ e $(\delta = 1/3, \delta' = -1)$, a melhor das duas soluções tem um valor esperado de pelo menos $(2/5 - o(1))$ vezes o valor ótimo f^*, onde f^* representa o valor ótimo da função objetivo f sobre X, como demonstrado por Feige, Mirrokni e Vondrák (2011). Portanto, o SLS é usado para construir um conjunto de decisões que maximize a função objetivo, considerando critérios de interpretabilidade e desempenho das previsões. O algoritmo é executado duas vezes com as duas configurações de parâmetros mencionadas, e a melhor solução é escolhida de acordo com a função objetivo.

Finalmente, o algoritmo `IDS` apresenta uma variedade de características que o torna uma escolha relevante dentro do escopo desta pesquisa. De acordo com a análise realizada por Boll e Clair (1995), o `IDS` tem como um de seus objetivos a construção de árvores rasas que generalizem eficientemente. Para alcançar esse objetivo, procura-se o melhor atributo de ramificação que mantenha a árvore o mais rasa possível, resultando em regras com poucas condições, o que as torna mais compreensíveis para os usuários. Além disso, esse algoritmo adota uma abordagem que equilibra acurácia, interpretabilidade e eficiência computacional, evitando a sobreposição de regras em uma mesma instância de dados, ao mesmo tempo em que leva em consideração as classes minoritárias e cobre todas as *features*.

Nesta pesquisa, foi utilizada a implementação do `IDS` realizada por Filip e Kliegr (2019)[3], que, de acordo com os autores, é até várias ordens de magnitude mais rápida do que a implementação original. Isso torna essa implementação uma escolha eficiente para aplicar o algoritmo `IDS` em seu contexto de pesquisa.

2.6.2.4 `RIPPER` O algoritmo de Poda Incremental Repetida para Produzir Redução de Erro (`RIPPER` - do inglês, *Repeated Incremental Pruning to Produce Error Reduction*) foi introduzido por Cohen (1995) e projetado especificamente para competir com as regras `C4.5`, oferecendo desempenho competitivo e, ao mesmo tempo, funcionando de forma mais eficiente, excedendo o desempenho das árvores de decisão. Esse algoritmo pode ser entendido em um processo de três etapas: crescer, podar e otimizar.

Segundo Cohen (1995), na primeira etapa, **crescer**, usa um método "separar e conquistar" (do inglês, *separate-and-conquer*, cunhado por Pagallo e Haussler (1990)) para adicionar condições a uma regra até que ela seja perfeitamente classificada como um subconjunto de dados. Logo em seguida, um critério de ganho de informações é usado para identificar o próximo atributo de divisão. Quando o aumento da especificidade de uma

[3]Disponível em <https://github.com/jirifilip/pyIDS>

regra não reduz mais a entropia, a regra é imediatamente **podada**. Essas duas etapas são repetidas até atingir o critério de parada, momento em que todo o conjunto de regras é **otimizado** usando uma variedade de heurísticas.

De acordo com Fürnkranz e Kliegr (2015), o algoritmo `RIPPER` é um sistema de aprendizado de regras que foi projetado para combater efetivamente o problema de *overfitting* em aprendizado de máquina. O `RIPPER` aborda esse problema por meio da técnica de poda incremental de erro reduzido (`I-REP` - do inglês, *Incremental Reduced Error Pruning*) proposta por Fürnkranz e Widmer (1994). A ideia central por trás da técnica é remover uma regra de um conjunto de regras previamente aprendido e tentar reaprendê-la não apenas no contexto de regras anteriores, mas também no contexto de regras subsequentes. O Algoritmo 6 apresenta uma versão de duas classes do `IREP`.

Algoritmo 6: IREP

```
   Input: Pos, Neg
   Output: RuleSet
 1 RuleSet = ∅;
 2 while Pos ≠ ∅ do
       // grow and prune a new rule
 3     spit(Pos, Neg) into (GrowPos, GrowNeg) and (PrunePos,
        PruneNeg);
 4     Rule = GrowRule(GrowPos, GrowNeg);
 5     Rule = PruneRule(Rule, PrunePos, PruneNeg);
 6     if the error rate of Rule on (PrunePos, PruneNeg) exceeds 50% then
 7         return RuleSet;
 8     else
 9         add Rule to RuleSet;
10         remove examples covered by Rule from (Pos, Neg);
11 return RuleSet;
```

O algoritmo `IREP` é iniciado com dois conjuntos de dados de entrada denominados "Pos" e "Neg". O conjunto "Pos" contém exemplos classificados como positivos, enquanto o conjunto "Neg" contém exemplos classificados como negativos. O algoritmo começa com a inicialização de um conjunto de regras vazio chamado "RuleSet" (na linha 1) e entra em um ciclo que continua enquanto ainda houver exemplos no conjunto "Pos" (na linha 2). Este ciclo tem a responsabilidade de crescer e podar novas regras.

Na linha 3, os exemplos descobertos são aleatoriamente divididos em dois pares de subconjuntos. Um par de conjuntos é designado para o crescimento e contém exemplos positivos ("GrowPos") e negativos ("GrowNeg"). O outro par de conjuntos é designado para a poda e também contém exemplos positivos ("PrunePos") e negativos ("PruneNeg"). O conjunto de crescimento representa 2/3 dos exemplos.

Na linha 4, uma nova regra é gerada a partir dos subconjuntos "GrowPos" e "GrowNeg" usando a função "GrowRule". Esta função adiciona repetidamente uma condição que maximiza o critério de ganho de informações do `FOIL` (QUINLAN, 1990), conforme

formulado na equação (21), até que a regra não cubra mais exemplos negativos no conjunto de dados de crescimento. À medida que a regra se torna mais complexa com condições adicionais, ela se torna mais específica e exclui cada vez mais exemplos da classe negativa.

A equação (21) é definida como:

(21)
$$p_0 \cdot [\log_2(\frac{p_1}{p_1 + n_1}) - \log_2(\frac{p_0}{p_0 + n_0})]$$

onde p_0 e n_0 representam o número de exemplos positivos e negativos, respectivamente, cobertos por uma regra existente, enquanto p_1 e n_1 representam os números cobertos pela nova regra proposta.

Depois de crescer uma regra, a regra é imediatamente podada usando a função "PruneRule", utilizando os subconjuntos "PrunePos"e "PruneNeg"(linha 5). Para podar uma regra, essa função considera excluir qualquer sequência final de condições da regra e escolhe a exclusão que maximiza a métrica de poda formulada em (22).

Após a etapa de crescimento de uma regra, ela é imediatamente podada usando a função “PruneRule” com base nos subconjuntos “PrunePos” e “PruneNeg” (na linha 5). Para podar uma regra, essa função considera a exclusão de qualquer sequência final de condições da regra e escolhe a exclusão que maximiza a métrica de poda definida na equação (22).

A equação (22) é definida como:

(22)
$$v = \frac{p - n}{p + n}$$

onde n representa o número de exemplos em “PruneNeg” cobertos pela regra e p representa o número de exemplos em “PrunePos” cobertos pela regra. Esse processo de poda é repetido até que nenhuma exclusão melhore o valor de v.

Na linha 6, o algoritmo verifica se a taxa de erro da regra em relação aos exemplos em “PrunePos” e “PruneNeg” excede 50%. Se isso acontecer, o algoritmo retorna o conjunto atual de regras (na linha 7), encerrando o processo de aprendizado. Se a taxa de erro for menor ou igual a 50%, a regra é adicionada ao conjunto de regras “RuleSet” (na linha 9). Além disso, os exemplos cobertos pela regra recém-gerada são removidos dos conjuntos “Pos” e “Neg” (na linha 10).

Após a conclusão do ciclo, quando não há mais exemplos positivos para construir regras, o algoritmo retorna o conjunto final de regras “RuleSet” (na linha 11), que representa o conjunto de regras aprendidas.

Para implementar o algoritmo `RIPPER`, Cohen (1995) aplicou os resultados do `IREP` em um otimizador, a fim de produzir um novo conjunto de regras otimizadas, conforme exemplificado no Algoritmo 7.

Algoritmo 7: RIPPER

Input: Pos, Neg, k
Output: RuleSet

```
1 RuleSet = IREP(Pos, Neg) ;                          // initial rule set
2 while k > 0 do
3 |   RuleSet = OptimizeRuleSet(RuleSet);
4 |   Add rules in RuleSet to cover any remaining positive examples using IREP;
5 |   k = k − 1;
6 return RuleSet;
```

O algoritmo `RIPPER` utiliza o `IREP` para criar um conjunto inicial de regras a partir dos exemplos positivos e negativos fornecidos (linha 1). Isso representa a etapa inicial de construção do conjunto de regras.

Em seguida, o algoritmo procede à otimização do conjunto de regras previamente gerado por meio de um algoritmo dedicado à otimização de regras (indicado na linha 3). Além disso, são incorporadas novas regras ao conjunto "RuleSet" com o propósito de cobrir quaisquer exemplos positivos remanescentes (como indicado na linha 4). Essa inclusão de novas regras é realizada também através do `IREP`. O algoritmo itera esse processo de otimização e acréscimo de novas regras por um total de k vezes (linha 2). Por fim, o algoritmo retorna o conjunto de regras "RuleSet" resultante (na linha 6).

A função de otimização de regras (linha 3) aprimora o conjunto de regras produzido pelo `IREP` por meio de pós-processamento. Cohen (1995) propõe um conjunto de etapas para otimizar um conjunto de regras: (i) cada regra é considerada na ordem em que foi aprendida, uma de cada vez. (ii) Para cada regra, são construídas duas alternativas: uma de "Substituição" e outra de "Revisão". A alternativa de substituição é formada pelo crescimento (começando com um conjunto de regras vazio) e, em seguida, pela poda de uma regra, com a poda orientada para minimizar a taxa de erro de todo o conjunto de regras nos dados de poda. A alternativa de revisão é construída de maneira semelhante à de substituição, exceto que a revisão é realizada pela adição gulosa de condições à regra, em vez de começar com uma regra vazia. (iii) Das três regras - a original e as duas alternativas - é escolhida aquela com menor tamanho em bits, seguindo o princípio de que a simplicidade é preferível.

Além de apresentar um desempenho superior, o algoritmo `RIPPER` possui outras vantagens em relação ao algoritmo de árvores de decisão `C4.5`. Enquanto o `C4.5` é considerado um método indireto de geração de regras, pois requer a construção prévia de uma árvore de decisão para derivar as regras, o `RIPPER` é um método direto, aprendendo as regras diretamente dos dados de treinamento. Essa característica torna o aprendizado de regras mais atraente, uma vez que, de acordo com Fürnkranz, Gamberger e Lavrač (2012), as árvores de decisão, mesmo aquelas podadas, tendem a ser complexas e de difícil interpretação.

São diversas as aplicações práticas que podem se beneficiar das vantagens do algoritmo `RIPPER`, uma vez que seu uso é bastante diversificado. Por exemplo, em um estudo

conduzido por Sasaki e Kita (1998), o RIPPER foi utilizado para categorizar hierarquias de documentos da Web, possibilitando uma recuperação eficaz de informações na Web. Já em outro estudo, Provost (1999) empregou o algoritmo para filtragem de spam em e-mails. Além disso, Seerat e Qamar (2015) desenvolveram um sistema de apoio à decisão clínica que auxilia no diagnóstico de doenças por meio da análise dos dados dos pacientes, utilizando o RIPPER. Também é possível mencionar o trabalho de Xu, Ding e Pan (2018), que desenvolveram um modelo interpretável de avaliação de crédito para prever o comportamento de usuários de cartão de crédito, utilizando o algoritmo RIPPER. O algoritmo também encontrou aplicação na área de física de partículas, como demonstrado em um estudo conduzido por Britsch, Gagunashvili e Schmelling (2009).

É verdade que, em algumas situações, o algoritmo RIPPER pode apresentar desempenho inferior a outros algoritmos mais populares, como o Naive Bayes. Um estudo conduzido por Provost (1999), por exemplo, comparou esses dois algoritmos em um sistema de classificação de spam e constatou que o Naive Bayes superou significativamente o RIPPER. O autor atribui essa vantagem à habilidade do Naive Bayes de utilizar contagens de palavras em *bags of words* (sacos de palavras) para calcular suas tabelas de probabilidade.

Além disso, em um estudo mais recente conduzido por Seerat e Qamar (2015), o algoritmo RIPPER mostrou desempenho inferior em comparação com outros algoritmos supervisionados, como Naive Bayes, kNN e SVM, em um sistema de diagnóstico de doenças baseado em análise de dados de pacientes. Os autores atribuíram essa inferioridade à capacidade limitada do RIPPER em lidar com dados faltantes.

No entanto, é importante destacar que o algoritmo RIPPER ainda é considerado competitivo, eficaz na prevenção de *overfitting* e capaz de gerar regras facilmente compreensíveis. De acordo com pesquisas, como o estudo de Asadi e Shahrabi (2016), o RIPPER é considerado um dos melhores algoritmos de aprendizado de regras atualmente em uso. Igualmente, estudos mais recentes, como o de Moscovitz (2019), afirmam que o RIPPER ainda é considerado o estado da arte nessa técnica.

2.7 CONSIDERAÇÕES FINAIS

Este capítulo apresentou alguns conceitos e o embasamento teórico relacionados ao tema deste trabalho. Foram introduzidas as entidades que compõem as relações temporais, como eventos e expressões temporais, e os tipos dessas relações. Além disso, foi apresentado o *corpus* anotado com informações temporais que foi utilizado na tarefa de identificação do tipo de relação temporal entre as entidades.

Também foi discutida a nossa abordagem baseada em regras, na qual investigaremos a aplicação de técnicas de aprendizagem de regras de classificação para construir nossos conjuntos de regras. A aprendizagem de regras de classificação é um tipo específico de aprendizagem de regras de associação, que busca encontrar relações frequentes entre objetos.

Por fim, foram apresentados quatro algoritmos que serão explorados para resolver problemas de classificação com regras de associação. A escolha desses algoritmos foi justificada principalmente pela capacidade de lidar de forma eficaz com o problema de

overfitting, pela robustez na presença de ruídos nos dados, pela habilidade de generalizar bem para dados não vistos e pela geração de regras compreensíveis para os seres humanos. A utilização desses algoritmos forneceu uma contribuição valiosa para enfrentar os desafios de pesquisa deste trabalho, especialmente em relação à identificação de tipos de relações temporais.

Capítulo 3

TRABALHOS RELACIONADOS

Os trabalhos descritos neste capítulo apresentam a tarefa de identificar o tipo da relação temporal existente entre um evento e uma expressão temporal que ocorre dentro da mesma frase, conforme definida por Verhagen et al. (2007) para a Tarefa A do SemEval 2007. Exploraremos, particularmente, as informações linguísticas utilizadas nesses trabalhos, bem como os métodos computacionais empregados com o objetivo de solucionar essa tarefa.

Este capítulo está organizado da seguinte forma. A primeira seção apresenta um conjunto de trabalhos com contribuições significativas para a identificação de tipos de relações temporais por meio da combinação de informações linguísticas, resultando em avanços no estado da arte para a língua inglesa. Na segunda seção, é apresentado um trabalho específico voltado para a língua portuguesa. Este trabalho é, até onde sabemos, o único estudo que aborda o tema para a língua portuguesa e contribui com diferentes abordagens para a combinação de informações linguísticas usadas na solução do problema.

3.1 IDENTIFICAÇÃO DE TIPOS DE RELAÇÕES TEMPORAIS

Nesta subseção, focamos nas contribuições dadas por cada autor sobre as informações linguísticas utilizadas em seus métodos. São apresentados os sinais temporais, eles são conjunções e advérbios com função de explicitar a natureza da relação temporal, além de um framework de tempos e aspectos verbais que fornece informações básicas sobre ordenação temporal. Ainda nessa subseção, é apresentada uma abordagem híbrida com estratégia de combinar regras manuais com aprendizagem de máquina, além de contribuições de outros trabalhos com mais informações linguísticas criativas.

Na pesquisa conduzida por Derczynski (2017), foi realizada uma investigação sobre a identificação de tipos de relações temporais em textos por meio de duas abordagens distintas. Um dos primeiros passos do autor consistiu na definição de relações temporais "difíceis", ou seja, aquelas que os sistemas desenvolvidos para as tarefas de avaliação do TempEval-2 não eram capazes de acertar de forma consistente. Nesse conjunto de relações difíceis, foram identificados grandes grupos de relações temporais usando sinais temporais explícitos e outros grupos usando mudança de tempo e aspecto. De acordo

com o autor, a superação dessas relações difíceis é essencial para o progresso da pesquisa nessa área.

3.1.1 Sinais Temporais

Na primeira abordagem para melhorar a identificação de tipos de relações temporais, Derczynski (2017) baseou-se em palavras e frases que explicitam a natureza de uma relação temporal, denominada de sinais temporais, que são conjunções e advérbios temporais. O autor utiliza os sinais temporais anotados no TimeBank para criar um classificador capaz de identificá-los automaticamente e associá-los a um par de entidades (evento ou expressão temporal), assim as anotações de sinais do *corpus* são aumentadas e a sua qualidade aperfeiçoada. Os sinais temporais foram apresentados como um novo recurso linguístico e sua utilização para ajudar a identificar o tipo da relação temporal alcançou mais de 53% de redução de erro quando comparada com o mesmo sistema sem informações de sinal. Esse trabalho resultou em um novo *corpus*, em inglês, que foi disponibilizado publicamente com o nome de TB-sig[1].

É importante destacar que os sinais temporais anotados no *corpus* TimeBank para a língua inglesa não foram anotados na tradução para o português no *corpus* TimeBankPT. Para contornar essa limitação, esta pesquisa adota uma abordagem que consiste na criação de uma lista de palavras que denotam tempo. Essa estratégia se fundamenta no trabalho realizado por Derczynski e Gaizauskas (2012), no qual são identificados os termos mais frequentemente utilizados na língua inglesa para expressar noções de tempo. Além disso, fizemos uso da lista de relacionadores de tempo fornecida por fontes de referência respeitáveis, como "A University Grammar of English" (QUIRK; GREENBAUM, 1972, p. 285–287), a fim de fundamentar a construção dessa lista. Essa abordagem permitirá a identificação de palavras e expressões relacionadas ao tempo no contexto específico da língua portuguesa.

A vinculação de um sinal temporal a pares de entidades tem a capacidade de descrever a natureza da relação existente entre o evento e a expressão temporal. Para ilustrar a influência exercida pelos sinais temporais no discurso, no exemplo (23a), ocorre um evento, a entrega das provas, que é temporalmente relacionado a uma expressão temporal, doze horas. Nesse cenário, a palavra antes desempenha o papel de um sinal que descreve a natureza da relação temporal entre essas entidades. Já no exemplo (23b), a função temporal do sinal temporal depois se torna evidente, estabelecendo um contexto onde um evento é ordenado em relação ao outro. Nesse caso, o sinal temporal indica a sequência em que os eventos ocorreram, deixando claro que primeiro Nanna trabalhou e, posteriormente, dormiu.

(23) a. *As provas foram entregues antes das doze horas.*

b. *Nanna dormiu depois de um longo dia de trabalho.*

De acordo com Derczynski (2017), esses sinais temporais podem surgir junto com conexões temporais e oferecem informações explícitas acerca do tipo de relação temporal.

[1]TB-sig disponível em <http://derczynski.com/sheffield/>

Para a realização da tarefa de associação de sinais, o autor empregou um classificador binário para determinar se o par constituído pela relação temporal e um sinal está vinculado. O autor utilizou as seguintes *features* em seu classificador.

- ***signal-event-closest-follow*:** *feature* booleana que verifica se esse evento é textualmente o mais próximo após o sinal
- ***signal-timex3-closest-follow*:** *feature* booleana que verifica se essa expressão temporal é textualmente o mais próximo após o sinal
- ***signal-event-closest-precede*:** *feature* booleana que verifica se esse evento é textualmente o mais próximo antes do sinal
- ***signal-timex3-closest-precede*:** *feature* booleana que verifica se essa expressão temporal é textualmente o mais próximo antes do sinal
- ***signal-ancestor-event*:** o sinal domina sintaticamente o evento?
- ***signal-ancestor-timex3*:** o sinal domina sintaticamente a expressão temporal?
- ***signal-text*:** texto do sinal
- ***signal-pos*:** classe gramatical do sinal
- ***signal-distance-event*:** distância em número de *tokens* do evento até o sinal
- ***signal-distance-timex3*:** distância em número de *tokens* da expressão temporal até o sinal
- ***signal-event-first-order*:** ordem textual entre o evento e o sinal. Verdadeiro se evento vier primeiro
- ***signal-timex3-first-order*:** ordem textual entre a expressão temporal e o sinal. Verdadeiro se expressão temporal vier primeiro
- ***signal-comma-between-event*:** há uma vírgula entre o evento e o sinal?
- ***signal-comma-between-timex3*:** há uma vírgula entre a expressão temporal e o sinal?
- ***signal-child-event*:** o sinal é regido sintaticamente pelo evento?
- ***signal-child-timex3*:** o sinal é regido sintaticamente pela expressão temporal?
- ***signal-is-event-head*:** o sinal é o regente direto do evento?
- ***signal-is-timex3-head*:** o sinal é o regente direto da expressão temporal?
- ***event-root*:** o evento é a raiz da árvore de dependência sintática?

- ***timex3-root*:** a expressão temporal é a raiz da árvore de dependência sintática?
- ***signal-dep-advmod-advcl-event*:** o evento está diretamente relacionado ao sinal com o tipo de dependência sintática `advmod` ou `advcl`?
- ***signal-dep-advmod-advcl-timex3*:** a expressão temporal está diretamente relacionada ao sinal com o tipo de dependência sintática `advmod` ou `advcl`?
- ***event-head-is-root*:** o evento modifica diretamente a raiz da sentença? (ex: o evento é um filho direto da raiz na arvore de dependência)
- ***timex3-head-is-root*:** a expressão temporal modifica diretamente a raiz? (ex: a expressão temporal é um filho direto da raiz na arvore de dependência)
- ***signal-head-is-event*:** o sinal modifica o evento diretamente? (ex: o sinal é um filho direto do evento)
- ***signal-head-is-timex3*:** o sinal modifica a expressão temporal diretamente? (ex: o sinal é um filho direto da expressão temporal)
- ***event-dep*:** que relação de dependência o evento tem com seu pai?
- ***timex3-dep*:** que relação de dependência a expressão temporal tem com seu pai?
- ***signal-dep-if-child-event*:** se o sinal é um filho do evento, qual é o tipo de relação de dependência sintática?
- ***signal-dep-if-child-timex3*:** se o sinal é um filho da expressão temporal, qual é o tipo de relação de dependência sintática?

Iremos incorporar algumas dessas *features* ao nosso conjunto de *features* para a tarefa de identificar o tipo da relação temporal, mesmo que originalmente elas tenham sido criadas para uma tarefa diferente: associar um sinal temporal a pares de entidades temporais. Nossa motivação para essa incorporação é que, segundo o autor, apesar de os sinais temporais serem complexos e úteis o suficiente para justificar uma investigação independente, o seu processamento pode potencialmente ser incluído como parte das *features* utilizadas para a tarefa de identificar o tipo da relação temporal. O autor justifica essa inclusão afirmando que, uma vez encontradas as *features* importantes para a representação do sinal, pode-se pular a tarefa de anotação do sinal e incluir estas *features* em um classificador de tipo de relação temporal. Relatamos na Seção 4.1.5, algumas alterações necessárias para adaptá-las a nossa abordagem.

Para a tarefa de identificar o tipo da relação temporal, o autor adota uma abordagem de aprendizagem de máquina, fazendo uso de classificadores de máxima entropia (JAYNES, 1982). As *features* utilizadas que se aplicam ao *corpus* em português são:

- ***event-text*:** texto do evento
- ***timex3-text*:** texto da expressão temporal

- ***event-tense*:** tempo verbal do evento
- ***event-aspect*:** aspecto do evento
- ***event-modal-verb*:** texto do verbo modal antes do evento
- ***event-has-modal-verb-precede*:** *feature* booleana que verifica se o evento possui verbos modais antes dele (esta *feature* está relacionada a *event-modal-verb* e foi utilizada por Kolya et al. (2013))
- ***event-polarity*:** polaridade do evento
- ***event-pos*:** classe gramatical do evento
- ***event-class*:** classe do evento anotado no *corpus*
- ***timex3-temporalfunction*:** `temporalFunction` da expressão temporal anotado no *corpus*
- ***timex3-value*:** valor normalizado da expressão temporal
- ***timex3-type*:** tipo da expressão temporal
- ***event-first-order*:** o evento precede textualmente a expressão temporal?

Em relação aos verbos modais, eles não foram anotados no TimeBankPT. Em nosso trabalho experimentaremos a utilização de lista de palavras que contêm os principais verbos modais na língua portuguesa e suas variações, por exemplo: *poder, dever, haver, precisar, ter, conseguir* e *querer*.

Em suma, segundo o autor, os resultados da adição de sinais temporais e das *features* derivadas dele foram positivas, melhorando o desempenho absoluto do classificador de tipo de relação temporal em 18% para links de *event-event* e 2,0% para links de *event-time*.

3.1.2 *Framework* de Reichenbach

A segunda abordagem de Derczynski (2017) para melhorar a extração de relações foca em um *framework* de tempo e aspecto verbal que pode fornecer informações básicas de ordenação temporal entre alguns eventos e expressões de tempo. Tempo e aspecto verbal são usados para descrever aspectos temporais de eventos que são expressos com verbos e é intuitivo que eles são relevantes para determinar tipos de relações temporais.

O autor faz uso do trabalho de Reichenbach (1947) que oferece uma estrutura teórica para análise de tempo e aspecto a ser usada para prever ordenação temporal entre eventos verbais e também entre expressão temporal e evento verbal.

O núcleo do *framework* de Reichenbach compreende três momentos abstratos: (i) momento da fala (S – do inglês, *speech time*) que representa o momento em que o verbo é pronunciado ou escrito, (ii) momento do evento (E – do inglês, *event time*) que é o momento em que o evento introduzido pelo verbo ocorre e (iii) momento de referência (R

– do inglês, *reference time*) que é um momento abstrato, a partir do qual os eventos são vistos. Esses três momentos abstratos estão relacionados entre si em termos de igualdade, precedência ou sucessão.

Na Tabela 3.1, temos os tempos detalhados por Reichenbach: passados, presentes ou futuros; eles podem tomar uma forma simples, anterior ou posterior. Há também os tempos equivalentes em inglês e em português.

Tabela 3.1: Os tempos de Reichenbach (REICHENBACH, 1947)

Relação	**Tempo de Reichenbach**	**Tempo em Inglês**	**Exemplo**	**Tempo em Português**
E<R<S	Passado Anterior	*Past perfect*	*I had slept*	Pretérito mais-que-perfeito composto
E=R<S	Passado Simples	*Simple past*	*I slept*	Pretérito perfeito
R<E<S R<S=E R<S<E	Passado Posterior		*I expected that I would sleep*	Futuro do pretérito
E<S=R	Presente Anterior	*Present perfect*	*I have slept*	Pretérito perfeito composto
S=R=E	Presente Simples	*Simple present*	*I sleep*	Presente
S=R<E	Presente Posterior	*Simple future*	*I will sleep (Je vais dormir)*	Futuro do presente
S<E<R S=E<R E<S<R	Futuro Anterior	*Future perfect*	*I will have slept*	Futuro do presente composto
S<R=E	Futuro Simples	*Simple future*	*I will sleep (Je dormirai)*	Futuro do presente
S<R<E	Futuro Posterior		*I shall be going to sleep*	

Fonte: Adaptado de Mani, Pustejovsky e Gaizauskas (2005)

Ao seguir os nomes dos tempos verbais de Reichenbach, é o caso de que, para os tempos passados, R sempre ocorre antes de S; no futuro, R está sempre depois de S; e no presente, R e S são simultâneos. Além disso, "anterior" sugere E antes de R, "simples" que R e E são simultâneos, e "posterior" que E depois de R. Utilizando os operadores lógicos, os tempos passados, presentes e futuros implicam $R < S$, $R = S$ e $S < R$, respectivamente. Os tempos anteriores, simples e posteriores implicam $E < R$, $E = R$ e $R < E$, respectivamente.

No entanto, o autor Derczynski (2017) reconhece a ausência de uma correspondência precisa entre os tempos verbais em inglês e os nove tempos especificados por Reichenbach. Um exemplo disso ocorre em situações em que o tempo verbal é o futuro simples, podendo os correspondentes de Reichenbach serem classificados como "presente posterior" ou "futuro simples", conforme apresentado na Tabela 3.1. O primeiro caso denota um futuro iminente, tal como exemplificado em *"Vou dormir"*, enquanto o segundo caso exprime um futuro mais distante, como ilustrado em *"Eu dormirei"*.

Apesar disso, o *framework* de Reichenbach fornece regras explícitas sobre o papel das datas e horas em relação ao verbo dentro de seu contexto temporal. Veremos um método básico de determinar o tipo da relação temporal entre expressão temporal e evento verbal, fazendo uso de propriedades especiais do momento de referência R.

Uma dessas propriedades refere-se a quando a expressão temporal (`TIMEX3` do tipo `DATE` ou `TIME`, mas não `DURATION`) ocorre na mesma frase de um evento verbal modificando sintaticamente o evento, ela é usada para posicionar o momento de referência R. Ou seja, nesse caso, o momento de referência R equivale à expressão temporal. Este princípio é nomeado como uso posicional do momento de referência. Já o momento da fala S é equivalente a Data de Criação do Documento (DCT), a menos que seja explicitamente posicionado pelo discurso. Este princípio é nomeado como uso posicional do momento da fala.

Com o intuito de exemplificar o uso posicional do momento de referência, na frase (24), o tempo explicitamente mencionado (10 horas) determina o nosso momento de referência R.

(24) *Eram 10 horas e Sarah tinha escovado os dentes.*

O tempo do grupo verbal "tinha escovado" é pretérito mais-que-perfeito composto, portanto, passado anterior, conforme Tabela 3.1; ou seja, E < R < S. O momento do evento E é concluído antes do momento de referência R – ou seja, a qualquer momento até às 10 horas – e assim a relação entre o evento e a expressão temporal pode ser determinada (escovado **antes** das 10 horas).

De acordo com o autor, o *framework* de Reichenbach é insuficiente para rotular relações temporais com base em regras. No entanto, vamos implementar as quatro *features* propostas pelo autor, descritas a seguir, para serem adicionadas ao conjunto geral de *features* que serão integradas à nossa abordagem de aprendizagem de regras.

- ***reichenbach-tense*:** como estamos ligando um evento a uma expressão temporal sob a suposição de que há o uso posicional do momento de referência R, ou seja, R será considerado equivalente à expressão temporal, a identificação do tipo da relação temporal será entre E e R. Observando a Tabela 3.1, vemos que o tempo verbal de Reichenbach para a ordenação de E/R ocorre entre os tempos anteriores onde E < R, os tempos simples onde E = R e os tempos posteriores onde E > R. Assim, o rótulo dessa *feature* determinando a relação E/R (que também é a relação entre o evento e a expressão temporal) assume o valor anterior, simples ou posterior
- ***timex3-dep*:** relação de dependência sintática da expressão temporal com seu regente (nó pai).

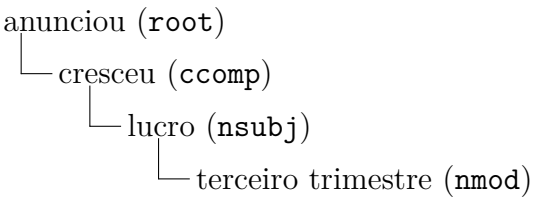
- ***reichenbach-direct-modification*:** a expressão temporal modifica diretamente o evento? (a expressão está no mesmo caminho de dependência do evento?)
- ***reichenbach-temporal-mod-function*:** Função de modificação temporal, existe uma relação `tmod` no caminho de dependência do evento à expressão temporal?

  Destacamos que a relação `tmod` não há na língua portuguesa, segundo a *Universal Dependencies*[2]. A relação `tmod` é um modificador temporal de outro nominal e é um subtipo da relação `nmod`[3] (modificador nominal), assim, iremos supor que a relação `nmod` possa substituir `tmod`, mesmo sendo mais abrangente.

(25) *A Du Pont Co.* ***anunciou*** *que o lucro do terceiro trimestre cresceu uns robustos 19% desde há um ano por causa da força dos negócios da companhia em vários produtos químicos e fibras, e em petróleo.*

Para ilustrar a aplicação desta *feature*, consideramos a sentença (25) do *corpus* TimeBankPT. Essa sentença, destaca a relação temporal entre o evento "anunciou" e a expressão temporal "terceiro trimestre".

Na análise da estrutura hierárquica subsequente e do caminho de dependência entre essas duas entidades, temos que o evento "anunciou" desempenha o papel de nó principal na árvore de dependência sintática, exercendo regência sobre o verbo "cresceu" por meio da relação de complemento clausal (`ccomp`). O verbo "cresceu", por sua vez, rege o substantivo "lucro" através da relação de sujeito nominal (`nsubj`). Nessa sequência, o substantivo "lucro" rege a expressão temporal "terceiro trimestre" por meio da relação de modificador nominal (`nmod`). Quando a relação `nmod` é encontrada no caminho de dependência do par *event-time*, o valor dessa *feature* é marcado como verdadeiro; caso contrário, é considerado falso.



A utilização do *framework* de Reichenbach proporcionou uma melhora de desempenho de 4,08% de redução de erro para um classificador de máxima entropia (JAYNES, 1982). O aumento absoluto da acurácia foi de aproximadamente 1,4%; um ganho modesto. A conclusão do autor foi que a utilização desse *framework* não foi tão útil por si só.

3.1.3 Regras com Aprendizado de Máquina

A pesquisa de D'Souza (2015) adotou uma abordagem híbrida, rica em conhecimento, baseada em aprendizagem de máquina e em regras para a identificação do tipo da relação temporal, explorando uma versão mais complexa e desafiadora do problema quando

[2] <https://github.com/UniversalDependencies/docs/tree/pages-source/_pt/dep>
[3] <https://universaldependencies.org/u/dep/nmod-tmod.html>

comparada, por exemplo, com a versão de Mirza e Tonelli (2014). Em sua abordagem, D'Souza (2015) buscou identificar todos os 14 tipos de relações temporais anotados no *corpus* TimeBank (PUSTEJOVSKY et al., 2003), no domínio de notícias, e 12 tipos de relações no *corpus* i2b2 (SUN; RUMSHISKY; UZUNER, 2013), no domínio clínico. O *design* das regras implementado foi parcialmente baseado na intuição e parcialmente orientado por dados: primeiro utilizou-se a intuição para criar uma regra e depois refiná-la manualmente com base nas observações feitas nos dados do *corpus*.

A combinação das regras e classificadores com aprendizado de máquinas se deu de duas formas. Na primeira, foram utilizadas todas as regras como *features* adicionais para o treinamento dos classificadores. O valor de cada *feature* foi o tipo da relação temporal predita pela regra correspondente. Na segunda forma, foi aplicado em cada instância o conjunto de regras composto apenas por regras que possuíam pelo menos 80% de acurácia. Se nenhuma das regras fossem aplicáveis, os classificadores seriam empregados.

Para identificar o tipo da relação temporal, a autora utilizou diversas informações linguísticas sofisticadas envolvendo semântica e discurso, além das informações lexicais e gramaticais comumente utilizadas. Dessa forma, foi possível capturar uma semântica mais rica para a tarefa. Entre as informações linguísticas sugeridas, utilizaram-se tipos de relações lexicais presentes no dicionário online *Webster*[4], como as relações de sinônimos, palavras relacionadas e antônimos. Relações lexicais do *WordNet* (MILLER, 1995) como hiperônimos, hipônimos e tropônimos, também foram utilizadas. Outra informação linguística proposta foi a utilização de relações predicado-argumento do tipo direcional, modo, temporal e causa extraídas do rotulador de funções semânticas (SRL - do inglês, *Semantic Role Labeler*) SENNA (COLLOBERT et al., 2011). Foi ainda proposta a utilização de relações de discurso extraídas do analisador de discurso *end-to-end* de Lin, Ng e Kan (2014) como as relações retóricas de causalidade, elaboração e capacitação que poderiam auxiliar no acompanhamento da progressão temporal do discurso. Por fim, a autora introduz *features* em pares calculadas com base em ambos os argumentos da relação supondo que isso poderia capturar melhor a relação temporal entre eles já que a relação temporal é computada com base em ambos os elementos. As principais informações coletadas em pares foram listas de preposições e de tempo e aspecto verbal ao longo do caminho entre os argumentos da relação.

As principais *features* propostas pela autora relacionadas ao nosso trabalho são:

- ***event-is-ancestor-timex3*:** o evento é a entidade regente na relação?
- ***event-is-child-timex3*:** o evento é a entidade dependente na relação?
- ***timex3-is-ancestor-event*:** a expressão temporal é a entidade regente na relação?
- ***timex3-is-child-event*:** a expressão temporal é a entidade dependente na relação?
- ***event-tense*:** tempo verbal do evento
- ***event-pos-token-i-precede*:** classe gramatical dos cinco *tokens* anteriores ao evento

[4]<https://www.merriam-webster.com>

- ***event-pos-token-i-follow*:** classe gramatical dos cinco *tokens* após o evento
- ***timex3-pos-token-i-precede*:** classe gramatical dos cinco *tokens* anteriores à expressão temporal
- ***timex3-pos-token-i-follow*:** classe gramatical dos cinco *tokens* após a expressão temporal
- ***event-head-pos*:** classe gramatical do nó pai de evento na árvore de dependência
- ***timex3-head-pos*:** classe gramatical do nó pai de expressão temporal
- ***event-class*:** classe do evento anotada no *corpus*
- ***event-timex3-distance*:** distância em número de *tokens* entre o evento e expressão temporal
- ***dct-value*:** valor normalizado da DCT anotada no *corpus*

Quanto ao resultado, quando todos os recursos são usados, o sistema alcançou uma acurácia de 53,4% no *corpus* do TimeBank, que representa uma redução relativa de erro de cerca de 16% quando comparado com o *baseline*. Quando são consideradas apenas a identificação de tipos de relações temporais entre evento e expressão temporal (*event-time*) a acurácia foi de 65,4%, que representa uma redução relativa de erro de apenas 9% no TimeBank, aproximadamente.

Considerando os experimentos realizados no conjunto de dados do TimeBank, a autora conclui que a adição de *features* em pares, relações de dependência e as relações de discurso melhoram significativamente a acurácia. No entanto, a adição das relações do *Webster*, das relações do *WordNet* e do predicado-argumento do SRL não foram úteis para a identificação dos tipos das relações temporais.

3.1.4 Outras Contribuições

Complementando nosso o conjunto de *features*, destacamos o trabalho de Mirza e Tonelli (2014) que adotaram uma abordagem baseada em classificadores de Máquina de Vetor de Suporte, utilizando *features* mais simples com o argumento de que é possível alcançar resultados comparáveis a outros trabalhos com *features* mais complexa, ao mesmo tempo em que reduz o tempo de processamento necessário.

Em sua abordagem de utilizar *features* mais simples, as autoras basearam-se em informações básicas sobre a posição de eventos e expressões de tempo, classe gramatical, análise de dependência e informações proveniente de lista de sinais temporais. As *features* principais foram: *event-first-order, event-lemma, timex3-lemma, event-text, timex3-text, event-pos, event-timex3-distance, event-class, timex3-type, timex3-value, dct-value, event-root, event-is-ancestor-timex3, event-is-child-timex3, timex3-is-ancestor-event* e *timex3-is-child-event*. Sendo as *features* a seguir inéditas:

- ***event-gov-verb-tense*:** estimativa do tempo verbal do evento não verbal da relação sob classificação, aquele enquadrado na categoria de substantivo, adjetivo e preposição. O seu valor é calculado com base no tempo do verbo que rege esse evento, com base na relação de dependência sintática. Se o evento for verbo, o valor é calculado com base no próprio evento

- ***event-gov-verb-aspect*:** semelhante à *feature* anterior, porém o valor extraído é o aspecto verbal do verbo que rege o evento não verbal. Se o evento for verbo, o valor é calculado com base no próprio evento

- ***event-timex3-dep*:** relação de dependência entre o evento e a expressão temporal ou entre a expressão temporal e o evento. A diferenciação é feita através das *features* *event-is-ancestor-timex3* e *event-is-child-timex3* que informa quem é o regente e o dependente.

A acurácia da melhor configuração para classificação de *event-time* foi de 66,62%, segundo as autoras.

Por fim, destacamos o trabalho realizado por Bethard e Martin (2007) que também adotaram uma abordagem baseada em Máquina de Vetor de Suporte, usando uma variedade de informações lexicais, sintáticas e semânticas para caracterizar os diferentes tipos de relações temporais, entre elas: *event-aspect, event-class, event-has-modal-verb-precede, event-lemma, event-polarity, event-pos, event-tense, timex3-temporalfunction, timex3-text* e *timex3-value*. Sendo as *features* a seguir, inéditas:

- ***event-gov-verb*:** verbo que rege o evento sob classificação. Para eventos que são verbos, essa feature é o próprio evento

- ***timex3-gov-verb*:** verbo que rege a expressão temporal sob classificação

- ***event-gov-verb-tense*:** tempo verbal do verbo que rege o evento. Se evento for verbo, o valor dessa *feature* é o tempo verbal do próprio evento. Para eventos não verbais, esse valor é uma tentativa de estimar do tempo verbal do evento

- ***timex3-gov-verb-tense*:** tempo verbal do verbo que rege a expressão temporal

- ***event-gov-verb-aux*:** verbos auxiliares e advérbios que modificam o verbo regente do evento sob classificação. Se evento é verbo, o verbo regente é o próprio evento

- ***timex3-gov-verb-aux*:** verbos auxiliares e advérbios que modificam o verbo regente da expressão temporal sob classificação

- ***event-preposition-gov*:** preposição que rege sintaticamente o evento

- ***timex3-preposition-gov*:** preposição que rege sintaticamente a expressão temporal

- ***timex3-pos*:** classe gramatical da expressão temporal

- ***timex3-between-quant*:** quantidade de expressões temporais existentes entre o evento e expressão temporal sob classificação.

Para a classificação de *event-time*, a acurácia alcançada pelos autores foi de 61%.

3.2 RELAÇÕES TEMPORAIS NA LÍNGUA PORTUGUESA

Nesta subseção, descrevemos o trabalho mais relevante para a identificação de tipos de relações temporais na língua portuguesa. Exploraremos, particularmente, as informações linguísticas utilizadas nele. As contribuições com informações linguísticas vão desde aquelas fornecidas por ferramentas de processamento raso, como POS tagger e analisador morfológico, passando por maneiras diferentes de codificar as anotações TimeML do *corpus* de forma a torná-las mais vantajosa, até a exploração de semântica lexical trazendo um pouco de conhecimento prévio sobre o mundo para a solução do problema.

O *LX-TimeAnalyzer* (COSTA, 2012) representa o primeiro trabalho publicado do qual temos conhecimento que se dedica à identificação de tipos de relações temporais, bem como à identificação de eventos e expressões temporais na língua portuguesa. O autor destaca que um dos principais obstáculos iniciais ao processamento temporal nessa língua foi a ausência de dados anotados com informações explícitas sobre o tempo. Para superar essa limitação, sua equipe desenvolveu o TimeBankPT (COSTA; BRANCO, 2012), um corpus de textos em português com anotações temporais, que foi apresentado na Seção 2.4.

O autor experimenta diferentes estratégias com potencial para melhorar a identificação de tipos de relações temporais e defende que diferentes níveis de informações são necessários. Essas informações vão desde recursos extraídos de ferramentas de processamento raso até recursos obtidos com ferramentas de linguagem natural que fornecem informações mais ricas, além de informações de conhecimento prévio sobre o mundo. A abordagem adotada foi a utilização de técnicas de aprendizagem de máquina e a seguir listaremos as *features* consideradas mais interessantes para a solução de nosso problema.

3.2.1 *Baseline*

A seguir, as *features* usadas pelo autor nos classificadores de referência.

- ***event-class*:** classe do evento anotada no *corpus*
- ***event-lemma*:** lema do evento
- ***event-aspect*:** aspecto do evento
- ***event-tense*:** tempo verbal do evento
- ***event-polarity*:** polaridade do evento
- ***event-pos*:** classe gramatical do evento
- ***timex3-type*:** tipo da expressão temporal

- ***event-first-order*:** se evento precede textualmente a expressão temporal na relação sob classificação

- ***event-between-order*:** se há outro evento entre o evento e a expressão temporal da relação sob classificação

- ***timex3-between-order*:** se há outra expressão temporal entre o evento e a expressão temporal da relação sob classificação

- ***event-timex3-no-between-order*:** verifica se não há evento ou expressão temporal entre o evento e a expressão temporal da relação sob classificação (é verdadeiro se e somente se *timex3-between-order* e *event-between-order* são falsos).

3.2.2 Processamento Raso

Adotando uma abordagem mais simples, Costa (2012) implementou algumas *features* que aproveitam informações fornecidas por ferramentas de processamento raso, como *POS tagger* e analisador morfológico. Elas são brevemente descritas em seguida.

- ***event-conjunction-closest-precede*:** conjunção mais próxima antes do evento da relação processada

- ***event-conjunction-closest-follow*:** conjunção mais próxima após o evento da relação processada

- ***timex3-relevant-lemmas*:** o valor dessa *feature* baseia-se na expressão temporal que denota o tempo relacionado com a relação temporal que está sendo processada. O valor é computado assim: (i) cada palavra na expressão temporal é substituída por seu lema; (ii) cada um deles é removido a menos que esteja contida em uma pequena lista de lema que têm algum conteúdo temporal e são frequentemente vistos nas expressões temporais nos dados de treinamento.

  Essa lista inclui palavras como: *ainda, amanhã, anterior, anteriormente, atual, breve, brevemente, logo, cada, corrente, atual, este, esse, futuro, haver, hoje, já, passado, posterior, presente, próximo, seguinte, todo, recente, recentemente* e *último*

- ***timex3-preposition-precede*:** tem como valor a preposição que imediatamente precede a expressão temporal no texto, ou o valor `NONE` se essa palavra não for uma preposição. Por exemplo, "*após dez anos*", o valor da *feature* é "*após*"

- ***event-preposition-precede*:** tem como valor a primeira preposição que precede textualmente o evento (esta *feature* está relacionada a *timex3-preposition-precede* e foi utilizada por Kolya et al. (2013)).

3.2.3 Conhecimento Prévio Sobre o Mundo

Na tarefa de identificação de tipos de relações temporais, o conhecimento prévio sobre o mundo pode ser utilizado como uma estratégia para auxiliar no processo de análise. O termo "conhecimento prévio sobre o mundo" refere-se à informação prévia que um falante possui sobre eventos, pessoas, lugares etc., em seu ambiente. Essa informação pode ser utilizada para inferir possíveis relações temporais entre eventos e expressões temporais, por exemplo. É importante destacar que algumas relações temporais são mais esperadas do que outras apenas devido ao contexto e itens lexicais empregados. Além disso, o significado isolado das palavras envolvidas pode fornecer pistas temporais relevantes para a tarefa de identificação de relações temporais, como apontado por Costa (2012), que considerou essa abordagem valiosa para explorar a detecção de relações temporais.

(26) *Os analistas previam [que em 1990 a BellSouth visse lucros na casa dos 3,90 dólares por ação.]*

- ***event-temporal-direction*:** considere o exemplo em (26), o fato de que a expressão temporal ocorre no complemento deste verbo (entre colchetes) é uma boa indicação de que o evento precede a data por causa do que o verbo significa: previsões são feitas antes do previsto acontecer, e como o que está previsto é o lucro de *BellSouth* em 1990, o evento de previsão deve ter ocorrido antes do momento em que esses ganhos são anunciados.

  O autor denominou essa *feature* de "*direção temporal*". Para obter essas informações, todos os lemas de eventos presentes nos dados de treinamento foram extraídos e um mapeamento foi criado manualmente entre eles e a relação temporal esperada com seu complemento. A ideia foi gravar esse tipo de informação em uma *feature* para os classificadores. Foi disponibilizada pelo autor uma lista com cerca de 300 eventos e suas respectivas relações temporais esperadas com seus complementos. Seguem alguns exemplos:

  - *acusar*: AFTER
  - *atrasar*: BEFORE
  - *organizar*: BEFORE
  - *relatar*: AFTER

  De acordo com os exemplos, eventos de acusação seguem os eventos que alguém é acusado de fazer, eventos de atraso precedem eventos atrasados, eventos de organização precedem eventos organizados e eventos de relatos seguem eventos relatados. Assim, essa *feature* registra conhecimento prévio sobre o mundo.

3.2.4 Explorando as Anotações do TimeML

As anotações do TimeML fornecem informações para muito mais *features* do que o pequeno conjunto utilizado no *baseline* definido pelo autor. Existem várias maneiras de

aproveitar as anotações. Por exemplo, (i) os valores de alguns dos atributos podem ser simplificados ou codificados de maneiras diferentes que possam ser mais vantajosos e (ii) outros eventos ou expressões temporais que ocorrem na mesma frase, além daqueles que participam da relação temporal sob classificação, podem conter informações relevantes para esta classificação. A seguir, são descritas algumas *features* adicionais que aproveitam essas ideias:

- ***event-closest-to-timex3-pos*:** classe gramatical do evento mais próximo da expressão temporal da relação temporal sob classificação
- ***event-closest-to-timex3-equal-pos*:** *feature* booleana que verifica se a classe gramatical do evento da relação sob classificação é igual à classe gramatical do evento mais próximo da expressão temporal da relação sob classificação
- ***event-closest-to-event-equal-lemma*:** *feature* booleana que verifica se o lema do evento da relação temporal sob classificação é igual ao lema de outro evento que é mencionado no texto mais próximo a ele
- ***event-closest-to-event-pos*:** classe gramatical do evento que está mais próximo do evento que está na relação temporal sob classificação
- ***event-closest-to-event-equal-pos*:** *feature* booleana que verifica se a classe gramatical do evento da relação temporal sob classificação é igual à classe gramatical do evento que é mencionado no texto mais próximo a ele
- ***event-closest-to-event-class*:** classe anotada do evento mais próximo do evento que está na relação temporal sob classificação
- ***event-closest-to-event-equal-class*:** *feature* booleana que verifica se o valor da classe do evento da relação temporal sob classificação é igual à classe do evento mais próximo a ele no texto
- ***event-closest-to-event-tense*:** tempo verbal do evento que está mais próximo do evento que está na relação temporal sob classificação
- ***event-closest-to-event-equal-tense*:** *feature* booleana que verifica se o tempo verbal do evento da relação temporal sob classificação é igual ao tempo verbal do evento mais próximo a ele no texto
- ***event-closest-to-event-temporal-direction*:** valor da direção temporal (conhecimento prévio sobre o mundo) para o evento mais próximo do evento da relação temporal sob classificação
- ***event-intervening-preceding-class*:** classe do evento que é mencionado entre a expressão temporal e o evento, nesta ordem textual, da relação temporal sob classificação e está mais próximo da expressão temporal.
  Por exemplo, `TIMEX -- event2.class ------ EVENT`

- ***event-intervening-following-tense*:** tempo verbal do evento que é mencionado entre o evento e a expressão temporal, nesta ordem textual, da relação temporal sob classificação e está mais próximo da expressão temporal.
  Por exemplo, `EVENT ------ event2.tense -- TIMEX`.

Para a obtenção dos resultados, o autor realizou diversas combinações de conjuntos de *features* e os submeteu a vários classificadores. Para os dados de treinamento com validação cruzada de 10 vezes, o melhor resultado para a identificação de tipos de relações temporais *event-time* foi o classificador *NaiveBayes* com 69% de acurácia. Esse valor representou uma melhoria absoluta de 10,7% em relação ao *baseline* com *features* simples. Para os dados de testes o melhor classificador foi a Máquina de Vetor de Suporte com 66,9% de acurácia, representando uma melhoria absoluta de 5,9% em relação ao *baseline* com *features* simples.

A seguir, o conjunto de *features* que apresentou melhor resultado para os dados de teste na tarefa de identificação de tipos de relações temporais *event-time*:

- *event-tense*
- *event-temporal-direction*
- *event-between-order*
- *event-intervening-preceding-class*
- *event-intervening-following-tense*
- *event-closest-to-timex-pos*
- *event-closest-to-timex-equal-pos*
- *event-closest-to-event-equal-pos*
- *event-closest-to-event-equal-class*
- *timex3-type*
- *timex3-relevant-lemmas*
- *timex3-preposition-precede*

3.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais trabalhos relacionados a esta pesquisa, buscando relacionar as principais informações linguísticas que contribuíram para a identificação de tipos de relações temporais entre evento e expressão temporal na mesma frase. Foi visto que os sinais temporais – conjunções e advérbios temporais – foram mais úteis para as relações *event-event* do que para relações *event-time*, alcançando uma melhoria absoluta de 18% e de 2%, respectivamente. Foi visto também que o *framework* de Reichenbach oferece uma estrutura teórica para análise de tempo e aspecto e que pode ser usado para prever relações temporais. Todavia, sua aplicação resultou em um aumento absoluto modesto, apenas 1,4%.

Também foi apresentada uma abordagem híbrida, na qual foram combinadas regras manuais com aprendizado de máquina, utilizando diversas informações linguísticas sofisticadas. Algumas foram consideradas úteis, pois elas melhoram significativamente a acurácia, como as *features* em pares, relações de dependência e as relações de discurso. Outras não foram consideradas úteis, como as relações lexicais presentes no dicionário online *Webster* e *WordNet* e rotulador de funções semânticas. Com essas informações, capturou-se uma semântica mais rica para a tarefa e alcançou uma redução relativa de erro de aproximadamente 9% nas relações *event-time* do domínio de notícias.

Por fim, o trabalho apresentado para a língua portuguesa empregou diversas técnicas de combinação de informações linguísticas, tais como a codificação dos atributos de maneiras distintas, a fim de tornar seu uso mais vantajoso, e a utilização de informações relevantes de outros eventos ou expressões temporais presentes na mesma frase. Ademais, foram empregados recursos de semântica lexical, com a finalidade de prever a relação temporal apenas pelo significado do verbo envolvido, trazendo, assim, conhecimento prévio sobre o mundo para o trabalho. Como resultado, obteve-se uma melhoria absoluta de 5,9% em relação ao *baseline* que utilizou informações linguísticas mais simples, totalizando 66,9% de acurácia e *F1-score* de 62,5% para as relações *event-time* nos dados de testes.

Na Tabela 3.2, reunimos os trabalhos relacionados, destacando os tipos de *features*, algoritmo de aprendizado de máquina e método utilizado.

Tabela 3.2: Resumo por tipo de *features* dos trabalhos relacionados.

Trabalho	**Tipo de *Features***	**Algoritmo**	**Método**
(DERCZYNSKI, 2017)	Sinais Temporais; Tempo de Aspecto de Reichenbach; Morfossintáticas; Anotações TimeML	Máxima Entropia	Aprendizado de máquina
(D'SOUZA, 2015)	Morfossintáticas; Anotações TimeML; Contextuais	SVM	Híbrido
(MIRZA; TONELLI, 2014)	Morfossintáticas; Anotações TimeML; Contextuais	SVM	Aprendizado de máquina
(BETHARD; MARTIN, 2007)	Morfossintáticas; Anotações TimeML; Contextuais	SVM	Aprendizado de máquina
(COSTA, 2012)	Léxicas; Morfossintáticas; Anotações TimeML; Contextuais; Conhecimento prévio sobre o mundo	SVM	Aprendizado de máquina

Fonte: Elaborada pelo autor.

Capítulo

4

IDENTIFICAÇÃO DE TIPOS DE RELAÇÕES TEMPORAIS

A tarefa abordada neste trabalho foi definida com base em Verhagen et al. (2007), que trata da identificação de relações temporais entre um evento e uma expressão temporal (*event-time*) presentes na mesma frase. Para a identificação do tipo de relação temporal, optou-se por uma abordagem baseada em regras. Inicialmente, foi realizado um processo manual para compor um conjunto de regras, ao qual foram posteriormente adicionadas técnicas de aprendizagem de regras, conforme descrito na Seção 2.6.2. As regras utilizadas levaram em consideração diversas informações linguísticas, como análise de dependência, classe gramatical, tempo verbal, lema de palavras flexionadas e combinações de posições de palavras no texto, além de informações anotadas no *corpus* TimebankPT.

Neste capítulo, é apresentado o método para identificar os tipos de relação temporal entre eventos e expressões temporais presentes na mesma frase, baseado em regras. O método envolve a criação de um conjunto de *features* abrangente, contendo informações linguísticas relevantes, que é utilizado para construir conjuntos de regras usando algoritmos de aprendizagem de regras. Os conjuntos de regras são aplicados individualmente aos pares *event-time* no *corpus*, bem como combinados de dois em dois e em conjunto, a fim de avaliar se o desempenho é melhor do que os conjuntos individuais. A aplicação dos conjuntos de regras é feita de duas maneiras: pela primeira regra acionada e por meio de votação. Além disso, foi empregada uma estratégia de aumento de dados, conhecida como fechamento temporal, que infere novas relações com base nas relações identificadas.

Neste contexto, na Seção 4.1 deste capítulo, é apresentado um conjunto amplo e diversificado de *features*, composto por informações linguísticas, juntamente com a descrição da engenharia de *features* empregada em sua composição. Na Seção 4.2, são abordados os procedimentos para a geração de conjuntos de regras abrangentes e eficientes na identificação de diferentes tipos de relações temporais. A Seção 4.3 descreve as formas como os conjuntos de regras podem ser aplicados nos pares *event-time*. Por fim, na Seção 4.4, é apresentada uma estratégia de aumento de dados que calcula o fechamento temporal de todos os pares *event-time* identificados, juntamente com suas respectivas relações identificadas.

4.1 CONJUNTO DE *FEATURES*

A partir da premissa apresentada por Derczynski (2017), que enfatiza a necessidade de recorrer a múltiplas fontes de informação linguística para compreender a ordenação temporal em textos escritos em língua portuguesa, realizamos uma revisão bibliográfica. O objetivo dessa revisão foi identificar conjuntos de *features* propostas por diversos autores que sejam úteis para a identificação de diferentes tipos de relações temporais. Cada *feature* é composta por informações linguísticas extraídas de eventos e expressões temporais, bem como das palavras que as acompanham, seus regentes e dependentes sintáticos na sentença. Na Tabela 4.1, estão listados os tipos de informações linguísticas utilizadas. Com base nesse levantamento, foi possível compilar um conjunto de 70 *features* que foram empregadas neste trabalho. Para obter uma descrição completa de todas as *features* e suas respectivas explicações, consulte o Apêndice A.

Tabela 4.1: Quantitativo de *features* por tipo de informação linguística.

Tipo de Informações Linguísticas	**Quantidade**
Informações morfológicas	26
Informações sintáticas	12
Contextual	11
Sinais temporais	10
Anotação TimeML	7
Conhecimento prévio sobre o mundo	2
Tempos verbais de Reichenbach	1
Informações lexicais	1
Total de *features*	**70**

Fonte: Elaborada pelo autor.

As *features* são representações dos dados que alimentam os algoritmos de aprendizagem e capturam informações relevantes e significativas que auxiliam o modelo a compreender e generalizar padrões nos dados. A engenharia de *features* envolve diferentes técnicas, como a seleção de atributos relevantes, a transformação de variáveis existentes e redução de dimensionalidade, conforme estudado no trabalho de Zheng e Casari (2018). A seguir, apresentamos as técnicas utilizadas neste trabalho.

4.1.1 *Features* Baseadas em Palavras

Nesta pesquisa, optamos por não usar *features* baseadas em palavras, lemas e datas específicas, porque acreditamos que elas não contribuem para a generalização do conjunto de regras. Costa (2012) argumenta que o seu uso poderia resultar em problemas de escassez de dados porque o conjunto de seus valores possíveis seria muito grande e muitos valores seriam raramente representados nos dados, uma vez que o *corpus* é relativamente

pequeno. As *features* com esse perfil são: lema do evento (*event-lemma*), texto do evento (*event-text*), texto da expressão temporal (*timex3-text*), valor normalizado da expressão temporal (*timex3-value*) e valor normalizado da Data de Criação do Documento (*dct-value*).

4.1.2 Escala de Distância

As *features* que utilizam distâncias em número de *tokens* necessitam ser agrupadas para reduzir a quantidade de variáveis. Esta forma funciona melhor nos algoritmos de aprendizagem de regras. A seguinte escala foi empregada para distâncias em número de *tokens*: "`perto`" para distancias de até 4 *tokens*, "`distância média`" de 5 a 9, "`longe`" para distâncias entre 10 e 14 e "`muito longe`" para distâncias maiores que 14 *tokens*. As *features* com esse perfil que foram adaptadas são: *event-timex3-distance, signal-precede-event-distance-event* e *signal-precede-timex3-dep-if-child-timex3*.

(27) a. *event-timex3-distance* = 'perto' and *event-class* = 'aspectual' ⇒ OVERLAP

b. *Ele também disse que as guerras de preços que estão a despontar em algumas partes da indústria informática* ***continuarão*** *no* <u>*próximo ano*</u>*.*

Um exemplo ilustrativo é fornecido através da aplicação da regra (27a). Nessa regra, a *feature* denominada *event-timex3-distance* é utilizada com o objetivo de estabelecer a condição de proximidade entre o evento e a expressão temporal em análise, definindo que essa proximidade deve ser "perto", implicando que a distância entre eles deve ser limitada a um máximo de 4 *tokens*, conforme escala previamente estabelecida. Na sentença exemplificada em (27b), o par *event-time* ("continuarão", "próximo ano") satisfaz essa condição estipulada. Além disso, se a classe do evento for determinada como "aspectual", o tipo de relação temporal será classificado como OVERLAP.

4.1.3 Tamanho da Janela do Contexto para Coleta de Informações

Algumas *features* coletam informações linguísticas de *tokens* próximos ao evento ou à expressão temporal envolvidos na relação temporal. Não há registro de qual seria o tamanho da janela ideal para a coleta dessas informações. Por exemplo, Chambers et al. (2014) usaram uma janela de 2 *tokens* para coletar a classe gramatical em torno do evento ou da expressão temporal da relação sob classificação. Já D'Souza (2015) usou uma janela de 5 *tokens*. Neste trabalho definimos em 3 *tokens* esse tamanho. As *features* com esse perfil são: *event-pos-token-i-precede, event-pos-token-i-follow, timex3-pos-token-i-precede* e *timex3-pos-token-i-follow*, onde o *i* representa a posição do *token* em relação às entidades envolvidas.

(28) a. *timex3-pos-token-1-follow* = 'ADV' and *event-root* = False ⇒ OVERLAP

b. <u>*Poucos dias*</u> *depois, a American Medical anunciou lucros acentuadamente mais baixos,* ***assumindo*** *encargos de 24 milhões de dólares por reservas de seguros e contratos de leasing imobiliário cancelados.*

Com o propósito de exemplificar a obtenção de informações em um contexto próximo das entidades envolvidas em uma relação temporal, é apresentada a regra (28a), na qual é utilizada a *feature timex3-pos-token-1-follow*. Essa *feature* tem como finalidade capturar a classe gramatical do primeiro *token* subsequente à expressão temporal e verificar se esse *token* é um advérbio. A referida regra também realiza uma verificação adicional para confirmar que o evento associado à relação temporal não é a raiz da árvore de dependência sintática.

No exemplo da sentença (28b), o primeiro *token* após a expressão temporal "Poucos dias" é o advérbio "depois", e o evento relacionado à relação temporal sob análise, "assumindo", não desempenha o papel de raiz na árvore de dependência; essa função é desempenhada pelo evento "anunciou". Consequentemente, as condições estabelecidas pela regra são cumpridas, e o tipo de relação temporal entre esse par é identificado como OVERLAP. É importante ressaltar que, como a janela de contexto foi definida com um tamanho de 3 *tokens* nesta pesquisa, há duas outras *features* com a mesma finalidade, as quais coletam informações dos segundo e terceiro *tokens* subsequentes à expressão temporal.

4.1.4 Distância do Contexto para Coleta de Informações

Em determinadas circunstâncias, é necessário buscar informações até uma determinada distância das entidades envolvidas na classificação, por exemplo, buscar a primeira preposição que precede o evento. No entanto, é necessário estabelecer até que distância do evento essa busca deve ser feita. Costa (2012) coleta o texto da preposição que imediatamente precede a expressão temporal no texto, o que corresponde a uma distância de um *token*. Já Kolya et al. (2013) coletam o texto da primeira preposição que precede a entidade, ou seja, até o início da sentença, que representa a distância máxima. No presente trabalho, a busca será limitada a até 5 *tokens*. As *features* que sofrerão alterações em decorrência da definição desse limite são: *event-preposition-precede* e *timex3-preposition-precede*.

(29) a. *timex3-preposition-precede* = 'no' and *event-between-order* = True and *timex3-relevant-lemmas* = 'próximo' ⇒ BEFORE

b. *Portanto, os seus altos executivos estão a* ***falar*** *abertamente da possibilidade de* ***recomprar*** *alguns dos 172,5 milhões de dólares da empresa em obrigações subordinadas convertíveis no próximo ano.*

Com o intuito de ilustrar essa abordagem, é apresentada a regra (29a), na qual é empregada a *feature timex3-preposition-precede*. Essa *feature* tem como objetivo rastrear a preposição que antecede a expressão temporal em até 5 *tokens* e verificar se esse valor corresponde a 'no'. A referida regra também avalia a presença de outro evento entre o evento e a expressão temporal da relação sob análise, bem como verifica se o lema da expressão temporal contém a palavra 'próximo'.

No exemplo fornecido pela sentença (29b), o par *event-time* em análise é composto por ("falar", "próximo ano"). A preposição 'no' antecede a expressão temporal em até 5

tokens, há a presença de outro evento entre o par analisado, neste caso o evento "recomprar". Adicionalmente, a palavra 'próximo' está presente no lema da expressão temporal. Como resultado, a relação temporal estabelecida é identificada como BEFORE, o que indica que o evento "falar" ocorreu antes do momento temporal representado por "próximo ano".

4.1.5 Posições Fixas dos Sinais Temporais

As *features* relacionadas aos sinais temporais foram inicialmente criadas para associar cada sinal temporal a pares de entidades temporais, como evento e expressão temporal. Entretanto, elas também podem ser utilizadas na tarefa de identificar o tipo de relação temporal, conforme discutido na Seção 3.1.1. Para adaptá-las a este trabalho, foram selecionados apenas os sinais temporais mais próximos encontrados antes do evento ou expressão temporal, sem a existência de outra entidade entre eles. Em outras palavras, o algoritmo procura um sinal temporal que precede o evento ou expressão temporal envolvidos na relação sob classificação, com a condição de parar ao encontrar outra entidade temporal. Caso o sinal temporal não seja encontrado, o valor será nulo para todas as *features* relacionadas a esse sinal. É importante ressaltar que também foi testada a utilização dos sinais temporais encontrados após o evento ou expressão temporal, porém, eles são mais raros e não demonstraram ser úteis neste contexto. As *features* relacionadas aos sinais temporais utilizadas neste trabalho estão listadas na Seção A.1.8.

Para exemplificar esta abordagem, optamos por destacar uma dessas *features*, denominada *signal-precede-timex3-text*. Essa *feature* reflete o texto do sinal temporal que antecede a expressão temporal. Conforme a regra (30a), quando esse texto corresponde ao sinal temporal "desde", o tipo de relação temporal entre o par *event-time* é identificado como AFTER. Essa situação é exemplificada pela sentença (30b), na qual fica evidente que o evento "inflacionados" ocorreu posteriormente ao momento representado pela expressão temporal "1 de junho", devido à presença do sinal "desde" antecedendo essa expressão.

(30) a. *signal-precede-timex3-text* = 'desde' ⇒ AFTER

b. *Ao mesmo tempo, os custos de produção, quando comparados aos de há um ano, foram* ***inflacionados*** *pelos custos mais altos com matérias-primas e com os salários que resultaram do novo pacto social da empresa em vigor desde 1 de junho.*

No entanto, as *features* que capturam a posição do sinal temporal em relação às entidades temporais, conforme mencionadas na Seção 3.1.1, não se mostraram necessárias para o desenvolvimento deste trabalho. Tal fato decorreu da fixação das posições desses sinais temporais antecedendo as entidades. As *features* específicas que não foram necessárias são as seguintes: *signal-event-closest-follow, signal-event-closest-precede, signal-event-first-order, signal-timex3-closest-follow, signal-timex3-closest-precede* e *signal-timex3-first-order*.

4.1.6 Redução de Dimensionalidade

Em conjuntos de dados com muitas *features*, pode ser vantajoso reduzir a dimensionalidade usando técnicas de seleção baseadas em medidas de relevância de *features*. Uma dessas medidas é o índice Gini, que avalia a pureza de um nó da árvore ao separar as amostras em classes diferentes. Quanto maior a redução na impureza de Gini ao dividir um nó com base em uma *feature*, maior a importância atribuída a essa *feature*. Este índice pode ser aplicado à técnica de eliminação recursiva de *features* com validação cruzada (RFECV - do inglês, *Recursive Feature Elimination with Cross-Validation*).

A RFECV é uma técnica comumente usada para selecionar *features* relevantes em problemas de aprendizado de máquina. Ela começa treinando um modelo de aprendizado de máquina com todas as *features* disponíveis e, em seguida, elimina iterativamente as *features* menos importantes com base em sua importância relativa calculada pelo índice Gini. Após a primeira iteração, a RFECV remove a *feature* menos importante e reavalia o desempenho do modelo usando validação cruzada. Esse processo é repetido até que o desempenho do modelo não melhore significativamente.

A validação cruzada é essencial para avaliar o desempenho do modelo em diferentes partições do conjunto de dados, evitando o fenômeno de *overfitting* e aprimorando a confiabilidade da seleção de *features*, conforme constatado por Hastie, Tibshirani e Friedman (2010).

Resumidamente, a RFECV é uma técnica que usa validação cruzada para eliminar iterativamente as *features* menos importantes, com o objetivo de selecionar um conjunto otimizado de *features* para treinar um modelo de aprendizado de máquina. Neste caso específico, das 70 *features* originais, a RFECV selecionou um total de 52 como as mais relevantes. Essas *features* selecionadas estão disponíveis na Seção A.2.

O uso dessa técnica permitiu que alguns algoritmos de aprendizagem de regras utilizados neste trabalho, como o `CN2` e o `IDS`, tomassem decisões mais precisas com base nas *features* mais informativas. Além disso, possibilitou o uso do `CBA`, que requer quantidades elevadas de memória, com os recursos disponíveis. Mais detalhes sobre a utilização desses algoritmos com redução de dimensionalidade serão apresentados na Seção 4.2.

Por fim, todas as *features* disponíveis no Apêndice A foram implementadas em nosso ambiente de desenvolvimento e utilizadas para gerar *dataset* contendo as informações linguísticas correspondentes a cada par de entidades que compõem as relações temporais *event-time* presentes nos dados de treinamento do *corpus*. Esse *dataset* foi empregado para a geração de novas regras, conforme apresentado na seção a seguir.

4.2 CONJUNTOS DE REGRAS

Após a seleção do conjunto de *features* relevantes para a tarefa de identificação de tipos de relações temporais, procedemos à descrição da geração dos conjuntos de regras e à apresentação dos hiperparâmetros de cada algoritmo gerador. Além disso, discutimos as modificações aplicadas em alguns algoritmos para ampliar a abrangência dos conjuntos de regras.

Em seguida, fornecemos tanto o conjunto de regras manual desenvolvido como os

conjuntos de regras gerados pelos algoritmos de aprendizado de regras, nomeadamente `CBA`, `CN2`, `IDS` e `RIPPER`. Todos os conjuntos de regras para a língua portuguesa podem ser encontrados em nosso site[1], enquanto o conjunto de melhor desempenho encontra-se disponível no Apêndice B.

4.2.1 Conjunto de Regras Manuais

Para a elaboração do nosso conjunto de regras manuais, extraímos informações lexicais, classe gramatical, informações morfológicas, relação de dependência sintática entre as entidades temporais e seus regentes na sentença, combinações contextuais e atributos anotados presentes no corpus. Esses elementos são essenciais para compor as regras que identificam tipos de relações temporais.

As regras desenvolvidas por D'Souza (2015) para a língua inglesa foram utilizadas como inspiração para a criação das regras específicas para a língua portuguesa. De acordo com a autora, essas regras foram inicialmente formuladas com base na intuição e, posteriormente, refinadas manualmente por meio da observação dos dados de treinamento presentes no *corpus*.

A fim de ilustrar como essas informações contribuem para a definição do tipo de relação temporal entre um evento e uma expressão temporal, apresentamos os seguintes exemplos concretos.

(31) Os preços do petróleo desceram desde meados de outubro a partir de um nível de vinte e dois dólares por barril para os catorze dólares que **vemos** hoje.

Na sentença (31), destaca-se a relação temporal que ocorre entre o evento "*vemos*" e a expressão temporal "*meados de outubro*", sendo caracterizada como do tipo AFTER. Pode-se formular uma regra que capture essa relação da seguinte maneira: se a preposição "*desde*" preceder uma expressão temporal do tipo "`DATE`", correspondendo, nesse contexto, à expressão "*meados de outubro*", e se o evento da relação, representado pelo termo "*vemos*", estiver conjugado no tempo verbal presente, então é altamente provável que a relação temporal estabelecida seja do tipo AFTER. Dessa forma, o evento "*vemos*" ocorre posteriormente a "*meados de outubro*".

(32) As **iniciativas** anteriores de redução de pessoal já eliminaram cerca de 300 empregos, disse o porta-voz.

Na sentença (32), é evidenciada as relação temporal que ocorre entre o evento "*iniciativas*" e a expressão temporal "*anteriores*", a qual é classificada como sendo do tipo OVERLAP. Pode-se estabelecer uma regra que capture essa relação da seguinte forma: quando se manifesta uma dependência sintática de caráter modificativo adjetival (`amod`) entre o evento e a expressão temporal em análise – que, nesse contexto, é representado pelo adjetivo "*anteriores*" atuando como modificador do substantivo "*iniciativas*" – e quando a expressão temporal se qualifica como pertencente aos tipos "`DATE`" ou "`TIME`",

[1] <https://github.com/FORMAS/temporal_relation>

e, simultaneamente, o evento se situa nas classes "`OCCURRENCE`" ou "`STATE`" – sendo que, neste exemplo, "*iniciativas*" se insere na classe "`OCCURRENCE`" –, além disso, se o evento não se caracterizar como um verbo, então é plausível que a relação temporal seja OVERLAP. Assim, podemos inferir que o evento "*iniciativas*" ocorreu durante o intervalo temporal correspondente à expressão "*anteriores*", a qual faz referência ao passado.

(33) A companhia disse que **pretende** aumentar a receita não GM para pelo menos 50% do seu volume total de negócios até ao fim de 1990.

Na sentença (33), uma das relações temporais presentes é entre o evento "*pretende*" e a expressão temporal "*fim de 1990*", sendo classificada como do tipo BEFORE. É possível estabelecer uma regra que capture essa relação da seguinte maneira: quando o tipo da expressão temporal for "`DATE`" e esta for antecedida pela preposição "*até*", e quando a classe gramatical do termo que rege o evento for um verbo, então é provável que o tipo da relação seja BEFORE. Desta forma, o evento "*pretende*" ocorreu antes do período definido como "*fim de 1990*".

O conjunto completo de regras proposto por D'Souza (2015) consiste em um total de 514 regras, abrangendo todas as tarefas do TempEval (VERHAGEN et al., 2007): relações *event-time, event-DCT* e *event-event*. A distribuição dessas regras de acordo com os tipos de informações linguísticas é apresentada na Tabela 4.2, conforme descrito na Seção 3.1.3.

Tabela 4.2: Quantidade de regras elaboradas por D'Souza (2015) por tipos de informações linguísticas.

Tipo de Informação Linguística	**Quantidade**
Informações lexicais, morfológicas e contextuais	205
Relações de discurso	190
Dependência sintática	47
Relação com a DCT	30
Rotulador de funções semânticas	24
WordNet (MILLER, 1995)	10
Merriam-Webster dictionary	8

Fonte: Elaborada pelo autor a partir de dados fornecidos por D'Souza (2015)

Dentre as regras aplicáveis às relações *event-time*, e que consideram as informações linguísticas utilizadas neste trabalho - dependência sintática, informações lexicais, morfológicas e contextuais - foram identificadas um total de 43 regras para a língua inglesa. Essas regras foram adaptadas para a língua portuguesa, resultando em um conjunto de 41 regras específicas.

4.2.2 Conjunto de Regras Gerado pelo Algoritmo CBA

Para a construção do conjunto de regras gerado pelo algoritmo `CBA` (LIU; HSU; MA, 1998), utilizamos como entrada o *dataset* construído na Seção 4.1. A implementação desse algoritmo permite a configuração de diversos hiperparâmetros que afetam o processo de geração de regras. Os hiperparâmetros considerados neste trabalho são: a confiança mínima a partir da qual a mineração de regras é iniciada (`init_conf`), o suporte mínimo a partir da qual a mineração de regras é iniciada (`init_support`), o comprimento mínimo das regras a serem extraídas (`minlen`), o comprimento máximo das regras a partir do qual a mineração é iniciada (`init_maxlen`) e o número máximo de iterações a serem realizadas antes de interromper a execução (`max_iterations`). De acordo com a documentação da implementação deste algoritmo, utilizada neste trabalho, os valores padrão para esses hiperparâmetros são os seguintes: `init_conf = 0,5`, `minlen = 2`, `init_maxlen = 3` e `max_iterations = 30`. Esses valores foram ajustados e a melhor configuração de hiperparâmetros utilizada para a geração das regras será apresentada na Seção 6.1.2. Por fim, ao término da geração das regras, é adicionada uma nova regra que atribui a classe majoritária, no caso a classe Overlap, como a classe padrão. Isso significa que as poucas instâncias que permanecerem sem classificação serão atribuídas a essa classe.

É importante destacar que o algoritmo `CBA` apresenta uma demanda significativa de recursos computacionais, especialmente em relação à memória. Isso ocorre devido ao crescimento exponencial do número de regras de associação de classe (CARs - do inglês, *Class Association Rules*) geradas à medida que o número de *features* aumenta.

Um exemplo ilustrativo dessa necessidade de recursos é observado na etapa de geração de todas as CARs, conforme apresentado na Seção 2.6.2.1. Nessa etapa, utilizando apenas 41 das 70 *features* existentes, foram geradas aproximadamente 19 milhões de CARs. É importante ressaltar que esse processo foi realizado em um computador com 32 GB de memória RAM, sendo essa a quantidade máxima de *features* viável para o referido algoritmo. As 41 *features* mais relevantes, selecionadas conforme técnica de eliminação recursiva de features com validação cruzada (RFECV), estão listadas nas primeiras posições da lista de *features* apresentada na Seção A.2.

4.2.3 Conjunto de Regras Gerado pelo Algoritmo CN2

O algoritmo `CN2` (CLARK; NIBLETT, 1989) possui diversos hiperparâmetros que podem ser configurados para ajustar o seu comportamento durante a geração das regras. No que se refere à ordenação das regras, podemos escolher entre dois modos: "`Ordered`" e "`Unordered`". No modo "`Ordered`" (padrão), as regras são induzidas seguindo uma lista de decisões, em que as condições da regra são encontradas e a classe majoritária é atribuída como a classe predita da regra. No modo "`Unordered`", são aprendidas regras para cada classe individualmente, em relação aos dados de aprendizado originais.

Outro hiperparâmetro importante se refere ao algoritmo de cobertura, que pode ser configurado como "`Exclusive`" ou "`Weighted`". No modo "`Exclusive`" (padrão), após cobrir uma instância de aprendizado, ela é removida das considerações futuras. Já no modo "`Weighted`", após cobrir uma instância de aprendizado, seu peso é diminuído (multiplicação por um fator gama configurável) e, consequentemente, seu impacto nas iterações

subsequentes do algoritmo é reduzido.

Além disso, o algoritmo `CN2` possui outros hiperparâmetros, como a medida de avaliação de hipóteses encontradas, que pode ser selecionada entre "`Entropy`" (padrão), "`Laplace Accuracy`" e "`Weighted Relative Accuracy`". Também podemos definir a largura do feixe de busca (`beam width`), que controla o número de alternativas monitoradas em cada iteração (padrão = 5). Há também os hiperparâmetros de filtragem de regras, como a cobertura mínima de regras (padrão = 1), que determina o número mínimo de instâncias cobertas por uma regra, e o comprimento máximo de regras (padrão = 5), que estabelece o número máximo permitido de condições em uma regra. Além disso, existem os hiperparâmetros de teste de significância para podar regras menos frequentemente aplicáveis com base na distribuição inicial de classes (`default alpha`) ou na distribuição de classe dos pais (`parent alpha`). Esses valores foram ajustados e a melhor configuração de hiperparâmetros utilizada para a geração das regras será apresentada na Seção 6.1.3. Por fim, ao final da geração das regras, é atribuído a classe majoritária como a classe padrão.

Destacamos que o conjunto de regras gerado pelo algoritmo `CN2` foi o único a apresentar operadores de desigualdade em suas condições. A inclusão desse operador traz consigo vantagens e desvantagens em relação a conjuntos de regras que se baseiam exclusivamente no operador de igualdade. Entre as vantagens, destaca-se a maior expressividade das regras, permitindo a definição de condições mais flexíveis e a captura de relações mais complexas entre as *features*. No entanto, é importante ressaltar que a interpretabilidade das regras pode ser comprometida, o que dificulta a compreensão dos padrões aprendidos e pode resultar em uma menor confiança nas decisões tomadas com base nessas regras.

4.2.4 Conjunto de Regras Gerado pelo Algoritmo IDS

O algoritmo `IDS` (LAKKARAJU; BACH; LESKOVEC, 2016) é composto por três fases principais. Na primeira fase, denominada "geração de conjuntos de itens frequentes", são extraídos os conjuntos de itens frequentes que atendem a um determinado limiar de suporte. Essa etapa utiliza o algoritmo Apriori para a mineração de conjuntos de itens frequentes.

Na segunda fase, chamada "geração de regras", para cada conjunto de itens frequentes de entrada, o `IDS` gera um conjunto de regras por classe. A geração é realizada por um algoritmo iterativo que é aplicado para ajustar iterativamente as restrições de confiança, suporte e comprimento máximo das regras. Esse processo continua até que seja gerado um número específico de regras definido pelo hiperparâmetro "`rule_cutoff`".

A terceira fase, conhecida como "seleção de regras", consiste em selecionar um subconjunto das regras geradas na etapa anterior para compor o conjunto de solução. Esse processo ocorre de forma iterativa, adicionando ou removendo regras com base na avaliação de uma função objetivo que deve ser maximizada. O `IDS` utiliza duas variantes do algoritmo de otimização para seleção de regras: *Deterministic Local Search* (DLS) e *Smooth Local Search* (SLS) que podem ser selecionados através do hiperparâmetro "`algorithm`". O DLS não possui garantia específica sobre a distância do ótimo, enquanto o SLS é garantido para encontrar uma solução com pelo menos 2/5 do valor ótimo, se-

gundo Feige, Mirrokni e Vondrák (2011). A escolha entre esses algoritmos depende do *trade-off* desejado entre velocidade e qualidade da solução.

A função objetivo que o algoritmo `IDS` busca maximizar consiste na combinação linear de sete objetivos, em que cada objetivo é ponderado por um fator de peso *lambda*. Os objetivos gerais a serem alcançados por essa função objetivo, visando a obtenção do conjunto final de regras, incluem a minimização do número de regras, a redução do número de condições em cada regra, a abrangência adequada do espaço de decisão e a minimização da sobreposição entre as regras.

O vetor de pesos *lambda* pode ser definido através do hiperparâmetro "`lambda_array`", que por sua vez, pode receber o valor um para cada peso do vetor, ou pode ser calculado através do algoritmo de otimização de subida de coordenadas (*Coordinate Ascent*). O algoritmo de otimização de subida de coordenadas é uma alternativa à descida do gradiente, no qual, em vez de minimizar, estamos maximizando o escore AUC (Área sob a Curva ROC) do modelo. Esse algoritmo busca encontrar a melhor combinação de pesos *lambda* que maximiza a métrica de desempenho AUC.

Em decorrência de nossos experimentos, constatamos que o algoritmo `IDS` é capaz de gerar regras de alta qualidade. No entanto, observamos que a taxa de cobertura do conjunto de dados foi bastante baixa, alcançando apenas cerca de 25%. A taxa de cobertura é definida como a quantidade de instâncias que foi aplicada alguma regra, dividido pela quantidade total de instâncias. Com o intuito de superar essa limitação, propomos e implementamos modificações no algoritmo que se mostraram eficientes.

Nossa proposta consiste em realizar iterações de treinamentos apenas nos dados não classificados, isto é, aqueles que não foram preditos por nenhuma regra. Em cada iteração, as regras geradas são acumuladas com as regras obtidas na iteração anterior, sendo que as novas regras são adicionadas após as já existentes. Adicionalmente, o hiperparâmetro "`lambda_array`" é calculado na primeira iteração de treinamento por meio do algoritmo de otimização "*Coordinate Ascent*", pois apresenta melhores resultados que os valores padrão. O valor encontrado é então utilizado nas demais iterações. Por fim, as regras duplicadas são removidas e é atribuída a classe majoritária como a classe padrão.

Os resultados dessas modificações são apresentados na Seção 6.1.4, fornecendo uma análise detalhada do desempenho do `IDS` após a implementação das alterações propostas.

4.2.5 Conjunto de Regras Gerado pelo Algoritmo RIPPER

O algoritmo `RIPPER` (COHEN, 1995) possui diversos hiperparâmetros que podem ser ajustados para otimizar seu desempenho. O hiperparâmetro "`k`" indica o número de iterações de otimização `RIPPERk` a serem executadas (padrão = 2). O hiperparâmetro "`prune_size`" determina a proporção do conjunto de treinamento a ser utilizada para poda durante o processo de construção das regras (padrão = 0,33).

Outro hiperparâmetro relevante é o "`dl_allowance`", que define um limite para a fase de crescimento do conjunto de regras. Caso seja encontrada uma descrição do conjunto de regras cujo comprimento seja superior ao menor comprimento encontrado até então somado ao valor de "`dl_allowance`", a fase de crescimento é interrompida precocemente (padrão = 64).

O `RIPPER` também oferece a opção de discretização automática de atributos numéricos. O hiperparâmetro "`n_discretize_bins`" determina o número máximo de intervalos discretos nos quais esses atributos serão ajustados, incluindo o limite superior do intervalo (padrão = 10), mas é possível desabilitar a discretização automática passando o valor "*None*".

Além desses hiperparâmetros, existem limites para interromper o treinamento antecipadamente. Esses limites têm como objetivo aprimorar a interpretabilidade do modelo e limitar o tempo de treinamento em conjuntos de dados ruidosos. No entanto, eles não são especificamente destinados a serem ajustados como hiperparâmetros, pois a poda já ocorre durante o treinamento. Ainda assim, é possível que ajustes nesses limites possam melhorar o desempenho do modelo. Os limites incluem o número máximo de regras (`max_rules`), o número máximo de condições por regra (`max_rule_conds`) e o número máximo de condições totais em todo o conjunto de regras (`max_total_conds`).

Assim como no algoritmo `IDS`, o `RIPPER` também é capaz de gerar regras de alta qualidade. No entanto, a cobertura do conjunto de dados também é baixa, alcançando menos de 50%. Além disso, na implementação[2] utilizada neste estudo, o algoritmo `RIPPER` resolve apenas problema de classificação binária, limitando-se a atribuir instâncias de dados a uma de duas classes possíveis. Essa abordagem é adequada quando o problema em questão envolve a divisão de instâncias em apenas duas categorias distintas, como sim/não, positivo/negativo, presente/ausente, entre outras combinações binárias possíveis. No entanto, é importante ressaltar que estamos resolvendo um problema de classificação multiclasse com 6 classes distintas. Para fins didáticos, dividimos essas classes que dois grupos, o primeiro composto pelas três classes mais frequente (Before, After e Overlap) e o segundo grupo composto pelas três classes menos frequente (Before-or-Overlap, Overlap-or-After e Vague). Dito isso, temos duas limitações a serem superadas: uma relacionada à baixa cobertura de dados e outra relacionada à classificação binária do algoritmo.

Em relação ao problema da classificação binária, no âmbito deste estudo, seguindo a abordagem descrita por Murphy (2012, p. 503), utilizamos a estratégia conhecida como "Um Contra Todo" (OvA - do inglês, *One-vs-All*) para aplicar algoritmos de classificação binária a problemas de classificação multiclasse. Nessa abordagem, treinamos um classificador binário para cada classe, em que os dados da classe em questão são considerados positivos, enquanto os dados de todas as outras classes são considerados negativos. Por exemplo, ao treinar o classificador para a classe Before, todas as demais classes seriam tratadas como "Não-Before". As regras resultantes do treinamento de cada classe são adicionadas ao conjunto de regras em ordem crescente de frequência da classe nos dados. Para ilustrar, no grupo das classes mais frequentes, as regras para a classe Before são incluídas no início, seguidas pelas regras para a classe After e, por último, as regras para a classe Overlap, que é a classe majoritária. Com essa adaptação, tornamos o algoritmo mais adequado para realizar classificação multiclasse, proporcionando uma abordagem mais ajustada ao cenário específico do nosso estudo.

Em relação ao problema de baixa cobertura, adotamos uma solução semelhante à

[2] <https://github.com/imoscovitz/wittgenstein>

adotada no algoritmo IDS, mas com uma diferença. Para obtermos regras de melhor qualidade, as iterações de treinamento com apenas os dados não classificados foram realizadas em duas etapas, resultando em dois conjuntos de regras: um conjunto que classifica apenas o grupo de classes mais frequentes e outro que classifica todas as seis classes. Para formar o primeiro conjunto, treinamos apenas o grupo de classes mais frequentes, assim, esse conjunto de regras classifica apenas essas classes. Em seguida, para formar o segundo conjunto, repetimos as iterações de treinamento, adicionando o grupo de classes menos frequentes após o grupo de classes mais frequentes, assim, esse segundo conjunto de regras classifica todas as seis classes. Posteriormente, ambos os conjuntos foram mesclados em um único conjunto de regras.

Para mesclar os dois conjuntos de regras, realizamos os seguintes passos. Primeiro, removemos do segundo conjunto todas as regras pertencentes ao grupo de classes mais frequentes, deixando apenas as regras do grupo menos frequente. Em seguida, adicionamos esse segundo conjunto, contendo apenas as regras do grupo de classes menos frequentes, ao final do primeiro conjunto, que originalmente possuía apenas as regras do grupo de classes mais frequentes, formando assim um único conjunto de regras. Após a mesclagem, removemos as regras duplicadas e atribuímos a classe majoritária como a classe padrão.

Vale ressaltar que em cada iteração de treinamento com o algoritmo RIPPER, utilizando apenas os dados não classificados (ou seja, aqueles que não foram previstos por nenhuma regra), acumulamos as regras geradas com as regras obtidas na iteração anterior, adicionando as novas regras após as já existentes. Esse processo é realizado de forma semelhante às iterações que implementamos no algoritmo IDS.

Os resultados dessas modificações são apresentados na Seção 6.1.5, fornecendo uma análise detalhada do desempenho do RIPPER após a implementação das alterações propostas.

4.2.6 Conjuntos de Regras Combinados

A hipótese deste trabalho afirma que a combinação de conjuntos de regras induzidas por algoritmos diferentes produz resultados superiores em comparação com os resultados individuais de cada conjunto de regras utilizado para identificar tipos de relações temporais em língua portuguesa. O objetivo dos conjuntos de regras combinados é verificar essa hipótese.

Foram gerados dois conjuntos de regras para realizar essa avaliação. O primeiro conjunto é composto por todos os conjuntos de regras individuais mencionados nas subseções anteriores: as regras manuais e as regras geradas pelos algoritmos CBA, CN2, IDS e RIPPER. O segundo conjunto de regras é formado pela melhor combinação de dois dos conjuntos individuais. Os detalhes sobre a composição desses dois conjuntos são apresentados na Seção 5.4.1.2.

Após a construção dos conjuntos de regras, procedemos à aplicação dessas regras em sentenças em português, com o objetivo de identificar os diferentes tipos de relações temporais, como descrito na Seção 4.3. Em seguida, realizamos o cálculo do fechamento temporal para todos os pares *event-time* da sentença e suas respectivas relações identifica-

das. Nessa etapa de pós-processamento, são derivadas novas relações que anteriormente não eram explícitas, conforme explanado na Seção 4.4. A Figura 4.1 ilustra o fluxo do método, e cada uma dessas etapas será detalhada a seguir.

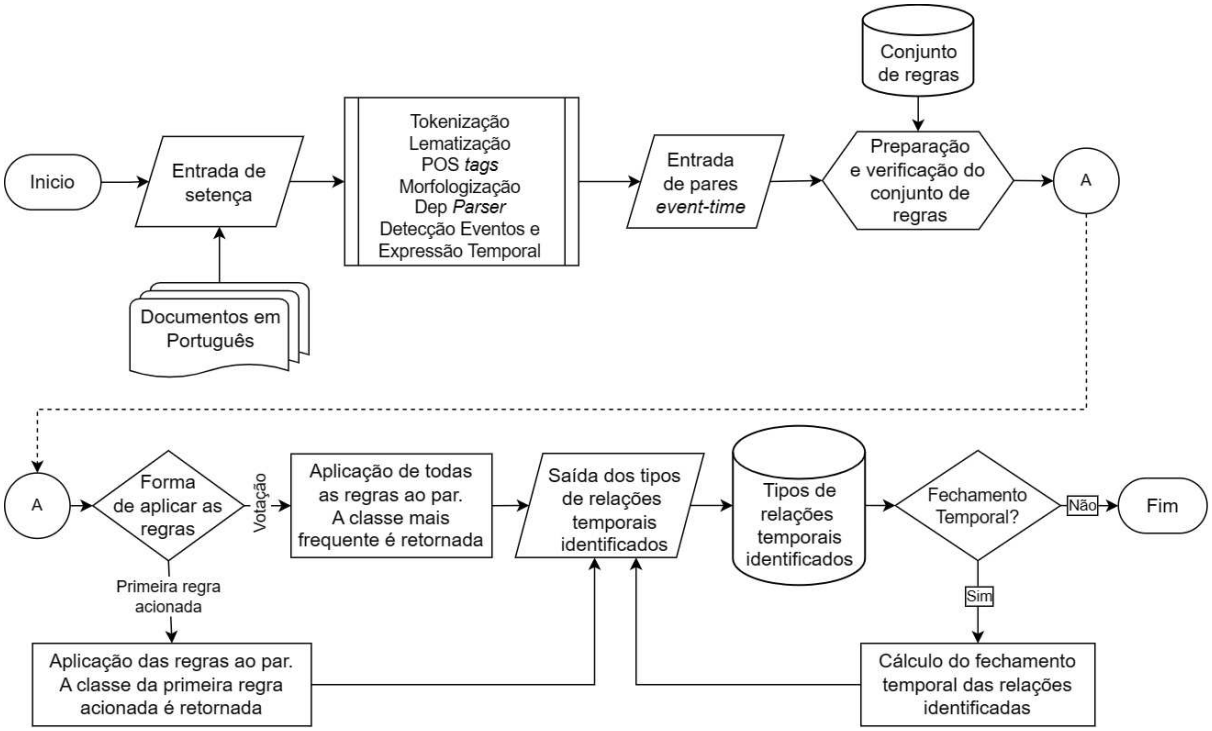


Figura 4.1: Fluxograma do processo de identificação do tipo de relação temporal.
Fonte: Elaborada pelo autor.

4.3 APLICAÇÃO DOS CONJUNTOS DE REGRAS

Nosso método recebe como entrada pares formados por evento/expressão temporal, sempre nessa ordem. Todos os pares *event-time* são submetidos à aplicação de um conjunto de regras, e a regra acionada retorna o tipo da relação temporal do par. Vale destacar que a quantidade de pares *event-time* em uma sentença é o produto da quantidade de eventos com a quantidade de expressões temporais, e todos eles são classificados. No entanto, é importante ressaltar que apenas os pares rotulados, ou seja, aqueles que possuem o tipo da relação temporal anotado no *corpus*, são avaliados. É importante mencionar que, conforme discutido na Seção 2.6.2, é possível que mais de uma regra cubra uma mesma instância de dados. Essa sobreposição de regras pode levar a uma indefinição na atribuição da classe, especialmente quando diferentes regras têm consequências diferentes.

Para resolver esse problema, a aplicação dos conjuntos de regras será realizada de duas formas diferentes. Na primeira forma, denominada "**primeira regra acionada**", a classe correspondente à primeira regra acionada é considerada a classe final para o par *event-time*. Na segunda forma, chamada de "**votação**", atribuímos votos para cada classe de acordo com as regras acionadas, e a classe mais frequente é atribuída como resultado.

A seguir, detalhamos as duas formas de processamento do conjunto de regras.

4.3.1 Primeira Regra Acionada

Utilizando essa forma de aplicar os conjuntos de regras, cada nova instância, que consiste em um par *event-time*, é classificada usando a primeira regra acionada no conjunto de regras. Após a classificação, a instância não é mais submetida às demais regras e o processamento avança para a próxima instância a ser classificada.

4.3.2 Votação

Nesta forma de aplicar as regras, todos os pares *event-time* são submetidos a todas as regras do conjunto de regras. Atribuímos votos para cada classe da regra acionada em cada par processado. A classe mais frequente é atribuída ao par *event-time*.

Encontramos na literatura um processo de votação semelhante realizado por Kolyal, Ekbal e Bandyopadhyay (2013). Os autores desenvolveram vários classificadores para identificar tipos de relações temporais. Eles utilizam técnicas de votação por maioria ou votação ponderada pelo *F1-score* para combinar os resultados dos classificadores.

4.4 FECHAMENTO TEMPORAL

O fechamento temporal é uma estratégia de aumento de dados que expande as relações temporais identificadas através do cálculo de raciocínio temporais em cada sentença. Essa estratégia é aplicada após a execução do conjunto de regras, quando todos os pares *event-time* já tiveram o tipo de sua relação temporal identificado. De acordo com Verhagen (2005), o fechamento temporal é um recurso que extrai novas relações temporais implícitas a partir das relações temporais conhecidas. Por exemplo, suponha que nosso sistema identifique as seguintes relações temporais: A ocorre antes de B e B ocorre antes de C. Com base nisso, podemos inferir que A ocorre antes de C, tornando essa relação explícita e considerada na avaliação.

Segundo Costa e Branco (2012), a precedência temporal (BEFORE) é transitiva, não reflexiva e assimétrica, enquanto a sobreposição temporal (OVERLAP) é reflexiva e simétrica. Apresentaremos a seguir as regras implementadas levando em consideração essas propriedades. Utilizaremos as letras A, B e C para representar um evento ou uma expressão temporal, e as setas '$\rightarrow$', '$\leftarrow$' e '$\leftrightarrow$' para representar os tipos de relação 'BEFORE', 'AFTER' e 'OVERLAP', respectivamente. Portanto, $A \rightarrow B$, $A \leftarrow B$ e $A \leftrightarrow B$ representam as relações de 'A *Before* B', 'A *After* B' e 'A *Overlap* B', respectivamente.

- Se $A \leftarrow B$, então $B \rightarrow A$
- Se $A \leftrightarrow B$, então $B \leftrightarrow A$
- Se $A \rightarrow B$ e $B \rightarrow C$, então $A \rightarrow C$
- Se $A \rightarrow B$ e $B \leftrightarrow C$, então $A \rightarrow C$
- Se $A \rightarrow B$ e $A \leftrightarrow C$, então $C \rightarrow B$
- Se $A \leftrightarrow B$ e $B \leftrightarrow C$, então $A \leftrightarrow C$

- Se $A \rightarrow B$ e $C \leftrightarrow B$, então $A \rightarrow C$
- Se $A \rightarrow B$ e $C \leftrightarrow A$, então $C \rightarrow B$
- Se $B \leftrightarrow A$ e $B \leftrightarrow C$, então $A \leftrightarrow C$
- Se $A \leftrightarrow B$ e $C \leftrightarrow B$, então $A \leftrightarrow C$
- Se $B \leftrightarrow A$ e $C \leftrightarrow B$, então $A \leftrightarrow C$

Portanto, a adição do fechamento temporal resulta em novas anotações úteis para a aplicação, enriquecendo o conjunto de relações previstas.

Todavia, é importante ressaltar que a aplicação do fechamento temporal nas relações anotadas no *corpus* não foi necessária, uma vez que, de acordo com Costa e Branco (2012), o fechamento temporal foi garantido durante a construção do TimebankPT. Essa constatação foi confirmada quando aplicamos nosso algoritmo nos dados do *corpus* e nenhuma nova relação foi gerada. Dessa forma, o fechamento temporal foi aplicado apenas aos pares *event-time* identificados pelos nossos conjuntos de regras.

4.5 CONSIDERAÇÕES FINAIS

Neste capítulo, é apresentado um método formal baseado em regras para identificar tipos de relações temporais entre eventos e expressões temporais na mesma frase. Esse método abrange estratégias para lidar com sobreposição de regras, que pode levar a uma indefinição da classe atribuída quando as regras sobrepostas geram tipos de relações diferentes para a mesma instância, e estratégias para ampliar, por meio do fechamento temporal, os pares *event-time* identificados.

Inicialmente, foi composto um conjunto de regras manuais. Em seguida, foi realizado um levantamento abrangente de informações linguísticas relevantes, resultando em um conjunto de *features* utilizado como entrada para algoritmos de aprendizagem de regras, tais como `CBA`, `CN2`, `IDS` e `RIPPER`, com o objetivo de gerar conjuntos de regras adicionais para o método.

Tanto o algoritmo `IDS` quanto o `RIPPER` apresentaram uma baixa cobertura de dados. Para contornar essa limitação, foram realizadas modificações nesses algoritmos, como a iteração de treinamentos exclusivamente nos dados não classificados. No caso específico do `RIPPER`, havia também a restrição de resolver apenas problemas de classificação binária. No entanto, uma abordagem denominada "Um Contra Todos" (OvA - do inglês, "*One-vs-All*") (MURPHY, 2012, p. 503) foi adotada para superar essa restrição.

Em resumo, foi apresentado um método abrangente para a identificação de tipos de relações temporais, que oferece um amplo conjunto de informações linguísticas. Além disso, os conjuntos de regras gerados pelos algoritmos mencionados, proporcionaram abordagens eficazes para a tarefa proposta.

Capítulo 5

AVALIAÇÃO EXPERIMENTAL

Neste capítulo, descreveremos a configuração experimental adotada em nosso estudo, que fez uso do *corpus* TimeBankPT. Apresentaremos as métricas de avaliação utilizadas para mensurar o desempenho do nosso método em relação ao *baseline* estabelecido. Além disso, forneceremos uma descrição detalhada dos experimentos conduzidos, abrangendo os procedimentos e abordagens adotadas.

5.1 CONJUNTO DE DADOS

Para conduzir a seleção dos parâmetros experimentais, procedemos à divisão dos documentos de treinamento do *corpus* TimeBankPT em duas partes distintas. A fração de 90% dos documentos foi destinada ao desenvolvimento das regras, enquanto a porção restante foi reservada para fins de validação. A representação gráfica da distribuição das classes pode ser visualizada na Figura 5.1. A partir dos resultados alcançados nos experimentos com os dados de validação, efetuamos a seleção das melhores configurações.

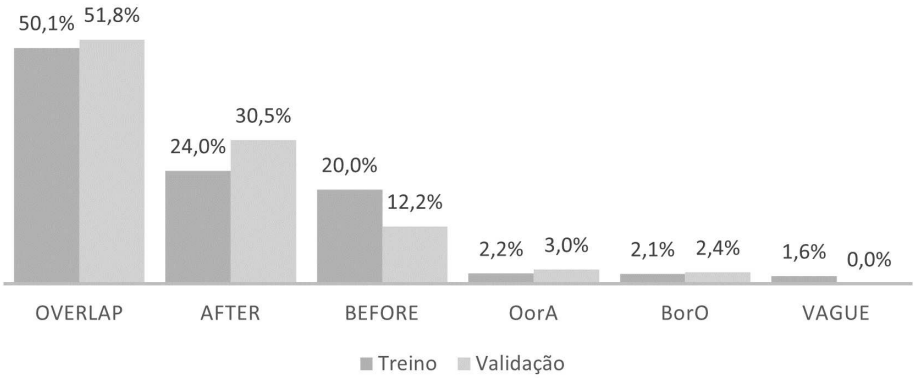


Figura 5.1: Distribuição das classes do *corpus* nos dados de validação.
Fonte: Elaborada pelo autor.

Durante o processo de divisão, consideramos a proporção de fontes presentes nos artigos que compõem o *corpus*, como *Associated Press Writer*, *CNN*, *The New York Times* e *Wall Street Journal*. Além disso, garantimos que a divisão ocorresse no nível do documento, de modo que todas as sentenças de um mesmo documento permanecessem no mesmo conjunto. Assim, levamos em conta esses aspectos ao dividir os dados, visando preservar a representatividade das fontes e a coesão dos documentos.

Para avaliar o desempenho final do nosso método, fizemos uso dos conjuntos de treinamento e teste do *corpus* TimeBankPT, que foram previamente particionados. Os dados de treinamento estendido compreendem 89% dos documentos do *corpus* e serão utilizados para treinar novamente os modelos selecionados com as melhores configurações. Por outro lado, os dados de teste correspondem a 11% dos documentos e serão empregados para a avaliação final do desempenho do nosso método. Dessa forma, poderemos verificar a eficácia e a generalização do método ao aplicá-lo em um conjunto de dados ainda não visto. A representação gráfica da distribuição das classes das relações *event-time* pode ser visualizada na Figura 5.2

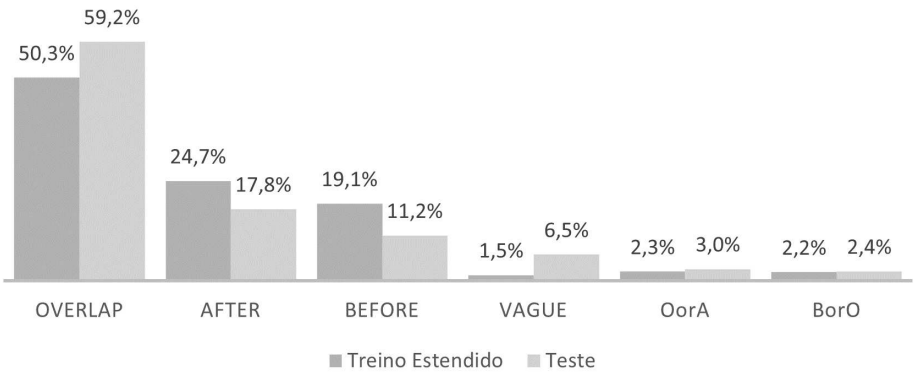


Figura 5.2: Distribuição de classes de todo o *corpus* com base nas relações *event-time*.
Fonte: Elaborada pelo autor.

No contexto do TempEval, foi estabelecido que a identificação dos tipos de relações temporais deveria ocorrer em todos os pares identificados na mesma frase. No entanto, estudos anteriores, como o realizado por Chambers (2013), relataram que os dados de treinamento e teste não seguiram essa definição devido à exclusão de muitos pares pelos anotadores, visando reduzir o esforço humano. Além disso, Chambers et al. (2014) observaram que apenas um subconjunto pequeno de pares recebeu rótulos, pois os anotadores consideraram apenas as relações mais centrais e óbvias, deixando o restante sem rótulos. A dificuldade em fornecer uma anotação completa também foi mencionada por Verhagen (2005), que apontou a impraticabilidade devido à complexidade, baixa velocidade de anotação e baixo acordo entre os anotadores.

Ao analisarmos o *corpus* TimebankPT, constatamos que apenas 45% das relações temporais *event-time* foram anotadas com seus respectivos tipos. Portanto, 55% dessas relações não possuem tal anotação. Neste trabalho, iremos avaliar apenas os dados ro-

tulados do *corpus*, seguindo a abordagem adotada por outros estudos que utilizaram o mesmo *corpus*.

5.2 MÉTRICA DE AVALIAÇÃO

Utilizamos a acurácia como métrica de avaliação para medir o desempenho do nosso método. Essa medida representa a porcentagem de instâncias classificadas corretamente em relação ao total de vezes que a regra é aplicada. É importante destacar que a acurácia é a métrica de avaliação padrão para a classificação de relações temporais no âmbito do TempEval, conforme ressaltado por D'Souza (2015) e Costa (2012) em suas respectivas teses. Além disso, o uso da acurácia nos permite comparar nossos resultados com os de outros estudos. No entanto, no Capítulo 6, também apresentaremos outras medidas, como o *F1-score*, e a matriz de confusão.

5.3 *BASELINE*

Utilizamos como *baseline* o *LX-TimeAnalyzer* proposto por Costa (2012). Esse trabalho representa o primeiro estudo publicado para a língua portuguesa que aborda a identificação de tipos de relações temporais. Até o momento, não encontramos outros trabalhos voltados para o domínio de notícias nesse idioma. Ao analisar a tarefa de identificação do tipo de relação temporal *event-time*, os resultados mostrados na Tabela 5.1 indicam que o classificador de melhor desempenho – *NaiveBayes* – obteve uma acurácia de 69% e *F1-score* de 66,1% nos dados de treinamento. Já nos dados de teste, a máquina vetorial de suporte alcançou uma acurácia de 66,9% e *F1-socre* de 62,5%, destacando-se como o classificador de maior desempenho.

Tabela 5.1: Resultado do *baseline* utilizado por esse trabalho.

	Acurácia	**F1**	**Melhor Classificador**
Treino	69,0	66,1	NaiveBayes
Teste	66.9	62,5	SVM

Fonte: Elaborada pelo autor.

5.4 DESCRIÇÃO DOS EXPERIMENTOS

Nessa seção são apresentados os experimentos realizados para validar o método proposto. O objetivo desses experimentos é determinar o conjunto de regras com melhor desempenho entre sete conjuntos distintos. Esses conjuntos consistem em cinco conjuntos individuais e dois conjuntos que são formados pela combinação dos conjuntos individuais. Entre os conjuntos individuais, temos um conjunto composto por regras manuais e outros quatro conjuntos gerados por algoritmos de aprendizado de regras, nomeadamente `CBA`, `CN2`, `IDS` e `RIPPER`. A seguir, apresentamos os procedimentos detalhados utilizados para obter esses

conjuntos de regras e forneceremos informações sobre como eles serão aplicados nos dados do *corpus*.

5.4.1 Seleção de Parâmetros Experimentais

Apresentamos os procedimentos empregados na obtenção das melhores configurações de parâmetros experimentais, que foram utilizados para gerar os sete conjuntos de regras propostos. A Figura 5.3 fornece uma visão geral da metodologia adotada, a qual é detalhada a seguir.

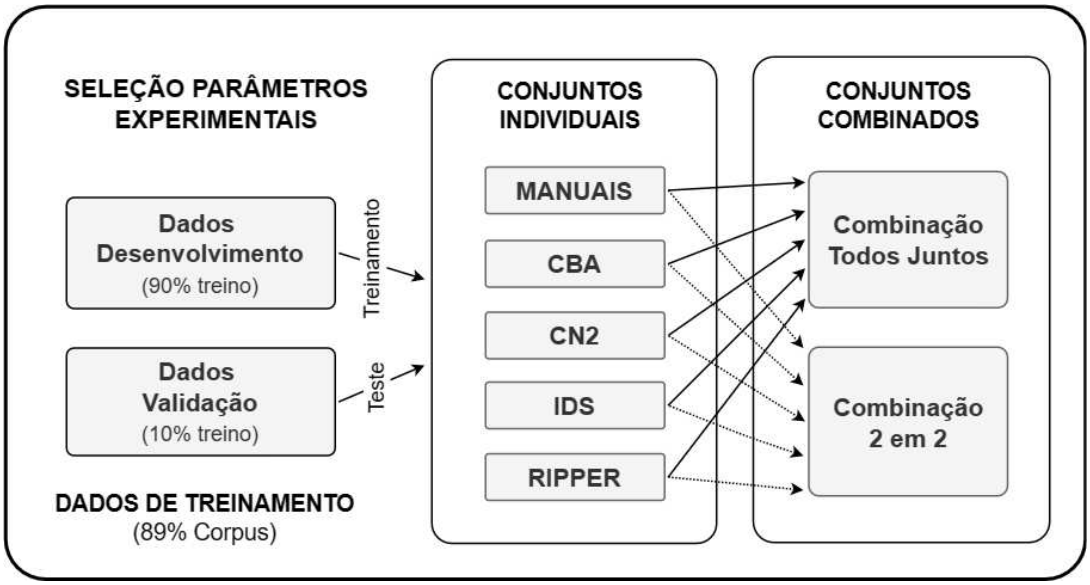


Figura 5.3: Visão geral da metodologia para seleção das melhores configurações de parâmetros experimentais.

Fonte: Elaborada pelo autor.

5.4.1.1 Formação dos Conjuntos de Regras Individuais Os parâmetros considerados em nosso arranjo experimental para selecionar os melhores conjuntos de regras individuais abrangem os hiperparâmetros individuais de cada algoritmo, conforme mencionado na Seção 4.2, bem como o limite de corte por acurácia da regra, a ordenação das regras e a quantidade de *features* utilizadas para gerar as regras. A acurácia de uma regra é definida como o número de vezes que a regra produz o tipo correto de relação temporal dividido pelo número de vezes que é aplicada, conforme medido na parte de dados de treinamento.

Para os limites de corte, foram considerados os valores de 0% (sem corte), 40%, 50% e 60% de acurácia, o que implica a remoção das regras com acurácia abaixo desses limites. Quanto à ordenação das regras, foram consideradas tanto a ordem original de geração das regras quanto a ordenação por acurácia da regra. Em relação à quantidade de *features*, foram consideradas todas as 70 *features* disponíveis, assim como as melhores *features* selecionadas pela técnica de eliminação recursiva de *features* com validação

cruzada apresentada na Seção 4.1.6. As melhores *features* selecionadas totalizaram 52, e estão detalhadas na Seção A.2. É importante ressaltar que, devido às limitações de recursos computacionais, para o algoritmo `CBA` foram consideradas apenas 41 *features*, conforme mencionado na Seção 4.2.2. Destacamos ainda que, para as regras manuais, aplicam-se somente o limite de corte por acurácia da regra e a ordenação das regras.

Realizamos os experimentos utilizando os dados de desenvolvimento para treinar diferentes combinações de hiperparâmetros. Os hiperparâmetros do modelo foram ajustados com base no desempenho obtido nos dados de validação. Cada combinação de hiperparâmetros foi avaliada nos dados de validação utilizando a acurácia como métrica de avaliação. Com base nos resultados obtidos na validação, selecionamos os melhores hiperparâmetros, os quais serão apresentados no Capítulo 6. Ao final dessa etapa, obtivemos os melhores conjuntos de regras individuais, considerando os resultados da avaliação nos dados de validação.

5.4.1.2 Formação dos Conjuntos de Regras Combinados Para a elaboração dos conjuntos combinados de regras, foram realizadas duas combinações com os conjuntos individuais. Na primeira abordagem, o conjunto de regras foi obtido combinando todos os conjuntos individuais. Na segunda abordagem de combinação, o conjunto de regras foi formado pela combinação de dois conjuntos individuais.

Para formar o conjunto que combina todos os conjuntos individuais, é importante definir a melhor ordem dos conjuntos individuais. Para essa finalidade, avaliamos a ordem decrescente dos conjuntos individuais de regras com base em sua acurácia e em seu coeficiente de variação das acurácias obtidas nos experimentos, além da ordem crescente por quantidade de regras de cada conjunto. No conjunto resultante da melhor ordem, exploramos diferentes limites de corte por acurácia da regra, especificamente os valores de 70%, 80% e 90% de acurácia, o que implica na remoção das regras com acurácia abaixo desses limites. Adicionalmente, consideramos a ordenação das regras do conjunto, tanto na ordem original quanto na ordenação por acurácia da regra.

Para formar o conjunto de regras que combina dois conjuntos individuais, consideramos todas as combinações possíveis entre os cinco conjuntos individuais. Para cada combinação, exploramos os mesmos limites de corte por acurácia utilizados no conjunto formado pela combinação de todos os conjuntos individuais: 70%, 80% e 90% de acurácia. Em todos os casos, as regras foram ordenadas por acurácia.

É importante salientar que as regras duplicadas ou aquelas que compartilhavam todas as condições similares com outra regra eram eliminadas ao término do procedimento de combinação.

Os melhores hiperparâmetros foram selecionados com base nos resultados obtidos nos dados de validação, utilizando a acurácia como métrica de avaliação. Essas escolhas visavam encontrar as configurações mais adequadas para a formação dos conjuntos de regras combinados. Ao final dessa etapa, obtemos os melhores conjuntos de regras combinados: um composto por todos os conjuntos individuais e outro formado pela combinação de dois conjuntos individuais, levando em consideração os resultados da avaliação nos dados de validação.

5.4.2 Avaliação Final de Desempenho do Método

Nesta subseção, descreveremos os procedimentos adotados para a avaliação do desempenho final do método e para obtenção dos sete conjuntos de regras de melhor desempenho com base dos dados de teste. A Figura 5.4 apresenta uma visão geral da metodologia adotada, a qual descreveremos em seguida.

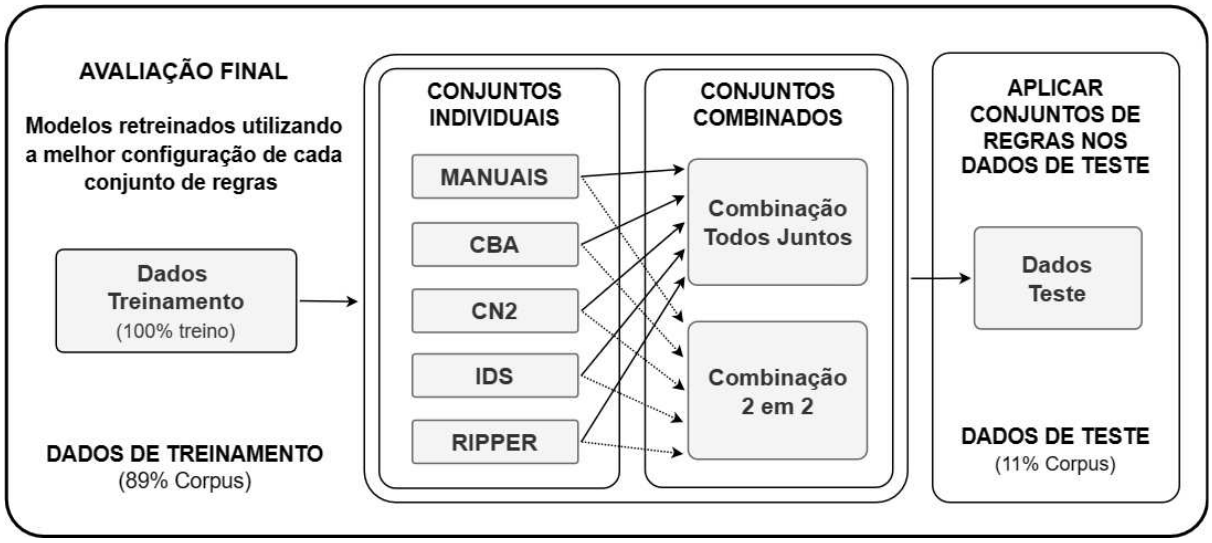


Figura 5.4: Visão geral da metodologia para avaliação final do método.
Fonte: Elaborada pelo autor.

Após a seleção dos melhores hiperparâmetros, procedemos combinando os conjuntos de dados de treinamento e validação em um único conjunto de treinamento expandido. Esse conjunto é composto pelos dados de treinamento do *corpus* previamente divididos. Utilizamos o conjunto de treinamento expandido para treinar o modelo final, empregando os melhores hiperparâmetros obtidos durante a seleção.

Posteriormente, realizamos a avaliação do desempenho final do modelo nos dados de teste, os quais ainda não haviam sido vistos pelo modelo. Essa avaliação é realizada por meio da métrica de acurácia, a qual possibilita a mensuração da eficácia e capacidade de generalização do modelo, juntamente com a métrica F1-score, a qual permite aferir a harmonia entre precisão e *recall* do modelo em relação a múltiplas classes.

A fusão dos conjuntos de treinamento e validação nessa etapa nos possibilita aproveitar todos os dados disponíveis para treinar o modelo final. Essa abordagem foi adotada visando utilizar o máximo de informações durante o treinamento do modelo, antes de testá-lo nos dados de teste ainda não vistos. Para garantir uma avaliação imparcial do desempenho final do modelo, os dados de teste foram mantidos isolados durante a seleção dos hiperparâmetros, assegurando que esses dados permanecessem sem uso até a fase final de avaliação. Ao final dessa etapa, obtivemos os sete melhores conjuntos de regras, levando em consideração os resultados da avaliação final nos dados de teste.

5.4.3 Aplicação dos Conjuntos de Regras

Após finalizar a versão final dos sete conjuntos de regras, procedemos à aplicação de cada um deles nos conjuntos de dados de treinamento e teste. Essa aplicação foi realizada de acordo com duas abordagens distintas apresentadas na Seção 4.3: a primeira baseada na "primeira regra acionada" e a segunda utilizando um sistema de votação. Ambas as abordagens foram aplicadas inicialmente sem a inclusão do fechamento temporal (Seção 4.4). Em seguida, repetimos o processo, porém, adicionando o fechamento temporal.

Ao concluir essa etapa, os resultados obtidos por cada conjunto de regras foram submetidos a testes estatísticos apropriados. Essa análise estatística permitiu identificar o conjunto de regras que apresentou o melhor desempenho entre os sete conjuntos distintos e o *baseline*. A comparação dos resultados foi realizada considerando duas métricas de avaliação: acurácia e *F1-score*. Dessa forma, por meio dessas métricas e testes estatísticos, foi possível determinar qual conjunto de regras apresentou o melhor desempenho.

5.5 CONSIDERAÇÕES FINAIS

Neste capítulo, descrevemos a configuração experimental adotada para avaliar a eficácia do nosso método de identificação de tipos de relações temporais. Utilizamos o *corpus* TimeBankPT como conjunto de dados e apresentamos as métricas de avaliação utilizadas, destacando a acurácia como métrica principal. Dividimos os documentos do *corpus* em conjuntos de treinamento, validação e teste, garantindo a representatividade das fontes e a coesão dos documentos.

Realizamos uma seleção de hiperparâmetros para obter os melhores conjuntos de regras individuais e combinados. Essa seleção envolveu a definição de limites de corte por acurácia das regras, ordenação das regras e seleção de *features*. Os conjuntos de regras foram treinados e avaliados nos dados de validação, e os melhores hiperparâmetros foram selecionados com base nos resultados obtidos.

Para avaliar o desempenho final do nosso método, utilizamos o conjunto de treinamento expandido para treinar o modelo final com os melhores hiperparâmetros selecionados. Em seguida, testamos o modelo nos dados de teste, que representam um conjunto de dados ainda não visto pelo modelo. A acurácia foi utilizada como métrica para medir a eficácia e a capacidade de generalização do modelo.

Ao analisar o *corpus* TimeBankPT, constatamos que nem todas as relações temporais foram rotuladas, o que limitou nossa avaliação aos dados rotulados disponíveis. No entanto, seguimos a abordagem adotada por outros estudos que utilizaram o mesmo *corpus* e focamos apenas nos dados rotulados.

Em resumo, este capítulo apresentou uma avaliação experimental detalhada do nosso método de identificação de tipos de relações temporais. Os resultados obtidos nos experimentos realizados nos conjuntos de treinamento, validação e teste forneceram *insights* sobre o desempenho do método e sua capacidade de generalização. No próximo capítulo, apresentaremos os resultados completos, incluindo outras medidas de avaliação, como o *F1-score*, e a matriz de confusão, para uma análise mais abrangente e comparativa do nosso método em relação ao *baseline* estabelecido: o *LX-TimeAnalyzer*.

Capítulo 6

RESULTADOS E DISCUSSÕES

Neste capítulo, são apresentados os resultados dos experimentos realizados nos conjuntos de dados de treinamento e teste para a tarefa de identificação de tipos de relação temporal entre evento e expressão temporal (*event-time*) dentro da mesma sentença. Cada conjunto de regras será analisado, destacando seus principais elementos. Além disso, serão apresentados os resultados da seleção otimizada dos hiperparâmetros para cada conjunto de regras, bem como os resultados obtidos ao aplicá-los nos dados do *corpus*. Os resultados considerarão duas abordagens distintas para a aplicação dos conjuntos de regras: a primeira regra acionada e o sistema de votação, vistos na Seção 4.3.

6.1 RESULTADOS DA APLICAÇÃO DOS CONJUNTOS DE REGRAS INDIVIDUAIS

Todos os conjuntos de regras gerados pelos algoritmos de aprendizado de regras investigados neste estudo obtiveram uma cobertura de 100%, abrangendo todas as instâncias de dados. Isso foi possível devido à designação da classe majoritária como classe padrão, conforme descrito na Seção 4.2. A classe majoritária é OVERLAP, conforme ilustrado na Figura 5.2. Mesmo quando consideramos apenas as regras, sem a aplicação da classe padrão, esses conjuntos de regras ainda alcançaram uma taxa de cobertura elevada. A fim de demonstrar a abrangência de cada conjunto de regras, mencionaremos a cobertura de cada conjunto na Seção 6.3, caso a classe padrão não tivesse sido utilizada.

Além disso, observamos que a aplicação do fechamento temporal, conforme descrito na Seção 4.4, não apresentou diferenças em comparação aos resultados sem sua utilização. Acreditamos que isso se deve ao emprego da classe padrão nos conjuntos de regras, que classifica todos os pares presentes na sentença. O fechamento temporal é uma etapa de pós-processamento que ocorre após a identificação dos pares *event-time* por meio das regras. Durante sua aplicação, quando um novo par é inferido e já foi identificado pelas regras, prevalece o tipo de relação temporal identificado pelas regras. Uma vez que todos os pares foram identificados pelas regras e possuem prioridade, as inferências do fechamento temporal não têm efeito. Em nossa implementação, priorizamos as identificações

realizadas pelas regras em vez das inferências realizadas pelo fechamento temporal, a fim de minimizar resultados incorretos e a propagação de erros causados pelo fechamento. Portanto, optamos por apresentar os resultados sem considerar o fechamento temporal, uma vez que são idênticos aos resultados obtidos sem sua aplicação, com exceção do conjunto de regras manuais, conforme descrito na Seção 6.1.1.

No entanto, é importante destacar que o uso da classe padrão em cada conjunto de regras mostrou-se mais vantajoso do que a aplicação do fechamento temporal, tanto em termos de cobertura quanto de outras métricas. A seguir, apresentaremos os resultados dos conjuntos de regras individuais.

6.1.1 Resultados da Aplicação do Conjunto de Regras Manuais

No que diz respeito ao conjunto de regras manuais, os resultados são apresentados de duas maneiras: sem a aplicação da classe padrão e com a sua aplicação. No entanto, devido à baixa cobertura desse conjunto de regras sem o uso da classe padrão em comparação com os demais conjuntos de regras sem a utilização da classe padrão, pretendemos avaliar também seu desempenho apenas nas instâncias cobertas. Assim, a aplicação do fechamento temporal apresenta diferenças nos resultados e também será apresentada.

Na Seção 5.4.1.1, verificamos que, em nosso experimento para o conjunto de regras manuais, foi considerada a aplicação de um limite de corte com base na acurácia da regra e a ordenação das regras. A Tabela 6.1 apresenta a melhor configuração determinada a partir dos dados de validação para esse conjunto de regras. Observou-se que o limite de corte mais adequado foi de 50% de acurácia, ou seja, as regras com desempenho inferior a esse limite foram excluídas. Em relação à ordenação das regras, verificou-se que a ordem original proporcionou resultados superiores quando comparada com a ordenação por acurácia. Nesse contexto específico, a ordem original refere-se à manutenção da mesma sequência das regras desenvolvidas por D'Souza (2015) para o idioma inglês, as quais serviram de base para a formulação do nosso conjunto de regras manuais. O conjunto completo de regras totalizou 35 regras, excluindo a regra para a classe padrão.

Tabela 6.1: Melhor arranjo experimental e detalhes do conjunto de regras manuais.

Experimento	**Valor**
Ordem das regras	original
Corte por acurácia	50%
Detalhes do Conjunto de Regras	
Número de regras	35

Fonte: Elaborada pelo autor.

6.1.1.1 Resultados Sem o Uso da Classe Padrão Inicialmente, apresentamos os resultados obtidos a partir dos experimentos realizados nos conjuntos de dados de trei-

namento e teste para analisar a aplicação do conjunto de regras manuais sem a utilização da classe majoritária, a classe OVERLAP, como classe padrão. A Tabela 6.2 apresenta os resultados com variações na forma de aplicar os conjuntos de regras (Seção 4.3) e no uso do fechamento temporal (Seção 4.4). As colunas mostram as métricas de acurácia, *F1-score* e cobertura para os dados de treinamento e teste.

Tabela 6.2: Resultados da aplicação do conjunto de regras manuais sem utilização da classe padrão.

	Dados Treinamento			**Dados Teste**		
	Acurácia	**F1**	**Cobertura**	**Acurácia**	**F1**	**Cobertura**
Baseline	69,0	66,1	100,0	66,9	62,5	100,0
Sem Fechamento Temporal						
Primeira Regra	74,9	70,6	32,9	**80,0**	**78,3**	35,5
Votação	**75,1**	**70,7**	32,9	**80,0**	**78,3**	35,5
Com Fechamento Temporal						
Primeira Regra	72,2	67,0	**35,2**	77,1	75,0	**41,4**
Votação	72,4	67,1	**35,2**	77,1	75,0	**41,4**

Fonte: Elaborada pelo autor.

Conforme observado na Tabela 6.2, a maior cobertura alcançada foi de 41,4% nos dados de teste e 35,2% nos dados de treinamento, especificamente quando o fechamento temporal foi aplicado. Por outro lado, as maiores acurácia e *F1-score* foram alcançadas quando o fechamento temporal não foi aplicado, sendo de 80% e 78,3% nos dados de teste, e 75,1% e 70,7% nos dados de treinamento, respectivamente. Isso indica que a aplicação do fechamento temporal, embora tenha aumentado a quantidade de instâncias classificadas, resultou em uma redução da acurácia e do *F1-score*. Foi inesperado constatar que a adição do fechamento temporal levou a uma diminuição na acurácia. Esperava-se um desempenho melhor ao tornar explícitas as relações implícitas do conjunto de classificação, semelhante ao que foi encontrado por Chambers et al. (2014). No entanto, ao analisarmos mais detalhadamente esse resultado, observamos que a quantidade de classificações corretas aumentou, mas a quantidade de classificações incorretas foi proporcionalmente maior.

Quanto às formas de aplicação do conjunto de regras, verificamos que a diferença entre a primeira regra acionada e o sistema de votação foi nula nos dados de teste e muito pequena nos dados de treinamento, com uma leve vantagem para o sistema de votação. Acreditamos que a razão pela qual a diferença foi tão pequena se deve à baixa taxa de cobertura, pois não houve classificações suficientes para que a diferença fosse mais significativa.

A Tabela 6.3 apresenta a matriz de confusão da classificação da configuração de melhor

desempenho nos dados de treinamento, que consiste na aplicação do conjunto de regras pelo sistema de votação sem a utilização do fechamento temporal, conforme indicado na Tabela 6.2. Ao analisarmos os valores contidos na matriz, podemos observar que as classes Before e After possuem um número limitado de instâncias corretamente classificadas e uma quantidade de erros maior do que de acertos, o que indica que o conjunto de regras enfrenta dificuldades em distinguir adequadamente entre as relações temporais Before e After.

Tabela 6.3: Matriz de confusão da melhor configuração do conjunto de regras manuais sem utilização da classe padrão nos dados de treinamento (votação, sem fechamento temporal).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	16	2	59	0	0	0
	Before	0	32	38	0	0	0
	Overlap	5	8	308	0	0	0
	Before-or-Overlap	0	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	0	0	6	0	0	0

Fonte: Elaborada pelo autor.

Por outro lado, a classe majoritária Overlap apresenta uma proporção maior de instâncias corretamente classificadas em comparação com as outras classes. Isso pode indicar que o modelo é mais eficaz na identificação de relações temporais desse tipo. As classes Before-or-Overlap (BorO), Overlap-or-After (OorA) e Vague não possuem nenhuma instância na matriz de confusão, pois não foram criadas regras para essas classes nesse conjunto de regras.

Na Tabela 6.4, apresentamos a matriz de confusão da configuração de melhor desempenho nos dados de teste, obtida sem a aplicação do fechamento temporal. Não foram observadas diferenças entre as formas de aplicar o conjunto de regras, conforme indicado na Tabela 6.2. Ao analisarmos os valores contidos na matriz, observamos resultados proporcionalmente semelhantes à matriz obtida nos dados de treinamento.

Ao não utilizar a classe padrão, estamos avaliando o desempenho das regras manuais apenas no conjunto de dados coberto por elas. Devido à baixa taxa de cobertura, o conjunto de regras não é capaz de abranger o maior número possível de instâncias do conjunto de dados. Portanto, não é possível comparar os resultados desse conjunto com outros conjuntos que alcançaram cobertura total, incluindo o nosso *baseline*.

6.1.1.2 Resultado Utilizando a Classe Padrão Para comparar os resultados do conjunto de regras manuais com os demais conjuntos de regras que alcançaram 100% de cobertura, foi atribuída a classe majoritária como a classe padrão para esse conjunto. Isso

Tabela 6.4: Matriz de confusão da melhor configuração do conjunto de regras manuais sem utilização da classe padrão nos dados de teste (primeira regra acionada, sem fechamento temporal).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	4	0	3	0	0	0
	Before	0	4	5	0	0	0
	Overlap	0	3	40	0	0	0
	Before-or-Overlap	0	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	0	0	1	0	0	0

Fonte: Elaborada pelo autor.

significa que todas as instâncias de dados não classificadas pelas regras serão classificadas como Overlap. Os resultados obtidos a partir dos experimentos realizados nos conjuntos de dados de treinamento e teste são apresentados na Tabela 6.5. As linhas da tabela variam de acordo com a forma de aplicação do conjunto de regras (Seção 4.3), e as colunas mostram as métricas de acurácia e *F1-score* para os conjuntos de dados de treinamento e teste.

Tabela 6.5: Resultados da aplicação do conjunto de regras manuais utilizando a classe padrão.

	Dados Treinamento		**Dados Teste**	
	Acurácia	**F1**	**Acurácia**	**F1**
Baseline	69,0	66,1	66,9	62,5
Primeira Regra	56,9	44,9	**66,9**	58,1
Votação	**57,0**	**44,9**	**66,9**	58,1

Fonte: Elaborada pelo autor.

É importante ressaltar que a aplicação do fechamento temporal não afetou os resultados dos conjuntos de regras que utilizam uma classe padrão. Além disso, esses conjuntos de regras sempre alcançam uma cobertura de 100%. Portanto, os resultados do fechamento temporal e da cobertura foram omitidos.

Conforme observado na Tabela 6.5, nos dados de treinamento, a maior acurácia e *F1-score* obtidos foi de 57% e 44,9%, respectivamente, quando o conjunto de regras foi aplicado pelo sistema de votação. Nos dados de teste, a acurácia foi de 66,9% em ambas as formas de aplicação, equiparando-se à acurácia do *baseline*. No entanto, o *F1-score* foi

inferior, atingindo 58,1%.

A atribuição da classe majoritária como a classe padrão em um conjunto de regras com baixa cobertura (máximo de 41,4%, conforme Tabela 6.2) pode causar sérias distorções nas classificações das outras classes, especialmente quando a classe majoritária representa a maioria dos dados. Em nosso caso específico, a classe Overlap representa 50,3% dos pares *event-time* nos dados de treinamento e 59,2% nos dados de teste, conforme dados apresentados na Figura 5.2.

A distorção é evidenciada na matriz de confusão dos dados de treinamento, conforme mostrado na Tabela 6.6. A classe Overlap foi classificada corretamente na maioria dos casos, mas as classes After e Before foram erroneamente classificadas como Overlap muitas vezes, resultando em uma redução significativa do seu *recall*. Essa situação já estava presente antes da aplicação da classe padrão, como mostrado na Tabela 6.3, e foi agravada pelo seu uso.

Tabela 6.6: Matriz de confusão da melhor configuração do conjunto de regras manuais utilizando a classe padrão nos dados de treinamento (votação).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	16	2	338	0	0	0
	Before	0	32	246	0	0	0
	Overlap	4	8	773	0	0	0
	Before-or-Overlap	0	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	0	0	21	0	0	0

Fonte: Elaborada pelo autor.

Ao observarmos a matriz de confusão dos dados de teste, na Tabela 6.7, podemos notar uma distorção semelhante, com a maioria das classificações das classes After e Before sendo atribuídas à classe Overlap. Além disso, não foram observadas previsões para as classes Vague, Before-or-Overlap (BorO) e Overlap-or-After (OorA) porque não foram criadas regras para identificá-las.

Portanto, apesar da acurácia ser igual à do *baseline*, o conjunto de regras manuais não pode ser equiparado ao *baseline*, não apenas por ter um *F1-score* inferior, mas principalmente por ter a maioria das instâncias classificadas com a classe majoritária, causando distorções nas outras classes.

6.1.2 Resultados da Aplicação do Conjunto de Regras `CBA`

O algoritmo `CBA` gerou um conjunto de regras com base nas regras de associação de classe (CARs - do inglês *Class Association Rules*, Seção 2.6.2.1) durante a etapa de construção de um classificador.

Tabela 6.7: Matriz de confusão do conjunto de regras manuais utilizando a classe padrão nos dados de teste.

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	4	0	26	0	0	0
	Before	0	4	16	0	0	0
	Overlap	0	3	105	0	0	0
	Before-or-Overlap	0	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	0	0	11	0	0	0

Fonte: Elaborada pelo autor.

Tabela 6.8: Elementos do conjunto de regras `CBA`.

Elementos do Conjunto de Regras	**Valor**
Quant. regras para a classe Overlap	285
Quant. regras para a classe Before	112
Quant. regras para a classe After	171
Quantidade total de regras	568
Quantidade de condições	2.123
Média de condições por regra	3,7

Fonte: Elaborada pelo autor.

A Tabela 6.8 mostra que nessa etapa, o algoritmo selecionou 568 regras para o classificador, tornando-se o maior conjunto individual de regras. As regras totalizam 2.123 condições, resultando em uma média de 3,7 condições por regra. Também é possível observar a quantidade de regras por classe.

É de relevância enfatizar que uma condição é constituída por uma combinação de *features*, operador e valor. O conjunto dessas condições conectadas por conjunções compõe o antecedente de uma regra, conforme explicado na Seção 2.6.1 e exemplificado no caso (34), extraído do conjunto de regras disponível na Seção B.1. Adicionalmente, o número médio de condições em uma regra reflete sua complexidade. De acordo com a análise realizada por Boll e Clair (1995), tal medida é considerada importante, pois as condições conferem maior especificidade às regras. Regras com um maior número de condições podem apresentar desempenho inferior em dados não observados, quando comparadas a regras com um menor número de condições.

(34)

$$\underbrace{\underbrace{\underbrace{\text{event-between-order}}_{feature}\ \underbrace{=}_{operador}\ \underbrace{\textit{False}}_{valor}}_{condição}\ \underbrace{\textit{and}}_{conjunção}\ \underbrace{\underbrace{\text{event-pos}}_{feature}\ \underbrace{=}_{operador}\ \underbrace{\textit{'Noun'}}_{valor}}_{condição}}_{antecedente} \Rightarrow \underbrace{\textsc{Overlap}}_{consequente}$$

Em nossos experimentos para selecionar os melhores hiperparâmetros para o conjunto de regras gerado pelo algoritmo `CBA`, conforme descrito na Seção 5.4.1.1, consideramos um limite de corte com base na acurácia da regra, na ordenação das regras e na quantidade de *features*, além dos hiperparâmetros do algoritmo mencionados na Seção 4.2.2. Com base nos dados de validação, foi identificada a melhor configuração para esse conjunto de regras, apresentada na Tabela 6.9.

Tabela 6.9: Seleção dos melhores hiperparâmetros do conjunto de regras `CBA`.

Hiperparâmetro	**Valor**
`init_conf`	0,7
`init_support`	0,6
`minlen`	3
`init_maxlen`	5
`max_iterations`	30
Limite de corte pela acurácia	50%
Ordenação das regras	acurácia
Quantidade de *features*	41

Fonte: Elaborada pelo autor.

Com o objetivo de destacar as *features* mais frequentemente utilizadas pelo algoritmo na geração das regras, é apresentada a Tabela 6.10 que exibe as dez *features* mais predominantes. Isso auxilia na compreensão das *features* que desempenham um papel significativo nas decisões tomadas pelo classificador. Vale ressaltar que a descrição de cada *features* pode ser encontrada na Seção A.1.

Ao observar a Tabela 6.10, é possível notar que as *features* mais recorrentes nas regras incluem as *features* contextuais *event-between-order*, que determinam se há outro evento entre o par *event-time* da relação, e *event-timex3-distance*, que informa a distância entre o evento e a expressão temporal do par da relação, além da classe do evento anotada no *corpus* (*event-class*).

Entre as *features* mais frequentes, aquelas que contêm informações morfológicas, como a *feature* que informa o tempo do verbo que rege a expressão temporal (*timex3-gov-verb-tense*), a classe gramatical do primeiro *token* após a expressão temporal (*timex3-pos-token-1-follow*) e a classe gramatical do evento mais próximo da expressão temporal do

Tabela 6.10: As dez *features* predominantes do conjunto de regras `CBA`

Feature	**Quantidade**
event-between-order	142
event-timex3-distance	119
event-class	97
timex3-gov-verb-tense	89
reichenbach-temporal-mod-function	83
reichenbach-direct-modification	83
timex3-pos-token-1-follow	77
event-closest-to-timex3-pos	76
event-pos-token-3-follow	68
event-closest-to-event-equal-class	67

Fonte: Elaborada pelo autor.

par da relação (*event-closest-to-timex3-pos*), ocupam uma proporção maior, correspondendo a cerca de 34,4% da frequência. Em segundo lugar, estão as *features* que contêm informações contextuais, representando aproximadamente 29% da frequência.

Na Tabela 6.11, são apresentados os resultados dos experimentos realizados nos conjuntos de dados de treinamento e teste para analisar a aplicação do conjunto de regras gerado pelo algoritmo `CBA`. As linhas representam as diferentes formas de aplicar o conjunto de regras (Seção 4.3), e as colunas mostram as métricas de acurácia e *F1-score* para os conjuntos de treinamento e teste.

Tabela 6.11: Resultados da aplicação do conjunto de regras gerado pelo algoritmo `CBA`.

	Dados Treinamento		**Dados Teste**	
	Acurácia	**F1**	**Acurácia**	**F1**
Baseline	69,0	66,1	66,9	62,5
Primeira Regra	**92,8**	**91,7**	62,7	59,9
Votação	91,0	90,0	**62,7**	**60,5**

Fonte: Elaborada pelo autor.

Conforme observado na Tabela 6.11, a maior acurácia e *F1-score* obtidos nos dados de treinamento foi de 92,8% e 91,7%, respectivamente, quando a primeira regra foi aplicada. Já nos dados de teste, os melhores resultados foram obtidos usando o sistema de votação, com uma acurácia de 62,7% e *F1-score* de 60,5%, apresentando um desempenho abaixo do *baseline*.

Quando um classificador apresenta um desempenho elevado nos dados de treinamento, mas não alcança resultados tão bons nos dados de teste, isso pode indicar a ocorrência

de *overfitting*. O overfitting acontece quando o modelo se ajusta muito bem aos dados de treinamento específicos, capturando os padrões e características individuais dos exemplos, mas não consegue generalizar essas aprendizagens para novos dados.

Nesse contexto, os valores elevados na matriz de confusão dos dados de treinamento, conforme mostrado na Tabela 6.12, sugerem que o conjunto de regras aprendeu detalhes e peculiaridades específicas dos dados de treinamento. Isso pode resultar em uma representação excessivamente específica dos dados de treinamento, tornando difícil a adaptação a novos dados que possam apresentar variações ou características diferentes.

Tabela 6.12: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `CBA` nos dados de treinamento (primeira regra acionada).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	346	6	14	0	0	0
	Before	25	242	23	0	0	0
	Overlap	2	1	749	0	0	0
	Before-or-Overlap	6	0	0	0	0	0
	Overlap-or-After	0	5	0	0	0	0
	Vague	4	5	12	0	0	0

Fonte: Elaborada pelo autor.

Como resultado, quando o modelo é aplicado a dados de teste que diferem dos dados de treinamento, sua capacidade de generalização pode ser comprometida. Ele pode ter dificuldade em classificar corretamente exemplos desconhecidos, pois está restrito aos padrões específicos dos dados de treinamento que não são universalmente aplicáveis, especialmente para as classes After e Before, como indicado na matriz de confusão da Tabela 6.13. É importante mencionar que não houve previsões para as classes minoritárias Before-or-Overlap (BorO), Overlap-or-After (OorA) e Vague porque o algoritmo não foi capaz de gerar regras para essas classes, conforme indicado na Tabela 6.8.

6.1.3 Resultados da Aplicação do Conjunto de Regras `CN2`

O conjunto de regras gerado pelo algoritmo `CN2` é composto por um total de 205 regras, contendo 515 condições no total, resultando em uma média de 2,5 condições por regra. Essas informações estão apresentadas na Tabela 6.14. É importante destacar que aproximadamente 51% das regras classificam a classe Overlap, cerca de 30% classificam a classe After e o restante classifica a classe Before.

Na Tabela 6.15, são exibidos os melhores hiperparâmetros selecionados para o conjunto de regras gerado pelo algoritmo `CN2`, levando em consideração os resultados obtidos nos dados de validação. Além dos hiperparâmetros do algoritmo mencionados na Se-

Tabela 6.13: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `CBA` nos dados de teste (votação).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	12	1	17	0	0	0
	Before	4	9	7	0	0	0
	Overlap	13	8	85	0	0	0
	Before-or-Overlap	1	0	0	0	0	0
	Overlap-or-After	0	1	0	0	0	0
	Vague	4	4	3	0	0	0

Fonte: Elaborada pelo autor.

Tabela 6.14: Elementos do conjunto de regras `CN2`.

Elementos do Conjunto de Regras	**Valor**
Quant. regras para a classe Overlap	105
Quant. regras para a classe Before	39
Quant. regras para a classe After	61
Quantidade total de regras	205
Quantidade de condições	515
Média de condições por regra	2,5

Fonte: Elaborada pelo autor.

ção 4.2.3, também foram considerados um limite de corte com base na acurácia da regra, a ordenação das regras e a quantidade de *features*.

A Tabela 6.16 apresenta as principais *features* predominantes no conjunto de regras, classificadas de acordo com sua frequência de ocorrência. Nessa tabela, são exibidas as dez *features* mais frequentes. A descrição detalhada de cada *feature* está disponível na Seção A.1.

Ao analisar a Tabela 6.16, pode-se observar que as três *features* mais frequentes coincidem com as do algoritmo `CBA`: *event-class, event-timex3-distance* e *event-between-order*. Em seguida, temos a *feature timex3-preposition-precede*, que indica a preposição que precede a expressão temporal no texto, e as *features event-pos-token-1-precede* e *event-pos-token-2-precede*, que fornecem a classe gramatical dos dois *tokens* que antecedem o evento.

Entre as *features* mais frequentes, aquelas que contêm informações contextuais, como *event-timex3-distance, event-between-order* e *timex3-preposition-precede*, correspondem a aproximadamente 34,7% da frequência total. Em segundo lugar, temos as *features* que

Tabela 6.15: Seleção dos melhores hiperparâmetros do conjunto de regras `CN2`.

Hiperparâmetros	**Valor**
Organização das regras	`Ordered`
Algoritmo de cobertura	`Weighted`
Gama	0,7
Medida de avaliação de hipóteses	`Laplace Accuracy`
Largura do feixe de busca	10
Cobertura mínima de regras	3
Comprimento máximo de regras	6
`default alpha`	0,7
`parent alpha`	0,7
Limite de corte pela acurácia	40%
Ordenação das regras	original
Quantidade de *features*	52

Fonte: Elaborada pelo autor.

Tabela 6.16: As dez *features* predominantes do conjunto de regras `CN2`.

Feature	**Quantidade**
event-class	36
event-timex3-distance	36
event-between-order	30
timex3-preposition-precede	27
event-pos-token-1-precede	23
event-pos-token-2-precede	22
timex3-relevant-lemmas	22
timex3-pos-token-1-follow	20
timex3-pos-token-2-precede	18
reichenbach-direct-modification	17
event-dep	17

Fonte: Elaborada pelo autor.

contêm informações morfológicas, representando cerca de 31% da frequência.

A seguir, apresentamos os resultados dos experimentos realizados nos conjuntos de dados de treinamento e teste, analisando a aplicação do conjunto de regras gerado pelo algoritmo `CN2`. Conforme demonstrado na Tabela 6.17, a maior acurácia e *F1-score* obtidos nos dados de treinamento foi de 75,8% e 73,9%, respectivamente, quando a abordagem da primeira regra acionada foi utilizada. Nos dados de teste, os melhores resultados também foram obtidos com a abordagem da primeira regra acionada, alcançando uma acurácia

de 65,7% e *F1-score* de 61,3%, porém sem atingir o desempenho do *baseline*.

Tabela 6.17: Resultados da aplicação do conjunto de regras gerado pelo algoritmo `CN2`.

	Dados Treinamento		**Dados Teste**	
	Acurácia	**F1**	**Acurácia**	**F1**
Baseline	69,0	66,1	66,9	62,5
Primeira Regra	**75,8**	**73,9**	**65,7**	**61,3**
Votação	64,6	57,9	65,1	57,6

Fonte: Elaborada pelo autor.

Ao analisar a matriz de confusão dos dados de treinamento apresentada na Tabela 6.18, pode-se observar que uma grande parte das classificações que deveriam ser das classes After (146) e Before (125) foram incorretamente atribuídas à classe Overlap, indicando dificuldade do conjunto de regras em distinguir entre essas classes. No entanto, em relação às previsões realizadas pelas regras para as classes After e Before, houve um alto índice de acerto. O algoritmo `CN2` não identificou padrões para as classes minoritárias Before-or-Overlap (BorO), Overlap-or-After (OorA) e Vague.

Tabela 6.18: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `CN2` nos dados de treinamento (primeira regra acionada).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	207	5	146	0	0	0
	Before	14	141	125	0	0	0
	Overlap	20	14	744	0	0	0
	Before-or-Overlap	3	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	3	1	17	0	0	0

Fonte: Elaborada pelo autor.

Ao analisar a matriz de confusão dos dados de teste apresentada na Tabela 6.19, observa-se um fenômeno semelhante, no qual muitas das classificações que deveriam pertencer às classes After (20) e Before (10) foram erroneamente atribuídas à classe Overlap. Esse problema pode ser resultado do desbalanceamento das classes, pois a classe Overlap possui um número significativamente maior de instâncias em comparação com as outras classes. Quando isso ocorre, pode haver uma tendência de classificar erroneamente outras classes como a classe majoritária.

Por outro lado, em relação a todas as previsões realizadas para a classe After, houve um desempenho inferior. O conjunto de regras não consegue generalizar satisfatoriamente as classificações para dados não vistos, especialmente para a classe After.

Tabela 6.19: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `CN2` nos dados de teste (primeira regra acionada).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	10	1	20	0	0	0
	Before	4	7	10	0	0	0
	Overlap	9	2	94	0	0	0
	Before-or-Overlap	1	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	4	0	7	0	0	0

Fonte: Elaborada pelo autor.

6.1.4 Resultados da Aplicação do Conjunto de Regras `IDS`

A Tabela 6.20 exibe os principais elementos do conjunto de regras gerado pelo algoritmo `IDS`. Este conjunto consiste em um total de 383 regras, com um total de 761 condições, resultando em uma média de duas condições por regra. Este é o conjunto de regras com o menor número de condições por regra, pois o algoritmo `IDS` prioriza a interpretabilidade das regras, buscando alcançar menos condições por regra, como mencionado na Seção 2.6.2.3.

Tabela 6.20: Elementos do conjunto de regras `IDS`.

Elementos do Conjunto de Regras	**Valor**
Quant. regras para a classe Overlap	269
Quant. regras para a classe Before	34
Quant. regras para a classe After	79
Quant. regras para a classe Vague	1
Quantidade total de regras	383
Quantidade de condições	761
Média de condições por regra	2,0

Fonte: Elaborada pelo autor.

Diferente dos outros dois algoritmos mencionados, o `IDS` foi capaz de identificar uma regra para uma das classes minoritárias, a classe Vague. Essa regra conseguiu identificar

três instâncias nos dados de treinamento, mas nenhuma nos dados de teste, conforme indicado na Tabela 6.24. As demais classes das regras estão distribuídas da seguinte forma: 70,2% para a classe OVERLAP, 20,6% para a classe AFTER e o restante para a classe BEFORE.

A Tabela 6.21 exibe os melhores hiperparâmetros selecionados para o conjunto de regras gerado pelo algoritmo `IDS`, considerando os resultados obtidos nos dados de validação. Além dos hiperparâmetros mencionados na Seção 4.2.4, também foram considerados os seguintes critérios: um limite de corte baseado na acurácia da regra, a ordenação das regras e a quantidade de *features*.

Tabela 6.21: Seleção dos melhores hiperparâmetros do conjunto de regras `IDS`.

Hiperparâmetros	**Valor**
`rule_cutoff`	200
`algorithm`	SLS
Limite de corte pela acurácia	50%
Ordenação das regras	acurácia
Quantidade de *features*	52

Fonte: Elaborada pelo autor.

A Tabela 6.22 apresenta as principais *features* predominantes no conjunto de regras, classificadas de acordo com sua frequência de ocorrência. As três *features* mais recorrentes nas regras incluem a *feature* contextual *timex3-preposition-precede*, que informa a preposição que antecede a expressão temporal do par da relação, e as *features* contendo informações sintáticas *reichenbach-direct-modification*, que indica se a expressão temporal modifica diretamente o evento do par da relação, e *timex3-preposition-gov*, que informa a preposição que rege sintaticamente a expressão temporal do par da relação.

Dentre as *features* mais recorrentes, as que contêm informações morfológicas têm a maior frequência, representando cerca de 42,3%. Em segundo lugar, estão as *features* que fornecem informações sintáticas, com uma frequência de aproximadamente 23,4%. É interessante notar que, entre as *features* contendo informações morfológicas, o algoritmo `IDS` deu prioridade àquelas que indicam a classe gramatical das palavras em torno do evento e da expressão temporal da relação. Por exemplo, *features* como *event-pos-token-1-precede, event-pos-token-1-follow, event-pos-token-3-follow, timex3-pos-token-3-follow* e *timex3-pos-token-2-precede*.

A seguir, apresentaremos os resultados dos experimentos realizados nos conjuntos de dados de treinamento e teste, analisando a aplicação do conjunto de regras gerado pelo algoritmo `IDS`. Conforme demonstrado na Tabela 6.23, a maior acurácia e *F1-score* obtidos nos dados de treinamento foi de 77,3% e 75,2%, respectivamente, quando a abordagem da primeira regra acionada foi utilizada. Nos dados de teste, os melhores resultados foram obtidos aplicando o sistema de votação, alcançando uma acurácia de 66,9% e *F1-score* de 60,7%. O conjunto de regras conseguiu alcançar o desempenho do *baseline* em termos de acurácia, porém não atingiu o mesmo desempenho quando medido pelo *F1-score*.

Tabela 6.22: As dez *features* predominantes do conjunto de regras `IDS`.

Feature	**Quantidade**
timex3-preposition-precede	50
reichenbach-direct-modification	43
timex3-preposition-gov	40
timex3-relevant-lemmas	39
event-pos-token-1-precede	35
timex3-pos-token-3-follow	34
event-class	33
event-pos-token-1-follow	27
event-pos-token-3-follow	27
timex3-pos-token-2-precede	27

Fonte: Elaborada pelo autor.

Tabela 6.23: Resultados da aplicação do conjunto de regras gerado pelo algoritmo `IDS`.

	Dados Treinamento		**Dados Teste**	
	Acurácia	**F1**	**Acurácia**	**F1**
Baseline	69,0	66,1	66,9	62,5
Primeira Regra	**77,3**	**75,2**	64,5	59,6
Votação	74,9	72,2	**66,9**	**60,7**

Fonte: Elaborada pelo autor.

Observando a matriz de confusão dos dados de treinamento na Tabela 6.24, notamos que, dentre as classes de maior frequência, a mais deficiente é a BEFORE, pois, embora todas as classificações feitas pelo conjunto de regras para esta classe estejam corretas (precisão máxima), esta foi a classe com menor *recall* (42,5%), principalmente devido às classificações errôneas para a classe BEFORE realizadas pelas regras que identificam a classe OVERLAP.

Já na matriz de confusão dos dados de teste apresentada na Tabela 6.25, podemos observar que a deficiência da classe AFTER é ainda maior que a da classe BEFORE. Além de ser a classe com menor *recall* (19,4%), as previsões realizadas pelo conjunto de regras para esta classe foram bastante imprecisas (precisão de 35,3%). O alto desempenho na classe OVERLAP não compensou o baixo desempenho nas outras classes.

6.1.5 Resultados da Aplicação do Conjunto de Regras `RIPPER`

A Tabela 6.26 exibe os principais elementos do conjunto de regras gerado pelo algoritmo `RIPPER`. Este conjunto consiste em um total de 146 regras, com um total de 354 con-

Tabela 6.24: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `IDS` nos dados de treinamento (primeira regra acionada).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	208	0	148	0	0	0
	Before	1	117	157	0	0	0
	Overlap	3	0	785	0	0	0
	Before-or-Overlap	0	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	0	0	18	0	0	3

Fonte: Elaborada pelo autor.

Tabela 6.25: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `IDS` nos dados de teste (votação).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	6	0	25	0	0	0
	Before	2	7	11	0	0	0
	Overlap	5	1	100	0	0	0
	Before-or-Overlap	1	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	3	1	7	0	0	0

Fonte: Elaborada pelo autor.

dições, resultando em uma média de 2,4 condições por regra. Este é o conjunto com a menor quantidade de regras quando comparado com os outros três conjuntos gerados por algoritmos de aprendizado de regras. Este conjunto também apresenta maior equilíbrio na proporção de regras para as classes de maior frequência. Foram geradas aproximadamente 37,7% para a classe Overlap, 24,7% para After, 22,6% para Before e 15,1% para as demais classes de menor frequência. É importante ressaltar que este conjunto de regras foi o único que gerou regras para classificar todas as classes, e acredita-se que o principal motivo seja as alterações realizadas no algoritmo, conforme apresentado na Seção 4.2.5.

A Tabela 6.27 exibe os melhores hiperparâmetros selecionados para o conjunto de regras gerado pelo algoritmo `RIPPER`, levando em consideração os resultados obtidos nos dados de validação. Além dos hiperparâmetros do algoritmo mencionados na Seção 4.2.5, também foram considerados os seguintes critérios: um limite de corte baseado na acurácia

Tabela 6.26: Elementos do conjunto de regras `RIPPER`.

Elementos do Conjunto de Regras	**Valor**
Quant. regras para a classe OVERLAP	55
Quant. regras para a classe BEFORE	33
Quant. regras para a classe AFTER	36
Quant. regras para a classe VAGUE	3
Quant. regras para BEFORE-OR-OVERLAP	7
Quant. regras para OVERLAP-OR-AFTER	12
Quantidade total de regras	146
Quantidade de condições	354
Média de condições por regra	2,4

Fonte: Elaborada pelo autor.

da regra, a ordenação das regras e a quantidade de *features*. É importante destacar que o `RIPPER` foi o único algoritmo que obteve melhores resultados nos dados de validação fazendo uso de todas as 70 *features* disponíveis, em vez de apenas as *features* mais importantes, segundo a aplicação da técnica de eliminação recursiva de *features* com validação cruzada (REFCV, Seção 4.1.6).

Tabela 6.27: Seleção dos melhores hiperparâmetros do conjunto de regras `RIPPER`.

Hiperparâmetros	**Valor**
`k`	2
`prune_size`	0,1
`dl_allowance`	1.024
`n_discretize_bins`	10
`max_rules`	100
`max_rule_conds`	15
`max_total_conds`	`None`
Limite de corte pela acurácia	50%
Ordenação das regras	original
Quantidade de *features*	70

Fonte: Elaborada pelo autor.

A Tabela 6.28 apresenta as principais *features* predominantes no conjunto de regras, classificadas de acordo com sua frequência de ocorrência. Entre as *features* mais recorrentes, destacam-se aquelas que verificam se há um evento entre o par *event-time* da relação (*event-between-order*), a classe do evento (*event-class*), o tempo do verbo que rege a expressão temporal (*timex3-gov-verb-tense*) e a distância entre o evento e a expressão temporal da relação (*event-timex3-distance*).

Tabela 6.28: As dez *features* predominantes do conjunto de regras `RIPPER`.

Feature	**Quantidade**
event-between-order	41
event-class	22
timex3-gov-verb-tense	15
event-timex3-distance	13
event-pos-token-1-follow	12
event-pos-token-1-precede	12
reichenbach-direct-modification	12
timex3-preposition-precede	12
timex3-pos-token-3-precede	11
event-pos	11

Fonte: Elaborada pelo autor.

Dentre as *features* mais recorrentes, aquelas que contêm informações contextuais, como as *features event-between-order, event-timex3-distance* e *timex3-preposition-precede*, têm a maior frequência, representando cerca de 41,0% da frequência. Em segundo lugar, estão as *features* que fornecem informações morfológicas, com uma frequência aproximada de 37,9%. Dentre as *features* que contêm informações morfológicas, observa-se a prevalência daquelas que informam a classe gramatical das palavras em torno do evento ou da expressão temporal, como *event-pos-token-1-follow, event-pos-token-1-precede* e *timex3-pos-token-3-precede*, além do tempo verbal que rege a expressão temporal, como *timex3-gov-verb-tense*.

A seguir, apresentaremos os resultados dos experimentos realizados nos conjuntos de dados de treinamento e teste, analisando a aplicação do conjunto de regras gerado pelo algoritmo `RIPPER`. Conforme demonstrado na Tabela 6.29, a maior acurácia e *F1-score* obtidos nos dados de treinamento foi de 78,1% e 77,1%, respectivamente, quando a abordagem da primeira regra acionada foi utilizada. Nos dados de teste, os melhores resultados foram obtidos aplicando o sistema de votação, alcançando uma acurácia de 69,2% e *F1-score* de 66,1%. Esse conjunto de regras superou o desempenho do *baseline* tanto em termos de acurácia quanto de *F1-score* nos dados de teste.

Entre os conjuntos de regras individuais, este foi o único conjunto que gerou regras para todas as classes. No entanto, ao observar as classes de menor frequência na matriz de confusão dos dados de treinamento na Tabela 6.30, as regras para a classe Before-or-Overlap não acertaram nenhuma instância, classificando incorretamente 3 registros como After. Isso ocorreu devido à posição dessas regras no conjunto de regras, onde as instâncias que deveriam ser classificadas corretamente nessa classe foram classificadas incorretamente por outras regras de prioridade mais alta. Por outro lado, as regras para a classe Overlap-or-After classificaram corretamente as 6 instâncias registradas na matriz de confusão. Já as regras para a classe Vague acertaram 2 instâncias, mas erraram 19, resultando em um baixo valor de *recall* de 9,5%.

Tabela 6.29: Resultados da aplicação do conjunto de regras gerado pelo algoritmo `RIPPER`.

	Dados Treinamento		**Dados Teste**	
	Acurácia	**F1**	**Acurácia**	**F1**
Baseline	69,0	66,1	66,9	62,5
Primeira Regra	**78,1**	**77,1**	65,1	64,2
Votação	74,2	72,5	**69,2**	**66,1**

Fonte: Elaborada pelo autor.

Tabela 6.30: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `RIPPER` nos dados de treinamento (primeira regra acionada).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	244	22	96	0	1	1
	Before	14	184	86	0	0	0
	Overlap	40	34	688	0	0	0
	Before-or-Overlap	3	0	0	0	0	0
	Overlap-or-After	0	0	0	0	6	0
	Vague	2	2	15	0	0	2

Fonte: Elaborada pelo autor.

Ao analisar a matriz de confusão dos dados de teste apresentada na Tabela 6.31, fica evidente que nenhuma instância foi corretamente classificada de acordo com as regras atribuídas às classes de menor frequência. Essa falha na classificação se deve principalmente à falta de dados rotulados adequados para essas classes específicas. Embora o algoritmo tenha sido capaz de gerar algumas regras para essas classes, sua capacidade de generalização para dados não observados mostrou-se insuficiente.

Ao examinarmos as estatísticas presentes nesta matriz de confusão dos dados de teste da Tabela 6.31, constatamos que as classes de maior frequência, especificamente as classes Before e After, apresentam um número restrito de instâncias corretamente classificadas, acompanhadas de uma quantidade ligeiramente superior de erros em relação aos acertos. Esses resultados sugerem que este conjunto de regras também enfrenta dificuldades em distinguir adequadamente entre as relações temporais Before e After, embora essa dificuldade seja menos acentuada em comparação com os demais conjuntos de regras avaliados.

Tabela 6.31: Matriz de confusão da melhor configuração do conjunto de regras gerado pelo algoritmo `RIPPER` nos dados de teste (votação).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	15	3	13	0	0	0
	Before	2	8	10	0	1	0
	Overlap	7	5	94	0	0	0
	Before-or-Overlap	0	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	4	2	5	0	0	0

Fonte: Elaborada pelo autor.

6.2 RESULTADOS DA APLICAÇÃO DOS CONJUNTOS DE REGRAS COMBINADOS

O objetivo dos conjuntos de regras combinados é avaliar se a fusão dos conjuntos de regras gerados por cada algoritmo resulta em resultados superiores em comparação com os conjuntos de regras individuais utilizados para identificar os tipos de relações temporais na língua portuguesa. Em outras palavras, busca-se verificar a hipótese deste trabalho.

A seguir, serão apresentados os resultados do conjunto de regras composto pela combinação de todos os conjuntos de regras individuais, bem como do conjunto de regras formado pela melhor combinação de dois conjuntos individuais. Os detalhes sobre a composição desses conjuntos podem ser encontrados na Seção 5.4.1.2.

6.2.1 Resultados da Aplicação do Conjunto de Regras Formado pela Combinação de Todos Conjuntos Individuais.

A Tabela 6.32 apresenta os principais elementos do conjunto de regras resultante da combinação de todos os conjuntos individuais. Esse conjunto consiste em um total de 980 regras, com um total de 2.917 condições, resultando em uma média de 3 condições por regra. É importante destacar que esse conjunto possui a maior quantidade de regras entre todos os conjuntos avaliados. Após a combinação, as regras foram distribuídas aproximadamente da seguinte forma: 64,4% para a classe Overlap, 24,9% para a classe After, 9,7% para a classe Before e 1% para as demais classes de menor frequência.

Na Tabela 6.33, são apresentados os melhores hiperparâmetros selecionados para esse conjunto de regras, levando em consideração os resultados obtidos nos dados de validação. A ordem dos conjuntos individuais foi definida como a ordem decrescente da acurácia de cada conjunto de regras. O limite de corte de acurácia da regra foi definido como 80%, o que implica na remoção das regras com acurácia abaixo desse limite. Por fim, as regras do conjunto foram ordenadas por acurácia da regra.

A Tabela 6.34 apresenta os resultados dos experimentos realizados nos conjuntos de

Tabela 6.32: Elementos do conjunto de regras formado pela combinação de todos os conjuntos individuais.

Elementos do Conjunto de Regras	**Valor**
Quant. regras para a classe OVERLAP	631
Quant. regras para a classe BEFORE	95
Quant. regras para a classe AFTER	244
Quant. regras para a classe VAGUE	2
Quant. regras BEFORE-OR-OVERLAP	3
Quant. regras OVERLAP-OR-AFTER	5
Quantidade total de regras	980
Quantidade de condições	2.917
Média de condições por regra	3,0

Fonte: Elaborada pelo autor.

Tabela 6.33: Seleção dos melhores hiperparâmetros do conjunto de regras formado por todos os conjuntos individuais

Hiperparâmetros	**Valor**
Ordem para junção	acurácia de cada conjunto
Limite de corte pela acurácia	80%
Ordenação das regras	acurácia

Fonte: Elaborada pelo autor.

dados de treinamento e teste, analisando a aplicação do conjunto de regras resultante da combinação de todos os conjuntos individuais. A maior acurácia e *F1-score* obtidos nos dados de treinamento foi de 89,3% e 88,5%, respectivamente, quando a abordagem da primeira regra acionada foi utilizada. Nos dados de teste, os melhores resultados foram obtidos aplicando o sistema de votação, alcançando uma acurácia de 67,5% e *F1-score* de 63,3%. Esse conjunto de regras superou o desempenho do *baseline* tanto em acurácia quanto em *F1-score* nos dados de teste, porém não superou o desempenho do conjunto de regras gerado pelo algoritmo `RIPPER`.

Conforme observado na matriz de confusão dos dados de treinamento na Tabela 6.35, esse conjunto de regras é capaz de classificar todas as classes, incluindo as classes de menor frequência. No entanto, ao analisar as classes de menor frequência, observa-se que as regras para as classes BEFORE-OR-OVERLAP e OVERLAP-OR-AFTER alcançaram altas taxas de acerto, enquanto a classe VAGUE teve um baixo *recall* de 19%, ou seja, errou mais do que acertou. Em relação às classes de maior frequência, a classe BEFORE apresentou maior dificuldade, com uma quantidade maior de erros de classificação em comparação com as classes AFTER e OVERLAP.

Tabela 6.34: Resultados da aplicação do conjunto de regras formado pela combinação de todos os conjuntos individuais.

	Dados Treinamento		**Dados Teste**	
	Acurácia	**F1**	**Acurácia**	**F1**
Baseline	69,0	66,1	66,9	62,5
Primeira Regra	**89,3**	**88,5**	63,3	59,6
Votação	85,2	84,1	**67,5**	**63,3**

Fonte: Elaborada pelo autor.

Tabela 6.35: Matriz de confusão da melhor configuração da combinação de todos os conjuntos de regras individuais nos dados de treinamento (primeira regra acionada).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	326	0	34	0	0	0
	Before	13	180	86	0	0	0
	Overlap	0	2	765	0	0	0
	Before-or-Overlap	2	0	0	6	0	0
	Overlap-or-After	0	0	0	0	5	0
	Vague	1	2	14	0	0	4

Fonte: Elaborada pelo autor.

Ao analisar a matriz de confusão dos dados de teste apresentada na Tabela 6.36, fica evidente que nenhuma instância foi corretamente classificada nas classes de menor frequência. Essa falha de classificação ocorre principalmente devido à falta de dados rotulados adequados para essas classes específicas. Embora a combinação de todos os conjuntos individuais tenha sido capaz de classificar corretamente algumas instâncias dos dados de treinamento para essas classes, sua capacidade de generalização para dados não observados mostrou-se insuficiente.

Ao examinar os dados presentes nessa matriz de confusão dos dados de teste na Tabela 6.36, é possível observar que as classes de maior frequência, especialmente a classe After, apresentam um baixo *recall*, ou seja, um número reduzido de instâncias corretamente classificadas. Além disso, entre todas as classificações realizadas pelas regras dessa classe, a quantidade de erros foi maior do que a de acertos, resultando também em uma baixa precisão. Em relação à classe Before, apesar de ter uma precisão superior, o *recall* também foi baixo. Esses resultados sugerem que esse conjunto de regras enfrenta dificuldades em distinguir adequadamente entre as relações temporais Before e After.

Tabela 6.36: Matriz de confusão da melhor configuração da combinação de todos os conjuntos de regras individuais nos dados de teste (votação).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	11	0	19	0	0	0
	Before	4	6	10	0	0	0
	Overlap	9	1	97	0	0	0
	Before-or-Overlap	1	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	6	1	4	0	0	0

Fonte: Elaborada pelo autor.

6.2.2 Resultados da Aplicação do Conjunto de Regras Formado pela Combinação de Dois Conjuntos Individuais.

A Tabela 6.37 apresenta os principais elementos do conjunto de regras resultante da melhor combinação de dois conjuntos individuais. Esse conjunto consiste em um total de 797 regras, com um total de 2.378 condições, resultando em uma média de 3 condições por regra. Após a combinação, as regras foram distribuídas aproximadamente da seguinte forma: 65,6% para a classe Overlap, 25,8% para a classe After, 8,4% para a classe Before e apenas uma regra para a classe Vague.

Tabela 6.37: Elementos do conjunto de regras formado pela combinação dos conjuntos individuais gerados pelo `IDS` e `CBA`.

Elementos do Conjunto de Regras	**Valor**
Quant. regras para a classe Overlap	523
Quant. regras para a classe Before	67
Quant. regras para a classe After	206
Quant. regras para a classe Vague	1
Quantidade total de regras	797
Quantidade de condições	2378
Média de condições por regra	3,0

Fonte: Elaborada pelo autor.

Na Tabela 6.38, são apresentados os melhores hiperparâmetros selecionados para esse conjunto de regras, considerando os resultados obtidos nos dados de validação. Todas as combinações de pares formados pelos conjuntos de regras individuais foram testadas, e o melhor resultado foi obtido com os conjuntos gerados pelos algoritmos `IDS` e `CBA`. No

conjunto resultante, o limite de corte de acurácia da regra selecionado foi de 80%, o que implica na remoção das regras com acurácia abaixo desse limite. Por fim, a ordenação selecionada para as regras do conjunto foi a ordenação por acurácia.

Tabela 6.38: Seleção dos melhores hiperparâmetros do conjunto de regras formado pela combinação dos conjuntos individuais gerados pelo `IDS` e `CBA`.

Hiperparâmetros	**Valor**
Melhor combinação dos pares	`IDS` e `CBA`
Limite de corte pela acurácia	80%
Ordenação das regras	acurácia

Fonte: Elaborada pelo autor.

A Tabela 6.39 apresenta os resultados dos experimentos realizados nos conjuntos de dados de treinamento e teste, analisando a aplicação do conjunto de regras resultante da combinação dos conjuntos de regras gerados pelos algoritmos `IDS` e `CBA`. A maior acurácia e *F1-score* obtidos nos dados de treinamento foi de 88,5% e 87,2%, respectivamente, quando a abordagem da primeira regra acionada foi utilizada. Nos dados de teste, os melhores resultados foram obtidos aplicando o sistema de votação, alcançando uma acurácia de 68% e *F1-score* de 63,4%. Esse conjunto de regras também superou o desempenho do *baseline* tanto em acurácia quanto em *F1-score* nos dados de teste, porém não superou o desempenho do conjunto de regras gerado pelo algoritmo `RIPPER`.

Tabela 6.39: Resultados da aplicação do conjunto de regras formado pela combinação dos conjuntos individuais gerados pelo `IDS` e `CBA`.

	Dados Treinamento		**Dados Teste**	
	Acurácia	**F1**	**Acurácia**	**F1**
Baseline	69,0	66,1	66,9	62,5
Primeira Regra	**88,5**	**87,2**	65,7	62,0
Votação	86,8	85,6	**68,0**	**63,4**

Fonte: Elaborada pelo autor.

Ao analisar a matriz de confusão dos dados de treinamento na Tabela 6.40, nota-se que a classe Before foi a mais deficiente, com o menor *recall* (59,5%), principalmente devido às classificações errôneas realizadas pelas regras que identificam a classe Overlap. As classes After e Overlap apresentaram bons índices de acerto.

Ao examinar a matriz de confusão dos dados de teste na Tabela 6.41, observa-se que muitas classificações que deveriam pertencer às classes After (21) e Before (11) foram erroneamente atribuídas à classe Overlap. Esse problema pode ser resultado

Tabela 6.40: Matriz de confusão do conjunto de regras formado pela combinação dos conjuntos individuais gerados pelo `IDS` e `CBA` nos dados de treinamento (primeira regra acionada).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	331	0	30	0	0	0
	Before	16	166	97	0	0	0
	Overlap	0	1	775	0	0	0
	Before-or-Overlap	2	0	0	0	0	0
	Overlap-or-After	0	1	0	0	0	0
	Vague	1	2	16	0	0	2

Fonte: Elaborada pelo autor.

do desbalanceamento das classes, uma vez que a classe Overlap possui um número significativamente maior de instâncias em comparação com as outras classes. Quando isso ocorre, pode haver uma tendência de classificar erroneamente outras classes como a classe majoritária. Além disso, uma grande proporção de classificações realizadas pelas regras que identificam a classe After foi atribuída erroneamente a outras classes, resultando em baixa precisão.

Tabela 6.41: Matriz de confusão do conjunto de regras formado pela combinação dos conjuntos individuais gerados pelo `IDS` e `CBA` nos dados de teste (votação).

		Previsto					
		After	Before	Overlap	BorO	OorA	Vague
Real	After	9	0	21	0	0	0
	Before	2	7	11	0	0	0
	Overlap	8	0	99	0	0	0
	Before-or-Overlap	1	0	0	0	0	0
	Overlap-or-After	0	0	0	0	0	0
	Vague	6	1	4	0	0	0

Fonte: Elaborada pelo autor.

6.3 DISCUSSÕES

No presente estudo, a Tabela 6.42 exibe um resumo dos resultados dos conjuntos de regras analisados neste capítulo, com base nos dados de teste. Ao examinar os resultados,

observa-se que o conjunto de regras gerado pelo algoritmo `RIPPER` alcançou o melhor desempenho, com uma taxa de acurácia de 69,2% e um valor de *F1-score* de 66,1%. Esses resultados sugerem a superioridade desse conjunto de regras em relação aos demais conjuntos analisados.

Tabela 6.42: Resultados de todos os conjuntos de regras com base nos dados de teste.

Conjunto de Regras	**Forma de Aplicar**	**Acurácia**	**F1**
Baseline	-	66,9	62,5
Manuais	Ambos	66,9	58,1
`CBA`	Votação	62,7	60,5
`CN2`	Primeira Regra	65,7	61,3
`IDS`	Votação	66,9	60,7
`RIPPER`	Votação	**69,2**	**66,1**
Combinação de Todos	Votação	67,5	63,3
Combinação `IDS` e `CBA`	Votação	68,0	63,4

Fonte: Elaborada pelo autor.

Em segundo lugar, encontra-se o conjunto de regras formado pela combinação dos conjuntos individuais gerados pelos algoritmos `IDS` e `CBA`, que alcançou uma taxa de acurácia de 68% e um valor de *F1-score* de 63,4%. Esses resultados indicam um desempenho satisfatório, embora inferior ao conjunto de regras gerado pelo algoritmo `RIPPER`.

Em terceiro lugar, está o conjunto de regras formado pela combinação de todos os conjuntos individuais, com uma taxa de acurácia de 67,5% e um valor de *F1-score* de 63,3%. Embora esse conjunto tenha obtido resultados próximos aos do segundo conjunto, sua performance é ligeiramente inferior.

Além disso, com base nesses resultados, os conjuntos de regras com melhor desempenho superaram o *baseline* tanto em acurácia quanto em *F1-score*, sugerindo sua eficácia em relação ao método de referência.

Para confirmar a significância estatística dos resultados obtidos, foi utilizada a análise de variância (ANOVA) de um fator (SNEDECOR; COCHRAN, 1989) e o teste de comparação múltipla de Tukey (TUKEY, 1953), com um nível de significância de 0,05. Essas análises visam determinar se existem diferenças estatisticamente significativas entre as médias do *F1-score* dos conjuntos de regras.

Para isso, foram construídos 10 subconjuntos a partir da divisão dos dados de teste em 10 partes iguais, excluindo-se uma parte diferente para cada subconjunto. Em seguida, cada conjunto de regras foi aplicado em cada um dos subconjuntos, obtendo-se 10 resultados diferentes para a métrica *F1-score* em cada conjunto de regras. O *F1-score* médio e a variância são apresentados na Tabela 6.43. Observa-se que não houve alterações entre o *F1-score* médio e os *F1-score* finais apresentados na Tabela 6.42.

Os testes estatísticos ANOVA e Tukey foram aplicados para verificar se existem diferenças significativas entre as médias dos experimentos, assumindo certas condições, tais

Tabela 6.43: Desempenho Médio (F1) e Variância das Subamostras de Teste para Cada Conjunto de Regra

	F1 Médio	**Variância**
MANUAIS	58,1	0,0002615
CBA	60,5	0,0004487
IDS	60,7	0,0003010
CN2	61,3	0,0002837
TODOS	63,3	0,0004397
IDS+CBA	63,4	0,0004168
RIPPER	66,1	0,0004014

Fonte: Elaborada pelo autor.

como a distribuição normal dos dados e a homogeneidade das variâncias. A normalidade dos dados foi avaliada por meio dos testes de Shapiro-Wilk (SHAPIRO; WILK, 1965) e Lilliefors (LILLIEFORS, 1967), que indicaram que não há evidências suficientes para rejeitar a hipótese de normalidade em cada experimento (valor-p médio de 0,561 e 0,557, respectivamente).

Para verificar a homogeneidade das variâncias, utilizou-se o teste de Bartlett (BARTLETT, 1937) e o teste de Levene (LEVENE et al., 1960), sendo este último menos sensível às violações da suposição de normalidade. Os resultados mostraram que não há evidências suficientes para rejeitar a hipótese de homogeneidade das variâncias das amostras dos experimentos (p = 0,971 e p = 0,991, respectivamente).

Atendendo a essas condições, foi realizado o teste ANOVA para determinar se há diferenças significativas entre as médias dos experimentos. O valor-p encontrado foi inferior ao nível de significância de 0,05, indicando que há diferença significativa (p = 2,08E-11). Com base nesse resultado, aplicou-se o teste de comparação múltipla de Tukey para identificar quais pares de experimentos têm médias significativamente diferentes, controlando o risco global de erro tipo I, ou seja, rejeitar erroneamente a hipótese nula (igualdade das médias) quando ela é verdadeira.

Os resultados são apresentados na Tabela 6.44. Valores de p inferiores ao nível de significância de 0,05 são destacados em cinza, indicando diferença estatisticamente significativa entre as médias. Valores superiores indicam que não há evidência suficiente para rejeitar a igualdade das médias.

Com base no teste de Tukey, as comparações entre os experimentos revelaram diferença estatisticamente significativa entre as médias do conjunto de regras gerado pelo algoritmo RIPPER e todos os outros conjuntos de regras, exceto o conjunto formado pela combinação dos algoritmos IDS e CBA, que apresentou valor-p de 0,055, indicando uma diferença sem significância estatística em relação ao RIPPER. Esse resultado fornece evidências estatísticas que confirmam o melhor desempenho global do conjunto de regras gerado pelo algoritmo RIPPER, inclusive superando o método de referência, conforme visualizado na Figura 6.1.

Tabela 6.44: Resultados do Teste de Tukey para Comparação Múltipla de Médias com Nível de Significância de 0.05 (valores de p)

	CBA	CN2	IDS	RIPPER	IDS+CBA	TODOS
MANUAIS	0,1005	0,0098	0,0722	1,58E-11	2,61E-06	4,35E-06
CBA	-	0,9762	1,0000	9,89E-07	0,0340	0,0481
CN2		-	0,9903	2,78E-05	0,2484	0,3136
IDS			-	1,71E-06	0,0490	0,0683
RIPPER				-	0,0550	0,0391
IDS+CBA					-	1,0000

Fonte: Elaborada pelo autor.

Observa-se também que o conjunto de regras formado pela combinação dos algoritmos IDS e CBA apresentou diferença significativa em relação aos conjuntos de regras gerados pelo CBA (p = 0,034), IDS (p = 0,049) e regras Manuais (p = 0,034). Já o conjunto formado pela combinação de todos os conjuntos individuais possui médias estatisticamente diferentes do conjunto de regras Manuais (p = 4,35E-06) e CBA (p = 0,048).

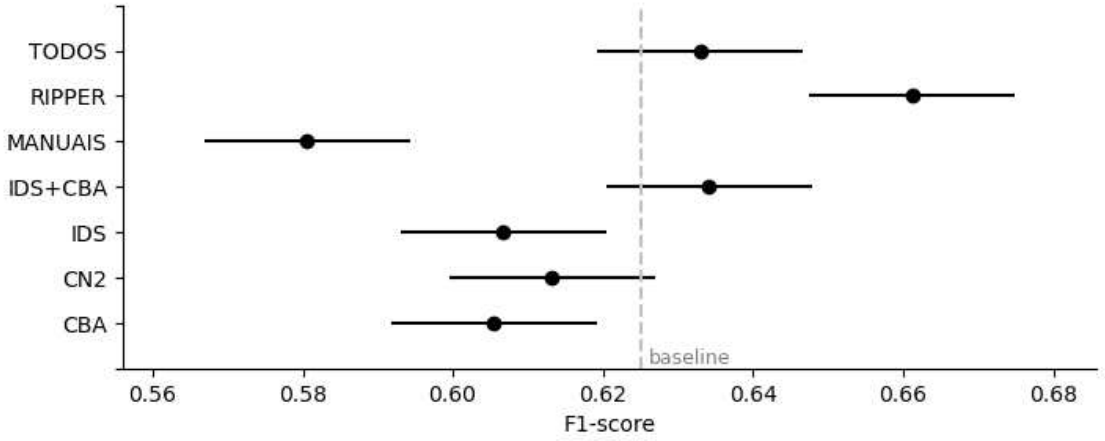


Figura 6.1: Comparação Simultânea de Médias pelo Teste de Tukey (0,05)
Fonte: Elaborada pelo autor.

A verificação das diferenças significativas entre as médias do *F1-score* das amostras também foi realizada por meio do teste ANOVA em todos os pares de experimentos. Os resultados indicaram que não havia evidências suficientes para rejeitar a hipótese de igualdade das médias apenas nos pares CBA e CN2 (p = 0,398), CBA e IDS (p = 0,892), CN2 e IDS (p = 0,426), e Combinação de Todos e Combinação de IDS+CBA (p = 0,905). Todos os outros pares apresentaram diferenças estatisticamente significativas.

Para comparar nossos resultados com o *baseline* estabelecido, foi aplicado o teste t de uma amostra, que é um teste de comparação única para avaliar se a média do experimento

é significativamente diferente do valor de referência. Os resultados indicaram que a média do conjunto de regras gerado pelo RIPPER difere significativamente do valor de referência (p = 0,00041), sugerindo que as diferenças são altamente improváveis de ocorrerem ao acaso e fornecendo evidências sólidas da superioridade desse conjunto de regras em relação ao *baseline*.

Diferenças significativas nas médias também foram encontradas nos conjuntos de regras gerados pelos algoritmos CBA (p = 0,02168) e IDS (p = 0,01141), bem como no conjunto de regras manuais (p = 0,000017), em relação ao *baseline*, indicando a superioridade do método de referência. Por outro lado, não há evidências suficientes para rejeitar a hipótese de igualdade das médias nos conjuntos de regras CN2 (p = 0,06555), Combinação de Todos (p = 0,28301) e Combinação de IDS+CBA (p = 0,21092), sugerindo que suas médias não diferem significativamente do *baseline*. Esses resultados foram confirmados pelo teste de Tukey, como pode ser visualizado na Figura 6.1, onde as igualdades são confirmadas onde há interseção com a linha que representa o *baseline*, e as diferenças significativas são indicadas nos conjuntos de regras em que não há interseção.

A análise dos resultados fornece evidências estatísticas que confirmam o melhor desempenho global do conjunto de regras produzido pelo algoritmo RIPPER, inclusive superando o método de referência. Essa constatação valida a eficácia desse conjunto de regras para identificar tipos de relações temporais *event-time*.

Em relação à forma de aplicar os conjuntos de regras, é relevante destacar que a abordagem mais eficaz foi a do sistema de votação, conforme demonstrado na Tabela 6.42. Nessa abordagem, todos os pares *event-time* são submetidos a todas as regras do conjunto, atribuindo-se votos para cada classe acionada em cada par processado. A classe mais frequente é atribuída ao par *event-time*. Essa abordagem mostrou-se mais eficaz em comparação com a abordagem da primeira regra acionada, que obteve melhor desempenho apenas para o conjunto de regras gerado pelo algoritmo CN2.

Além disso, é importante ressaltar que a hipótese estabelecida neste trabalho, a qual afirmava que a combinação de conjuntos de regras induzidas por algoritmos diferentes resultaria em resultados superiores em comparação com os resultados individuais de cada conjunto de regras na identificação de tipos de relações temporais em língua portuguesa, não foi confirmada pela análise dos resultados obtidos. O conjunto de regras gerado pelo algoritmo RIPPER obteve o melhor desempenho, superando as combinações de conjuntos de regras. Embora as combinações tenham alcançado a segunda e terceira posição em termos de desempenho, o fato de um conjunto de regras individual superar essas combinações indica que a hipótese não foi confirmada.

Portanto, com base nos resultados obtidos, não se pode afirmar que a combinação de conjuntos de regras induzidas por algoritmos diferentes produza consistentemente resultados superiores em comparação com os resultados individuais de cada conjunto de regras na tarefa de identificação de tipos de relações temporais em língua portuguesa. Esse resultado aponta para a necessidade de revisar ou reformular a hipótese inicialmente proposta para investigações futuras nessa área de estudo.

Por fim, a Tabela 6.45 apresenta a cobertura dos conjuntos de regras sem a utilização da classe majoritária como classe padrão. Os valores demonstram que, mesmo sem a classe padrão, as regras geradas pelos algoritmos de aprendizagem de regras foram capazes de

abranger uma parte significativa das instâncias nos conjuntos de dados de treinamento e teste.

Tabela 6.45: Cobertura dos conjuntos de regras se não fosse aplicada a classe padrão.

Conjunto de Regras	**Treinamento**	**Teste**
`CN2`	99,7	98,8
Combinação de Todos	94,6	96,4
`CBA`	97,8	94,7
Combinação `IDS` e `CBA`	89,4	94,1
`RIPPER`	92,3	93,5
`IDS`	65,8	78,1

Fonte: Elaborada pelo autor.

A inclusão da classe majoritária como classe padrão é uma estratégia importante para lidar com as instâncias que não são abrangidas por regras específicas. Essas instâncias podem ser consideradas como casos atípicos ou casos difíceis, e a atribuição da classe majoritária permite classificá-las de forma coerente.

Ao adicionar a classe padrão, o sistema de regras torna-se mais abrangente e robusto, pois é capaz de classificar instâncias desconhecidas que não foram abrangidas por regras específicas. Isso melhora a capacidade de generalização do sistema e torna sua classificação mais coerente e confiável.

Portanto, a cobertura dos conjuntos de regras sem a utilização da classe majoritária como classe padrão destaca a eficácia desses conjuntos em abranger uma parte considerável das instâncias. Por outro lado, a estratégia de atribuição da classe majoritária como padrão fortalece a abrangência do sistema, permitindo classificar de forma coerente as instâncias não abrangidas por regras específicas. Essa combinação de regras específicas e classe padrão contribui para a robustez e eficácia do sistema de classificação baseado em regras.

6.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais resultados dos experimentos realizados nos conjuntos de dados de treinamento e teste para a tarefa de identificação de tipos de relação temporal *event-time* dentro da mesma sentença, utilizando métricas de avaliação como acurácia e *F1-score*, e matriz de confusão. O objetivo foi analisar o desempenho da aplicação dos sete conjuntos de regras estudados.

Os resultados obtidos demonstraram que o método baseado em regras proposto foi eficaz na identificação de tipos de relações temporais em textos em língua portuguesa. O conjunto de regras gerado pelo algoritmo `RIPPER` mostrou o melhor desempenho geral, superando os conjuntos combinados de regras. Esses resultados indicam que, no contexto específico deste trabalho, a hipótese de que a combinação de conjuntos de regras induzidas por algoritmos diferentes resultaria em resultados superiores não foi confirmada.

Além disso, é importante destacar que o conjunto de regras gerado pelo algoritmo `RIPPER` apresentou um desempenho estatisticamente superior em relação ao *baseline*, o que valida a superioridade desse conjunto de regras na identificação de relações temporais. Esse resultado indica que as regras geradas para esse conjunto possuem *features* adicionais relevantes e demonstram uma capacidade de generalização eficaz.

Entre as abordagens de aplicação dos conjuntos de regras em novos dados, o sistema de votação mostrou-se a mais eficaz. Essa abordagem, que considera todas as regras para cada par *event-time*, proporcionou resultados superiores em relação à abordagem da primeira regra acionada. É ainda importante destacar que a aplicação do fechamento temporal não apresentou diferenças em comparação aos resultados sem sua utilização.

A inclusão da classe majoritária como classe padrão fortaleceu a abrangência e robustez do sistema de classificação baseado em regras. Essa estratégia permitiu classificar de forma coerente as instâncias não cobertas por regras específicas, contribuindo para a capacidade de generalização do sistema.

Em suma, os resultados dos experimentos indicam que o método baseado em regras, especialmente o conjunto de regras gerado pelo algoritmo `RIPPER`, é eficaz na identificação de tipos de relação temporal *event-time* em textos em língua portuguesa. A abordagem de votação mostrou-se a mais adequada para aplicar os conjuntos de regras em novos dados, enquanto a inclusão da classe majoritária fortaleceu a capacidade de generalização do sistema. Esses resultados contribuem para o avanço da área de processamento de linguagem natural e têm potenciais aplicações em diferentes domínios que envolvam a análise de relações temporais em textos.

No entanto, é importante ressaltar que a avaliação de desempenho de conjuntos de regras é um processo complexo e depende de diversos fatores, como o tamanho e a qualidade do conjunto de dados, a escolha das métricas de avaliação e a configuração dos algoritmos. Portanto, é recomendável que esses resultados sejam interpretados com cautela e considerados em conjunto com outras evidências e estudos relacionados.

Capítulo 7

CONCLUSÕES

No presente estudo, foi desenvolvido um método computacional para a identificação de tipos de relações temporais entre evento e expressão temporal (*event-time*) em textos em português. Ao abordar esse problema específico, exploramos técnicas de aprendizagem de regras, sendo este o primeiro trabalho conhecido a utilizar essa abordagem para resolver essa tarefa.

Nossa abordagem se baseou na criação de um conjunto abrangente de *features* contendo informações linguísticas relevantes. Essas *features* foram utilizadas na construção de conjuntos de regras, empregando algoritmos de aprendizagem de regras como `CBA`, `CN2`, `IDS` e `RIPPER`, além de um conjunto de regras manuais. Os conjuntos de regras foram aplicados individualmente aos pares *event-time*, assim como em combinação. Duas estratégias de aplicação foram adotadas: a primeira regra acionada e um sistema de votação.

Os resultados obtidos demonstraram a eficácia da nossa abordagem baseada em regras, com destaque para o conjunto de regras gerado pelo algoritmo `RIPPER`, que obteve o melhor desempenho. Esse conjunto de regras alcançou uma acurácia de 69,2% e um *F1-score* de 66,1%, superando o método de referência com uma diferença estatisticamente significativa (p = 4,17E-04). Observou-se um aumento absoluto de 2,3 pontos percentuais em acurácia e 3,6 pontos percentuais em *F1-score* nos dados de teste. Além disso, verificou-se que a aplicação dos conjuntos de regras pelo sistema de votação foi mais eficaz em dados não vistos.

A diferença significativa entre o desempenho dos conjuntos de regras e o *baseline* utilizado ressalta a importância das *features* adicionais empregadas pelos conjuntos de regras na identificação das relações temporais. Essas *features* forneceram informações complementares e permitiram uma análise mais abrangente e precisa dos dados.

A capacidade de generalização dos conjuntos de regras é outro aspecto relevante. Eles conseguiram capturar padrões e regularidades nos dados que podem ser aplicados a novas instâncias, possibilitando previsões precisas. Isso demonstra a utilidade e eficácia dos conjuntos de regras como uma abordagem robusta para lidar com a complexidade das relações temporais em um contexto linguístico.

7.1 CONTRIBUIÇÕES

Esta dissertação apresentou uma série de contribuições significativas para o avanço das aplicações de processamento de linguagem natural, fornecendo uma compreensão aprimorada e explicável das relações temporais em textos em português. Dentre as principais contribuições, destacam-se:

1. **Conjunto Abrangente de *Features*:** Para a construção dos conjuntos de regras, foi elaborado um conjunto abrangente de *features* que incorpora informações linguísticas relevantes e diversificadas. Essas *features* revelaram-se fundamentais para a precisão e abrangência na identificação das relações temporais, permitindo análises mais completas dos dados.

2. **Conjuntos de Regras Interpretáveis:** Os conjuntos de regras gerados pelos algoritmos de aprendizagem de regras e as regras manuais são facilmente interpretáveis, proporcionando uma compreensão clara dos critérios utilizados para identificar diferentes tipos de relações temporais em português. Essa interpretabilidade é crucial para a confiança e aplicabilidade prática da abordagem.

Com base nessas contribuições, a pesquisa avança no entendimento e tratamento das relações temporais em textos e proporciona uma base sólida para futuras investigações nesse campo. Ao continuar refinando e expandindo essa pesquisa, busca-se desvendar novas possibilidades para a compreensão temporal em textos e promover avanços contínuos nas aplicações práticas do processamento de linguagem natural.

7.2 TRABALHOS FUTUROS

Este estudo proporcionou *insights* valiosos sobre a identificação de relações temporais em textos em português, mas também revelou algumas direções promissoras para trabalhos futuros. Dentre as possíveis extensões e aprimoramentos da pesquisa, destacam-se:

1. **Ampliação dos Dados Anotados:** Uma limitação deste estudo reside na escassez de dados anotados em português. No entanto, essa circunstância se apresenta como uma perspectiva promissora para pesquisas futuras. Nesse contexto, é recomendável proceder à anotação dos tipos das relações temporais presentes no *corpus* TimeBankPT, fazendo uso da nossa ferramenta de identificação de tipos de relações temporais, a qual é automatizada, embora não seja perfeita. Tal procedimento permitirá a elaboração de processos de anotação de dados de forma semiautomática, contribuindo assim para o aprimoramento da eficácia e da capacidade de generalização inerentes à abordagem proposta.

2. **Relações Temporais entre Eventos e Data de Criação do Documento:** O estudo abordou a identificação de tipos de relações temporais entre eventos e expressões temporais. No entanto, trabalhos futuros podem explorar a possibilidade de estender a abordagem para identificar tipos de relações temporais entre eventos e a data de criação do documento. Isso pode ser relevante para enriquecer a análise temporal dos textos.

3. **Relações Temporais entre Pares de Eventos:** Outro aspecto a ser explorado é a identificação de tipos de relações temporais estabelecidas entre pares de eventos presentes em frases distintas. Essa extensão da abordagem permitirá uma compreensão ainda mais abrangente da organização temporal das informações nos textos.

4. **Ampliar as Formas de Aplicação das Regras:** Este estudo investigou duas modalidades distintas para aplicar as regras nos conjuntos de dados. A primeira forma consiste em considerar a classe da primeira regra acionada como a classe final atribuída ao par *event-time*. A segunda forma adota uma abordagem de votação por maioria. Não obstante, trabalhos futuros podem abarcar a exploração de outras maneiras para a aplicação das regras. Por exemplo, é possível conceber um sistema de votação ponderado, no qual métricas, como a acurácia ou o *F1-score* de cada regra acionada, são utilizadas para influenciar o peso dos votos. A incorporação dessa abordagem pode contribuir para aprimorar o método e potencialmente levar a resultados mais consistentes.

5. **Análise Qualitativa:** Este estudo conduziu uma análise quantitativa dos resultados obtidos nos experimentos. Entretanto, uma abordagem qualitativa, acompanhada por uma análise minuciosa de um linguista, visando validar as regras com base nos dados e no conhecimento linguístico, representaria uma contribuição substancial para o enriquecimento da nossa abordagem explicável das relações temporais. Tal abordagem permitiria a identificação de padrões e princípios gerais relacionados às relações temporais, além de potencialmente inspirar futuras pesquisas na área da linguística, promovendo uma compreensão mais profunda e fundamentada desses fenômenos linguísticos complexos.

6. **Avaliação de Outras Técnicas de Aprendizagem de Regras:** Embora os conjuntos de regras tenham se mostrado eficazes nesta dissertação, futuros estudos podem avaliar outras técnicas de aprendizagem de regras ou até mesmo explorar abordagens híbridas que possam melhorar ainda mais o desempenho na identificação das relações temporais.

7. **Comparação com Técnicas de Aprendizado de Máquina:** Uma linha de pesquisa importante para trabalhos futuros consiste em realizar comparações detalhadas entre a técnica baseada em regras apresentada nesta dissertação e as técnicas de aprendizado de máquina, como redes neurais e modelos pré-treinados. Essa comparação deve abranger aspectos como desempenho, interpretabilidade, necessidade de dados anotados e aplicabilidade em contextos multilíngues.

As descobertas desta dissertação têm implicações práticas significativas em diversas áreas do processamento de linguagem natural. Entre elas, destacam-se a descrição de cenas, compreensão de histórias, resumo de documentos, representação da estrutura temporal de prontuários médicos e análise de notícias.

Em resumo, este trabalho apresentou uma abordagem baseada em regras para a identificação de tipos de relações temporais em textos em português. Embora os resultados

obtidos tenham sido promissores, o campo ainda oferece inúmeras oportunidades para pesquisas futuras. Espera-se que este trabalho inspire e motive novos estudos nessa área, contribuindo para avanços contínuos e para uma melhor compreensão do processamento de informações temporais em textos em português.

REFERÊNCIAS BIBLIOGRÁFICAS

ACE. *ACE (Automatic Content Extraction) English Annotation Guidelines for Events*. 5.4.3 2005.07.01. ed. [S.l.], 2005. Disponível em: <https://www.bibsonomy.org/bibtex/202c002e12b4c5abf91b96d2865227ae2/jnothman>.

AGRAWAL, R.; IMIELIŃSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. [S.l.: s.n.], 1993. p. 207–216.

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: SANTIAGO, CHILE. *Proc. 20th int. conf. very large data bases, VLDB*. [S.l.], 1994. v. 1215, p. 487–499.

ALLEN, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, ACM New York, NY, USA, v. 26, n. 11, p. 832–843, 1983.

ASADI, S.; SHAHRABI, J. Ripmc: Ripper for multiclass classification. *Neurocomputing*, v. 191, p. 19–33, 2016. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231216000576>.

BACH, E. The algebra of events. *Linguistics and Philosophy*, v. 9, p. 5–16, 02 1986.

BARTLETT, M. S. Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, The Royal Society London, v. 160, n. 901, p. 268–282, 1937.

BETHARD, S.; MARTIN, J. H. Cu-tmp: Temporal relation classification using syntactic and semantic features. In: *Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007)*. [S.l.: s.n.], 2007. p. 129–132.

BOLL, E. M.; CLAIR, D. C. S. Analysis of rule sets generated by the cn2, id3, and multiple convergence symbolic learning methods. In: *Proceedings of the 1995 ACM 23rd annual conference on Computer science*. [S.l.: s.n.], 1995. p. 48–55.

BRIN, S. et al. Dynamic itemset counting and implication rules for market basket data. In: *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. [S.l.: s.n.], 1997. p. 255–264.

BRITSCH, M.; GAGUNASHVILI, N.; SCHMELLING, M. Application of the rule-growing algorithm ripper to particle physics analysis. *arXiv preprint arXiv:0910.1729*, 2009.

CENDROWSKA, J. Prism: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, v. 27, n. 4, p. 349–370, 1987. ISSN 0020-7373. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020737387800032>.

CHAMBERS, N. *Navytime: Event and time ordering from raw text*. [S.l.], 2013.

CHAMBERS, N. et al. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, MIT Press, v. 2, p. 273–284, 2014.

CHANG, A. X.; MANNING, C. D. Sutime: A library for recognizing and normalizing time expressions. In: *Lrec*. [S.l.: s.n.], 2012. v. 3735, p. 3740.

CHITICARIU, L.; LI, Y.; REISS, F. Rule-based information extraction is dead! long live rule-based information extraction systems! In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. [S.l.: s.n.], 2013. p. 827–832.

CLARK, P.; BOSWELL, R. Rule induction with cn2: Some recent improvements. In: SPRINGER. *European Working Session on Learning*. [S.l.], 1991. p. 151–163.

CLARK, P.; NIBLETT, T. The cn2 induction algorithm. *Machine learning*, Springer, v. 3, n. 4, p. 261–283, 1989.

COHEN, W. W. Fast effective rule induction. In: *Machine learning proceedings 1995*. [S.l.]: Elsevier, 1995. p. 115–123.

COLLOBERT, R. et al. Natural language processing (almost) from scratch. *Journal of machine learning research*, v. 12, n. ARTICLE, p. 2493–2537, 2011.

COSTA, F.; BRANCO, A. Timebankpt: A timeml annotated corpus of portuguese. In: *LREC*. [S.l.: s.n.], 2012. v. 12, p. 3727–3734.

COSTA, F. N. Q. M. C. *Processing Temporal Information in Unstructured Documents*. Tese (Doutorado) — Universidade de Lisboa (Portugal), 2012. Disponível em: <https://repositorio.ul.pt/handle/10451/8639>.

CUNHA-FILHO, O. A. L.; ROCHA-JUNIOR, J. B. Encontrando regras de associação sem especificar suporte mínimo e confiança mínima. In: SBC. *Anais Estendidos do XXXVII Simpósio Brasileiro de Bancos de Dados*. [S.l.], 2022. p. 168–174.

DERCZYNSKI, L.; GAIZAUSKAS, R. A corpus-based study of temporal signals. *arXiv preprint arXiv:1203.5066*, 2012.

DERCZYNSKI, L. R. *Automatically ordering events and times in text*. [S.l.]: Springer, 2017.

D'SOUZA, J. *Extracting Time and Space Relations from Natural Language Text*. Tese (Doutorado) — The University of Texas at Dallas, ago. 2015. Disponível em: <https://doi.org/10.13140/RG.2.2.20018.89288>.

FEIGE, U.; MIRROKNI, V. S.; VONDRáK, J. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, v. 40, n. 4, p. 1133–1153, 2011. Disponível em: <https://doi.org/10.1137/090779346>.

FILIP, J.; KLIEGR, T. *Classification based on associations (CBA) - a performance analysis*. [S.l.], 2018.

FILIP, J.; KLIEGR, T. Pyids-python implementation of interpretable decision sets algorithm by lakkaraju et al, 2016. In: *RuleML+ RR (Supplement)*. [S.l.: s.n.], 2019.

FÜRNKRANZ, J.; GAMBERGER, D.; LAVRAČ, N. *Foundations of rule learning*. [S.l.]: Springer Science & Business Media, 2012.

FÜRNKRANZ, J.; KLIEGR, T. A brief overview of rule learning. In: SPRINGER. *International symposium on rules and rule markup languages for the semantic web*. [S.l.], 2015. p. 54–69.

FüRNKRANZ, J.; WIDMER, G. Incremental reduced error pruning. In: COHEN, W. W.; HIRSH, H. (Ed.). *Machine Learning Proceedings 1994*. San Francisco (CA): Morgan Kaufmann, 1994. p. 70–77. ISBN 978-1-55860-335-6. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9781558603356500179>.

GUIZZARDI, G. et al. Towards ontological foundations for the conceptual modeling of events. In: NG, W.; STOREY, V. C.; TRUJILLO, J. C. (Ed.). *Conceptual Modeling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 327–341. ISBN 978-3-642-41924-9.

HASANPOUR, H.; MEIBODI, R. G.; NAVI, K. Improving rule-based classification using harmony search. *PeerJ Computer Science*, PeerJ Inc., v. 5, p. e188, 2019. Disponível em: <https://peerj.com/articles/cs-188.pdf>.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The elements of statistical learning: data mining, inference, and prediction*. [S.l.]: Springer, 2010.

HOVY, E. et al. Ontonotes: the 90% solution. In: *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*. [S.l.: s.n.], 2006. p. 57–60.

HÜHN, J.; HÜLLERMEIER, E. Furia: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, Springer, v. 19, p. 293–319, 2009.

JAYNES, E. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, v. 70, n. 9, p. 939–952, 1982.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. [S.l.]: Prentice-Hall, 2000.

KLIEGR, T.; KUCHAR, J. Tuning hyperparameters of classification based on associations (cba). In: *ITAT*. [s.n.], 2019. p. 9–16. Disponível em: <https://ceur-ws.org/Vol-2473/paper8.pdf>.

KLÖSGEN, W. Explora: A multipattern and multistrategy discovery assistant. In: _____. *Advances in Knowledge Discovery and Data Mining*. USA: American Association for Artificial Intelligence, 1996. p. 249–271. ISBN 0262560976.

KOLYA, A. K. et al. Ju_cse: A crf based approach to annotation of temporal expression, event and temporal relations. In: *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. [S.l.: s.n.], 2013. p. 64–72.

KOLYAL, A.; EKBAL, A.; BANDYOPADHYAY, S. Using voting approach for event extraction and event-dct, event-time relation identification. *International Journal of Artificial Intelligence and Applications*, v. 4, p. 65–83, 01 2013.

KONONENKO, I.; BRATKO, I.; ROSKAR, E. Experiments in automatic learning of medical diagnostic rules (tech. rep.). *Ljubjana, Yogoslavia: Jozef Stefan Institute*, 1984.

KUMBHARE, T. A.; CHOBE, S. V. An overview of association rule mining algorithms. *International Journal of Computer Science and Information Technologies*, Citeseer, v. 5, n. 1, p. 927–930, 2014.

KUMI, S.; LIM, C.; LEE, S.-G. Malicious url detection based on associative classification. *Entropy*, v. 23, n. 2, 2021. ISSN 1099-4300. Disponível em: <https://www.mdpi.com/1099-4300/23/2/182>.

LAKKARAJU, H.; BACH, S. H.; LESKOVEC, J. Interpretable decision sets: A joint framework for description and prediction. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2016. p. 1675–1684.

LEVENE, H. et al. Contributions to probability and statistics. *Essays in honor of Harold Hotelling*, v. 278, p. 292, 1960.

LILLIEFORS, H. W. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, Taylor & Francis, v. 62, n. 318, p. 399–402, 1967. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1967.10482916>.

LIN, Z.; NG, H. T.; KAN, M.-Y. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, Cambridge University Press, v. 20, n. 2, p. 151–184, 2014.

LIU, B.; HSU, W.; MA, Y. Integrating classification and association rule mining. In: *Proceedings of the fourth international conference on knowledge discovery and data mining*. [S.l.: s.n.], 1998. p. 80–86.

MANI, I.; PUSTEJOVSKY, J.; GAIZAUSKAS, R. *The Language of Time: A Reader*. [S.l.]: Oxford University Press UK, 2005.

MARSIC, G. *Temporal Processing of News: Annotation of Temporal Expressions, Verbal Events and Temporal Relations*. Tese (Doutorado) — University of Wolverhampton, 2011. Disponível em: <http://hdl.handle.net/2436/209933>.

MARTIN, B. *INSTANCE-BASED LEARNING: Nearest Neighbour with Generalisation*. Tese (Doutorado) — University of Waikato Hamilton, New Zealand, 1995.

MICHALSKI, R. S. On the quasi-minimal solution of the general covering problem. In: *Proceedings of the 5th International Symposium on Information Processing*. [s.n.], 1969. v. 3, p. 125–128. Disponível em: <http://ebot.gmu.edu/bitstream/handle/1920/1507/69-02.pdf?sequence=2&isAllowed=y>.

MILLER, G. A. Wordnet: A lexical database for english. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 38, n. 11, p. 39–41, nov 1995. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/219717.219748>.

MIRZA, P.; TONELLI, S. Classifying temporal relations with simple features. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. [S.l.: s.n.], 2014. p. 308–317.

MOSCOVITZ, I. *How to Perform Explainable Machine Learning Classification - Without Any Trees*. 2019. <https://towardsdatascience.com/how-to-perform-explainable-machine-learning-classification-without-any-trees-873db4192c68>.

MURPHY, K. P. *Machine learning: a probabilistic perspective*. [S.l.]: MIT press, 2012.

PAGALLO, G.; HAUSSLER, D. Boolean feature discovery in empirical learning. *Machine learning*, Springer, v. 5, p. 71–99, 1990.

PALANISAMY, S. K. *Association rule based classification*. Tese (Doutorado) — Worcester Polytechnic Institute Worcester, 2006.

PORTNER, P. The (temporal) semantics and (modal) pragmatics of the perfect. *Linguistics and philosophy*, Springer, v. 26, p. 459–510, 2003.

PROVOST, J. Naıve-bayes vs. rule-learning in classification of email. *University of Texas at Austin*, 1999.

PUSTEJOVSKY, J. et al. The timebank corpus. *Proceedings of Corpus Linguistics*, 01 2003.

PUSTEJOVSKY, J. et al. *The Specification Language TimeML*. [S.l.]: Citeseer, 2004.

PUSTEJOVSKY, J. et al. Iso-timeml: An international standard for semantic annotation. In: *LREC*. [S.l.: s.n.], 2010. v. 10, p. 394–397.

QUINLAN, J. R. Learning efficient classification procedures and their application to chess end games. In: MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. (Ed.). *Machine Learning*. San Francisco (CA): Morgan Kaufmann, 1983. p. 463–482. ISBN 978-0-08-051054-5. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780080510545500194>.

QUINLAN, J. R. Learning logical definitions from relations. *Machine learning*, Springer, v. 5, p. 239–266, 1990.

QUINLAN, J. R. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 1558602402.

QUIRK, R.; GREENBAUM, S. A university grammar of english, 1973. *Harlow: Addison Wesley Longman Limited*, p. 285–287, 1972. Disponível em: <https://xn--webducation-dbb.com/wp-content/uploads/2020/11/Randolph-Quirk-A-University-Grammar-of-English-Longman-1973.pdf>.

RAI, V. K.; CHAKRABORTY, S.; CHAKRABORTY, S. Association rule mining for prediction of covid-19. *Decision Making: Applications in Management and Engineering*, v. 6, n. 1, p. 365–378, Apr. 2023. Disponível em: <https://dmame-journal.org/index.php/dmame/article/view/319>.

RAJAK, A.; GUPTA, M. K. Association rule mining: applications in various areas. In: *Proceedings of international conference on data management, Ghaziabad, India*. [S.l.: s.n.], 2008. p. 3–7.

REICHENBACH, H. Elements of symbolic logic. 1947.

RIVEST, R. L. Learning decision lists. *Sponsored Paper, NSF Grant DCR-8607494, ARO Grant DAAL03-86-K-0171 ve Siemens Corporation*, 2001.

SASAKI, M.; KITA, K. Rule-based text categorization using hierarchical categories. In: *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*. [S.l.: s.n.], 1998. v. 3, p. 2827–2830 vol.3.

SAURI, R. et al. *TimeML Annotation Guidelines Version 1.2.1*. [S.l.]: January, 2006.

SEERAT, B.; QAMAR, U. Rule induction using enhanced ripper algorithm for clinical decision support system. In: *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*. [S.l.: s.n.], 2015. p. 83–91.

SETZER, A.; GAIZAUSKAS, R.; HEPPLE, M. The role of inference in the temporal annotation and analysis of text. *Language Resources and Evaluation*, Springer, v. 39, n. 2, p. 243–265, 2005.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, [Oxford University Press, Biometrika Trust], v. 52, n. 3/4, p. 591–611, 1965. ISSN 00063444. Disponível em: <http://www.jstor.org/stable/2333709>.

SNEDECOR, G. W.; COCHRAN, W. G. Statistical methods, eight edition. *Iowa state University press, Ames, Iowa*, v. 1191, n. 2, 1989.

SUN, W.; RUMSHISKY, A.; UZUNER, O. Evaluating temporal relations in clinical text: 2012 i2b2 Challenge. *Journal of the American Medical Informatics Association*, v. 20, n. 5, p. 806–813, 04 2013. ISSN 1067-5027. Disponível em: <https://doi.org/10.1136/amiajnl-2013-001628>.

TUKEY, J. W. The problem of multiple comparisons. *Multiple comparisons*, Chapman and Hall, 1953.

UZZAMAN, N. et al. Tempeval-3: Evaluating events, time expressions, and temporal relations. *arXiv preprint arXiv:1206.5333*, 2012.

VERHAGEN, M. Temporal closure in an annotation environment. *Language Resources and Evaluation*, JSTOR, v. 39, n. 2/3, p. 211–241, 2005.

VERHAGEN, M. et al. SemEval-2007 task 15: TempEval temporal relation identification. In: *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. Prague, Czech Republic: Association for Computational Linguistics, 2007. p. 75–80. Disponível em: <https://aclanthology.org/S07-1014>.

VERHAGEN, M. et al. The tempeval challenge: Identifying temporal relations in text. *Language Resources and Evaluation*, v. 43, p. 161–179, 06 2009.

VERHAGEN, M. et al. Semeval-2010 task 13: Tempeval-2. In: *Proceedings of the 5th international workshop on semantic evaluation*. [S.l.: s.n.], 2010. p. 57–62.

XU, P.; DING, Z.; PAN, M. A hybrid interpretable credit card users default prediction model based on ripper. *Concurrency and Computation: Practice and Experience*, v. 30, n. 23, p. e4445, 2018. E4445 cpe.4445. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4445>.

ZHENG, A.; CASARI, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2018. ISBN 1491953241.

Apêndice

A

CONJUNTO DE *FEATURES*

Apresentamos o nosso conjunto de *features* utilizado no desenvolvimento das regras aplicadas nesse trabalho. O conjunto de *features* apresentado no Capítulo 3 é composto por 100 *features* únicas, as quais foram reduzidas para 70 *features* após aplicação de nossa engenharia de *features*, descrita na Seção 4.1.

A.1 *FEATURES* POR CATEGORIA

A seguir, relacionamos as *features* organizadas por tipo de informação linguística, as quais foram utilizadas na geração dos conjuntos de regras empregados neste trabalho, conforme exposto no Capítulo 4:

A.1.1 Anotação TimeML

1. ***event-class*:** classe do evento anotada no *corpus*
2. ***event-closest-to-event-class*:** classe anotada do evento mais próximo do evento que está na relação temporal sob classificação
3. ***event-closest-to-event-equal-class*:** verifica se o valor da classe do evento da relação temporal sob classificação é igual à classe do evento mais próximo a ele no texto
4. ***event-intervening-preceding-class*:** classe do evento que é mencionado entre a expressão temporal e o evento, nesta ordem textual, da relação temporal sob classificação e está mais próximo da expressão temporal.
   Por exemplo, `TIMEX -- event2.class ------ EVENT`
5. ***event-polarity*:** polaridade do evento
6. ***timex3-temporalfunction*:** `temporalFunction` da expressão temporal anotado no *corpus*
7. ***timex3-type*:** tipo da expressão temporal

A.1.2 Conhecimento Prévio Sobre o Mundo

8. ***event-temporal-direction*:** registra a direção temporal do evento. Representa a relação temporal esperada entre o evento e seu complemento. Mais detalhe na Seção 3.2.3.

9. ***event-closest-to-event-temporal-direction*:** valor da direção temporal para o evento mais próximo do evento da relação temporal sob classificação

A.1.3 Contextual

10. ***event-between-order*:** se há outro evento entre o evento e a expressão temporal da relação sob classificação

11. ***event-conjunction-closest-follow*:** conjunção mais próxima após o evento da relação processada

12. ***event-conjunction-closest-precede*:** conjunção mais próxima antes do evento da relação processada

13. ***event-first-order*:** o evento precede textualmente a expressão temporal?

14. ***event-has-modal-verb-precede*:** verifica se o evento possui verbos modais antes dele

15. ***event-modal-verb*:** texto do verbo modal antes do evento

16. ***event-preposition-precede*:** tem como valor a preposição que precede o evento no texto

17. ***event-timex3-distance*:** distância entre o evento e expressão temporal.

18. ***event-timex3-no-between-order*:** verifica se não há evento ou expressão temporal entre o evento e a expressão temporal da relação sob classificação

19. ***timex3-between-order*:** se há outra expressão temporal entre o evento e a expressão temporal da relação sob classificação

20. ***timex3-preposition-precede*:** tem como valor a preposição que precede a expressão temporal no texto

A.1.4 Informações Lexicais

21. ***timex3-relevant-lemmas*:** o valor do lema da expressão temporal se ele estiver em uma lista de lemas que têm algum conteúdo temporal e são frequentemente vistos nas expressões temporais. Mais detalhes na Seção 3.2.2

A.1.5 Informações Morfológicas

22. ***event-closest-to-event-equal-pos*:** verifica se a classe gramatical do evento da relação temporal sob classificação é igual à classe gramatical do evento que é mencionado no texto mais próximo a ele

23. ***event-closest-to-event-equal-tense*:** verifica se o tempo verbal do evento da relação temporal sob classificação é igual ao tempo verbal do evento mais próximo a ele no texto

24. ***event-closest-to-event-pos*:** classe gramatical do evento que está mais próximo do evento que está na relação temporal sob classificação

25. ***event-closest-to-event-tense*:** tempo verbal do evento que está mais próximo do evento que está na relação temporal sob classificação

26. ***event-closest-to-timex3-equal-pos*:** verifica se a classe gramatical do evento da relação sob classificação é igual à classe gramatical do evento mais próximo da expressão temporal da relação sob classificação

27. ***event-closest-to-timex3-pos*:** classe gramatical do evento mais próximo da expressão temporal da relação temporal sob classificação

28. ***event-gov-verb-aspect*:** aspecto verbal do verbo que rege o evento. Se evento for verbo, o valor dessa *feature* é o aspecto verbal do próprio evento

29. ***event-gov-verb-tense*:** tempo verbal do verbo que rege o evento. Se evento for verbo, o valor dessa *feature* é o tempo verbal do próprio evento

30. ***event-head-pos*:** classe gramatical do nó pai do evento na árvore de dependência

31. ***event-intervening-following-tense*:** tempo verbal do evento que é mencionado entre o evento e a expressão temporal, nesta ordem textual, da relação temporal sob classificação e está mais próximo da expressão temporal.
    Por exemplo, `EVENT ------ event2.tense -- TIMEX`.

32. ***event-pos*:** classe gramatical do evento

33. ***event-pos-token-1-follow*:** classe gramatical do 1º *token* após o evento

34. ***event-pos-token-2-follow*:** classe gramatical do 2º *token* após o evento

35. ***event-pos-token-3-follow*:** classe gramatical do 3º *token* após o evento

36. ***event-pos-token-1-precede*:** classe gramatical do 1º *token* anterior ao evento

37. ***event-pos-token-2-precede*:** classe gramatical do 2º *token* anterior ao evento

38. ***event-pos-token-3-precede*:** classe gramatical do 3º *token* anterior ao evento

39. ***timex3-pos-token-1-follow*:** classe gramatical do 1º *token* após a expressão temporal

40. ***timex3-pos-token-2-follow*:** classe gramatical do 2º *token* após a expressão temporal

41. ***timex3-pos-token-3-follow*:** classe gramatical do 3º *token* após a expressão temporal

42. ***timex3-pos-token-1-precede*:** classe gramatical do 1º *token* anterior à expressão temporal

43. ***timex3-pos-token-2-precede*:** classe gramatical do 2º *token* anterior à expressão temporal

44. ***timex3-pos-token-3-precede*:** classe gramatical do 3º *token* anterior à expressão temporal

45. ***timex3-gov-verb-tense*:** tempo verbal do verbo que rege a expressão temporal

46. ***timex3-head-pos*:** classe gramatical do nó pai de expressão temporal

47. ***timex3-pos*:** classe gramatical da expressão temporal

A.1.6 Informações Sintáticas

48. ***event-dep*:** relação de dependência que o evento tem com seu regente

49. ***event-head-is-root*:** o evento modifica diretamente a raiz? (ex: o evento é um filho direto da raiz da árvore de dependência?)

50. ***event-is-ancestor-timex3*:** o evento é a entidade regente na relação?

51. ***event-preposition-gov*:** preposição que rege sintaticamente o evento

52. ***event-root*:** o evento é a raiz da árvore de dependência sintática?

53. ***event-timex3-dep*:** relação de dependência entre o evento e a expressão temporal ou entre a expressão temporal e o evento

54. ***reichenbach-direct-modification*:** a expressão temporal modifica diretamente o evento? (a expressão está no mesmo caminho de dependência do evento?)

55. ***reichenbach-temporal-mod-function*:** função de modificação temporal, existe uma relação `nmod` no caminho de dependência do evento à expressão temporal?

56. ***timex3-dep*:** relação de dependência sintática da expressão temporal com seu regente (nó pai).

57. ***timex3-head-is-root*:** a expressão temporal modifica diretamente a raiz? (ex: a expressão temporal é um filho direto da raiz da arvore de dependência)

58. ***timex3-is-ancestor-event*:** a expressão temporal é a entidade regente na relação?

59. ***timex3-preposition-gov*:** preposição que rege sintaticamente a expressão temporal

A.1.7 Reichenbach

60. ***reichenbach-tense*:** tempo verbal de Reichenbach. Assume o valor anterior, simples ou posterior. Mais detalhes na Seção 3.1.2

A.1.8 Sinais Temporais

Os sinais temporais consistem em conjunções e advérbios temporais, conforme conceituado na Seção 3.1.1. É importante mencionar que se não existir sinal que preceda o evento ou a expressão temporal sob classificação, o valor será nulo para todas as features relacionadas a este sinal.

61. ***signal-precede-event-child-event*:** o sinal que precede o evento é regido sintaticamente por este evento?

62. ***signal-precede-timex3-child-timex3*:** o sinal que precede a expressão temporal é regido sintaticamente por esta expressão?

63. ***signal-precede-event-dep-if-child-event*:** se o sinal que precede o evento for regido por este evento, qual é o tipo de relação de dependência sintática?

64. ***signal-precede-timex3-dep-if-child-timex3*:** se o sinal que precede a expressão temporal for regido por esta expressão, qual é o tipo de relação de dependência sintática?

65. ***signal-precede-event-distance-event*:** distância entre o evento e o sinal temporal que o precede

66. ***signal-precede-timex3-distance-timex3*:** distância entre a expressão temporal e o sinal temporal que a precede

67. ***signal-precede-event-pos*:** classe gramatical do sinal que precede o evento

68. ***signal-precede-timex3-pos*:** classe gramatical do sinal que precede a expressão temporal

69. ***signal-precede-event-text*:** texto do sinal que precede o evento

70. ***signal-precede-timex3-text*:** texto do sinal que precede a expressão temporal

Ressaltamos que os nomes das *features* não refletem necessariamente aos nomes dados originalmente pelos seus autores. Criamos nossa própria semântica de nomenclatura.

A.2 *FEATURES* ORDENADAS POR IMPORTÂNCIA

A seguir, relação das 52 *features* mais relevantes selecionadas conforme técnica de eliminação recursiva de *features* com validação cruzada (RFECV, do inglês *Recursive Feature Elimination with Cross-Validation*), discutida na Seção 4.1.6.

1. *event-between-order*
2. *reichenbach-direct-modification*
3. *event-class*
4. *event-closest-to-event-pos*
5. *event-closest-to-timex3-pos*
6. *event-dep*
7. *timex3-dep*
8. *event-pos-token-1-precede*
9. *event-pos-token-1-follow*
10. *event-pos-token-2-precede*
11. *event-pos-token-2-follow*
12. *event-pos-token-3-precede*
13. *event-pos-token-3-follow*
14. *timex3-pos-token-1-follow*
15. *timex3-pos-token-2-precede*
16. *timex3-pos-token-2-follow*
17. *timex3-pos-token-3-precede*
18. *timex3-pos-token-3-follow*
19. *event-preposition-precede*
20. *timex3-preposition-precede*
21. *event-timex3-distance*
22. *timex3-relevant-lemmas*
23. *event-gov-verb-tense*
24. *timex3-gov-verb-tense*
25. *timex3-head-pos*
26. *signal-precede-timex3-text*
27. *signal-precede-timex3-pos*
28. *reichenbach-tense*
29. *timex3-pos*
30. *event-closest-to-event-class*
31. *event-temporal-direction*
32. *event-modal-verb*
33. *event-closest-to-event-temporal-direction*
34. *timex3-type*
35. *event-conjunction-closest-follow*
36. *signal-precede-event-child-event*
37. *event-closest-to-event-equal-class*
38. *timex3-pos-token-1-precede*
39. *event-closest-to-event-tense*
40. *reichenbach-temporal-mod-function*
41. *event-preposition-gov*
42. *timex3-between-order*
43. *signal-precede-timex3-dep-if-child-timex3*
44. *event-intervening-preceding-class*
45. *timex3-preposition-gov*
46. *signal-precede-event-text*
47. *event-closest-to-event-equal-pos*
48. *event-polarity*
49. *event-closest-to-timex3-equal-pos*
50. *event-conjunction-closest-precede*
51. *timex3-is-ancestor-event*
52. *event-closest-to-event-equal-tense*

Apêndice

B

CONJUNTO DE REGRAS

Apresentamos o conjunto de regras que demonstrou o melhor desempenho na tarefa de identificação de tipos de relações temporais entre eventos e expressões temporais na mesma frase. Conforme evidenciado no Capítulo 6, o conjunto de regras mais eficaz foi gerado pelo algoritmo `RIPPER`, que consiste em um total de 146 regras. Os detalhes referentes à composição desse conjunto foram devidamente descritos na Seção 4.2.5.

B.1 MELHOR CONJUNTO DE REGRAS

Cada regra possui o seguinte formato:

(35) *TIPO DA RELAÇÃO TEMPORAL: expressão lógica que representa a regra* |
(total de acertos / número de vezes que foi acionada = acurácia)

Em (35), a expressão lógica que representa a regra é composta por uma sucessão de conjunções de condições. Cada uma destas condições é construída a partir de uma ***`feature`***, como delineado no Apêndice A, um `operador` que, neste conjunto de regras específico, é representado exclusivamente pelo operador de igualdade, e o `valor` correspondente à *feature*. Adicionalmente, a interpretação minuciosa dos valores atribuídos às *features* que envolvem a classe gramatical[1] (`POS`) e a árvore de dependência[2] (`DEP`) encontra-se disponível no endereço da *Universal Dependencies*[3].

1. *BEFORE: event-between-order == True and event-closest-to-timex3-pos == 'VERB' and event-conjunction-closest-follow == 'que' and timex3-pos-token-1-follow == 'PUNCT' and timex3-relevant-lemmas == 'próximo'* | (15/15 = 1)
2. *BEFORE: event-between-order == True and signal-precede-timex3-pos == 'ADP' and timex3-preposition-gov == 'até'* | (29/39 = 0,7436)
3. *BEFORE: event-between-order == True and event-pos-token-1-follow == 'SCONJ' and event-pos-token-1-precede == 'PRON'* | (14/19 = 0,7368)

[1] <https://universaldependencies.org/u/pos/>
[2] <https://universaldependencies.org/u/dep/all.html>
[3] <https://github.com/UniversalDependencies/docs/tree/pages-source/_pt>

4. *BEFORE: event-between-order == True and event-closest-to-timex3-pos == 'VERB' and event-first-order == True and timex3-pos-token-2-precede == 'PUNCT'* | (20/34 = 0,5882)
5. *BEFORE: event-gov-verb-tense == 'PAST' and reichenbach-direct-modification == False and timex3-relevant-lemmas == 'próximo'* | (18/18 = 1)
6. *BEFORE: timex3-preposition-precede == 'até'* | (21/33 = 0,6364)
7. *BEFORE: timex3-dep == 'DEP'* | (3/6 = 0,5)
8. *AFTER: event-between-order == True and event-class == 'REPORTING' and reichenbach-tense == 'simples' and timex3-gov-verb-tense == 'PAST' and timex3-pos-token-2-follow == 'NUM'* | (17/17 = 1)
9. *AFTER: event-between-order == True and timex3-relevant-lemmas == 'haver'* | (36/41 = 0,878)
10. *AFTER: event-between-order == True and reichenbach-tense == 'simples' and signal-precede-timex3-distance-timex3 == 'perto' and timex3-gov-verb-tense == 'PAST'* | (29/37 = 0,7838)
11. *AFTER: event-class == 'REPORTING' and event-closest-to-event-equal-tense == True and reichenbach-direct-modification == False and reichenbach-temporal-mod-function == True and timex3-gov-verb-tense == 'PAST'* | (20/20 = 1)
12. *AFTER: event-between-order == True and event-closest-to-timex3-pos == 'NOUN' and event-is-ancestor-timex3 == True and timex3-dep == 'NMOD' and timex3-pos-token-1-follow == 'ADP'* | (23/28 = 0,8214)
13. *AFTER: event-class == 'REPORTING' and event-pos-token-1-precede == 'VERB' and reichenbach-direct-modification == False* | (10/10 = 1)
14. *AFTER: event-pos-token-3-follow == 'ADJ' and event-preposition-precede == 'em' and reichenbach-direct-modification == False* | (8/8 = 1)
15. *OVERLAP: event-between-order == False and event-closest-to-event-equal-pos == False and reichenbach-direct-modification == True* | (146/154 = 0,9481)
16. *OVERLAP: event-between-order == False and event-pos-token-3-precede == 'NOUN' and reichenbach-direct-modification == True* | (55/57 = 0,9649)
17. *OVERLAP: event-between-order == False and event-closest-to-event-equal-class == False and event-pos == 'NOUN' and timex3-pos-token-2-precede == 'NOUN'* | (32/32 = 1)
18. *OVERLAP: event-between-order == False and event-pos-token-1-follow == 'ADV' and reichenbach-direct-modification == True* | (38/39 = 0,9744)
19. *OVERLAP: event-first-order == True and event-head-is-root == False and event-timex3-no-between-order == True* | (102/111 = 0,9189)
20. *OVERLAP: event-pos-token-1-precede == 'NOUN' and event-timex3-distance == 'perto'* | (57/69 = 0,8261)
21. *OVERLAP: event-first-order == True and event-pos == 'NOUN' and event-timex3-distance == 'perto'* | (75/81 = 0,9259)
22. *OVERLAP: event-first-order == False and event-head-is-root == True and event-pos-token-2-follow == 'NUM'* | (41/44 = 0,9318)
23. *OVERLAP: event-closest-to-event-pos == 'NOUN' and event-pos-token-2-precede == 'DET' and event-root == False* | (17/18 = 0,9444)
24. *OVERLAP: event-class == 'ASPECTUAL' and event-timex3-distance == 'perto'* | (28/30 = 0,9333)
25. *OVERLAP: timex3-preposition-precede == 'neste'* | (22/25 = 0,88)
26. *OVERLAP: event-root == False and timex3-pos-token-1-follow == 'ADV'* | (16/22 = 0,7273)

27. *OVERLAP: timex3-pos-token-1-precede == 'ADJ'* | (11/13 = 0,8462)
28. *BEFORE: event-between-order == True and event-closest-to-event-equal-tense == False and event-closest-to-timex3-pos == 'VERB' and event-pos-token-1-follow == 'SCONJ' and signal-precede-timex3-distance-timex3 == 'perto' and timex3-pos == 'NUM'* | (12/16 = 0,75)
29. *BEFORE: event-between-order == True and event-pos-token-1-follow == 'SCONJ' and timex3-relevant-lemmas == 'próximo'* | (18/19 = 0,9474)
30. *BEFORE: event-between-order == True and event-class == 'REPORTING' and event-pos-token-1-precede == 'PRON'* | (9/9 = 1)
31. *BEFORE: event-class == 'REPORTING' and reichenbach-direct-modification == False and timex3-relevant-lemmas == 'próximo'* | (17/17 = 1)
32. *BEFORE: event-modal-verb == 'ter' and signal-precede-timex3-child-timex3 == True* | (6/11 = 0,5455)
33. *AFTER: event-class == 'REPORTING' and event-temporal-direction == 'before' and reichenbach-temporal-mod-function == True and timex3-gov-verb-tense == 'PAST'* | (22/23 = 0,9565)
34. *AFTER: event-between-order == True and event-class == 'REPORTING' and reichenbach-tense == 'simples' and timex3-gov-verb-tense == 'PAST'* | (58/76 = 0,7632)
35. *AFTER: event-closest-to-event-tense == 'PRES' and event-pos-token-1-follow == 'DET' and event-temporal-direction == 'before'* | (9/10 = 0,9)
36. *AFTER: event-polarity == 'NEG' and timex3-gov-verb-tense == 'PAST'* | (7/9 = 0,7778)
37. *AFTER: signal-precede-timex3-text == 'desde'* | (11/13 = 0,8462)
38. *OVERLAP: event-between-order == False and reichenbach-direct-modification == True* | (250/275 = 0,9091)
39. *OVERLAP: event-between-order == False and event-closest-to-event-equal-pos == False and event-pos == 'NOUN' and timex3-pos-token-2-precede == 'NOUN'* | (41/43 = 0,9535)
40. *OVERLAP: event-between-order == False and event-pos == 'NOUN'* | (118/139 = 0,8489)
41. *OVERLAP: event-timex3-distance == 'perto' and timex3-pos-token-3-precede == 'NOUN'* | (74/101 = 0,7327)
42. *OVERLAP: event-timex3-dep == 'OBL'* | (168/197 = 0,8528)
43. *OVERLAP: event-class == 'ASPECTUAL'* | (48/67 = 0,7164)
44. *OVERLAP: event-preposition-precede == 'nos'* | (9/10 = 0,9)
45. *BEFORE: event-between-order == True and timex3-preposition-precede == 'no' and timex3-relevant-lemmas == 'próximo'* | (12/14 = 0,8571)
46. *BEFORE: event-class == 'REPORTING' and timex3-relevant-lemmas == 'próximo'* | (17/17 = 1)
47. *AFTER: event-class == 'REPORTING' and reichenbach-temporal-mod-function == True and timex3-gov-verb-tense == 'PAST' and timex3-pos-token-2-follow == 'NUM'* | (21/21 = 1)
48. *AFTER: event-first-order == True and event-timex3-distance == 'muito-longe' and timex3-gov-verb-tense == 'PAST'* | (61/93 = 0,6559)
49. *AFTER: event-class == 'REPORTING' and event-closest-to-event-equal-pos == False and reichenbach-temporal-mod-function == True* | (37/55 = 0,6727)
50. *AFTER: event-between-order == True and timex3-dep == 'OBL' and timex3-pos-token-2-follow == 'NOUN'* | (12/17 = 0,7059)

51. *AFTER: event-class == 'REPORTING' and event-pos-token-1-precede == 'VERB'* | (11/11 = 1)
52. *AFTER: event-pos-token-3-follow == 'VERB' and timex3-pos-token-2-follow == 'PROPN'* | (5/6 = 0,8333)
53. *AFTER: timex3-pos-token-1-follow == 'ADJ'* | (2/3 = 0,6667)
54. *OVERLAP: event-between-order == False and reichenbach-direct-modification == True and reichenbach-tense == 'simples'* | (134/145 = 0,9241)
55. *OVERLAP: signal-precede-timex3-text == 'anualmente'* | (2/2 = 1)
56. *BEFORE: event-between-order == True and event-closest-to-event-equal-pos == False and event-conjunction-closest-follow == 'que' and event-pos-token-1-follow == 'SCONJ' and timex3-temporalfunction == False* | (13/16 = 0,8125)
57. *BEFORE: event-closest-to-timex3-equal-pos == False and timex3-pos-token-3-precede == 'SCONJ'* | (5/5 = 1)
58. *BEFORE: timex3-preposition-precede == 'à'* | (5/8 = 0,625)
59. *BEFORE: event-preposition-gov == 'nesse'* | (1/1 = 1)
60. *AFTER: event-class == 'REPORTING' and event-closest-to-event-tense == 'PAST' and reichenbach-temporal-mod-function == True and timex3-gov-verb-tense == 'PAST'* | (21/22 = 0,9545)
61. *AFTER: event-class == 'REPORTING' and event-dep == 'XCOMP' and event-temporal-direction == 'before'* | (10/10 = 1)
62. *AFTER: event-preposition-precede == 'no' and timex3-pos-token-3-precede == 'ADP'* | (11/19 = 0,5789)
63. *AFTER: event-dep == 'APPOS'* | (2/4 = 0,5)
64. *OVERLAP: event-between-order == False and event-closest-to-event-class == 'REPORTING' and event-is-ancestor-timex3 == True* | (49/54 = 0,9074)
65. *OVERLAP: timex3-dep == 'ADVMOD' and timex3-gov-verb-tense == 'PRES' and timex3-pos-token-1-follow == 'VERB'* | (20/23 = 0,8696)
66. *OVERLAP: event-temporal-direction == 'before' and timex3-pos-token-1-follow == 'SCONJ'* | (15/17 = 0,8824)
67. *BEFORE: event-between-order == True and event-pos-token-1-follow == 'SCONJ' and timex3-gov-verb-tense == 'PRES'* | (20/38 = 0,5263)
68. *BEFORE: event-between-order == True and event-closest-to-event-equal-pos == False and event-closest-to-timex3-pos == 'VERB' and event-dep == 'ROOT'* | (32/53 = 0,6038)
69. *BEFORE: event-between-order == True and timex3-preposition-precede == 'em' and timex3-temporalfunction == False* | (16/29 = 0,5517)
70. *BEFORE: event-between-order == True and event-pos-token-1-precede == 'PRON' and timex3-head-is-root == False* | (27/45 = 0,6)
71. *BEFORE: timex3-preposition-gov == 'até'* | (36/51 = 0,7059)
72. *BEFORE: timex3-pos-token-3-precede == 'SCONJ'* | (11/22 = 0,5)
73. *BEFORE: event-closest-to-timex3-pos == 'X'* | (2/2 = 1)
74. *OVERLAP: event-timex3-no-between-order == True and reichenbach-direct-modification == True* | (246/269 = 0,9145)
75. *OVERLAP: event-between-order == False and event-pos == 'NOUN' and timex3-between-order == False and timex3-is-ancestor-event == False* | (110/124 = 0,8871)

76. *OVERLAP: event-pos-token-3-follow == 'ADP' and timex3-preposition-precede == 'de'* | (28/40 = 0,7)
77. *BEFORE: event-between-order == True and event-closest-to-event-equal-pos == False and event-pos-token-1-precede == 'PRON' and timex3-head-is-root == False and timex3-temporalfunction == True* | (11/14 = 0,7857)
78. *AFTER: event-class == 'OCCURRENCE' and event-head-pos == 'VERB' and reichenbach-temporal-mod-function == False and timex3-pos-token-3-follow == 'ADV'* | (8/16 = 0,5)
79. *AFTER: event-closest-to-event-class == 'ASPECTUAL' and timex3-pos-token-3-follow == 'DET'* | (3/4 = 0,75)
80. *OVERLAP: event-pos == 'NOUN' and event-timex3-no-between-order == True* | (113/130 = 0,8692)
81. *OVERLAP: event-pos-token-1-follow == 'ADP' and timex3-pos-token-3-precede == 'DET'* | (37/50 = 0,74)
82. *OVERLAP: event-pos-token-2-precede == 'VERB' and timex3-pos-token-2-follow == 'DET'* | (30/37 = 0,8108)
83. *OVERLAP: timex3-pos-token-3-precede == 'SYM'* | (1/2 = 0,5)
84. *BEFORE: event-between-order == True and event-pos-token-1-precede == 'PRON'* | (30/55 = 0,5455)
85. *BEFORE: event-between-order == True and timex3-pos-token-2-precede == 'DET'* | (8/16 = 0,5)
86. *OVERLAP: event-between-order == False and event-pos == 'NOUN' and timex3-pos-token-2-precede == 'NOUN'* | (69/75 = 0,92)
87. *OVERLAP: timex3-pos-token-1-follow == 'AUX'* | (11/17 = 0,6471)
88. *OVERLAP: timex3-pos-token-1-follow == 'DET'* | (21/29 = 0,7241)
89. *OVERLAP: timex3-pos-token-2-precede == 'AUX'* | (9/15 = 0,6)
90. *OVERLAP: event-pos-token-3-follow == 'NOUN' and event-pos-token-3-precede == 'NUM'* | (5/6 = 0,8333)
91. *OVERLAP: event-pos-token-1-follow == 'PRON'* | (8/14 = 0,5714)
92. *BEFORE: event-pos-token-1-follow == 'SCONJ' and event-pos-token-1-precede == 'PRON'* | (15/20 = 0,75)
93. *AFTER: event-between-order == True and event-class == 'REPORTING' and event-pos == 'VERB' and timex3-gov-verb-tense == 'PAST'* | (66/99 = 0,6667)
94. *AFTER: event-timex3-distance == 'muito-longe' and timex3-pos-token-2-precede == 'PROPN'* | (12/23 = 0,5217)
95. *AFTER: timex3-preposition-precede == 'com'* | (3/6 = 0,5)
96. *OVERLAP: event-between-order == False and event-gov-verb-tense == 'PAST' and reichenbach-direct-modification == True and timex3-temporalfunction == True* | (129/142 = 0,9085)
97. *OVERLAP: event-between-order == False and event-pos == 'NOUN' and timex3-pos-token-3-precede == 'ADP'* | (47/52 = 0,9038)
98. *OVERLAP: event-between-order == False and event-class == 'OCCURRENCE' and timex3-preposition-precede == 'no'* | (70/75 = 0,9333)
99. *OVERLAP: event-closest-to-event-temporal-direction == 'before' and timex3-relevant-lemmas == 'ano'* | (21/36 = 0,5833)

100. *BEFORE: event-between-order == True and event-closest-to-timex3-equal-pos == True and event-gov-verb-tense == 'PAST' and event-is-ancestor-timex3 == False and timex3-pos-token-1-follow == 'PUNCT'* | (14/25 = 0,56)
101. *BEFORE: event-conjunction-closest-follow == 'que' and timex3-pos-token-3-precede == 'VERB'* | (9/16 = 0,5625)
102. *AFTER: timex3-dep == 'ADVCL'* | (31/43 = 0,7209)
103. *OVERLAP: event-dep == 'CCOMP' and event-pos-token-3-follow == 'ADP'* | (33/49 = 0,6735)
104. *BEFORE: event-dep == 'DEP'* | (2/3 = 0,6667)
105. *AFTER: event-between-order == True and event-closest-to-event-tense == 'PRES' and event-dep == 'XCOMP'* | (12/19 = 0,6316)
106. *AFTER: event-between-order == True and event-class == 'REPORTING' and timex3-gov-verb-tense == 'PAST'* | (67/103 = 0,6505)
107. *AFTER: event-pos-token-1-precede == 'SCONJ' and timex3-pos-token-2-follow == 'NOUN'* | (5/10 = 0,5)
108. *OVERLAP: event-closest-to-event-equal-pos == False and event-first-order == True and event-timex3-distance == 'perto'* | (130/152 = 0,8553)
109. *OVERLAP: event-pos-token-1-precede == 'PROPN' and event-pos-token-2-precede == 'ADP'* | (36/54 = 0,6667)
110. *OVERLAP: event-intervening-following-tense == 'FUT'* | (5/8 = 0,625)
111. *OVERLAP: timex3-pos-token-1-precede == 'ADV'* | (7/13 = 0,5385)
112. *OVERLAP: timex3-preposition-precede == 'da'* | (14/21 = 0,6667)
113. *AFTER: event-timex3-distance == 'muito-longe' and reichenbach-temporal-mod-function == True and timex3-gov-verb-tense == 'PAST'* | (33/45 = 0,7333)
114. *AFTER: event-pos-token-1-follow == 'DET' and event-temporal-direction == 'before'* | (51/85 = 0,6)
115. *OVERLAP: event-is-ancestor-timex3 == True and event-pos == 'NOUN'* | (89/113 = 0,7876)
116. *OVERLAP: event-closest-to-timex3-equal-pos == False and event-closest-to-timex3-pos == 'VERB' and event-head-is-root == True* | (32/54 = 0,5926)
117. *OVERLAP: event-closest-to-timex3-equal-pos == False and event-conjunction-closest-precede == 'e'* | (12/23 = 0,5217)
118. *OVERLAP: signal-precede-event-text == 'já'* | (3/3 = 1)
119. *BEFORE: event-closest-to-event-class == 'OCCURRENCE' and event-pos-token-1-follow == 'SCONJ' and event-pos-token-3-precede == 'PROPN'* | (9/18 = 0,5)
120. *AFTER: event-pos == 'VERB' and event-timex3-distance == 'muito-longe' and timex3-gov-verb-tense == 'PAST' and timex3-pos-token-2-precede == 'NOUN'* | (27/41 = 0,6585)
121. *AFTER: event-pos-token-2-follow == 'ADV' and timex3-pos-token-2-follow == 'ADP'* | (6/11 = 0,5455)
122. *OVERLAP: event-timex3-no-between-order == True* | (418/540 = 0,7741)
123. *OVERLAP: event-conjunction-closest-precede == 'que'* | (85/162 = 0,5247)
124. *BEFORE-OR-OVERLAP: event-between-order == False and event-closest-to-event-pos == 'VERB' and timex3-between-order == True and timex3-pos == 'NUM' and timex3-pos-token-3-precede == 'ADP'* | (3/3 = 1)

125. *OVERLAP-OR-AFTER: event-timex3-distance == 'longe' and timex3-relevant-lemmas == 'passado'* | (3/5 = 0,6)
126. *OVERLAP-OR-AFTER: event-closest-to-event-tense == 'PRES' and timex3-pos-token-1-follow == 'CCONJ'* | (3/6 = 0,5)
127. *OVERLAP-OR-AFTER: event-dep == 'CCOMP' and event-pos-token-3-precede == 'NUM'* | (3/6 = 0,5)
128. *OVERLAP-OR-AFTER: event-pos-token-2-follow == 'NUM' and timex3-preposition-precede == 'desde'* | (2/2 = 1)
129. *OVERLAP-OR-AFTER: event-conjunction-closest-precede == 'para' and timex3-preposition-precede == 'em'* | (2/3 = 0,6667)
130. *VAGUE: event-conjunction-closest-follow == 'com'* | (2/2 = 1)
131. *VAGUE: timex3-dep == 'CCOMP'* | (2/3 = 0,6667)
132. *BEFORE-OR-OVERLAP: event-class == 'OCCURRENCE' and timex3-dep == 'CONJ' and timex3-preposition-precede == 'entre'* | (3/4 = 0,75)
133. *OVERLAP-OR-AFTER: signal-precede-event-text == 'recente'* | (1/1 = 1)
134. *BEFORE-OR-OVERLAP: event-class == 'OCCURRENCE' and timex3-dep == 'CONJ'* | (3/6 = 0,5)
135. *BEFORE-OR-OVERLAP: event-conjunction-closest-precede == 'por'* | (1/1 = 1)
136. *OVERLAP-OR-AFTER: event-pos-token-1-precede == 'SCONJ' and timex3-pos-token-3-precede == 'PRON'* | (2/2 = 1)
137. *BEFORE-OR-OVERLAP: event-pos-token-3-follow == 'NUM' and event-timex3-distance == 'longe' and timex3-pos-token-1-precede == 'VERB'* | (2/2 = 1)
138. *BEFORE-OR-OVERLAP: event-conjunction-closest-follow == 'a' and signal-precede-event-text == 'em'* | (2/4 = 0,5)
139. *OVERLAP-OR-AFTER: event-gov-verb-aspect == 'PROGRESSIVE' and event-pos-token-3-follow == 'DET'* | (2/3 = 0,6667)
140. *BEFORE-OR-OVERLAP: event-class == 'OCCURRENCE' and reichenbach-temporal-mod-function == True and timex3-pos-token-1-precede == 'VERB'* | (1/2 = 0,5)
141. *OVERLAP-OR-AFTER: event-closest-to-event-equal-pos == True and event-first-order == True and event-timex3-distance == 'longe' and timex3-head-is-root == True* | (3/4 = 0,75)
142. *OVERLAP-OR-AFTER: event-conjunction-closest-precede == 'a' and event-intervening-following-tense == 'IMP'* | (2/2 = 1)
143. *OVERLAP-OR-AFTER: event-timex3-distance == 'longe' and timex3-pos-token-3-precede == 'PRON'* | (2/2 = 1)
144. *VAGUE: event-closest-to-event-temporal-direction == 'after' and event-intervening-preceding-class == 'I-STATE'* | (2/3 = 0,6667)
145. *OVERLAP-OR-AFTER: event-closest-to-event-class == 'STATE' and event-pos-token-3-precede == 'VERB' and timex3-dep == 'OBL'* | (2/3 = 0,6667)
146. *OVERLAP: True == True* | (classe padrão)