
**UNIVERSIDADE FEDERAL DO CEARÁ**

**CENTRO DE CIÊNCIAS**

**DEPARTAMENTO DE COMPUTAÇÃO**

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**ABELARDO VIEIRA MOTA**

# UMA SOLUÇÃO BASEADA EM TEMPLATE E MULTI-SOLUÇÃO PARA GERAÇÃO DE TEXTOS A PARTIR DE TRIPLAS

**FORTALEZA**

**2020**

ABELARDO VIEIRA MOTA

# UMA SOLUÇÃO BASEADA EM TEMPLATE E MULTI-SOLUÇÃO PARA GERAÇÃO DE TEXTOS A PARTIR DE TRIPLAS

Dissertação apresentada ao Curso de do Programa de pós-graduação em Ciência da Computação do Centro de ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência de Dados

Orientador: Prof. Dr. José Antônio Fernandes de Macêdo

Coorientadora: Prof[a]. Dr[a]. Ticiana Linhares Coelho da Silva

FORTALEZA

2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

V713s Vieira Mota, Abelardo.
Uma solução baseada em template e multi-solução para geração de textos a partir de triplas / Abelardo Vieira Mota. – 2020.
87 f. : il.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2020.
Orientação: Prof. Dr. José Antônio Fernandes de Macêdo.
Coorientação: Profa. Dra. Ticiana Linhares Coelho da Silva.

1. Geração de linguagem natural. 2. Bases de conhecimento. 3. Geração dados-para-texto. I. Título.
CDD 005

---

ABELARDO VIEIRA MOTA

UMA SOLUÇÃO BASEADA EM TEMPLATE E MULTI-SOLUÇÃO PARA GERAÇÃO DE TEXTOS A PARTIR DE TRIPLAS

Dissertação apresentada ao Curso de do Programa de pós-graduação em Ciência da Computação do Centro de ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência de Dados

Aprovada em: 15/07/2020.

BANCA EXAMINADORA

---

Prof. Dr. José Antônio Fernandes de Macêdo (Orientador)
Universidade Federal do Ceará (UFC)

---

Profª. Drª. Ticiana Linhares Coelho da Silva (Coorientadora)
Universidade Federal do Ceará (UFC)

---

Prof. Dr. César Lincoln Cavalcante Mattos
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Régis Pires Magalhães
Universidade Federal do Ceará (UFC)

## AGRADECIMENTOS

Agradeço a Deus pelas oportunidades e pelo apoio nos momentos difíceis.

À minha família pela criação, inspiração e incentivo.

Ao meu orientador Prof. Dr. José Antônio Fernandes de Macêdo e aos meus coorientadores Profª. Drª. Ticiana Linhares Coelho da Silva e Dr. Igo Ramalho Brilhante, por compartilharem comigo a sabedoria que possuem e me guiarem nesse caminho.

Aos colegas de trabalho e de laboratório pela convivência e pelas trocas de experiência.

Aos meus amigos, incluindo todos já citados, por enriquecerem minha vida com significado.

Por fim, ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

## RESUMO

Com a evolução dos estudos sobre Web Semântica, o formato de representação de dados RDF foi proposto e diversas bases de conhecimento foram criadas com ele. Com isso, muita informação se tornou disponível para processamento por computador e para usuários especialistas, mas ao mesmo tempo essa informação é inacessível para um usuário leigo. Uma das formas de tornar esse tipo de dado mais acessível é representá-lo utilizando linguagem natural, o que pode ser custoso se feito manualmente. Esse contexto motiva o desenvolvimento de soluções que gerem automaticamente textos a partir de bases de dados RDF. Existem várias soluções integradas propostas para esse fim, mas elas exigem grandes bases de dados de alta qualidade para treinamento, carecem de controles sobre como o texto será gerado e não oferecem garantias de que o texto gerado verbaliza todos e somente os dados de entrada. Este trabalho aborda esses problemas propondo uma solução modular baseada em template e com abordagem de múltiplas soluções, que gera múltiplos textos durante o processo de geração. O uso de templates exige uma quantidade de dados proporcional à variedade de textos que se quer gerar, e a modularização da solução permite um maior controle sobre o processo de geração, bem como a possibilidade de dar maiores garantias quanto ao resultado final. Os experimentos realizados para validar a solução proposta utilizaram uma base de dados real e demonstram que a abordagem proposta gera textos de melhor qualidade e supera alguns modelos do estado da arte, em relação aos métodos de avaliação BLEU, METEOR e TER. Além da avaliação com métodos automáticos, foi feita também uma inspeção em uma amostra dos textos gerados de forma a evidenciar limitações e possíveis pontos de melhoria.



**Palavras-chave:** Geração de linguagem natural. Bases de conhecimento. Geração dados-para-texto.

## ABSTRACT

With the evolution of studies on the Semantic Web, the RDF data representation format was proposed and several knowledge bases were created with it. As a result, a lot of information has become available for computer processing and for expert users, but at the same time this information is inaccessible to a lay user. One way to make this type of data more accessible is to represent it using natural language, which can be costly if done manually. This context motivates the development of solutions that automatically generate texts from RDF databases. There are several integrated solutions proposed for this purpose, but they require large, high-quality databases for training, they lack controls over how the text will be generated and do not offer guarantees that the generated text verbalizes all and only the input data. This work addresses these problems by proposing a modular template-based solution based and with a multiple solution approach, which generates multiple texts during the generation process. The use of templates requires an amount of data proportional to the variety of texts to be generated, and the modularization of the solution allows greater control over the generation process, as well the possibility of giving greater guarantees over the final result. The experiments carried out to validate the proposed solution used a real database and demonstrate that the proposed approach generates better quality texts and outperforms some state-of-the-art solutions, regarding the BLEU, METEOR and TER evaluation methods. In addition to the evaluation with automatic methods, an inspection was also carried out on a sample of the generated texts in order to highlight limitations and possible improvement points.



**Keywords:** Natural language generation. Knowledge bases. Data to text generation.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ALGORITMOS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| BLEU | BiLingual Evaluation Understudy |
| GCN | Graph Convolutional Network |
| GLN | Geração de Linguagem Natural |
| GRU | Gated Recurrent Unit |
| HTER | Human-targeted Translation Edit Rate |
| lemon | Lexicon Model for Ontologies |
| LSTM | Long short-term memory |
| METEOR | Metric for Evaluation of Translation With Explicit ORdering |
| PLN | Processamento de Linguagem Natural |
| RDF | Resource Description Framework |
| SMS | Short Message Service |
| TER | Translation Edit Rate |
| URI | Uniform Resource Identifier |

**SUMÁRIO**

# 1 INTRODUÇÃO

Existem várias formas de representar informações, como através de imagens, tabelas ou textos. Um dos atributos com o qual elas podem ser avaliadas refere-se a quão fácil é para uma pessoa recuperar a informação representada. Algumas formas de representação foram desenvolvidas com o objetivo de serem melhor consumidas por computadores, o que, em contrapartida, pode torná-las menos apropriadas para o consumo por pessoas. Esse é o caso da Resource Description Framework (RDF), uma forma de representar informações através de triplas $\langle sujeito, predicado, objeto \rangle$, utilizada para armazenar informações em bases de conhecimento, como a DBpedia (LEHMANN *et al.*, 2015), que contém dados extraídos da Wikipedia. Uma forma de tornar esse tipo de base de conhecimento mais acessível consiste em verbalizar seus dados, ou seja, representá-los utilizando texto em linguagem natural, que assume-se, a princípio, ser mais acessível para uma pessoa.

Apesar de ser possível transformar manualmente bases RDF em textos, essa abordagem depende da disponibilidade de pessoas com conhecimento sobre o domínio dos dados e com habilidades tanto para interpretar dados em formato RDF, quanto para transformá-los em textos. Essa dependência de pessoas pode trazer problemas de disponibilidade, caso haja poucas pessoas capacitadas; de custo e de velocidade. Esses problemas motivam o desenvolvimento de métodos de geração automática de textos a partir de bases RDF. Geração de Linguagem Natural (GLN) é uma subárea de Inteligência Artificial e Linguística Computacional que engloba o estudo sobre desenvolvimento de sistemas computacionais para automatizar a geração de texto em linguagem natural a partir de dados representados de formas não linguísticas (REITER; DALE, 1997). Sistemas de GLN podem trazer benefícios como, por exemplo, o caso da *Associated Press* (INSIGHTS, 2020a), que automatizou a geração de coberturas esportivas, conseguindo tanto aumentar a cobertura de jogos, quanto liberar seus jornalistas para se dedicarem a atividades mais críticas; ou o desenvolvimento de *bots* para gerar textos informativos sobre a evolução da pandemia de Covid-19 no Brasil, em 2020 (CORONAREPORTER, 2020).

Considere, por exemplo, que uma pessoa busque na DBpedia informações sobre o Barack Obama. Suponha que o seguinte conjunto de triplas RDF seja retornado[1]:

```
<Barack_Obama, birthDate, 1961-8-4>
```

[1] Para simplificar, são apresentadas apenas a última parte da Uniform Resource Identifier (URI) que identifica os recursos referenciados nas triplas. Por exemplo, a URI http://dbpedia.org/resource/Barack_Obama está representada como Barack_Obama

```
<Barack_Obama, birthPlace, Hawaii>
<Hawaii, capital, Honolulu>
```

Há várias formas de representar esses dados como texto. Uma delas, em português, é *"Barack Obama nasceu em Honolulu, Hawaii, em 4 de agosto de 1961."*. Outra é *"Honolulu, capital do Hawaii, é a cidade onde Obama nasceu, em 1961/08/04."*. Esses textos, apesar de verbalizarem o mesmo conjunto de dados, diferem quanto à ordenação dos dados, em como as entidades são referenciadas e em como os relacionamentos entre elas são expressos. Todos esses aspectos de um texto devem ser levados em consideração por um sistema de GLN.

O artigo (REITER; DALE, 1997) informa que na época estava sendo formado um consenso sobre quais decisões básicas um sistema de GLN deveria tomar: (i) *Determinação de Conteúdo*, que consiste em decidir qual subconjunto dos dados serão verbalizados; (ii) *Planejamento de Discurso*, ou seja, decidir em que ordem os dados serão verbalizados, e o discurso a ser utilizado; (iii) *Agregação em Sentenças*, que decide como particionar os dados, de forma que cada parte seja verbalizada em uma sentença; (iv) *Lexicalização*, que consiste em decidir quais palavras utilizar para expressar conceitos e relacionamentos entre entidades; (v) *Geração de Expressões de Referência*, ou seja, decidir quais palavras utilizar para referenciar entidades; e, finalmente, (vi) *Realização Linguística*, que consiste em decidir sobre aspectos gramaticais, como concordância verbal e pontuação.

Vários trabalhos recentemente publicados lidam com a geração de textos a partir de bases RDF usando uma arquitetura integrada, em que todas essas tarefas são executadas por um modelo só (JAGFELD; JENNE; VU, 2018), (DISTIAWAN *et al.*, 2018), (ZHU *et al.*, 2019), (MARCHEGGIANI; PEREZ-BELTRACHINI, 2018) e técnicas de Aprendizagem Profunda (GOODFELLOW; BENGIO; COURVILLE, 2016), que fazem uso de algoritmos para aprender representações hierárquicas dos dados, seguindo uma tendência que outras tarefas de Processamento de Linguagem Natural (PLN), como Tradução Automática, seguem (GATT; KRAHMER, 2018). Arquiteturas integradas usando Aprendizagem Profunda têm a vantagem de requererem pouco conhecimento especialista no domínio dos dados, ou em linguística, mas têm como desvantagens a necessidade de uma base de dados grande e de alta qualidade para treinamento de modelos (DUŠEK; HOWCROFT; RIESER, 2019), além de não oferecerem controles sobre aspectos do texto a ser gerado, bem como não garantirem que o texto gerado verbalizará todos os dados de entrada, e apenas eles. Uma alternativa a arquiteturas integradas

tem sido explorada, e consiste em dividir a solução em módulos, utilizando para todos os módulos Aprendizagem Profunda (FERREIRA *et al.*, 2019), ou utilizando Aprendizagem Profunda apenas no último módulo, responsável por gerar um texto a partir de um plano de texto, que especifica que dados serão verbalizados, em que ordem e organizados em que sentenças (MORYOSSEF; GOLDBERG; DAGAN, 2019).

Uma alternativa ao uso de Aprendizagem Profunda consiste em adotar uma arquitetura modular e baseada em templates de textos, definidos em (DEEMTER; THEUNE; KRAHMER, 2005) como um sistema que mapeia entradas não linguísticas diretamente em textos, potencialmente contendo lacunas, a serem preenchidas com os dados de entrada. A abordagem modular e baseada em templates é uma alternativa viável ao uso de Aprendizagem Profunda, à medida que é mais simples e permite maior controle sobre o processo de geração de textos, o que é um requisito para alguns sistemas em produção (HARRIS, 2008).

Um problema que pode ocorrer com arquiteturas modulares é a existência de restrições sobre o texto que deverá ser gerado e que não podem ser satisfeitas criando restrições sobre as saídas de cada módulo individualmente, como garantir que o texto gerado respeite um tamanho máximo (REITER, 2000). Esse problema é chamado Lacuna Geracional, e ocorre quando decisões em etapas iniciais do processo de geração de texto têm consequências imprevisíveis nas demais etapas (GATT; KRAHMER, 2018). Este trabalho lida com o problema Lacuna Geracional fazendo uso de uma arquitetura modular com abordagem de múltiplas soluções, onde cada módulo gera uma ou mais saídas a partir de cada entrada que recebem, ao fim do qual são gerados múltiplos textos que são então rankeados por um modelo que determinará qual deles será retornado.

Este trabalho propõe uma solução modular baseada em templates e com abordagem de múltiplas soluções para a geração automatizada de textos a partir de bases RDF. Essa solução é avaliada com o uso da base de dados *WebNLG* (COLIN *et al.*, 2016), utilizada na *shared task*[2] de mesmo nome que ocorreu em 2017, comparando com outras soluções que também fizeram uso dessa base de dados.

[2] *Shared task* são competições que têm como objetivo fomentar o estudo e desenvolvimento de soluções para um problema específico.

## 1.1 Objetivo Geral

Este trabalho tem como objetivo geral avaliar uma arquitetura modular e baseada em templates para geração de textos a partir de triplas e fazendo uso de uma abordagem de múltiplas soluções.

## 1.2 Objetivos Específicos

Este trabalho tem como objetivos específicos:

- Especificar e implementar uma arquitetura modular, baseada em templates e com abordagem de múltiplas soluções para o problema de geração de textos a partir de triplas;
- Comparar, em relação à qualidade dos textos, a arquitetura proposta e o estado da arte para esse problema;
- Analisar e identificar deficiências dessa arquitetura, propondo melhorias;
- Analisar o *tradeoff*, dessa arquitetura, entre a quantidade de textos explorados e a qualidade do texto final;

## 1.3 Contribuições

As principais contribuições deste trabalho são:

- Uma proposta e uma implementação de uma arquitetura modular baseada em templates e com abordagem de múltiplas soluções para geração automatizada de textos a partir de bases RDF;
- Implementação para facilitar a manipulação da base de dados *WebNLG* e a avaliação de modelos de GLN.

A implementação da solução e do suporte à manipulação da base de dados e avaliação de modelos encontra-se hospedada em um repositório do GitHub[3].

## 1.4 Publicações

O seguinte artigo, que faz parte deste trabalho, foi aceito para publicação:

- MOTA, A. V.; SILVA, T. L. c. da; MACEDO, J. A. de. Template-Based Multi-Solution Approach for Data-to-Text Generation. In: ADBIS, 2020 (a publicar).

[3] https://github.com/abevieiramota/masters

## 1.5 Estrutura da Dissertação

Esta dissertação está organizada nos seguintes capítulos: o Capítulo 2 apresenta a área de estudo GLN, o formato de representação de dados RDF e modelos N-grama; o Capítulo 3 apresenta os trabalhos relacionados à solução proposta; o Capítulo 4 apresenta a solução proposta e o Capítulo 5 apresenta a experimentação realizada para avaliá-la; finalmente, o Capítulo 6 apresenta um resumo desta dissertação, bem como propõe trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

A solução estudada neste trabalho lida com o problema de geração automática de textos, estudado pela área GLN. Nessa área são estudadas as tarefas que normalmente são executadas por uma solução de GLN, como implementá-las e que tipos de arquiteturas podem ser utilizadas para organizá-las. Além disso, nela são estudados métodos de avaliação da qualidade dos textos gerados, que servem tanto para avaliar as soluções desenvolvidas, quanto para fornecer *feedback* para soluções baseadas em Aprendizado de Máquina. Diferentes formatos de dados de entrada são possíveis, e este trabalho lida com dados em formato de tripla, como o RDF, fazendo uso de modelos baseados em N-grama.

Este Capítulo apresenta a fundamentação teórica que embasa a solução estudada nesta dissertação e está estruturado da seguinte forma: a Seção 2.1 apresenta a área GLN, que estuda soluções de geração automática de textos; a Seção 2.2 apresenta os meios com os quais soluções de GLN podem ser avaliadas; a Seção 2.3 descreve o tipo de dado a partir do qual a solução estudada gera textos; e, finalmente, a Seção 2.4 apresenta um algoritmo para treinar modelos utilizados para implementar os módulos da solução.

## 2.1 Geração de Linguagem Natural

GLN é uma área de estudo cujo objeto é a automatização do processo de geração de textos em linguagem natural. Não há um consenso quanto se tarefas como Tradução Automática, de geração de textos a partir de textos de entrada, estão inclusas em GLN, mas Gatt e Krahmer (2018) adotam a definição de que GLN se refere apenas à geração de textos a partir de dados representados de forma não linguística, o que inclui, por exemplo, imagens e tabelas. Este trabalho também adota essa definição. Esta Seção inicialmente motiva o estudo de GLN, depois apresenta quais tarefas normalmente são executadas por uma solução de GLN e conclui falando de arquiteturas de soluções de GLN propostas na literatura.

### *2.1.1 Por que estudar GLN?*

A representação de informação utilizando linguagem natural é justificada principalmente quando o público alvo dessa informação tem maior facilidade de recuperá-la a partir de representações linguísticas, do que a partir de representações não linguísticas. Há alguns motivos para alguém ter maior dificuldade de acessar uma informação representada de forma

não linguística: ausência de conhecimento sobre como interpretar os dados, que é o caso de uma pessoa leiga em cardiologia tentando recuperar informações de um eletrocardiograma; presença de deficiência que impeça ou dificulte o consumo de certos formatos, como o acesso a dados em formato de imagem por portadores de deficiência visual; limitações do meio de representação dos dados, como é o caso de Short Message Service (SMS), que não podem representar imagens, a não ser através de uma descrição textual.

Dado que há a necessidade de representar dados através de textos, cabe então decidir se o processo de geração será manual ou automatizado. Para a tomada dessa decisão, é necessário comparar as duas abordagens em relação a diversos aspectos, como o custo por texto, a disponibilidade de recursos, a velocidade de geração dos textos e a necessidade de controle sobre o resultado final.

Os custos da abordagem manual giram em torno principalmente de recursos humanos especializados para gerar o tipo de texto desejado; já em uma abordagem automatizada, o custo se divide entre custo de desenvolvimento da solução e custo de operação. Mesmo sendo o caso de o custo de desenvolvimento de uma solução de GLN ser alto, considerando o tempo em que será utilizada, e o custo de operação, talvez essa abordagem seja mais barata que a geração manual de textos.

Outra situação em que a abordagem automatizada pode ser mais viável ocorre quando há pouca disponibilidade de pessoas capacitadas para a escrita dos textos, como em cenários em que um grande volume de textos deverá ser gerado, ou quando o domínio dos textos possui poucos especialistas. Ainda será necessário o trabalho de especialistas nos textos para orientar o desenvolvimento do sistema e desenvolver o *corpus* a ser utilizado no treinamento de modelos, no caso de soluções baseadas em dados, mas espera-se que a dependência de especialistas diminua na operação do sistema. Foi o caso da *Associated Press* (INSIGHTS, 2020a), que após a adoção de tecnologias de GLN, com o suporte da empresa *Automated Insights* (INSIGHTS, 2020b), conseguiu aumentar a cobertura jornalística sobre jogos de basquete, além de liberar 20% do tempo dos jornalistas para atividades mais críticas. Outras empresas existem atualmente vendendo soluções de GLN, como a *ArriaNLG* (ARRIA, 2020) e a *Narrative Science* (SCIENCE, 2020).

A velocidade com que o texto será gerado pode ser um requisito essencial em alguns contextos, como na divulgação de notícias sobre a ocorrência de eventos imprevisíveis. Gatt e Krahmer (2018) citam o caso do jornal *Los Angeles Times*, que reportou um terremoto em 2014

apenas 3 minutos após a sua ocorrência, usando um *bot* que converte automaticamente em texto dados de sensores, com o uso de templates (TIMES, 2020). Outro uso de GLN no jornalismo foi o desenvolvimento de um *bot* para o *Twitter* para reportar dados sobre a evolução da pandemia de Covid-19 no Brasil, em 2020, gerando textos como *"São registrados 4.822 novos casos de COVID19 no Brasil, totalizando 96.559 casos confirmados. Com 366 novos óbitos, o país chega a marca de 6.750 mortos. O país registra alta diária de 5% na quantidade de casos e de 6% na quantidade de óbitos por COVID19."* (CORONAREPORTER, 2020).

Finalmente, pode ser necessário um nível de controle sobre o texto gerado que pode ser difícil alcançar em um processo manual. É o caso da geração de múltiplos textos a partir de um mesmo conjunto de dados, quando, por exemplo, é necessário transmitir a mesma informação para perfis de usuários distintos, que exijam o uso de línguas ou vocabulários diferentes; ou quando é necessário que os textos possuam pouca variabilidade, como no contexto de relatórios meteorológicos (REITER *et al.*, 2005).

Além desses aspectos, Bouayad-Agha, Casamayor e Wanner (2014) definem outros, como o volume e o tipo dos dados a serem verbalizados, se o sistema deverá ser flexível o suficiente para se adequar a mais de um domínio, o perfil do público alvo, o gênero textual e a modalidade da comunicação, se composta apenas por textos, ou se contendo outras formas, como imagens ou tabelas.

A pesquisa na área de GLN tem explorado soluções para diversos domínios, desde a geração de relatórios sobre partidas de futebol (GATTI; LEE; THEUNE, 2018) (LEE; KRAHMER; WUBBEN, 2017), sobre as condições clínicas de um paciente (BANAEE; AHMED; LOUTFI, 2013) ou sobre previsões de clima (MEI; BANSAL; WALTER, 2015), à geração de textos para persuadir pessoas a pararem de fumar (REITER; ROBERTSON; OSMAN, 2003) ou para gerar explicações sobre uma conclusão de um sistema especialista (SWARTOUT, 1983).

### *2.1.2 Tarefas executadas na geração de um texto*

O processo de gerar um texto a partir de um conjunto de dados envolve um conjunto de tarefas que definirão aspectos do texto que será gerado. Deve-se decidir, por exemplo, que conjunto de dados será verbalizado, em que ordem e com qual vocabulário. Reiter e Dale (1997) caracterizaram um conjunto dessas tarefas, a seguir descritas:

- **Determinação de Conteúdo**: um dos aspectos de um texto é o seu conteúdo, que deverá ser determinado a partir de fontes de dados, de forma a melhor atender a um objetivo,

como responder a uma pergunta. Geralmente, esta tarefa consiste na aplicação de filtros ou sumarização sobre as fontes de dados, e é influenciada pelo nível de conhecimento do público alvo, pelo objetivo a ser alcançado com o texto ou pelas interações prévias que os usuários tiveram com o sistema. Para responder à pergunta *"Quem foi Barack Obama?"*, por exemplo, pode-se consultar uma base de conhecimento, como a DBpedia, e selecionar um conjunto de dados que contenham as informações mais relevantes para que o usuário possa decidir sobre seu voto para presidente; para gerar a descrição de uma imagem, é necessário identificar os elementos presentes nela que melhor a descrevem.

- **Planejamento de Discurso**: também chamada por Estruturação do Texto por Gatt e Krahmer (2018), esta tarefa consiste na estruturação da informação de forma que possa ser representada como um texto, que é uma representação em formato de sequência, contendo sentenças, parágrafos, dentre outras estruturas. Para tanto, deve-se definir em que ordem os dados serão verbalizados e como eles serão conectados através de relações de discurso, como no seguinte exemplo, presente em (REITER; DALE, 1997), de duas sentenças em que a segunda exemplifica uma informação verbalizada na primeira:
  1. *"Eu gosto de colecionar guitarras Fender antigas."*
  2. *"Meu instrumento favorito é uma Stratocaster 1951."*

  Problemas com essa tarefa podem fazer com que o texto possua baixa coesão e seja de difícil entendimento.
- **Agregação em Sentenças**: um texto, além de ser uma sequência de palavras e pontuação, também é uma sequência de sentenças, e esta tarefa é responsável pela decisão de como estruturar os dados em sentenças. A solução mais simples é verbalizar cada unidade de informação em uma sentença separada, mas dessa forma o texto resultante pode parecer artificial e pouco fluente. Existem várias formas de agregar duas sentenças, como:
  1. Utilizar uma conjunção, unindo duas sentenças como *"Carlos comprou banana."* e *"João comprou maçã."* na sentença *"Carlos comprou banana e João comprou maçã."*;
  2. Utilizar elipse, como na sentença *"João comprou banana e foi para casa"*, em que fica implícito o sujeito da segunda oração;
  3. Utilizar uma conjunção, unindo os objetos de sentenças similares, como transformando *"Carlos comprou banana."* e *"Carlos comprou maçã."* na sentença *"Carlos comprou banana e maçã."*;

  Além de agregação sintática, quando a informação é verbalizada alterando-se apenas a

forma, Gatt e Krahmer (2018) também falam sobre agregação semântica, quando também há alteração na informação verbalizada. É o caso de, por exemplo, no lugar de usar a sentença *"Carlos comprou banana e maçã."* usar *"Carlos comprou frutas."*.

- **Lexicalização**: a decisão de quais palavras utilizar pode ser dividida em decidir que palavras utilizar para expressar relações entre entidades, conhecida como Lexicalização, e em decidir como referenciar entidades, conhecida como Geração de Expressões de Referência. Por exemplo, deve-se decidir que palavras utilizar para expressar que o preço do produto X é Y, podendo ser verbalizado como *"X custa Y."* ou *"O preço de X é Y."*. Aspectos do texto como a variabilidade e a simplicidade são impactados por esta tarefa.
- **Geração de Expressões de Referência**: parte de um texto faz referência a entidades. É o caso de falar sobre a Universidade Federal do Ceará, que pode ser feito tanto com o uso da sequência de palavras *"Universidade Federal do Ceará"*, com o uso da sigla *"UFC"*, com o uso da descrição *"a universidade federal do estado do Ceará"*, ou com o uso do pronome pessoal *"ela"*, dado que já haja uma referência prévia no texto a ela. De acordo com (REITER; DALE, 1997), essa tarefa normalmente é formalizada como uma tarefa de discriminação, responsável por referenciar uma entidade de forma tal que ela possa ser identificada, não havendo ambiguidade.
- **Realização Linguística**: dado que foi decidido quais palavras utilizar e como estruturá-las, é necessário tomar decisões relativas a aspectos gramaticais do texto, como a flexão dos verbos, concordância nominal e o uso de pontuação, além de aspectos visuais, como que fonte utilizar ou que trechos gerar em negrito. Reiter e Dale (1997) enfatizam que uma vantagem de implementar essa tarefa em um módulo separado dos demais é a de que se pode desenvolver várias versões desse módulo de forma que para um mesmo dado de entrada sejam geradas várias versões de textos, como na verbalização dos dados em várias línguas.

### *2.1.3 Arquiteturas de GLN*

Gatt e Krahmer (2018) classificam as principais formas de organizar uma solução de geração automatizada de textos em três tipos de arquiteturas:

- **Modular**, em que as tarefas são executadas por módulos, cada um com responsabilidades distintas. Reiter e Dale (1997) propõem o uso de três módulos dispostos em um *pipeline*: (i) Planejamento de Texto, englobando as tarefas Determinação de Conteúdo e Plane-

jamento de Discurso; (ii) Planejamento de Sentença, englobando as tarefas Agregação em Sentenças, Lexicalização e Geração de Expressões de Referência; e o módulo (iii) Realização Linguística, englobando a tarefa de mesmo nome. Já foram propostas tanto soluções baseadas em regras manualmente desenvolvidas, como (MILLE; DASIOPOULOU, 2017), quanto soluções em que os módulos são implementados como modelos treinados com dados, como (DUMA; KLEIN, 2013), que faz uso de templates extraídos de forma semi-automática, ou (FERREIRA *et al.*, 2019), que faz uso de técnicas de Aprendizagem Profunda para implementar cada módulo.

- **Planejamento**, em que o problema é modelado como um problema de planejamento, em que se deve identificar uma sequência de ações, com precondições e efeitos, para alcançar um determinado objetivo de comunicação. Por exemplo, Rieser e Lemon (2009) modelam o problema de geração de texto como um problema de planejamento sob incerteza (BLYTHE, 1999), quando não há informações suficientes sobre o efeito das ações ou seus efeitos podem mudar com o tempo, e faz uso de técnicas de Aprendizagem por Reforço.
- **Integrada**, em que o problema de geração de texto não é subdivido em tarefas, sendo resolvido de forma integrada e normalmente através da identificação de correspondências estatísticas entre os dados de entrada e textos de referência. Esse tipo de arquitetura, implementada com técnicas de Aprendizagem Profunda, tem sido uma tendência, dada a crescente adoção desse tipo de técnica em problemas de PLN em geral. Das 9 soluções que participaram da *shared task* WebNLG, 5 delas seguiam uma arquitetura integrada e utilizando técnicas de Aprendizagem Profunda (GARDENT *et al.*, 2017). Apesar dessa tendência, trabalhos recentes, como (FERREIRA *et al.*, 2019) e (MORYOSSEF; GOLDBERG; DAGAN, 2019), têm comparado arquiteturas integradas e arquiteturas modulares, e têm tido resultados a favor de arquiteturas mais modulares.

A divisão do problema de geração de textos em subproblemas a serem resolvidos por módulos pode causar o problema conhecido por Lacuna Geracional, que ocorre quando decisões tomadas no início do processo de geração de texto impactam negativamente as decisões tomadas no fim do processo (GATT; KRAHMER, 2018). Por exemplo, a geração de um *tweet* descrevendo uma pessoa cujos dados estão presentes numa base de conhecimento. Até o fim do processo de geração do texto, quando todas as decisões forem tomadas, não é possível determinar qual o tamanho final do texto e, portanto, se ele satisfará a limitação de um *tweet* ter no máximo 280 caracteres. Não tendo essa informação no momento da seleção dos dados

a serem verbalizados, pode ocorrer que ao fim do processo o texto gerado tenha um tamanho maior que o permitido.

Reiter (2000) avaliou três abordagens para lidar com o problema de gerar textos respeitando um tamanho máximo: abordagem de solução única, em que uma heurística foi utilizada para estimar o tamanho final do texto a partir de um plano de documento, que é gerado no início da geração do texto e informa o conteúdo e a estrutura do texto a ser gerado; abordagem de múltiplas soluções, em que são gerados vários planos de documento e a partir deles vários textos, retornando aquele com tamanho mais próximo do desejado; e usar um processo de revisão, em que textos são gerados até satisfazerem o tamanho limite - quando um texto gerado não satisfaz, o plano de documento que foi utilizado é revisado e um novo texto é gerado. Foi avaliado, para um conjunto de dados de teste, a quantidade de textos que satisfizeram o limite de tamanho do texto, para cada solução. Além disso foi mensurado o tempo médio para a geração dos textos. A solução baseada em revisão foi a que alcançou a maior taxa de sucesso, mas também foi a que mais demorou para gerar os textos. Em contrapartida, a abordagem de múltiplas soluções ficou em segundo lugar e gerou os textos utilizando menos tempo. A abordagem de solução única ficou em último lugar, confirmando a expectativa de haver um *tradeoff* entre a qualidade do texto gerado e o tempo gasto para gerá-lo.

## 2.2 Avaliação de soluções de GLN

A capacidade de avaliar a qualidade de soluções de GLN se faz necessária principalmente quando é preciso avaliar se elas são adequadas para resolver determinado problema e se serão mais vantajosas que abordagens manuais. Dado que se concluiu por utilizar uma solução de GLN, é necessário determinar qual arquitetura e algoritmos utilizar. Desenvolvida a solução, é necessário então avaliar se ela realmente atende aos requisitos estabelecidos. Quando é então colocada em produção, é necessário avaliar se a solução continua atendendo aos requisitos e se é necessário realizar ajustes. Além dessas motivações, abordagens baseadas em Aprendizado de Máquina do tipo supervisionado exigem, no desenvolvimento, algum mecanismo de *feedback* sobre a qualidade dos textos gerados.

A avaliação de uma solução de GLN pode se referir tanto a aspectos não-funcionais de um sistema de informação, como o tempo para gerar um texto ou facilidade de controlar aspectos dos textos gerados, quanto a aspectos funcionais, relativos a atributos dos textos em si. Em relação ao texto gerado, chamado texto hipótese, Gatt e Krahmer (2018) classificam as

formas de avaliação em dois conjuntos: avaliações intrínsecas lidam com aspectos do problema de geração de texto, como adequação, que é quão bem o texto verbaliza todos e somente os dados que deveria verbalizar, fluência, e corretude gramatical; e avaliações extrínsecas avaliam quão bem os textos gerados otimizam algum objetivo em seu uso, como convencer o leitor a tomar alguma ação, ou auxiliá-lo a tomar decisões mais rapidamente.

Um desafio que surge com a avaliação da qualidade de textos é a imprecisão com que certos aspectos são definidos. Não há uma definição objetiva de como mensurar a fluência de um texto, por exemplo, tornando difícil definir uma medida que calcule um escore, ou garantir que um escore dado por duas pessoas diferentes representa o mesmo julgamento do nível de fluência. Outro desafio está relacionado com o fato de que normalmente, para um conjunto de dados, existem vários textos que podem ser gerados e que o verbalizarão corretamente.

O processo de avaliação de uma solução de GLN pode ser executado tanto por pessoas, quanto por computadores. A avaliação por pessoas sofre dos mesmos problemas de custo, disponibilidade de recursos, velocidade e controle, referentes à geração de textos por pessoas, o que também motiva o estudo e adoção de métodos de avaliação automatizados. Mas, apesar desses problemas, a avaliação manual ainda é considerada essencial, visto que o uso de métodos automatizados ainda não é suficiente para decidir se uma solução é melhor que outra (GATT; KRAHMER, 2018).

### *2.2.1 Avaliação manual*

De acordo com (GATT; KRAHMER, 2018), uma das formas de conduzir uma avaliação de um sistema de GLN com o uso de pessoas consiste em gerar um conjunto de textos e expô-los a um indivíduo para que ele forneça uma nota para cada texto, baseada em algum critério, como quão fluente ele julga que o texto é, e então inferir a capacidade do sistema de gerar textos fluentes a partir das notas dadas ao conjunto de textos. Dado que vários aspectos de um texto que devem ser avaliados não possuem uma definição clara e objetiva e que isso pode levar a uma alta variabilidade da avaliação, dependendo da subjetividade do avaliador, normalmente são coletadas notas de mais de um avaliador.

Uma decisão que deve ser tomada diz respeito à escala da nota a ser utilizada, podendo ser ou um ordinal, como uma escala de 1 a 3, ou uma escala contínua, que pode ser apresentada com o uso de um *slider*. Para reduzir o baixo consenso que pode haver entre os avaliadores, decorrente da imprecisão na definição do que está sendo avaliado, uma solução

consiste em usar um processo de avaliação iterativo, consistindo de uma etapa de atribuição de notas, seguida por uma etapa de discussão das notas atribuídas e finalmente por uma etapa de revisão.

Outro desafio que pode ocorrer com o uso de notas consiste na falta de clareza sobre o significado de uma nota em específica, o que pode ser reduzido apresentando para o avaliador, antes da avaliação, exemplos de textos e respectivas notas para que o avaliador utilize como parâmetro. Para comparar sistemas, uma alternativa ao uso de notas consiste em expor o avaliador aos textos gerados pelos sistemas sob avaliação, para uma mesma entrada, e solicitar que ele os ordene de acordo com algum critério, posteriormente realizando testes estatísticos para avaliar se é possível afirmar que um sistema performa melhor que outro. Um problema que surge na comparação de sistemas por pessoas é a dificuldade de reproduzir os experimentos, dado que o julgamento dos textos muda de pessoa para pessoa, e, para a mesma pessoa, com o tempo.

### *2.2.2 Avaliação automática*

Com a evolução e maior uso de abordagens baseadas em Aprendizado de Máquina do tipo supervisionado, a importância de métodos de avaliação automatizados tem aumentado, dado que essas abordagens exigem *feedbacks* frequentes sobre a qualidade dos textos que geram em tempo de treinamento. Uma das abordagens de avaliação automatizada consiste em desenvolver um corpus contendo parte dos textos que são considerados adequados para cada conjunto de dados, chamados de textos de referência. Para problemas em que a quantidade de soluções é única ou pequena essa abordagem pode ser utilizada valendo-se de medidas como acurácia, precisão ou revocação. Ocorre que os problemas de geração de textos, de uma forma geral, normalmente permitem diversas soluções para um conjunto de dados de entrada. Uma alternativa consiste em desenvolver um corpus contendo apenas um subconjunto de textos de referência, para cada entrada, e utilizar alguma medida de similaridade entre o texto que se quer avaliar e os textos de referência correspondentes, buscando-se maximizar a correlação entre o escore retornado pela métrica e o real nível de qualidade do texto avaliado.

Gatt e Krahmer (2018) ressaltam que essa forma de avaliação, que consiste na comparação entre textos, avalia apenas indiretamente se o conteúdo do texto condiz com o conteúdo dos dados de entrada, dado que o texto gerado pode ser bastante diferente dos textos de referência e mesmo assim expressar os dados de entrada. Essa abordagem de avaliação

começou a ser explorada inicialmente em outras áreas de pesquisa, como Tradução Automática e Sumarização Automática, de forma que várias métricas utilizadas para avaliar soluções de GLN provêm dessas áreas.

#### *2.2.2.1 BLEU*

O método de avaliação automática BiLingual Evaluation Understudy (BLEU) foi proposto em (PAPINENI *et al.*, 2002) como uma alternativa à avaliação manual de sistemas de Tradução Automática. BLEU é um método baseado na similaridade entre o texto hipótese e um conjunto de textos de referência, e faz isso partindo da ideia de que os textos adequados para uma determinada entrada diferem entre si em relação às posições das palavras e à escolha das palavras utilizadas. Como forma de avaliar essa similaridade, BLEU é baseado em uma medida de precisão sobre N-grama dos textos, ou seja, o quanto dos N-gramas do texto sob avaliação estão contidos no conjunto de N-gramas dos textos de referência. BLEU faz uso de uma média geométrica dos valores de precisão de N-gramas para $N$ variando de 1 a 4. Para lidar com problemas decorrentes do uso apenas de uma medida de precisão, como favorecer textos curtos, BLEU inclui uma penalidade proporcional ao tamanho dos textos, além de limitar a contagem de N-gramas repetidos no texto hipótese ao máximo de repetições nos textos de referência. Seus valores variam de 0 a 100, sendo valores maiores melhor. Ele foi proposto como um método de avaliação de sistemas, significando que ele reporta um escore para um sistema como um todo, levando em consideração os textos por ele gerados. Trabalhos posteriores, como (CHEN; CHERRY, 2014) propuseram versões de BLEU para avaliar textos individuais.

#### *2.2.2.2 METEOR*

O método de avaliação automática Metric for Evaluation of Translation With Explicit ORdering (METEOR) foi proposto como uma alternativa ao BLEU, buscando resolver algumas de suas deficiências: a ausência de uma medida de revocação no seu cálculo; a ausência de verificação da ordem das palavras, só o fazendo indiretamente ao usar N-gramas; e o uso de média geométrica dos valores de precisão, que faz com que o valor de BLEU seja igual a 0 caso o texto hipótese não possua nenhum N-grama dentre os N-gramas dos textos de referência, para algum $N$ utilizado (BANERJEE; LAVIE, 2005). Diferente de BLEU, que trabalha apenas com correspondência exata entre N-gramas, o método METEOR é calculado verificando a correspondência entre palavras do texto hipótese e dos textos de referência, fazendo uso de

diversos esquemas de correspondência, como correspondência exata, caso duas palavras sejam iguais, ou correspondência por estemização, caso duas palavras retornem o mesmo valor após passarem por um processo de estemização. Além disso, no lugar de avaliar a similaridade do texto hipótese para todos os textos de referência como um todo, METEOR calcula um escore entre o texto hipótese e cada texto de referência, retornando o maior valor. Verificada a correspondência entre palavras do texto hipótese com as palavras dos textos de referência, são então calculadas medidas de precisão, definida como a razão entre o número de palavras, do texto hipótese, com correspondência e o total de palavras do texto hipótese, e de revocação, definida como a razão entre o número de palavras, do texto hipótese, com correspondência e o total de palavras do texto de referência. A combinação do valor de precisão e de revocação, através de uma média harmônica, é então utilizada, multiplicada por um fator cuja finalidade é penalizar textos cujas palavras estão ordenadas diferentemente de como estão no texto de referência. Seus valores variam de 0 a 1, sendo valores maiores melhor.

#### *2.2.2.3 TER*

O método de avaliação automática Translation Edit Rate (TER), proposto como alternativa ao BLEU e ao METEOR para avaliação de soluções de Tradução Automática, objetiva mensurar a quantidade mínima de operações de edição que uma pessoa faria sobre um texto hipótese para transformá-lo em um texto de referência (SNOVER *et al.*, 2006). Diferente dos métodos BLEU e METEOR, cujos escores não possuem um significado isolado claro, servindo apenas para rankear soluções, o escore retornado pelo TER informa quão distante o texto hipótese está de uma das referências, do ponto de vista de esforço necessário para corrigí-lo. As operações de edição consideradas são inserção, remoção e substituição de palavras, e troca de posição de sequências de palavras, todas recebendo o mesmo peso. Seu cálculo é feito através de uma aproximação, dado que seu cálculo completo é NP-Completo. Dada a possibilidade de que haja uma quantidade enorme de boas traduções para um dado texto de entrada, e que o conjunto de textos de referência normalmente contém apenas uma pequena quantidade desse conjunto, os autores propuseram uma adaptação da TER chamada Human-targeted Translation Edit Rate (HTER), em que o texto hipótese e os textos de referência são apresentados para uma pessoa, que deverá escrever um novo texto que verbalize o mesmo conteúdo dos textos de referência e seja o mais próximo o possível do texto hipótese. Após a escrita, esse texto é utilizado como texto de referência utilizando o escore TER. Os valores de TER e de HTER variam de 0 a 1,

sendo valores menores melhor.

#### *2.2.2.4 Avaliação baseada em gramática*

Outra abordagem de avaliação automática de textos consiste no uso de uma medida que avalie apenas aspectos do texto, independente do conteúdo que ele deve verbalizar. Exemplos de medidas seguindo essa abordagem são a quantidade de erros gramaticais, a taxa de erros gerados durante o parse do texto utilizando ferramentas como o parser *Stanford* (CHEN; MANNING, 2014), e a *Flesch Reading Ease*, para avaliar a inteligibilidade de um texto, fazendo uso de medidas de tamanho das palavras e das frases (FLESCH, 1948).

## 2.3 RDF

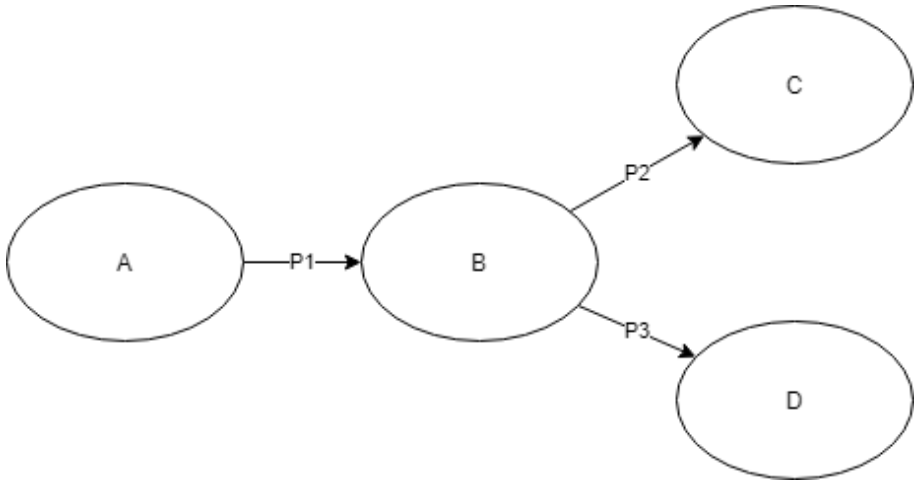
Breitman, Casanova e Truszkowski (2007) afirmam que à época havia em torno de 8 bilhões de páginas Web e que elas ainda eram, em sua maioria, desenvolvidas utilizando representações de dados mais adequadas para o consumo por pessoas e menos por computadores, como com o uso de textos. Para fazer uso desses dados utilizando computadores é então necessário um trabalho de extração dos dados dessas páginas, o que pode ser impreciso. Essa situação se reflete nos problemas que os sistemas de buscas de páginas Web sofrem por terem que lidar, no caso de busca textual, com aspectos linguísticos que apenas acidentalmente fazem parte do problema. É o caso, por exemplo, de pesquisar *"UFC"* na internet, buscando informações sobre a Universidade Federal do Ceará, e receber como resposta uma lista de sites com notícias sobre *Ultimate Fighting Championship*.

O uso de linguagem natural para representar dados incorre nesse tipo de problema, dado que é comum haver expressões cuja semântica é ambígua. Breitman, Casanova e Truszkowski (2007) caracterizam essa situação como Web Sintática, dado que a interpretação dos dados na Web são na maior parte feitas por pessoas. Em contrapartida, a Web Semântica propõe meios para facilitar o processamento dos dados disponibilizados na Web em um nível semântico. Um desses meios consiste no uso da linguagem de representação de dados RDF, que modela informações com o uso de triplas $\langle sujeito, predicado, objeto \rangle$, em que $sujeito$ é uma URI que identifica um recurso, $objeto$ é ou uma URI identificado um recurso ou um literal, e $predicado$ é uma URI que identifica um recurso que informa como $sujeito$ está relacionado com $objeto$.

Dado que um mesmo recurso pode aparecer como $objeto$ de uma tripla e $sujeito$ de

outra tripla, podemos tratar um conjunto de triplas RDF como um grafo rotulado e direcionado. Para tanto, dada uma tripla, basta considerarmos que há uma aresta de *sujeito* para *objeto*, com um rótulo identificado por *predicado*. A Figura 1 apresenta uma visualização para o seguinte conjunto de triplas $\{\langle A, P1, B\rangle, \langle B, P2, C\rangle, \langle B, P3, D\rangle\}$:

Figura 1 – Exemplo de grafo RDF




Fonte: Gerado pelo Autor.

Um exemplo de base de dados representados em RDF é a DBpedia, que na versão de 2014 continha 580 milhões de triplas sobre 4.58 milhões de recursos extraídas da Wikipedia (DBPEDIA, 2020). Outros exemplos de bases de dados em RDF públicas são a *MusicBrainz*[1], contendo dados sobre música, e a *LinkedGeoData*[2], contendo aproximadamente 20 bilhões de triplas RDF com informações geolocalizadas.

## 2.4 Modelos N-grama

Um modelo N-grama estima a probabilidade de uma palavra $w_{m+1}$ vir logo após uma sequência de palavras $w = [w_1, w_2, ..., w_m]$ utilizando o pressuposto de *Markov*, olhando apenas para suas últimas $N-1$ palavras, isto é, $[w_{m-N+1}, ..., w_m]$, denominadas contexto, enquanto a variável $N$ é a ordem do modelo. Seja $w_i^j = [w_i, w_{i+1}, ..., w_j]$, onde $i \leq j$, denota a subsequência de $w$, do i-ésimo elemento ao j-ésimo elemento; e $w_i^j w_k$ a concatenação da subsequência $w_i^j$ e a palavra $w_k$. Adiciona-se no início e no fim da sequência $w$ a sequência $[\langle s\rangle, ..., \langle s\rangle]$, com $N-1$ elementos, sendo $\langle s\rangle$ um símbolo especial utilizado para representar o contexto que há antes e

[1] https://musicbrainz.org/. Acesso em: 20 jun. 2020
[2] http://linkedgeodata.org/About. Acesso em: 20 jun. 2020

após a sequência original, e denomina-se $W$ a sequência resultante. Usando a regra da cadeia e um contexto de $N-1$ palavras, pode-se estimar a probabilidade da sequência $w$:

$$P(w) = P(W) \approx \prod_{k=N}^{|W|} P(W_k | W_{k-N+1}^{k-1}) \tag{2.1}$$

Dado que $C(W_i^j)$ seja o número de ocorrências da sequência de palavras arbitrária $W_i^j$ em um corpus, pode-se estimar $P(W_k | W_{k-N+1}^{k-1})$ utilizando estimativa por máxima verossimilhança (JURAFSKY; MARTIN, 2008):

$$P(W_k | W_{k-N+1}^{k-1}) = \frac{C(W_{k-N+1}^{k-1} W_k)}{C(W_{k-N+1}^{k-1})} \tag{2.2}$$

Técnicas de suavização podem ser utilizadas para mover parte da probabilidade dos N-gramas mais frequentes para os N-gramas que não estão presentes no corpus e que, de outra forma, receberiam probabilidade estimada igual a 0. Uma das técnicas mais simples é chamada suavização de *Laplace* e consiste em adicionar em 1 a contagem de todo N-grama possível, de forma que aqueles que não estão presentes no corpus terão uma contagem igual a 1, e aqueles presentes no corpus terão suas contagens acrescidas de 1. A suavização de *Laplace* tem como desvantagem mover muita da probabilidade de N-gramas presentes no corpus para N-gramas não presentes. Outra técnica, chamada suavização de *Kneser-Ney* (KNESER; NEY, 1995), é bastante utilizada e parte da hipótese de que quanto maior a quantidade de contextos diferentes que precedem uma palavra no corpus, maior a chance de ela aparecer após um contexto não presente no corpus (JURAFSKY; MARTIN, 2008). Ela é baseada em um desconto constante da contagem dos N-gramas presentes no corpus, adicionado de um fator que é função dessa quantidade de contextos diferentes que precedem a palavra para a qual se está calculando a probabilidade. Uma versão melhorada dela, chamada de suavização de *Kneser-Ney* modificada, consiste em usar fatores de desconto diferentes de acordo com a ordem dos N-gramas utilizada (CHEN; GOODMAN, 1999).

# 3 TRABALHOS RELACIONADOS

Antes do aumento na adoção de técnicas de Aprendizado de Máquina, era predominante o uso de soluções de GLN modulares e baseadas em regras (GATT; KRAHMER, 2018). Com a disponibilidade cada vez maior de recursos computacionais e o desenvolvimento de pesquisas sobre Aprendizado de Máquina, os estudos de soluções adotando arquiteturas integradas com uso de Aprendizagem Profunda aumentou. Esse movimento da pesquisa, de arquiteturas modulares para arquiteturas integradas, tem sido contestado recentemente por trabalhos que reportam melhores resultados de soluções mais modularizadas, como (MORYOSSEF; GOLDBERG; DAGAN, 2019) e (FERREIRA *et al.*, 2019). Este capítulo apresenta trabalhos que foram estudados no desenvolvimento desta dissertação, relacionados a GLN e, mais específico, a geração de textos a partir de dados em formato de triplas. Na Seção 3.1 são apresentados trabalhos que fazem uso de regras manualmente desenvolvidas ou de templates; já na Seção 3.2 são apresentados trabalhos que fazem uso de técnicas de Aprendizagem Profunda; finalmente, na Seção 3.3 é feito um comparativo entre os trabalhos relacionados e a solução proposta.

## 3.1 Soluções modulares baseadas em regras ou templates

Nesta Seção são apresentados trabalhos que propõem soluções que fazem uso ou de regras manualmente desenvolvidas, ou de templates.

### *3.1.1 Generating Natural Language from Linked Data: Unsupervised template extraction*

Como exemplo de solução modular, Duma e Klein (2013) propõe LOD-DEF, um sistema que gera textos curtos descrevendo entidades representadas com triplas RDF e fazendo uso de templates. Recebendo como entrada um URI de uma entidade da DBpedia, a solução proposta primeiro seleciona as 5 melhores classes às quais a entidade pertence, de acordo com um escore que leva em consideração a frequência com que os rótulos associados a uma classe aparecem na primeira sentença da Wikipedia para essa entidade; utilizando modelos de classes que especificam, para uma determinada classe de entidade quais predicados são relevantes para descrevê-la e a ordem e templates com os quais eles podem ser verbalizados, é então selecionado o modelo de classe cujos templates verbalizem a maior quantidade de triplas da entidade de entrada; esses templates são então preenchidos com uma *string* gerada a partir do $objeto$ das triplas, gerando o texto a ser retornado. São utilizadas apenas triplas cujo $sujeito$ é a entidade de

entrada, de forma que o grafo RDF composto pelas triplas utilizadas possui profundidade 1.

Os templates utilizados foram extraídos através de um processo semi-automático, primeiro identificando a ocorrência do *objeto* das triplas RDF de uma entidade no texto da Wikipedia dessa entidade, seguido de um processo automático de poda, onde foram removidas trechos do texto que continham alguma palavra funcional que não pudesse ser associado a alguma tripla da entidade; e finalmente, após substituídos no texto as referências a *objeto* das triplas, esse texto passou por uma revisão manual para ajuste de aspectos gramaticais, como correção no uso de pontuação.

A solução proposta foi avaliada por 25 pessoas, que julgaram com notas de 1 a 5 os textos gerados em relação a corretude gramatical, redundância e coerência. Ela foi comparada com textos gerados por pessoas, e com textos gerados por uma solução *baseline*, que inicializa o texto com uma sentença que informa a classe da entidade e continua com uma sentença por tripla *t*, usando a estrutura *"**pronome_possessivo t.predicado** is **t.objeto**."*, como *"Her birth date is 6 July 1958."*. Como esperado pelos autores, os textos gerados por pessoas obtiveram notas melhores que os gerados pelas duas outras soluções, sendo que a LOD-DEF obteve notas melhores em redundância e coerência que a solução *baseline*, ficando empatada com ela em relação a corretude gramatical. Os autores indicam como trabalho futuro o uso de abordagens baseadas em dados para treinamento de cada módulo do sistema. Diferente do LOD-DEF, a solução proposta nesta dissertação trabalha com grafos de profundidade maior que 1, segue uma abordagem de múltiplas soluções e faz uso de uma base de expressões de referência.

### *3.1.2 Template-based multilingual football reports generation using Wikidata as a knowledge base*

Uma das vantagens de soluções baseadas em template é a facilidades de adaptá-las para outras linguagens, como mostrado por Gatti, Lee e Theune (2018). Eles adaptaram do holandês para o inglês uma solução para geração de relatórios sobre partidas de futebol, traduzindo os templates do holandês para o inglês através de um processo semi-automático, usando uma ferramenta de Tradução Automática e manualmente corrigindo as traduções. Ela é composta por diversos módulos responsáveis por selecionar quais templates serão utilizados, de acordo com os dados disponíveis e a quantidade de informações que os templates verbalizam, e recuperá-los de uma base de dados de templates; por ordenar os dados e respectivos templates de forma cronológica; por estruturar os textos em título, introdução, sumário e conclusão; por

evitar repetição de templates e de expressões de referência. Para a tradução dos templates eles utilizaram o *Google Translate* e corrigiram tanto falhas na tradução dos textos, quanto problemas gerados pelo fato de os templates conterem lacunas e não constituírem um texto completo. Apesar dessa etapa manual de correção, os autores citam (FEDERICO; CATTELAN; TROMBETTI, 2012) e (GREEN; HEER; MANNING, 2013) que concluem que esse processo de tradução semi-automática é mais eficiente que uma tradução totalmente manual. Apesar de os autores não avaliarem a qualidade dos textos gerados para o inglês, eles sugerem que uma avaliação similar à feita com a versão do sistema para o holandês deve ser feita, avaliando a clareza, fluência e adequação dos textos.

### *3.1.3 Exploiting ontology lexica for generating natural language texts from RDF data*

O artigo (CIMIANO *et al.*, 2013) apresenta uma solução de verbalização de triplas RDF utilizando uma base de dados léxicos modelada com o formato Lexicon Model for Ontologies (lemon) (MCCRAE; SPOHR; CIMIANO, 2011) e desenvolvida manualmente, contendo formas de verbalizar, em alemão, os recursos do domínio de receitas culinárias. Para a geração de um texto detalhando uma receita culinária, os autores extraíram árvores sintáticas a partir de uma base de receitas, substituindo suas folhas por lacunas, e armazenando informações estatísticas que são utilizadas durante o processo de geração de texto.

A arquitetura da solução segue a arquitetura proposta em (REITER; DALE, 1997), a menos de que todos os dados de entrada devem ser verbalizados e a ordem de verbalização já é pré definida como a ordem das ações da receita, de forma que as tarefas Determinação de Conteúdo e Planejamento de Discurso não são executadas. As tarefas Agregação em Sentenças e Geração de Expressões de Referência são implementadas com uso de regras manualmente desenvolvidas. Já a tarefa Lexicalização é implementada utilizando informações estatísticas coletadas a partir da base de receitas, selecionando expressões para verbalizar as ações e ingredientes a partir da base de dados léxica. Por último a tarefa Realização Linguística consiste em selecionar, também baseado em informações estatísticas, uma árvore sintática a ser utilizada para verbalizar os dados, usando as expressões léxicas definidas nas etapas anteriores.

Uma das capacidades que a solução proposta possui é a de permitir a personalização do texto a partir de restrições informadas na base de dados léxicos, indicando que formas de verbalizar os conceitos é mais adequada para que perfis de texto. Nesse trabalho os autores avaliaram a geração de receitas em versões tanto para usuários leigos, como para usuários

especialistas, no domínio culinária. Para avaliar a solução proposta, 93 avaliadores, dos quais 95% possuíam alemão nativo, analisaram, cada um, os textos gerados para 20 receitas, dando notas para fluência e adequação, numa escala de 4 pontos, bem como informando se consideravam o texto mais adequado para um leigo ou para um especialista em culinária. Os autores concluem a partir dos resultados que a solução proposta é capaz de gerar textos fluentes, adequados e personalizados. A solução proposta nesta dissertação não depende do uso de uma base de dados léxica, associando os dados de entrada à forma de verbalizá-los utilizando bases de dados de templates e de expressões de referência. Além disso, ela não está restrita a um domínio apenas, como é o caso, para o domínio de receitas culinárias.

## 3.2 Soluções baseadas em Aprendizagem Profunda

Nesta Seção são apresentados trabalhos que modelam o problema de geração de textos como um problema de tradução dos dados de entradas em textos, fazendo uso de arquiteturas integradas e Aprendizagem Profunda.

### *3.2.1 Sequence-to-Sequence Models for Data-to-Text Natural Language Generation: Word- vs. Character-based Processing and Output Diversity*

Outra abordagem para geração de textos a partir de dados não linguísticos consiste em primeiro representá-los como uma sequência de palavras, e então utilizar abordagens de Tradução Automática. No artigo (JAGFELD; JENNE; VU, 2018) foram comparados modelos de Tradução Automática *sequence-to-sequence* baseados em palavras e baseados em caracteres. Além dessa comparação, os autores avaliaram a capacidade de modelos *sequence-to-sequence* gerarem textos com estruturas diferentes das estruturas dos textos utilizados no treinamento, o que seria uma vantagem em relação a soluções baseadas em templates, que geram textos com pouca diversidade. Os autores citam (ORABY *et al.*, 2018), que obteve resultados apontando que a entropia de N-gramas dos textos gerados por modelos baseados em redes neurais é significantemente menor que a dos textos utilizados no treinamento.

Para os experimentos foram utilizadas as bases de dados *WebNLG* e E2E (NOVIKOVA; DUŠEK; RIESER, 2017), ambas disponibilizadas em *shared tasks*, onde foram divididas em conjuntos de *treino*, *desenvolvimento* e *teste*; e os métodos de avaliação automática BLEU e ROUGE-L (LIN, 2004), além da realização de avaliação manual. Para seleção dos hiper-

parâmetros os modelos foram treinados com o conjunto *treino* e avaliados com o conjunto *desenvolvimento*, sendo utilizadas as combinações de hiperparâmetros que alcançaram maior escore BLEU.

Para a base de dados E2E os modelos avaliados conseguiram escores próximos dos escores obtidos pelos melhores competidores, e o modelo baseado em palavras obteve escores melhores que a versão baseada em caracteres. Já para a base de dados WebNLG o modelo baseado em palavras obteve escore BLEU superior ao do melhor competidor, enquanto o modelo baseado em caracteres obteve um maior escore de ROUGE-L que a versão baseada em palavras.

Para entender melhor o comportamento dos métodos de avaliação automática, os autores fizeram um experimento, utilizando os conjuntos *desenvolvimento* de ambas as bases de dados, em que removeram aleatoriamente um texto do conjunto de textos de referência, gerados por pessoas, que utilizaram como texto hipótese. Para a base de dados E2E ambos os modelos propostos pelos autores geraram textos com escores superiores aos textos removidos aleatoriamente, sugerindo que as medidas BLEU e ROUGE-L têm utilidade limitada para avaliar a qualidade de textos. Os autores indicam que isso ocorre porque os textos gerados por pessoas possuem maior variedade e criatividade, divergindo mais dos padrões encontrados na base de textos de referência.

Para a avaliação manual, os autores selecionaram aleatoriamente 15 entradas para cada tamanho de entrada, do conjunto *desenvolvimento*, num total de 90 entradas para a base E2E e 105 para a base WebNLG. Eles avaliaram tanto a corretude gramatical quanto adequação dos textos, constatando que de 70 a 80% dos textos gerados pelos modelos estavam gramaticalmente corretos, e que todos os modelos tiveram problemas com adequação, gerando apenas 31.4% dos textos corretamente, no caso do modelo baseado em caracteres e para a base de dados WebNLG, e no máximo 70%, no caso do modelo baseado em caracteres e para a base de dados E2E. O erro mais comum detectado foi de omissão de conteúdo. Já erros de informação adicionada ou repetida só foram constatados nos textos gerados para a base WebNLG.

Além desses experimentos, os autores treinaram um modelo *sequence-to-sequence* baseado em palavras utilizando textos gerados a partir de templates, e verificaram que não apenas o modelo conseguiu gerar textos seguindo os templates utilizados, mas também textos utilizando combinações desses templates.

### *3.2.2 Triple-to-Text: Converting RDF Triples into High-Quality Natural Languages via Optimizing an Inverse KL Divergence*

O artigo (ZHU *et al.*, 2019) explora a ideia de que é possível aumentar a qualidade dos textos gerados a partir de triplas por um modelo *sequence-to-sequence* diminuindo a diversidade dos textos gerados. No lugar de treinar o modelo com estimativa por máxima verossimilhança, o modelo é treinado buscando minimizar a inversa da divergência de *Kullback-Leibler*. A solução proposta foi avaliada utilizando as bases de dados WebNLG, SemEval-2010 (HENDRICKX *et al.*, 2010) e Baidu SKE[1], sendo comparada com *baselines* propostos pelos autores. Ela foi avaliada com os métodos de avaliação automática BLEU, METEOR e TER, e obteve melhores escores que os baselines avaliados. Da mesma forma ocorreu na avaliação manual, realizada apenas com textos gerados para a base de dados WebNLG, onde foram avaliados 20 textos gerados por cada solução, por 10 pessoas, que observaram a corretude gramatical e a adequação dos textos. Os autores concluem que a estratégia de treinamento do modelo proposta não se limita à tarefa de gerar textos a partir de triplas, e propõem como trabalhos futuros a aplicação dela em problemas como Tradução Automática.

### *3.2.3 Deep Graph Convolutional Encoders for Structured Data to Text Generation*

No lugar de simplesmente mapear um conjunto de triplas RDF em uma sequência de palavras e utilizá-la como entrada de um modelo *sequence-to-sequence*, (DISTIAWAN *et al.*, 2018) e (MARCHEGGIANI; PEREZ-BELTRACHINI, 2018) propõem usar formas diferentes de codificar triplas RDF, melhor preservando informações sobre sua estrutura de grafo. (MARCHEGGIANI; PEREZ-BELTRACHINI, 2018) faz uso de Graph Convolutional Network (GCN) como codificador, e através de experimentos utilizando as bases de dados WebNLG e SR11Deep (BELZ *et al.*, 2011), comparando com um modelo tradicional baseado em Long short-term memory (LSTM), concluem que o uso de um codificador que leve em consideração a estrutura em grafo dos dados de entrada contribui para a geração de textos com maior qualidade. Foram utilizados os métodos de avaliação automática BLEU, METEOR e TER, bem como os autores fizeram uma inspeção manual nos textos, levantando que ambas as soluções geraram textos com informações repetidas ou faltantes, sendo esses problemas menos presentes na solução proposta. Além disso eles observaram a ocorrência de alucinação, quando o texto verbaliza

[1] http://ai.baidu.com/broad/download?dataset=sked. Acesso em: 20 jun. 2020

informações não presentes nos dados de entrada, nos textos gerados pela solução baseada em LSTM.

### *3.2.4 GTR-LSTM: A Triple Encoder for Sentence Generation from RDF Data*

Já Distiawan *et al.* (2018) propõem um codificador de triplas que busca preservar informações de relacionamentos entre as triplas, utilizando ele com uma arquitetura LSTM, que eles denominaram *GTR-LSTM*. Além disso, as referências a entidades nos textos utilizados foram substituídas por marcadores contendo um identificador e classes às quais as entidades pertencem, o que eles chamaram de mascaramento de entidades. Por exemplo, *"John Doe"* foi substituído por *"ENT-1 PERSON GOVERNOR"*, em que *"ENT-1"* identifica a entidade e *"PERSON"* e *"GOVERNOR"* são classes às quais a entidade *"John Doe"* pertence. Os autores argumentam que esse pré-processamento auxilia o modelo a lidar melhor com entidades não presentes na base de dados de treino.

Para identificar as referências a entidades nos textos os autores utilizaram três estratégias, baseadas em correspondência exata entre partes do texto e a representação das entidades nas triplas, em similaridade de strings baseada em N-grama, e utilizando a árvore de parse do texto.

Para avaliar a solução proposta, os autores utilizaram tanto a base de dados WebNLG, quanto uma base de dados por eles desenvolvida, consistindo de pares de triplas RDF e textos extraídos da Wikipedia e posteriormente filtrados por pessoas para remover textos que não se referem aos dados das respectivas triplas. Comparando com modelos tradicionais de Aprendizagem Profunda e de Tradução Automática Estatística, e utilizando os métodos de avaliação BLEU, METEOR e TER, os autores obtiveram melhores escores com a solução proposta. Além disso, eles também treinaram um modelo sem realizar o mascaramento de entidades, e concluíram que esse pré-processamento também contribui para a geração de textos de maior qualidade, principalmente quando eles se referem a entidades não presentes na base de dados utilizada no treinamento, além de melhorar o tempo de execução do modelo, ao exigir um menor vocabulário.

Os autores também conduziram uma avaliação manual das soluções analisadas, solicitando a 5 pessoas que analisassem os textos gerados por cada solução, para um total de 100 entradas aleatoriamente selecionadas, em relação aos aspectos adequação, corretude gramatical e fluência, utilizando uma escala de 1 a 3. Novamente, a solução proposta obteve os melhores escores. Finalmente, para entender melhor a qualidade da solução proposta, os autores também

selecionaram aleatoriamente 100 entradas e analisaram os textos gerados, encontrando que 8% dos textos continham informações erradas e 15% continham informações duplicadas

### *3.2.5 A Semi-Supervised Approach for Low-Resourced Text Generation*

Modelos *sequence-to-sequence* dependem de uma grande quantidade de dados rotulados, o que pode ser uma limitação, dado o custo de rotulá-los. Pensando nisso e na maior disponibilidade de dados não rotulados, Zang e Wan (2019) exploram uma estratégia de treinamento semi-supervisionado de modelos *sequence-to-sequence* fazendo uso dos dados não rotulados com técnicas de Aprendizado por Reforço.

Para avaliar a solução proposta, os autores realizaram experimentos utilizando as bases de dados WebNLG e Newsela (XU; CALLISON-BURCH; NAPOLES, 2015), esta última voltada para a tarefa de simplificação de texto. Os métodos de avaliação automáticos utilizados para avaliar os textos gerados para a base WebNLG foram BLEU, METEOR, TER e perplexidade. Além da avaliação automática, foi conduzida uma avaliação manual, em que foram selecionados aleatoriamente 50 textos gerados pelos modelos treinados e submetidos a avaliação por 5 pessoas, utilizando o *Amazon Mechanical Turk*[2], devendo elas dar um escore de 1 a 5 de acordo com quão bem o texto verbalizava os dados de entrada.

A solução fazendo uso de Aprendizagem por Reforço obteve escores melhores, comparado à versão sem Aprendizagem por Reforço, tanto em relação à avaliação automática, quanto na avaliação manual. Para entender melhor o impacto, na qualidade dos textos, da quantidade de dados rotulados e de dados não rotulados, os autores treinaram a solução proposta fixando a quantidade de dados rotulados e variando a quantidade de dados não rotulados, e vice-versa. Em todos os casos o aumento na quantidade de dados utilizados no treinamento do modelo levou a um aumento no escore BLEU, a menos de quando a solução foi treinada com uma quantidade de dados rotulados fixa e usou 10.000 e 14.000 dados não rotulados. Diferente do esperado, a solução treinada com 10.000 dados não rotulados obteve resultado melhor, no que os autores levantaram a hipótese de ser por conta de uma limitação na função de recompensa utilizada.

[2] https://www.mturk.com/. Acesso em: 20 jun. 2020

### *3.2.6 Neural data-to-text generation: A comparison between pipeline and end-to-end architectures*

Apesar de haver resultados indicando que soluções adotando arquitetura integrada com técnicas de Aprendizagem Profunda geram textos mais fluentes que soluções baseadas em templates (PUZIKOV; GUREVYCH, 2018), trabalhos mais recentes propuseram o uso de técnicas de Aprendizagem Profunda com arquiteturas mais modulares, explorando a ideia de que modelar o problema de geração de textos em subproblemas mais simples pode evitar problemas que têm sido reportados, como a geração de textos contendo informações repetidas ou informações não presentes no dado de entrada (WISEMAN; SHIEBER; RUSH, 2017). O artigo (FERREIRA *et al.*, 2019) explora a geração de textos a partir de triplas comparando o uso das técnicas de Aprendizagem Profunda Gated Recurrent Unit (GRU) (CHO *et al.*, 2014) e *Transformer* (VASWANI *et al.*, 2017) para implementar os módulos de uma arquitetura com estrutura em *pipeline*, inspirada nas tarefas definidas em (REITER; DALE, 1997), e para implementar uma arquitetura integrada.

Para os experimentos, utilizaram a base de dados WebNLG e realizaram avaliação tanto com métodos automáticos, quanto com métodos manuais. Para a avaliação automática foram utilizadas os métodos BLEU e METEOR, e as soluções utilizando arquitetura modular alcançaram melhores resultados que as soluções utilizando arquitetura integrada. Para a avaliação manual foi utilizado o *Amazon Mechanical Turk*, recrutando 35 avaliadores, que ficaram responsáveis por dar escores de 1 a 7, sobre a fluência e a semântica (se o texto expressava claramente o conteúdo das triplas), para os textos gerados pelas soluções propostas e concorrentes, para 223 conjuntos de triplas.

A solução UPF-FORGe, baseada em templates, e proposta na *shared task* WebNLG, foi a melhor colocada, seguida das soluções propostas com arquitetura em *pipeline*. Além disso, os autores analisaram os textos que essas soluções geraram para 75 conjuntos de triplas, avaliando a ocorrência de problemas gramaticais e, no caso das soluções com arquitetura em *pipeline*, se os textos gerados respeitaram as decisões tomadas em cada módulo. Novamente, os textos gerados pela solução UPF-FORGe foram os melhores avaliados, não possuindo erros gramaticais e em 91% dos casos verbalizando corretamente os dados das triplas. Os autores observaram que menos da metade dos textos gerados com os modelos utilizando arquitetura integrada verbalizaram todas as triplas de entrada, além de em vários casos verbalizarem informações não presentes na entrada.

### *3.2.7 Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation*

O artigo (MORYOSSEF; GOLDBERG; DAGAN, 2019) também propõe uma arquitetura em *pipeline* para geração de texto a partir de triplas, mas com apenas dois módulos, sendo o primeiro responsável por planejar o texto, definindo a ordem com que as triplas serão verbalizadas, como elas serão agrupadas em sentenças e se serão verbalizadas na voz passiva ou na voz ativa, e o segundo módulo responsável por transformar esse plano em um texto.

O primeiro módulo foi implementado como um modelo que rankeia todos os possíveis planos de textos da entrada e seleciona o melhor, de acordo com um escore calculado como o produto de probabilidades de algumas características dos textos, como a frequência com que as triplas eram agrupadas em sentenças. Diferente de (FERREIRA *et al.*, 2019), apenas o último módulo é implementado com técnicas de Aprendizagem Profunda.

Para a execução dos experimentos eles utilizaram a base de dados *WebNLG*, usando na avaliação dos textos gerados tanto métodos automáticos, como métodos manuais. Para a avaliação automática utilizaram a ferramenta *nlg-eval* (SHARMA *et al.*, 2017), reportando os resultados para os métodos BLEU, ROUGE-L (LIN, 2004), METEOR e CIDEr (VEDANTAM; ZITNICK; PARIKH, 2015). Para a avaliação manual, os autores analisaram os textos gerados pela solução proposta para um conjunto de 139 entradas, e contaram as ocorrências de omissão de dados, dados verbalizados incorretamente e repetições.

Assim como (FERREIRA *et al.*, 2019), eles também obtiveram resultados que sugerem que arquiteturas mais modularizadas performam melhor que arquiteturas integradas, apontando que isso ocorre porque uma arquitetura modular distribui as responsabilidades do processo de geração nos módulos, criando subproblemas que são mais simples de serem resolvidos individualmente. Mais especificamente, o módulo implementado com técnicas de Aprendizagem Profunda, diferente de outros trabalhos, não possui responsabilidades como a de ordenar e particionar os dados. Outra conclusão desse artigo é a de que arquiteturas modulares permitem um maior controle sobre os textos gerados. A solução proposta nesta dissertação, diferente da proposta nesse trabalho, não leva em consideração a decisão sobre que voz verbal utilizar para verbalizar os dados.

### *3.2.8 Automated learning of templates for data-to-text generation: comparing rule-based, statistical and neural methods*

O artigo (LEE; KRAHMER; WUBBEN, 2018) explora a ideia de combinar o uso de templates e modelos baseados em dados, utilizando modelos de Aprendizagem Profunda ou de Tradução Automática Estatística para gerar templates a partir de dados de entrada, no lugar de gerarem logo o texto a ser retornado. Esses templates são então preenchidos, num passo seguinte, com referências às entidades dos dados de entrada.

Para avaliar essa ideia, eles utilizaram modelos de Aprendizagem Profunda, utilizando o OpenNMT (KLEIN *et al.*, 2017), e de Tradução Automática Estatística, utilizando o Moses *toolkit* (KOEHN *et al.*, 2007), ambos treinados tanto numa versão integrada, quanto na versão proposta, com dois módulos, um para gerar um template e outro para preencher suas lacunas. Foram utilizadas 4 bases de dados, sendo 2 delas contendo textos gerados automaticamente por sistemas e 2 delas contendo textos escritos por pessoas. Foi utilizado BLEU para avaliação automática. Já para avaliação manual os textos gerados foram submetidos à avaliação por 24 pessoas, em relação à fluência, clareza e adequação.

Os resultados obtidos de BLEU sugerem que modelos baseados em Aprendizado de Máquina têm melhores desempenhos quando avaliados utilizando bases de dados geradas por computador, sendo uma das possíveis explicações o fato de que o problema, nesse caso, se torna realizar uma engenharia reversa do programa que gerou os textos. Além disso, as versões dos modelos que fizeram uso de templates obtiveram escores BLEU similares ou menores do que as versões que não fizeram uso. Já na avaliação manual as versões dos modelos que fizeram uso de templates obtiveram os melhores resultados em relação aos três aspectos avaliados, o que os autores indicam sugerir que o uso de templates pode contribuir para uma maior qualidade dos textos gerados.

## 3.3 Comparativo entre as soluções

A tabela 1 apresenta um comparativo entre os trabalhos relacionados e a solução proposta nesta dissertação, de acordo com os atributos “Profundidade do grafo de triplas", que indica se a solução possui restrição quanto à profundidade do grafo de triplas de entrada; “Regras específicas de um domínio", que indica se a solução faz uso de regras manualmente desenvolvidas e específicas de um determinado domínio; “Garantias de adequação", que indica se a solução

fornece garantias de que o texto gerado irá verbalizar todos e apenas os dados de entrada; e “Múltiplas soluções", que indica se a solução gera múltiplos textos, ao fim do qual seleciona um a ser retornado. Os trabalhos relacionados que adotam Aprendizagem Profunda e foram descritos na Seção 3.2 estão todos representados pela linha identificada por “Aprendizagem Profunda", dado que eles compartilham os mesmos valores de atributos.

Tabela 1 – Comparativo entre trabalhos relacionados e a solução proposta.

| | **Profundidade do grafo de triplas** | **Regras específicas de um domínio** | **Garantias de adequação** | **Múltiplas soluções** |
|---|---|---|---|---|
| (DUMA; KLEIN, 2013) | No máximo 1 | Não | Sim | Não |
| (CIMIANO *et al.*, 2013) | Sem restrição | Sim | Sim | Não |
| Aprendizagem Profunda | Sem restrição | Não | Não | Não |
| **Solução proposta** | Sem restrição | Não | Sim | Sim |

Como se pode verificar, a solução proposta é a única que reúne as seguintes características: recebe como entrada grafos de triplas de profundidades arbitrárias, permitindo de grafos simples, com profundidade 1, a grafos com estruturas mais complexas; não faz uso de regras manualmente desenvolvidas e específicas de um domínio, tendo todo seu comportamento treinado a partir de uma base de dados de treinamento; fornece garantias de adequação, ao fazer uso de templates que garantem que todo o dado de entrada será verbalizado; e segue uma abordagem de múltiplas soluções, em que cada decisão explora uma ou mais possíveis soluções, gerando, ao fim do processo, um ou mais textos, dos quais é selecionado aquele que será retornado.

# 4 ARQUITETURA BASEADA EM TEMPLATES E ABORDAGEM DE MÚLTIPLAS SOLUÇÕES

A arquitetura utilizada neste trabalho faz uso de templates como meio para geração dos textos a partir de dados em formato de triplas, bem como, durante seu processo, explora diversas possibilidades de gerar um texto, variando, por exemplo, a ordem com que os dados são verbalizados. Esse processo é estruturado em módulos, cada qual responsável por uma tarefa e capaz de retornar uma ou mais saídas, selecionadas com o uso de modelos N-grama.

Este Capítulo é estruturado da seguinte forma: a Seção 4.1 motiva o uso da arquitetura adotada e a descreve; as Seções 4.2 e 4.3 definem triplas e templates, respectivamente, que são as principais estruturas de dados utilizadas na solução; finalmente, a Seção 4.4 detalha o funcionamento de cada módulo, através de um exemplo.

## 4.1 Motivação

O problema de gerar um texto para verbalizar um determinado conjunto de dados de entrada pode ser entendido como o problema de buscar uma sequência de palavras, dentro de um conjunto de sequências possíveis, garantindo que essa sequência satisfaça requisitos como verbalizar os dados de entrada, ser fluente e ser gramaticalmente correta. Considerando, por exemplo, um vocabulário hipotético contendo $10^3$ palavras diferentes[1], e que queremos gerar uma sentença com 10 palavras. Há portanto $10^{30}$ sequências possíveis de serem formadas. Dado que se possua um critério para determinar quais dessas sequências verbalizam os dados de entrada, é possível adotar uma abordagem de força bruta, gerando todas as sequências possíveis e retornando a primeira válida. Ocorre que a maior parte das sequências possíveis sequer constituem textos gramaticalmente corretos, como é o caso das sequências formadas pela mesma palavra repetida 10 vezes. Uma forma de reduzir esse espaço de busca é usar templates de sentenças, que podem ser considerados sequências de palavras com lacunas, que, se corretamente preenchidas, geram sentenças válidas. O processo de geração do texto pode ser então definido como um conjunto de passos: (1) decidir a ordem em que os dados serão verbalizados; (2) decidir quais deles serão verbalizados em uma mesma sentença; (3) decidir que templates serão utilizados para cada sentença; (4) finalmente, decidir que valores serão utilizados para preencher as lacunas dos templates.

[1] Para comparação, o dicionario de inglês de Oxford contém em torno de 600.000 palavras (DICTIONARY, 2020).

Considere o cenário hipotético, com os dados representados como triplas $\langle sujeito, predicado, objeto \rangle$:

1. Para qualquer sequência de triplas existem $\boldsymbol{T}$ templates diferentes que podem ser utilizados para verbalizá-las;
2. Cada template possui uma lacuna para o $sujeito$ e uma para o $objeto$ para cada tripla que ele verbalize (por exemplo, um template que verbalize uma sequência de 3 triplas terá 6 lacunas);
3. Para qualquer lacuna em um template há $\boldsymbol{R}$ valores diferentes que podem ser utilizados para preenchê-la.

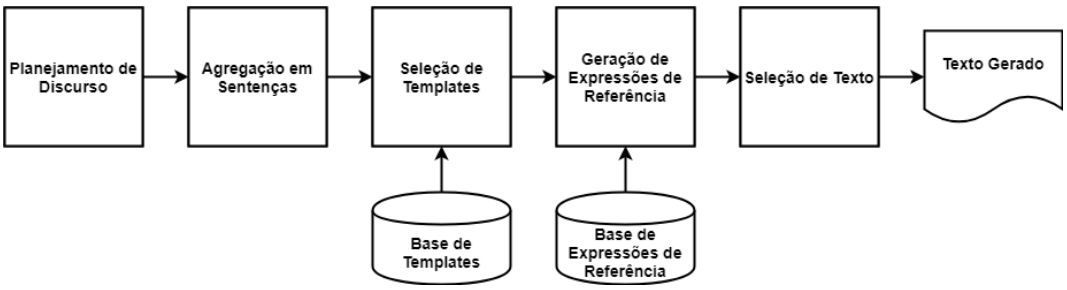
A quantidade total de textos que poderão ser gerados a partir de um conjunto de triplas com tamanho $n$ é dada pela seguinte fórmula:

$$n!R^{2n}\sum_{i=0}^{n-1}\binom{n-1}{i}T^{(i+1)} \tag{4.1}$$

Considerando, por exemplo, $n = 5$, $T = 3$ e $R = 3$, temos um total de 113.374.080 textos possíveis. Mesmo com o auxílio de templates, ainda há cenários em que a quantidade de textos que verbalizam um dado de entrada é muito grande. Se o único objetivo do texto for verbalizar os dados de entrada e dado que esse processo apresentado necessariamente gere um texto que verbalize os dados de entrada, não há necessidade de explorar todos os textos que podem ser gerados. Ocorre que outros requisitos podem existir, como o de que o texto deve ser fluente ou possuir um vocabulário mais adequado para um determinado público alvo. Isso faz com que simplesmente retornar o primeiro texto explorado pode não ser a melhor solução e disso surge a necessidade de explorar mais textos. Uma forma de diminuir a quantidade de textos explorados consiste em explorar apenas um subconjunto das possíveis decisões em cada etapa. Explorar, por exemplo, apenas 10 formas de ordenar as 5 triplas, no lugar de explorar todas as $5! = 120$ formas. Para maximizar a chance de que, gerando apenas um pequeno subconjunto do espaço de textos, seja possível selecionar textos que satisfaçam os requisitos, as decisões que serão exploradas em cada etapa do processo de geração devem ser bem selecionadas. É baseado nessa ideia que este trabalho propõe a arquitetura discutida nesta dissertação.

A arquitetura adotada na solução e apresentada na Figura 2 é modular, em formato de *pipeline*, faz uso de templates e lida com o problema de gerar um texto para verbalizar um conjunto de triplas de entrada. Seus módulos foram definidos com base na divisão em tarefas definidas em (REITER; DALE, 1997): são utilizados módulos específicos para as tarefas Planejamento de Discurso, Agregação em Sentenças e Geração de Expressões de Referência; não

Figura 2 – Arquitetura da solução




Fonte: Elaborado pelo autor

há módulo para a tarefa Determinação de Conteúdo, dado que o problema abordado neste trabalho consiste em verbalizar todo o conteúdo da entrada; e as tarefas Lexicalização e Realização Linguística são executadas pelo módulo Seleção de Templates, que é responsável por selecionar os templates a serem utilizados para gerar os textos. Sendo uma arquitetura de múltiplas soluções, cada módulo recebe como entrada ou o conjunto de triplas de entrada, ou a saída do módulo anterior, e retorna uma ou mais possíveis saídas. Por exemplo, o módulo Planejamento de Discurso, responsável por decidir em que ordem as triplas de entrada serão verbalizadas, recebe como entrada o conjunto de triplas de entrada e retorna uma ou mais formas de ordená-las. Ao fim do processo, são gerados um ou mais textos que são então rankeados pelo módulo Seleção de Texto, sendo retornado o texto melhor rankeado.

## 4.2 Tripla

O problema abordado neste trabalho possui como dados de entrada um conjunto de triplas $t = \langle sujeito,\ predicado,\ objeto \rangle$, onde $sujeito$, $predicado$ e $objeto$ são *strings*, e representam que a informação identificada por $sujeito$ está relacionada com a informação identificada por $objeto$ através da relação identificada por $predicado$. Por exemplo, pode-se expressar a informação verbalizada em *"O Abelardo possui o título de mestre."* através de uma tripla $\langle Abelardo,\ titulo,\ mestre \rangle$. Esta é a definição de tripla adotada neste trabalho, que é mais abrangente que a de uma tripla RDF, dado que ela segue essa definição, mas também segue um conjunto de regras que detalham sua sintaxe e semântica, como, por exemplo, o uso de ontologias e a identificação de recursos usando URI. Dessa forma, além de a solução proposta aceitar como entrada triplas RDF, ela também aceita, por exemplo, tabelas de dados, dado que estas sejam representadas como triplas de dados.

O fato de este trabalho adotar essa definição mais ampla que a de uma tripla RDF

gera alguns desafios relacionados à imprecisão dos dados. Por exemplo, não há consistência sobre se um *predicado* informa um relacionamento ainda vigente ou que já foi concluído. É o caso do predicado *club*, utilizado na base de dados WebNLG, que é verbalizado tanto como indicando que o jogador identificado pelo $sujeito$ da tripla atualmente é jogador do clube identificado pelo $objeto$ da tripla, como em *"Christian Panucci plays at the A.S. Roma."*, quanto como se ele tivesse sido jogador do clube, mas essa relação já foi interrompida, como em *"Christian Panucci played football for Genoa C.F.C."*. Outro problema, exemplificado por esse mesmo *predicado club*, é o de que há *predicado* com mais de um significado: *club* pode tanto indicar um relacionamento de "jogador do clube", quanto de "técnico do clube", como no texto *"Christian Panucci has been manager of A S Livorno Calcio."*. O Capítulo 5 discute mais sobre esses problemas de imprecisão da representação de dados da base de dados utilizada nos experimentos e o impacto desses problemas na geração de textos.

Neste trabalho utilizaremos a seguinte notação para uma tripla:
$t = \langle sujeito,\ predicado,\ objeto \rangle$, tal que $t.sujeito$ é o sujeito da tripla; $t.predicado$ é o predicado da tripla; e $t.objeto$ é o objeto da tripla.

## 4.3 Template

Uma das principais características da arquitetura utilizada neste trabalho é o uso de templates de sentenças[2]. Neste trabalho, um template é definido como uma tupla $\langle s,\ Z \rangle$, onde $s = [\omega_1, \omega_2, ..., \omega_n]$ é a estrutura do template e $Z$ é o texto do template. A estrutura do template especifica quais sequências de triplas podem ser verbalizadas pelo template e é composta por uma sequência de triplas $\langle sujeito,\ predicado,\ objeto \rangle$, em que ambos o $sujeito$ e o $objeto$ são variáveis que identificam lacunas no texto, representadas por $slot\_i, i > 0$, e *predicado* é uma *string* identificado uma relação entre informações. Por exemplo, considere o seguinte template $\langle s,\ Z \rangle$:

$\langle s = [\langle$ **slot_1**, *birthDate*, **slot_2**$\rangle$,
$\langle$ **slot_1**, *birthPlace*, **slot_3**$\rangle$,
$\langle$ **slot_3**, *capital*, **slot_4**$\rangle]$,
$Z =$"**slot_1** *was born in* **slot_4**, **slot_3**, *on* **slot_2**."$\rangle$

[2] Para simplificar, template de sentença será referido simplesmente por template.

A estrutura desse template define que ele pode ser utilizado para verbalizar uma sequência com 3 triplas, em que o *predicado* das triplas devem ser, respectivamente, *birthDate*, *birthPlace* e *capital*; além disso, o *sujeito* das duas primeiras triplas devem ser iguais, o que é representado pela variável **slot_1** sendo utilizada como *sujeito* das duas primeiras triplas da estrutura do template. De forma similar, a variável **slot_3** define que o *objeto* da segunda tripla deve ser o *sujeito* da primeira tripla. Já o texto do template é uma *string* com lacunas que fazem referência às variáveis da estrutura do template. Para verbalizar uma lista de triplas, é necessário primeiro alinhar a estrutura do template com essa lista de triplas, associar cada variável a um valor de *sujeito* ou de *objeto* nessas triplas e decidir como eles serão verbalizados, preenchendo então as lacunas do texto do template.

Dada a sequência de triplas:

$[\langle Barack\_Obama,\ birthDate,\ 1961\text{-}8\text{-}4\rangle,$
$\langle Barack\_Obama,\ birthPlace,\ Hawaii\rangle,$
$\langle Hawaii,\ capital,\ Honolulu\rangle]$

O seguinte mapeamento entre as variáveis da estrutura do template e os *sujeito* e *objeto* da sequência de triplas pode ser definido: $slot\_1 = Barack\_Obama$, $slot\_2 = 1961\text{-}8\text{-}4$, $slot\_3 = Hawaii$, $slot\_4 = Honolulu$.

Para gerar então a sentença para verbalizar essas triplas, basta que sejam selecionadas expressões de referência para os *sujeito* e *objeto* das triplas, e substituir no texto do template, de acordo com o mapeamento. Por exemplo, pode-se referenciar a entidade *Barack_Obama* com a expressão *"Barack Obama"*; as entidades *Honolulu* e *Hawaii* com os próprios valores; e o valor *1961-8-4* com "*August 4, 1961*". Substituindo, no texto do template, as variáveis pelas expressões de referência, o seguinte texto: "*Barack Obama was born in Honolulu, Hawaii, on August 4, 1961.*" é gerado.

A verificação de se um template é adequado para verbalizar uma lista de triplas consiste em checar se essa lista de triplas está estruturada de acordo com a estrutura do template. Mais formalmente, um template $\langle s,\ Z\rangle$ é considerado adequado para verbalizar uma sequência de triplas $T = [t_1, t_2, ..., t_m]$ se:

- $|T| = |s|$
- $\forall i \in [1, |T|]\ s[i].predicado = T[i].predicado$

- $\forall i, j \in [1, |T|]\ s[i].sujeito = s[j].sujeito \rightarrow T[i].sujeito = T[j].sujeito$
- $\forall i, j \in [1, |T|]\ s[i].objeto = s[j].objeto \rightarrow T[i].objeto = T[j].objeto$
- $\forall i, j \in [1, |T|]\ s[i].sujeito = s[j].objeto \rightarrow T[i].sujeito = T[j].objeto$

Isto é, a sequência de triplas tem o mesmo tamanho da estrutura do template; suas triplas possuem os mesmos predicados; e se há duas triplas em $s$ que possuem o mesmo $sujeito$, o mesmo $objeto$, ou o $sujeito$ de uma é igual ao $objeto$ da outra, o mesmo deve ocorrer com as respectivas triplas em $T$.

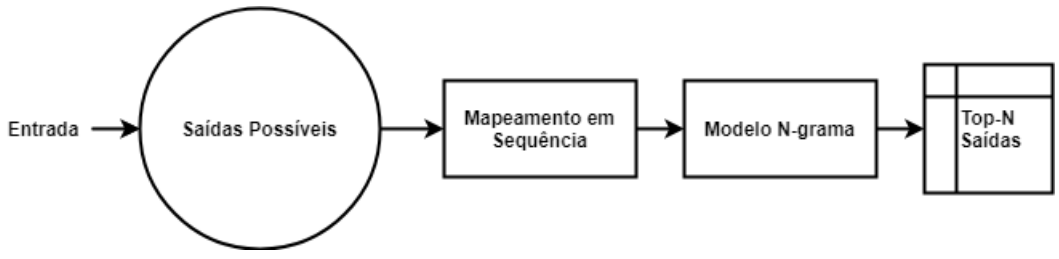
## 4.4 Módulos

Dado que são utilizados templates, o processo de gerar um texto foi definido como:

1. Definir a ordem com que as triplas serão verbalizadas;
2. Particionar as triplas de forma que cada parte será verbalizada em uma sentença;
3. Selecionar um template para cada sentença;
4. Selecionar como serão referenciadas as entidades e literais;
5. Preencher os templates, compondo o texto final.

A arquitetura proposta, por ser de múltiplas soluções, retorna em cada módulo uma quantidade de saídas definida por um parâmetro, e elas se tornam entrada do módulo seguinte. Para tanto, dada uma entrada, cada módulo irá expandir todas as possíveis saídas, mapeá-las em sequências, que serão utilizadas como entrada para o modelo N-grama do módulo e, rankeando elas de acordo com as probabilidades retornadas, são então selecionadas as melhores para serem retornadas. Esse processo está esquematizado na Figura 3. Ao fim desse processo são gerados um ou mais textos, que são então submetidos a um rankeamento, sendo retornado o texto melhor rankeado.

Figura 3 – Ideia geral de funcionamento dos módulos




Fonte: Elaborado pelo autor

Os modelos N-grama utilizados em cada módulo são treinados da maneira usual,

utilizando como base de dados um conjunto de sequências de *tokens*. Eles diferirão em relação a que *tokens* serão utilizados e seus significados. Por exemplo, o modelo N-grama utilizado pelo módulo Seleção de Texto é um modelo N-grama comumente utilizado para modelagem de linguagem, em que os *tokens* são palavras. Já o modelo N-grama utilizado pelo módulo Planejamento de Discurso lida com sequência de *tokens* que identificam *predicado* de triplas. Mais detalhes sobre o processo de extração dessas sequências são apresentados no Capítulo 5, na Seção 5.3.

A seguir, para melhor explicar o funcionamento de cada módulo e a abordagem de múltiplas soluções da arquitetura, será detalhada a geração do texto para um dado conjunto de triplas que faz parte da base de dados utilizada nos experimentos e contém as seguintes triplas, sobre o astronauta *Elliot See*:

$\{\langle Elliot\_See,\ almaMater,\ University\_of\_Texas\_at\_Austin\rangle,$
$\langle St.\_Louis,\ leaderName,\ Francis\_G.\_Slay\rangle,$
$\langle Elliot\_See,\ deathPlace,\ St.\_Louis\rangle\}$

Os seguintes textos serão utilizados como textos de referência para esse conjunto de triplas:

- "*Elliot See, a past graduate of the University of Texas, Austin, died in St Louis where the leader is Francis G Slay.*"
- "*Elliot See, who attended the University of Texas at Austin, died in St. Louis, whose Mayor is Francis G. Slay.*"
- "*Elliot See was a student at University of Texas at Austin and died in St. Louis led by Francis G. Slay.*"

Eles diferem, por exemplo, em relação à forma com que as triplas de *predicado* *almaMater* e *leaderName* são verbalizadas, bem como em relação ao uso de um ponto após abreviações (*"St Louis"* e *"St. Louis"*, por exemplo).

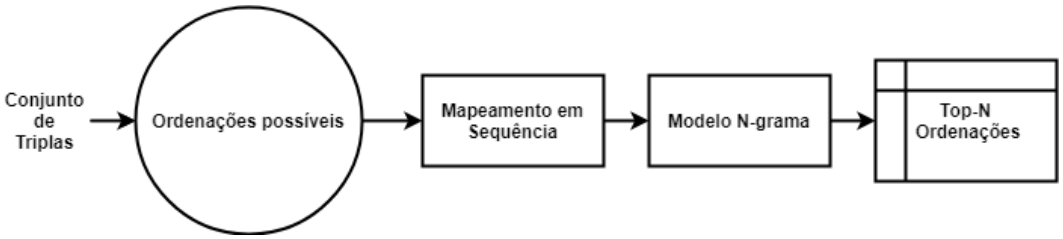
### *4.4.1 Módulo Planejamento de Discurso*

O módulo Planejamento de Discurso recebe como entrada um conjunto de triplas e tem como objetivo definir a ordem com que essas triplas serão verbalizadas. O número de

ordenações possíveis para os elementos de um conjunto $X$ qualquer é $|X|!$. O conjunto de triplas do exemplo possui 3 triplas e portanto 3! = 6 formas diferente de ordená-las.

Para definir quais ordenações serão retornados, todas as ordenações possíveis são geradas e então rankeadas, sendo retornadas apenas as $max\_dp$ melhores, onde $max\_dp$ é um parâmetro do módulo. Para rankear uma ordenação de triplas, a solução faz uso de um modelo N-grama treinado sobre sequências de *predicado*, utilizando o escore que ele retorna para a sequência de *predicado* das triplas. Esse processo está esquematizado na Figura 4.

Figura 4 – Funcionamento do módulo Planejamento de Discurso




Fonte: Elaborado pelo autor

Para o exemplo, as seguintes sequências serão geradas (estão rankeadas do maior escore para o menor):

1. [*almaMater, deathPlace, leaderName*]
2. [*deathPlace, leaderName, almaMater*]
3. [*leaderName, almaMater, deathPlace*]
4. [*leaderName, deathPlace, almaMater*]
5. [*leaderName, deathPlace, almaMater*]
6. [*almaMater, leaderName, deathPlace*]

Considerando $max\_dp = 2$, apenas as duas ordenações melhor rankeadas serão retornadas para o próximo módulo.

A definição da ordem de verbalização das triplas considerando apenas seus predicados ignora informações que possivelmente poderiam contribuir para a geração de textos melhores, como a existência de um encadeamento de triplas, quando o $objeto$ de uma tripla é o $sujeito$ da próxima tripla na ordem. Outra limitação é a possibilidade de haver duas triplas que tenham o mesmo *predicado*. Por exemplo, para as triplas:

$$\{\langle Adirondack\_Regional\_Airport,\ cityServed,\ Lake\_Placid, \_New\_York\rangle,$$
$$\langle Adirondack\_Regional\_Airport,\ cityServed,\ Saranac\_Lake, \_New\_York\rangle\}$$

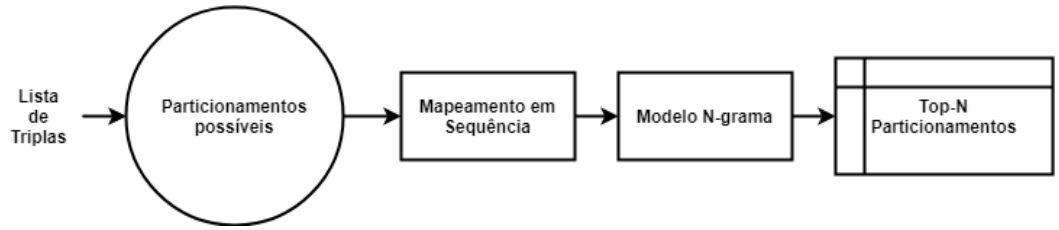
As duas formas possíveis de ordená-las receberão escores iguais e o rankeamento final será aleatório.

### *4.4.2 Módulo Agregação em Sentenças*

O módulo Agregação em Sentenças recebe como entrada uma lista de triplas e tem como objetivo definir como essa lista será particionada, de forma que cada parte seja verbalizada por uma sentença. O número de formas de particionar uma lista de triplas $T$ qualquer é $2^{|T|-1}$. Para uma lista de 5 triplas, por exemplo, há um total de 16 formas diferentes de particioná-la.

Para definir quais formas de particionar a lista de entrada serão retornadas, todas as formas de particioná-la possíveis são geradas e então rankeadas, sendo retornadas apenas as *max_sa* melhores, onde *max_sa* é um parâmetro do módulo. Para rankear uma forma de particionamento, a solução utiliza um modelo N-grama treinado sobre sequências de predicados e dos símbolos "_", representando que os predicados adjacentes aparecem na mesma sentença, e "|", representando que os *predicado* adjacentes aparecem em sentenças separadas. Por exemplo, dado que deve-se particionar uma sequência com duas triplas, com os seguintes predicados $[birthDate,\ birthPlace]$, pode-se tanto colocar cada tripla em uma sentença, resultando na sequência $[birthDate,\ \_,\ birthPlace]$, quanto colocá-las em sentenças separadas, resultando na sequência $[birthDate,\ |,\ birthPlace]$. O escore do modelo N-grama sobre essa sequência é então utilizado para rankear a forma de particionar as triplas. Esse processo está esquematizado na Figura 5.

Figura 5 – Funcionamento do módulo Agregação em Sentenças




Fonte: Elaborado pelo autor

Considerando que a ordenação melhor rankeada pelo módulo anterior é $[almaMater,\ deathPlace,\ leaderName]$, as formas de particioná-la são (estão rankeadas do maior escore para o menor):

1. [*almaMater, _, deathPlace, _, leaderName*]
2. [*almaMater, _, deathPlace, |, leaderName*]

3. [*almaMater,* |*, deathPlace, _, leaderName*]
4. [*almaMater,* |*, deathPlace,* |*, leaderName*]

Considerando $max\_sa = 2$, apenas as duas primeiras formas de particionamento serão retornadas para o próximo módulo.

Similar ao módulo Planejamento de Discurso, ao levar em consideração apenas os *predicado* das triplas, este módulo ignora informações que podem ser relacionadas a aspectos do texto final. Além disso, triplas com *predicado* iguais serão tratadas como triplas iguais.

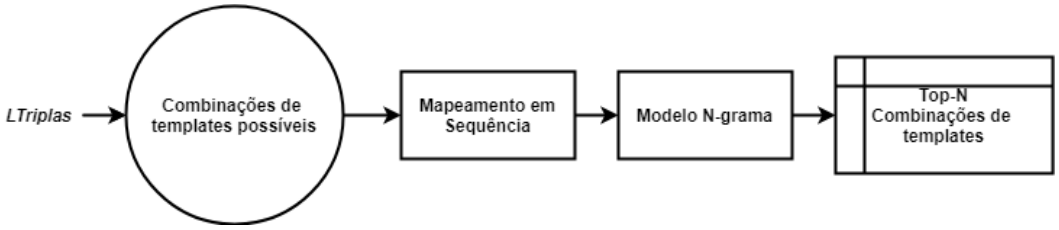
### *4.4.3 Módulo Seleção de Templates*

O módulo Seleção de Templates recebe como entrada $LTriplas$, uma lista de listas de triplas, e tem como objetivo retornar listas de templates $LTemplates$ de forma que $|LTemplates| = |LTriplas|$ e $LTemplates[i]$ é adequado para $LTriplas[i]$, $\forall i \in [1, |LTriplas|]$. Supondo que $|LTriplas| = 4$, e portanto serão verbalizadas 4 listas de triplas; que para cada lista de triplas em $LTriplas$ haja 3 templates adequados. Haverá então um total de $3^4 = 81$ combinações de templates para verbalizar $LTriplas$. Para controlar o total de combinações a serem retornadas, o parâmetro $max\_tems$ é utilizado, limitando o máximo de templates a serem considerados, por lista de triplas. No exemplo, com $max\_tems = 2$, haveria no máximo 2 templates por lista de triplas, totalizando $2^4 = 16$ combinações de templates.

Além de limitar o número de templates utilizados na combinação, o módulo também limita o número de combinações retornadas, também utilizando o parâmetro $max\_tems$. Logo, apesar de haver $2^4 = 16$ combinações de templates, o módulo retornaria apenas as $max\_tems = 2$ melhores rankeadas.

O rankeamento de templates é feito por um modelo N-grama, que atribui um escore à sequência formada pelo texto do template, substituídas as lacunas pelos $sujeito$ e $objeto$ das respectivas triplas. Observe que as lacunas não são substituídas por expressões de referência para os $sujeito$ e $objeto$, mas por eles próprios, que são identificadores de informações, como *"Barack_Obama"*. Já para rankear uma combinação de templates, seu escore é calculado como o produto dos escores de seus templates. Esse processo está esquematizado na Figura 6.

Figura 6 – Funcionamento do módulo Seleção de Templates




Fonte: Elaborado pelo autor

**Algoritmo 1:** Abstração de triplas

**Entrada:** *triplas*

**Saída:** *triplas_abstraidas*

```
1  mapa_entidade_em_id := {};
2  triplas_abstraidas := [];
3  id_disponivel := 0;
4  para tripla ∈ triplas faça
5      se tripla.sujeito ∉ mapa_entidade_em_id então
6          slot := "slot_"+ id_disponivel
7          map_entidade_em_id[tripla.sujeito] := slot
8          id_disponivel := id_disponivel + 1
9      fim
10     se tripla.objeto ∉ mapa_entidade_em_id então
11         slot := "slot_"+ id_disponivel
12         map_entidade_em_id[tripla.objeto] := slot
13         id_disponivel := id_disponivel + 1
14     fim
15     sujeito_abstraido := mapa_entidade_em_id[tripla.sujeito]
16     objeto_abstraido := mapa_entidade_em_id[tripla.objeto]
17     tripla_abstraida := Tripla(sujeito_abstraido, tripla.predicado, objeto_abstraido)
18     triplas_abstraidas.add(tripla_abstraida)
19     fim
20     retorne triplas_abstraidas
```

Para definir que templates serão utilizados, todos os templates adequados para cada

lista de triplas da entrada são rankeados, sendo retornados apenas os *max_tems* melhores, onde *max_tems* é um parâmetro do módulo. O primeiro passo, portanto, consiste em identificar na base de dados de templates, quais deles são adequados para a lista de triplas de entrada. Esse processo consiste em transformar os *sujeito* e *objeto* das triplas da lista de triplas em variáveis, de forma que o resultado será igual à estrutura de um template, caso esse template seja adequado para essa lista de triplas. Esse procedimento é o mesmo utilizado na extração dos templates e de suas estruturas. Denomina-se "lista de triplas abstraídas" a lista de triplas resultante desse processo, cujos *sujeito* e *objeto* foram substituídos por identificadores de lacunas. O Algoritmo 1 apresenta como esse procedimento é realizado. Ele recebe como entrada a lista de triplas *triplas* e faz uso de um mapa de *sujeito*/*objeto* em identificadores de lacunas, representado por *mapa_entidade_em_id*. Para cada tripla em *triplas* (linha 4) primeiro são verificados se seu *sujeito* já está mapeado em um identificador de lacuna (linhas 5 a 9) - em caso negativo, uma nova entrada é adicionada ao mapa mapeando o *sujeito* no próximo identificador disponível; o mesmo é feito com o *objeto* da tripla (linhas 10 a 14). Feito isso, uma nova tripla é então gerada, representada por *tripla_abstraida* (linha 17), tendo como *sujeito* o identificador de lacuna associado ao *sujeito* da tripla da iteração (linha 15), como *predicado* o mesmo *predicado* da tripla da iteração, e como *objeto* o identificador de lacuna associado ao *objeto* da tripla da iteração (linha 16). A *tripla_abstraida* é então adicionada a uma lista (linha 18), representada por *triplas_abstraidas*, que é retornada ao fim (linha 20).

Feito isso, e dado que a base de dados de templates permite recuperar templates a partir de suas estruturas, para que os templates para uma dada lista de triplas sejam recuperados basta que o Algoritmo 1 seja aplicado e então seja verificado se a lista de triplas abstraída resultante está na base de dados, retornando os templates associados, em caso positivo. Em caso negativo, e tendo a lista de triplas tamanho 1, o módulo utilizará um template *fallback* com texto *"**slot_1** predicado_processado **slot_2**."*, onde *predicado_processado* é o predicado da tripla da lista, aplicado um processamento que transforma o símbolo *"_"* em espaço, e separa termos que estão em *CamelCase*. Por exemplo, para a tripla $\langle Elliot\_See, almaMater, University\_of\_Texas\_at\_Austin \rangle$, o texto a ser gerado pelo template *fallback* será *"Elliot See alma mater University of Texas at Austin"*.

Para exemplificar o funcionamento deste módulo será utilizada a segunda forma de particionamento[3] retornada pelo módulo anterior e representada pela sequência

[3] Não foi utilizada a primeira forma de particionamento retornada pelo módulo anterior pois esta, representada pela sequência [*almaMater, _ , deathPlace, _ , leaderName*], é bastante simples, colocando todas as triplas em

[*almaMater, _ , deathPlace,* | *, leaderName*], contendo duas partes, sendo a primeira parte:

$[\langle Elliot\_See,\ almaMater,\ University\_of\_Texas\_at\_Austin\rangle,$
$\langle Elliot\_See,\ deathPlace,\ St.\_Louis\rangle]$

Enquanto a segunda parte é:

$[\langle St.\_Louis,\ leaderName,\ Francis\_G.\_Slay\rangle]$

Para a primeira parte são encontrados os templates com os seguintes textos:

1. *"**slot_1** graduated from **slot_2** , and died in **slot_3** ."*
2. *"**slot_1** graduated **slot_2** and has died in **slot_3** ."*
3. *"**slot_1** attended **slot_2** and died in **slot_3** ."*

Eles são então preenchidos utilizando os valores de *sujeito* e *objeto* das respectivas triplas, gerando assim as seguintes respectivas sequências, que serão rankeadas pelo modelo N-grama:

1. *"**Elliot_See** graduated from **University_of_Texas_at_Austin** , and died in **St._Louis** ."*
2. *"**Elliot_See** graduated **University_of_Texas_at_Austin** and has died in **St._Louis** ."*
3. *"**Elliot_See** attended **University_of_Texas_at_Austin** and died in **St._Louis** ."*

Considerando $max\_tems = 2$, apenas os 2 melhores templates serão utilizados. Para o exemplo acima, o segundo e o terceiro templates serão utilizados. Estes serão os templates que serão utilizados para verbalizar a primeira parte do particionamento das triplas. Para a segunda parte, que consiste apenas de uma tripla com o predicado *leaderName*, os seguintes 2 templates são os melhores rankeados e serão utilizados:

- *"The name of **slot_1** leader is **slot_2** ."*;
- *"The leader of **slot_1** was **slot_2** ."*.

Finalmente, as 2 melhores combinações possíveis entre os 2 templates selecionados para a primeira parte e os 2 templates selecionados para a segunda parte são:

- *"**slot_1** attended **slot_2** and died in **slot_3** ."* e *"The name of **slot_1** leader is **slot_2** ."*;
- *"**slot_1** attended **slot_2** and died in **slot_3** ."* e *"The leader of **slot_1** was **slot_2** ."*.

Diferente dos módulos anteriores, não há garantias de que esse módulo retornará alguma saída, pois ele depende da disponibilidade de templates na base de dados de templates. Caso para alguma lista de triplas não seja possível definir templates adequados para verbalizá-la,

uma mesma sentença.

seja com templates da base de dados de templates ou utilizando o template *fallback*, o processo de geração de texto, para a $LTriplas$ de entrada, é interrompido. Dependendo da configuração de parâmetros, outras $LTriplas$ serão exploradas. No caso extremo de não haver templates para nenhuma forma de ordenar e particionar as triplas, a solução irá retornar um texto gerado a partir da aplicação do template *fallback* a todas as triplas.

### *4.4.4 Módulo Geração de Expressões de Referência*

O módulo Geração de Expressões de Referência recebe como entrada $LTriplas$, uma lista de listas de triplas, e $LTemplates$, uma lista de templates apropriados para as listas de triplas em $LTriplas$, e tem como objetivo retornar uma expressão de referência[4] para preencher cada lacuna de cada template, de acordo com o respectivo $sujeito$ ou $objeto$ das triplas em $LTriplas$. Seja, por exemplo, um template com 4 *slots*, e supondo que para cada lacuna existam 3 referências que podem ser utilizadas para preenchê-lo. Haverá então um total de $3^4 = 81$ combinações de referências para preencher o template. Para controlar o total de combinações a serem retornadas, o módulo usa o parâmetro $max\_refs$, que limita o máximo de referências a serem consideradas por lacuna. No exemplo, com $max\_refs = 2$, haveria no máximo 2 referências por lacuna, totalizando $2^4 = 16$ combinações de referências.

Além de limitar o número de referências utilizadas nas combinações, o módulo também limita o número de combinações retornadas, também utilizando o parâmetro $max\_refs$. Logo, apesar de haver $2^4 = 16$ combinações de referências, o módulo retornará apenas as $max\_refs = 2$ melhores combinações.

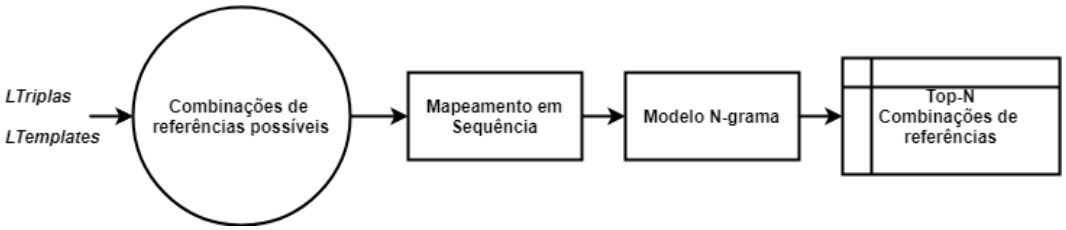
Para definir quais referências serão retornadas, todas as referências adequadas para cada lacuna são rankeadas, sendo retornadas apenas as $max\_refs$ melhores, onde $max\_refs$ é um parâmetro do módulo. Para recuperar as referências para um dado identificador de informação, o módulo pesquisa na base de dados de referências, que associa um identificador de informação a um conjunto de referências que podem ser utilizadas para referenciar a respectiva informação. Além disso, a base de referências é composta por dois subconjuntos, um contendo referências que podem ser utilizadas para introduzir uma informação no texto, e o outro contendo referências que podem ser utilizadas para referenciar uma informação que já foi previamente introduzida no texto. Enquanto no primeiro subconjunto normalmente encontra-se o nome de entidades, no segundo subconjunto há pronomes pessoais ou descrições. Caso seja a primeira vez em que a

[4] Para simplificar o texto, “referência" será utilizado como sinônimo de “expressão de referência".

informação será referenciada no texto, o módulo considerará o primeiro conjunto, considerando o segundo subconjunto, caso contrário. Ao conjunto de referências a serem rankeadas também é adicionado o identificador de informação, processado. Esse processamento consiste de substituir símbolos *"_"* por espaço, ou separar palavras em *CamelCase*.

O rankeamento de referências é feito por um modelo N-grama, que atribui um escore à sequência formada pelo texto do template, com uma das lacunas preenchidos com uma referência, enquanto as demais lacunas são mantidas não preenchidas. Diferente do módulo anterior, as lacunas não são substituídas por identificadores de informações, mas sim por referências. Já para rankear uma combinação de referências, seu escore é calculado como o produto dos escores de cada referência. Esse processo está esquematizado na Figura 7.

Figura 7 – Funcionamento do módulo Seleção de Expressões de Referência




Fonte: Elaborado pelo autor

Considerando a melhor combinação de *templates* retornada pelo módulo anterior, *"**slot_1** attended **slot_2** and died in **slot_3** ."* e *"The name of **slot_1** leader is **slot_2** ."*, o módulo Geração de Expressões de Referência irá selecionar expressões de referência para cada uma das lacunas. Para o primeiro template, há o seguinte mapeamento entre as lacunas e os *sujeito* e *objeto* das triplas: **slot_1** = *Elliot_See*, **slot_2** = *University_of_Texas_at_Austin*, **slot_3** = *St._Louis*. Para selecionar referências para o **slot_1**, a base de referências é consultada e retorna, para *Elliot_See* as seguintes referências:

- *"elliot see"*;
- *"test pilot elliot see"*;
- *"eliot see"*;
- *"elliot see's"*.

Serão geradas então as seguintes sequências, a serem rankeadas pelo modelo N-grama (considerar o espaço como separador entre os elementos):

- *"**elliot_see** attended **slot_1** and died in **slot_2** ."*;
- *"**test_pilot_elliot_see** attended **slot_1** and died in **slot_2** ."*;

- *"**eliot_see** attended **slot_1** and died in **slot_2** ."*;
- *"**elliot_see_'_s** attended **slot_1** and died in **slot_2** ."*.

Considerando $max\_refs = 2$, são selecionadas as duas referências melhores rankeadas, que são *"elliot see"* e *"elliot see's"*. O mesmo procedimento é feito para as demais lacunas dos dois templates, ao fim do que haverá até 2 referências por lacuna. Da mesma forma que com o módulo Seleção de Templates, no lugar de retornar todas as combinações de referências, são retornadas apenas as $max\_refs$ com melhor escore. Para o exemplo, os textos gerados com as 2 melhores combinações de referências são:

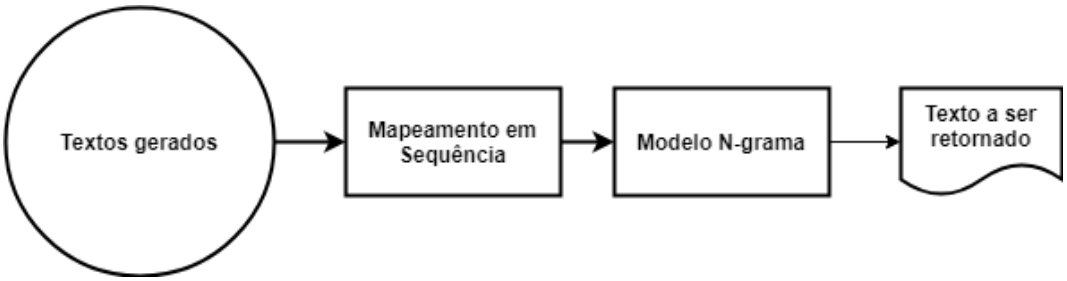
- *"elliot see attended university of texas at austin and died in st . louis . The name of st louis ' leader is francis g . slay ."*;
- *"elliot see attended university of texas at austin and died in st louis . The name of st louis ' leader is francis g . slay ."*.

A única diferença entre os dois textos gerados está em um símbolo de ponto na referência a *"St._Louis"*, referenciado como *"st . louis"* e como *"st louis"*.

### *4.4.5 Módulo Seleção de Texto*

O módulo Seleção de Texto recebe como entrada um conjunto de textos, gerados a partir das saídas dos módulos anteriores, e tem como objetivo selecionar qual deles será retornado pelo *pipeline*. Para tanto ele faz uso de um modelo N-grama treinado sobre sequências de palavras, como um tradicional modelo de linguagem baseado em N-grama. A cada texto é atribuído um escore e o melhor é então retornado. Esse processo está esquematizado na Figura 8.

Figura 8 – Funcionamento do módulo Seleção de Texto




Fonte: Elaborado pelo autor

Para o exemplo, este módulo irá rankear todos os 39 textos que foram gerados, sendo os 3 melhores textos os seguintes:

1. *“elliot see died in st louis , whose mayor is francis g slay , and graduated from university of texas at austin ."*;
2. *“elliot see died in st louis , whose mayor is francis g slay , and graduated from the university of texas , austin ."*;
3. *“elliot see died in st . louis with francis g . slay . elliot see attended the university of texas in austin ."*.

# 5 EXPERIMENTOS

A capacidade da solução proposta de gerar textos de qualidade foi avaliada utilizando uma base de dados publicada em uma *shared task* de GLN. Os textos gerados com ela foram comparados, com o uso de métodos de avaliação automáticos, com os textos gerados pelas soluções que competiram na *shared task*, bem como com soluções propostas em trabalhos posteriores. Além da avaliação automática, também foi realizada uma análise manual dos textos a fim de identificar possíveis problemas e pontos de melhoria.

Este Capítulo apresenta a experimentação realizada para avaliar a solução proposta da seguinte forma: a Seção 5.1 descreve como os experimentos foram realizados; a Seção 5.2 apresenta informações sobre a base de dados utilizada nos experimentos, enquanto a Seção 5.3 detalha como os modelos N-grama foram treinados; finalmente, a Seção 5.4 apresenta os resultados da avaliação automática e a Seção 5.5 apresenta os resultados da análise manual dos textos.

## 5.1 Metodologia

Para avaliar a solução, foi utilizada uma base de dados que contém conjuntos de triplas e, associados a eles, conjuntos de textos de referência, particionada pelos autores da base de dados em três subconjuntos, *treinamento*, *desenvolvimento* e *teste*. Foram executadas as seguintes etapas:

1. Foi executado um *Grid Search* treinando os modelos com o subconjunto *treinamento* e avaliando com o subconjunto *desenvolvimento*, com o objetivo de determinar as ordens dos modelos N-grama. Foram utilizados valores de 3 a 5 para cada modelo N-grama, bem como foi fixado em 3 a quantidade máxima de saídas de cada módulo. Esse valor foi escolhido por ao mesmo tempo permitir que os módulos retornem várias saídas e não exigir um custo alto de tempo de treinamento. Dessa forma foram exploradas $3^5 = 243$ combinações;
2. Após a execução do *Grid-Search*, foi selecionada a combinação de ordens dos modelos N-grama que resultou no melhor escore BLEU, e com esses parâmetros foram então treinados novos modelos, agora utilizando os subconjuntos *treinamento* e *desenvolvimento*, e avaliando com o subconjunto *teste*. Dessa vez a quantidade máxima de saídas de cada módulo foi variada, para que a relação entre o tempo gasto na geração dos textos e a

qualidade dos textos retornados fosse avaliada, bem como para avaliar a capacidade da solução proposta de gerar textos de qualidade;

3. Os modelos treinados na etapa anterior foram então avaliados com os mesmos métodos automáticos e pré-processamento utilizados na *shared task* que originou a base de dados utilizada nos experimentos. Os métodos de avaliação utilizados foram BLEU, METEOR e TER, e o pré-processamento consistiu em alterar todas as letras para caixa baixa, *tokenizar* os termos, utilizando uma expressão regular, e substituir caracteres não ASCII por caracteres ASCII similares. São apresentados os resultados tanto para diversas variações da solução proposta, quanto para os competidores da *shared task* e para outras soluções publicadas posteriormente.
4. Além da avaliação com métodos automáticos, uma amostra aleatória de textos gerados pela solução foi manualmente analisada pelo autor deste trabalho, buscando identificar problemas de qualidade dos textos, referentes a aspectos como gramaticalidade, adequação aos dados de entrada e fluência.

## 5.2 Dados utilizados nos experimentos

A base de dados utilizada nos experimentos foi disponibilizada como parte da *shared task* WebNLG de 2017 (COLIN *et al.*, 2016)[1]. Ela contém 9674 entradas na forma $\langle categoria,\ tripleset,\ referências \rangle$, onde *categoria* é uma de 15 categorias, como *"Astronaut"* e *"University"*; $tripleset$ é um conjunto de triplas $\langle sujeito,\ predicado,\ objeto \rangle$, contendo de 1 a 7 triplas, em que *sujeito*, *predicado* e *objeto* são *strings* e representam que a informação identificadas por *sujeito* está relacionada com a informação identificada por *objeto* através da relação identificada por *predicado*; e *referências* é um conjunto de textos em inglês verbalizando as informações representadas pelas triplas em $tripleset$, contendo de 1 a 8 textos.

Foi utilizado o mesmo particionamento da base de dados utilizado na *shared task*, em que há o subconjunto *treinamento*, com 6940 entradas (72% dos dados), o subconjunto *desenvolvimento*, com 872 entradas (9% dos dados), e o subconjunto *teste*, com 1862 entradas (19% dos dados). O subconjunto *teste* pode ser dividido em mais dois subconjuntos: *teste-visto*, com 971 entradas, cujas categorias também estão presentes nos subconjuntos *treinamento* e *desenvolvimento*, e *teste-não-visto*, com 891 entradas cujas categorias não estão presentes

[1] Existe mais de uma versão da base de dados, que diferem em relação a correções que foram feitas com o tempo. Nós utilizamos a versão *release_v1*, a mesma utilizada na *shared task* e nos trabalhos relacionados, e que pode ser acessada em https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/release_v1. Acesso em: 20 jun. 2020

nos subconjuntos *treinamento* e *desenvolvimento*. A intenção dos organizadores dessa base de dados ao adicionar as entradas do subconjunto *teste-não-visto* foi avaliar a capacidade dos competidores de gerar textos de domínios diferentes dos presentes nos dados utilizados para treinar seus modelos. Este trabalho, e outros como (MORYOSSEF; GOLDBERG; DAGAN, 2019), consideram como objeto de estudo apenas a geração de textos de domínios para os quais seus modelos foram treinados, portanto reportando apenas os resultados da avaliação para o subconjunto *teste-visto*.

Abaixo há um exemplo de entrada contida nessa base de dados:

- **Categoria**: *"Airport"*
- **Tripleset**:

  $\{\langle Aarhus\_Airport,\ location,\ Tirstrup\rangle,$
  $\langle Tirstrup,\ country,\ Denmark\rangle,$
  $\langle Denmark,\ capital,\ Copenhagen\rangle,$
  $\langle Tirstrup,\ isPartOf,\ Central\_Denmark\_Region\rangle\}$

- **Referências**:
  - *"Aarhus airport is located in Tirstrup, part of the Central Region of Denmark which has the capital city of Copenhagen."*
  - *"Copenhagen is the capital of Denmark where Aarhus airport is located in Tirstrup which is part of the Central Denmark region."*
  - *"Aarhus Airport is located in Tirstrup, part of the Central Denmark region. The capital of the country is Copenhagen."*

Desse exemplo pode-se extrair algumas características dessa base de dados:

- A ordem com que as triplas estão verbalizadas nos textos de referência não necessariamente coincide com a ordem com a qual estão serializadas na base de dados - todos os três textos de referência seguem ordens diferentes;
- As triplas podem estar verbalizadas em mais de uma sentença: os dois primeiros textos de referência contêm apenas uma sentença, enquanto o terceiro texto contém duas sentenças;
- Os relacionamentos podem estar verbalizados de mais de uma forma, como é o caso do predicado *capital* da tripla $\langle Denmark,\ capital,\ Copenhagen\rangle$, verbalizado no primeiro

texto como *"which has the capital city of"*, e no segundo texto como *"is the capital of"*;

- Os valores de $sujeito$, $predicado$ e $objeto$ das triplas são similares às *strings* utilizadas para verbalizar as informações que eles representam - é o caso de *"Denmark"* que pode ser utilizada para se referir ao $sujeito$ da tripla $\langle Denmark,\ capital,\ Copenhagen\rangle$; ou *"Central Region of Denmark"*, que é uma *string* similar ao $objeto$ da tripla $\langle Tirstrup,\ isPartOf,\ Central\_Denmark\_Region\rangle$, diferindo apenas no uso do caractere *"_"* no lugar de espaço.

A geração dessa base de dados foi realizada de forma semi-automática, consistindo de uma etapa de seleção automática de conjuntos de triplas RDF da DBpedia e substituição das URI das triplas por *strings* que fazem referência às respectivas entidades ou literais. Seguiu-se então com etapas manuais, via *crowdsourcing*: primeiro solicitando aos colaboradores que verbalizassem triplas individuais; depois, para a verbalização de conjuntos de triplas de tamanho maior que um, foi solicitado aos colaboradores que fizessem a fusão das sentenças associadas às triplas individuais, que foram geradas na etapa anterior; finalmente, os textos gerados foram filtrados a partir de uma avaliação de novos colaboradores, relacionada a fluência, adequação e corretude gramatical. Antes da disponibilização da base de dados, alguns ajustes foram feitos para tornar as triplas mais claras: por exemplo, o predicado de nome *west* foi substituído por *has to its west*; alguns predicados foram fundidos em um só, por possuírem semântica similar, como *club* e *team*, sendo os dois mapeados em *club*; além de correções sobre valores erroneamente extraídos da DBpedia. A Tabela 2 apresenta algumas informações estatísticas sobre a base de dados WebNLG.

Tabela 2 – Informações estatísticas sobre os subconjuntos da base de dados WebNLG

| | *treinamento* | *desenvolvimento* | *teste* |
|---|---|---|---|
| Triplas distintas | 2090 | 1102 | 2331 |
| Textos de referência distintos | 18051 | 2261 | 4922 |
| Sujeitos distintos | 430 | 304 | 575 |
| Predicados distintos | 246 | 186 | 300 |
| Objetos distintos | 1619 | 918 | 1888 |
| Mediana da quantidade de textos por sujeitos | 51 | 12 | 14 |
| Mediana da quantidade de textos por predicados | 59 | 13 | 15 |
| Mediana da quantidade de textos por objetos | 18 | 5 | 5 |

Uma diferença das triplas da base de dados WebNLG para triplas RDF consiste em que, no lugar de literais ou URI identificando recursos, as triplas são compostas por *strings* que referenciam esses recursos ou literais, mas não o identificam unicamente. Por exemplo, no

lugar de uma tripla se referir ao recurso da DBpedia que representa a entidade Barack Obama, o ex-presidente dos Estados Unidos da América, utilizando a sua URI http://dbpedia.org/resource/Barack_Obama, ela se referirá a ele utilizando a *string "Barack_Obama"*. Uma implicação disso, considerando o exemplo, é que não há um identificador que permita recuperar facilmente da DBpedia mais informações sobre o Barack Obama, como as formas com as quais ele é chamado, ou seu gênero, que poderiam ser utilizadas para melhor decidir que pronome utilizar em um texto para referenciá-lo.

## 5.3 Implementação dos módulos

Para treinar os modelos N-grama de cada módulo, foi utilizada uma base de dados derivada da base de dados WebNLG, publicada por (FERREIRA *et al.*, 2018) e utilizada no trabalho (FERREIRA *et al.*, 2019). Essa base de dados foi desenvolvida através da identificação manual, para cada texto de referência, da ordem e do particionamento com que as triplas foram verbalizadas em sentenças, bem como que *substrings* do texto representam expressões de referências para os $sujeito$ ou $objeto$ das triplas - essas *substrings* foram então substituídas por *strings* nos formatos *AGENT-i*, *BRIDGE-i* ou *PATIENT-i*, com $i > 0$, representando lacunas. A *string AGENT-i* foi utilizada para identificar uma referência a uma informação que aparece apenas como $sujeito$ nas triplas em *tripleset*; a *string BRIDGE-i* foi utilizada para identificar uma referência a uma informação que aparece tanto como $sujeito$ como $objeto$ nas triplas em *tripleset*; e *PATIENT-i* foi utilizada para identificar referências a informações que aparecem apenas como $objeto$ nas triplas em *tripleset*.

A seguir é apresentado como os dados utilizados para treinar os modelos N-grama de cada módulo foram extraídos, utilizando como exemplo um dos textos de referência do exemplo de entrada apresentado na Seção 5.2, bem como o respectivo texto com a lacunas e o particionamento das triplas em sentenças.

- **Texto de referência**: *"Aarhus Airport is located in Tirstrup, part of the Central Denmark region. The capital of the country is Copenhagen."*
- **Texto com lacunas**: *"**AGENT-1** is located in **BRIDGE-1**, part of **PATIENT-1**. The capital of **BRIDGE-2** is **PATIENT-2**."*
- **Particionamento das triplas**:

```
Parte 1: [<Aarhus_Airport, location, Tirstrup>,
          <Tirstrup, isPartOf, Central_Denmark_Region>]
Parte 2: [<Tirstrup, country, Denmark>,
          <Denmark, capital, Copenhagen>]
```

Abaixo há a descrição de como são extraídas sequências, a partir do exemplo acima, para o treinamento dos modelos N-grama:

- **Módulo Planejamento de Discurso**: a seguinte sequência será gerada, contendo os *predicado* das triplas ordenados de acordo com a ordem em que foram verbalizadas:

  [*location, isPartOf, country, capital*]

- **Módulo Agregação em Sentenças**: a seguinte sequência será gerada, similar à do módulo anterior, mas com símbolos indicando se os predicados aparecem ou não na mesma sentença:

  [*location, _, isPartOf,* |*, country, _, capital*]

- **Módulo Seleção de Templates**: a seguinte sequência será gerada, preenchendo as lacunas do texto com os respectivos valores de $sujeito$ e $objeto$ das triplas. Por exemplo, *AGENT-1* é substituído por *"Aarhus_Airport"*, *BRIDGE-1* por *"Tirstrup"*, *PATIENT-1* por *"Central_Denmark_Region"*, *BRIDGE-2* por *"Denmark"*, e finalmente *PATIENT-2* por *"Copenhagen"*:
  ["**Aarhus_Airport**", "is", "located", "in", "**Tirstrup**", ",", "part", "of", "**Central_Denmark_Region**", ".", "The", "capital", "of", "**Denmark**", "is", "**Copenhagen**", "."]

  Além de serem extraídas sequências para treinar o modelo N-grama, também são extraídos templates, que são adicionados à base de dados de templates. O processo de extração consiste basicamente em mapear os rótulos *AGENT-i*, *BRIDGE-i* e *PATIENT-i* nos rótulos utilizado nos templates propostos, no formato *slot_i*.

  Como a solução faz uso de templates de sentenças e há duas sentenças, dois templates ponderiam ser extraídos. Ocorre que a segunda sentença falha em verbalizar todas as duas triplas correspondentes: a tripla $\langle Tirstrup, country, Denmark \rangle$ não está verbalizada

na sentença *"The capital of the country is Copenhagen"*, e seu *sujeito Tirstrup* não é referenciado. As sentenças que não fazem referência a todos os *sujeito* e *objeto* das respectivas triplas, como é o caso quando ocorre elipse do sujeito, são descartadas. Para a primeira sentença, o template extraído é:

$\langle s = [\langle$**slot_1**, *location*, **slot_2**$\rangle$,

$\langle$**slot_2**, *isPartOf*, **slot_3**$\rangle]$,

$Z =$ "**slot_1** *is located in* **slot_2**, *part of* **slot_3**."$\rangle$

- **Módulo Geração de Expressões de Referência**: a seguinte sequência será gerada, considerando cada expressão de referência como um item da sequência. Por exemplo, a expressão *"the Central Denmark region"*, no lugar de ser tratada como quatro elementos separados, ["the", "Central", "Denmark", "region"], é tratada como um elemento só ["the Central Denmark region"]:
  ["**Aarhus Airport**", "is", "located", "in", "**Tirstrup**", ",", "part", "of",
  "**Central Denmark region**", ".", "The", "capital", "of", "**the country**", "is", "**Copenhagen**", "."]
  Além de serem extraídas sequências para treinar o modelo N-grama, também são extraídas expressões de referência. A base de dados de referências é composta por dois subconjuntos: referências introdutórias, contendo referências que introduzem pela primeira vez uma informação num texto; referências não introdutórias, contendo referências a informações que já foram introduzidas no texto. Do exemplo, as seguintes expressões de referência serão extraídas e adicionadas à base de dados de referências introdutórias:
  - *"Aarhus Airport"* para *Aarhus_Airport*
  - *"Tirstrup"* para *Tirstrup*
  - *"the Central Denmark region"* para *Central_Denmark_Region*
  - *"the country"* para *Denmark*
  - *"Copenhagen"* para *Copenhagen*

  Desse exemplo, podem-se fazer duas observações: há referência que podem ser geradas a partir de um pré processamento simples sobre o *sujeito* ou *objeto* que referenciam, que é o caso de *"Aarhus Airport"* que é basicamente *Aarhus_Airport*, substituindo o *underline* por um espaço; isso não é necessariamente o caso, mas é comum, dado que os textos foram gerados por pessoas cujo objetivo era verbalizar as triplas, e, supõe-se, que muitas derivaram expressões de referência a partir dos *sujeito* e *objetos* referenciados.

Outra observação está relacionada à estratégia de dividir a base de referências em dois subconjuntos, um deles contendo expressões de referência que podem ser utilizadas para introduzir uma informação no texto. Ocorre que a primeira referência a *Denmark* no texto é *"the country"*, que não é adequada para ser utilizada como primeira referência, dado que não identifica a informação a que se refere. Isso decorre de um problema do próprio texto de referência, evidenciado no item anterior, e pode trazer prejuízo para a solução, dado que esses casos não são filtrados.

- **Módulo Seleção de Texto**: o próprio texto será utilizado como uma sequência: [“Aarhus", “Airport", “is", “located", “in", “Tirstrup", “,", “part", “of", “Central", “Denmark", “region", “.", “The", “capital", “of", “the", “country", “is", “Copenhagen", “."]

O código da solução foi escrito com a linguagem Python 3.6 e os modelos N-grama foram treinados com a ferramenta *KenLM* (HEAFIELD *et al.*, 2013), que faz uso da técnica de suavização de *Kneser-Ney*.

## 5.4 Resultados da avaliação automática

Após a execução do *Grid-Search*, treinando com o subconjunto *treinamento* e avaliando com o subconjunto *desenvolvimento*, a melhor combinação dos valores de ordens para os modelos N-grama de cada módulo foi selecionada e então novos modelos foram treinados, agora com os subconjuntos *treinamento* e *desenvolvimento*. A solução então foi executada com esses novos modelos N-grama, variando a quantidade de saídas máximas de cada módulo, a fim de avaliar a relação entre a quantidade de textos gerados ao fim do processo e a qualidade deles.

Tabela 3 – Os 3 melhores e os 3 piores modelos treinados no *Grid-Search*, de acordo com o escore BLEU

| DP | SA | TEMS | REG | TXS | BLEU | METEOR | TER |
|---|---|---|---|---|---|---|---|
| 4 | 5 | 3 | 3 | 5 | **61,70** | 0,444180 | 0,358201 |
| 4 | 4 | 3 | 3 | 5 | 61,65 | 0,444372 | 0,359761 |
| 4 | 5 | 5 | 3 | 5 | 61,65 | **0,446424** | **0,357811** |
| 5 | 5 | 3 | 5 | 3 | 59,45 | 0,434315 | 0,364393 |
| 3 | 3 | 3 | 5 | 3 | 59,42 | 0,433559 | 0,368391 |
| 5 | 4 | 3 | 5 | 3 | 59,40 | 0,433147 | 0,364929 |

A Tabela 3 apresenta as 3 melhores e as 3 piores combinações de ordens dos modelos N-grama do *Grid-Search*, de acordo com o escore *BLEU*. A variação entre o valor mínimo

e máximo de *BLEU* foi de $61,70 - 59,40 = 2,30$ pontos. Para entender melhor o que essa variação representa, foram verificados os valores de *BLEU*, *METEOR* e *TER* quando os textos avaliados são os textos de referência, realizadas algumas alterações. No caso, foram selecionados aleatoriamente uma quantidade de textos e removidas aleatoriamente deles 4 palavras, utilizando como base de dados o subconjunto *teste*. Por exemplo, considerando o texto *"John **Madin** is an architect **who** was born **in** Birmingham, England and **designed** 103 Colmore Row."*, após removermos 4 palavras aleatoriamente o resultado seria *"John is an architect was born Birmingham, England and 103 Colmore Row."*.

Tabela 4 – Escores para o subconjunto *teste* de uma quantidade variável de textos alterados

| **Quantidade de textos alterados** | **BLEU** | **METEOR** | **TER** |
|---|---|---|---|
| 100 | 97,24 | 0,718396 | 0,017056 |
| 200 | 94,64 | 0,668297 | 0,033941 |
| 400 | 89,26 | 0,600745 | 0,067881 |
| 600 | 83,30 | 0,550446 | 0,101907 |

A Tabela 4 apresenta os resultados dos métodos de avaliação quando aplicada essa alteração a 100, 200, 400 e 600 textos aleatoriamente selecionados, de um total de 1862 textos. Os autores deste trabalho assumem que a diferença entre a melhor e a pior combinação no *Grid-Search* é equivalente a alterar aleatoriamente 100 textos, removendo 4 palavras, dado que a diferença de escores BLEU entre essas combinações, de $2,30$ pontos, é próxima da diferença de escores BLEU resultante de alterar 100 textos e alterar 200 textos, de $97,24 - 94,64 = 2,60$ pontos.

Tabela 5 – Valores de métodos de avaliação automáticas para soluções testadas com a base WebNLG, subconjunto *teste-visto* (ordenado por BLEU do melhor para o pior)

| | **BLEU** | **METEOR** | **TER** |
|---|---|---|---|
| adaptcentre | **60,59** | **0,445389** | 0,378763 |
| **TBMS_9** | 59,64 | 0,438606 | **0,369851** |
| deepnlg-e2ernn | 58,36 | 0,415778 | 0,397737 |
| seq2seq_wc_word | 55,82 | 0,405280 | 0,399997 |
| gcn | 55,35 | 0,390372 | 0,397695 |

Para comparar a solução proposta neste trabalho com outras já propostas, foram treinados modelos utilizando a melhor combinação de ordens dos modelos N-grama, de acordo com o *Grid-Search*, e então a quantidade máxima de saídas de cada módulo foi variada de 1 a 10.

A Tabela 5 apresenta os valores dos escores dos métodos de avaliação automática referentes ao subconjunto *teste-visto* para os 5 melhores resultados dentre a solução proposta e as soluções dos competidores da *shared task* e dos trabalhos publicados. A solução proposta está identificada com o nome TBMS_*n*, onde *n* é número máximo de saídas de cada módulo. TBMS_*3*, por exemplo, significa que cada módulo retornou no máximo as 3 saídas melhores rankeadas.

### *5.4.1 Competidores*

Já as soluções dos competidores e dos trabalhos publicados estão listados abaixo, com o identificador utilizado nas tabelas e uma breve descrição:

- **adaptcentre**[2]: arquitetura integrado com modelo de Aprendizagem Profunda, utilizando GRU e *subword encoding*;
- **deepnlg-e2ernn**: arquitetura integrada com modelo de Aprendizagem Profunda, utilizando GRU (FERREIRA *et al.*, 2019);
- **seq2seq-wc-word**: arquitetura integrada com modelo de Aprendizagem Profunda, utilizando LSTM (JAGFELD; JENNE; VU, 2018);
- **gcn**: arquitetura integrada com modelo de Aprendizagem Profunda, utilizando GCN (MARCHEGGIANI; PEREZ-BELTRACHINI, 2018);
- **biu-nmt**: arquitetura integrada com modelo de Aprendizagem Profunda, utilizando LSTM (MORYOSSEF; GOLDBERG; DAGAN, 2019);
- **upf-forge**: arquitetura modular e baseado em regras, foi o melhor colocado na avaliação humana da *shared task* (MILLE; DASIOPOULOU, 2017);
- **melbourne**[3]: arquitetura integrada com modelo de Aprendizagem Profunda, utilizando LSTM (HOCHREITER; SCHMIDHUBER, 1997) bidirecional e *Glove* (PENNINGTON; SOCHER; MANNING, 2014);
- **tilburg-smt**[4]: utilizou técnicas de tradução automática estatística baseada em frases, com o *Moses toolkit* (KOEHN *et al.*, 2007);

### *5.4.2 Discussão*

Para o subconjunto *teste-visto*, contendo apenas categorias presentes nos subconjuntos utilizados no treinamento dos modelos, o melhor resultado obtido ocorreu quando cada

[2] https://webnlg-challenge.loria.fr/files/ADAPTcentre_report.pdf. Acesso em: 20 jun. 2020
[3] https://webnlg-challenge.loria.fr/files/melbourne_report.pdf. Acesso em: 20 jun. 2020
[4] https://webnlg-challenge.loria.fr/files/tilburg_report.pdf. Acesso em: 20 jun. 2020

Tabela 6 – Escores para a solução variando a quantidade de saídas máximas por módulo.

| | BLEU | METEOR | TER | Tempo de geração dos textos (s) |
|---|---|---|---|---|
| **TBMS_9** | 59,64 | 0,438606 | 0,369851 | 190 |
| **TBMS_8** | 59,41 | 0,438666 | 0,370576 | 143 |
| **TBMS_10** | 59,33 | 0,437968 | 0,371258 | 230 |
| **TBMS_6** | 59,19 | 0,438503 | 0,371386 | 72 |
| **TBMS_7** | 58,96 | 0,437352 | 0,374328 | 97 |
| **TBMS_5** | 58,78 | 0,438228 | 0,374968 | 42 |
| **TBMS_4** | 58,45 | 0,436717 | 0,374115 | 26 |
| **TBMS_3** | 57,33 | 0,431618 | 0,385884 | 13 |
| **TBMS_2** | 55,57 | 0,424378 | 0,403664 | 7 |
| **TBMS_1** | 50,28 | 0,407020 | 0,455002 | 3 |

módulo retornou no máximo as 9 saídas melhores rankeadas, obtendo um escore *BLEU* de $59,64$, apenas $0,95$ pontos menor que o melhor escore obtido até o momento pelas soluções propostas. A Tabela 6 apresenta os escores obtidos pela solução proposta, variando a quantidade máxima de saídas de cada módulo. É possível observar que há uma tendência de aumento no escore *BLEU* com o aumento na quantidade máxima de saídas dos módulos. Mas, diferente do esperado por conta dessa tendência, o modelo testado que retorna a maior quantidade de saídas, TBMS_10, não foi o que obteve o melhor escore *BLEU*. Uma explicação para esse ocorrido é o fato de que o último módulo, Seleção de Texto, é responsável por selecionar o texto que será retornado, dentre todos que foram gerados. Dessa forma, pode ocorrer que gerar mais textos implique em gerar mais textos de baixa qualidade e que, por uma deficiência do módulo Seleção de Texto, podem ser melhor rankeados e retornados. Além dos escores, a Tabela 6 apresenta também o tempo de execução para a geração dos textos, mostrando a relação entre a quantidade de textos explorados e a quantidade de tempo gasto.

## 5.5 Resultados da análise manual dos textos

Dado que os métodos de avaliação automática utilizados apenas fornecem indícios do nível de qualidade dos textos avaliados, também foi realizada uma análise manual dos textos gerados pela solução proposta. Foram selecionadas aleatoriamente 10 entradas do subconjunto *teste* para cada tamanho de conjunto de triplas, que varia de 1 a 7, totalizando 70 entradas. A avaliação foi feita pelo autor da dissertação, que possui conhecimento avançado em inglês, a língua dos textos avaliados, com o objetivo de identificar problemas de gramaticalidade, adequação ou fluência, bem como analisar possíveis causas. Abaixo seguem os problemas

identificados nos textos da solução proposta (TBMS_9) [5]:

- **Orações mal formadas**: os textos *"californian gemstone benitoite ."* e *"pacific grove in california ."* não são orações bem formadas. Ambos os problemas decorrem de falhas no processo de geração da base de templates, seja por não conter templates suficientes, seja por conter templates de baixa qualidade.
- **Textos contendo mais informação que os dados de entrada**: o texto *"adare manor is located in adare , county limerick ."* informa uma hierarquia entre as localizações referenciadas por *"adare"* e *"county limerick"*, o que não está informado nos dados de entrada. Este problema tem origem no próprio template utilizado, que foi corretamente extraído, mas que pressupõe uma hierarquia entre os locais informados nos dados de entrada.
- **Expressões de referência confusas**: o texto *"johns hopkins university is the parent company of johns hopkins university"* referencia de forma incompleta a informação identificada por *Johns_Hopkins_University_Press*, tornando o texto confuso. Essa referência está presente no *corpus* de onde foi extraída a base de expressões de referência, de forma que o problema decorre dos dados utilizados no treinamento da solução.
- **Expressões de referência erradas**: o texto *"atlanta is from the united states where african americans are an ethnic group and the capital is washington d . c ."* referencia de forma errada a informação identificada por *Asian_Americans* utilizando a referência *"african americans"*. Da mesma forma com o problema anterior, esse problema também decorre dos dados utilizados no treinamento da solução.
- **Predicados verbalizados com mais de um tempo verbal**: o texto *"massimo drago plays for s . s . chieti calcio . he currently plays for u . s castrovillari calcio and manages a . c . cesena ."* verbaliza o *predicado club* com *"plays for"*, no presente, mas há textos, no *corpus* utilizado, que verbalizam esse *predicado* no passado, como *"he once also was a player for US Castrovillari Calcio"*. Novamente, esse comportamento decorre do *corpus* utilizado, que possui textos que verbalizam um mesmo *predicado* utilizando tempos verbais diferentes.
- **Baixa fluência**: o texto *"john cowper powys died in blaenau ffestinlog . he was born in shirley , derbyshire . he wrote a glastonbury romance ."* verbaliza cada tripla com uma sentença, o que o torna menos fluente que, por exemplo, o texto de referência *"John*

[5] A análise detalhada encontra-se no Apêndice A

*Cowper Powys, author of A Glastonbury Romance, was born in Shirley, Derbyshire, and died in Blaenau Ffestiniog.".*

- **Medidas informadas sem a unidade**: o trecho *"serving the city of amsterdam , amsterdam airport schipol is - 3 . 3528 above sea level . "* verbaliza uma informação de altitude sem especificar a unidade de medida. Foi verificado que no *corpus* utilizado há textos que também verbalizam essa medida sem informar a unidade.
- **Expressões de referência repetidas**: o texto *"antioch , california time zone offset is - 7 . the population of antioch , california is 102372 . 925 are area codes for antioch , ca . the total area of antioch , california is 75 . 324 square km ."* faz referência ao *sujeito Antioch,_California* diversas vezes utilizando a mesma expressão de referência *"antioch , california"*, tornando o texto pouco fluente. Dado que seja necessário referenciar uma mesma entidade mais de uma vez e a base de expressões de referência contenha apenas uma expressão para essa entidade, o texto gerado irá repetir essa expressão.
- **Informações verbalizadas de forma incompleta**: o trecho *"buzz aldrin was born in glen ridge new jersey in 1930"* contém a informação representada na tripla $\langle Buzz\_Aldrin,\ birthDate,\ 1930\text{-}01\text{-}20\rangle$, mas verbaliza a data *1930-01-20* informando, de forma incompleta, apenas o ano *"1930"*. Foi verificado que no *corpus* utilizado há textos que também verbalizam essa data informando apenas o ano.

Abaixo seguem dois exemplos de textos gerados pela solução e que são gramaticalmente corretos, adequados e fluentes, bem como as triplas de entrada e os respectivos textos de referência.

Exemplo 1:

**Triplas**:

$\{\langle United\_States,\ language,\ English\_language\rangle,$

$\langle Akron,\_Ohio,\ country,\ United\_States\rangle\}$

**Texto gerado**:

*"akron , ohio is from the united states where the language is english ."*

**Textos de referência**:

- *"The country of Akron, Ohio, is United States, where English is the language spoken."*
- *"Akron, Ohio is located within the United States, where English is the language spoken."*
- *"Akron,Ohio is located within the United States where English is the official language."*

Exemplo 2:

**Triplas**:

$\{\langle 200\_Public\_Square, location, Cleveland\rangle,$

$\langle 200\_Public\_Square, completionDate, 1985\rangle,$

$\langle Cleveland, isPartOf, yahoga\_County\rangle,$

$\langle Cleveland, isPartOf, Ohio\rangle,$

$\langle Cleveland, country, United\_States\rangle\}$

**Texto gerado**:

*"200 public square is located in cleveland , cuyahoga county , ohio , united states . the building was completed in 1985 ."*

**Textos de referência**:

- *"The 200 Public Square was completed in 1985 in Cuyahoga County, Cleveland, Ohio, United States."*
- *"200 Public Square completed in 1985 is in Cleveland (part of Cuyahoga County) in Ohio, U.S."*
- *"200 Public Square was completed in 1985, and is located in Cleveland, Cuyahoga County, Ohio, United States."*

### *5.5.1 Discussão*

Parte dos problemas detectados são causados ou por problemas de qualidade com os textos de referência utilizados para extrair as bases de templates e de referências, ou com o próprio processo de extração dessas bases a partir dos textos de referência. Um texto contendo mais informações que os dados que ele verbaliza consiste em um problema de adequação e gerará um template que ao ser utilizado também irá gerar textos com problemas de adequação. Sendo uma solução orientada por dados, a qualidade deles impacta diretamente na qualidade da solução. Para a criação de um corpus para desenvolvimento de uma solução de GLN, Reiter e Dale (1997) propõem que os textos a serem utilizados sejam segmentados de acordo com a relação entre o segmento e os dados, sendo uma das categorias de segmentos a que inclui trechos cuja informação não provém dos dados. De acordo com os autores, a ocorrência de textos contendo informações não presentes nos dados disponíveis é bastante frequente, no que eles propõem três

soluções, no contexto de desenvolvimento de uma solução de GLN: enriquecer a base de dados, de forma que ela contenha todos os dados verbalizados; remover do texto esses trechos, de forma que o sistema de GLN não os gerará; ou delegar a uma pessoa a responsabilidade por escrever esses trechos.

Outros problemas decorreram da modelagem dos dados de entrada, tanto por possuir ambiguidades, não definindo precisamente que informação representam, mas também por não informar o nível de detalhamento com o qual deverão ser verbalizados. O nível de detalhamento com o qual a informação deve ser verbalizada é um aspecto que normalmente está pouco definido no problema de geração de textos, e se reflete tanto em se um texto deve se referir a uma pessoa utilizando seu nome completo ou apenas o primeiro nome, quanto no exemplo reportado de dados não atômicos. De fato, o problema de uma tarefa de PLN estar mal definida já foi discutido em (III; MARCU, 2004), tratando da tarefa de fusão de sentenças, que consiste em gerar uma sentença verbalizando todo o conteúdo de um conjunto de sentenças. Os autores concluem que essa tarefa está mal definida a partir de experimentos com avaliação manual, onde constataram um alto grau de discordância entre os avaliadores.

# 6 CONCLUSÕES E TRABALHOS FUTUROS

O Capítulo 4 apresentou a arquitetura proposta neste trabalho, bem como os experimentos para validar a sua capacidade de gerar textos. Ela faz uso de uma arquitetura modular, baseada em templates e segue uma abordagem de múltiplas soluções, em que o texto retornado é selecionado a partir de múltiplos textos gerados, variando em relação a ordem de verbalização dos dados, a segmentação em sentenças, os templates e referências utilizados. A fim de reduzir a quantidade de textos gerados, cada módulo utiliza um modelo de rankeamento para selecionar quais as melhores saídas deverão ser retornadas.

Os experimentos apresentados no Capítulo 5, fazendo uso da base de dados WebNLG, validam a capacidade dessa arquitetura de gerar textos com qualidade. Na avaliação com métodos automáticos foi verificado que ela é capaz de superar a maior parte dos trabalhos com as quais foi comparada, em sua maioria fazendo uso de técnicas de Aprendizagem Profunda. Sendo uma solução baseada em múltiplas soluções, também foi avaliada a relação entre a quantidade de textos gerados, representada pelo parâmetro que determina a quantidade máxima de saídas de cada módulo, e a qualidade dos textos. Apesar de o aumento da quantidade de textos gerados acarretar num ganho nos escores para a maior parte dos casos, os melhores escores não foram obtidos com a geração da maior quantidade de textos gerados. Isso aponta para uma deficiência no módulo Seleção de Texto, que não foi capaz de selecionar os melhores textos. Além disso é importante minimizar a quantidade de textos que podem ser gerados mas que não têm qualidade suficiente para serem retornados. Isso inclui tanto a remoção de templates e de referências de baixa qualidade, quanto a identificação de decisões que não são adequadas, como, por exemplo, a verbalização de cada tripla em uma sentença separada, ou a repetição de expressões de referência.

Os templates utilizados foram extraídos a partir de uma base de dados contendo textos anotados, indicando quais trechos fazem referência a entidades. Ao fazer uso de bases de templates e de referências, a solução proposta pressupõe a extração dessas bases. Da mesma forma com a geração e avaliação de textos, o processo de extração dessas bases conduzido manualmente pode ser custoso, o que revela a necessidade de métodos automáticos de alinhamento entre textos e dados, identificando, nos textos, que trechos verbalizam que dados. Esta é uma tarefa que não foi explorada neste trabalho, mas que seria útil na aplicação da arquitetura proposta em um novo domínio, para o qual não há textos anotados. Ela já foi estudada noutros trabalhos, como (MENDES *et al.*, 2011), que explora técnicas de alinhamento automático para anotar textos com referências a entidades presentes na DBPedia. Além da extração automática

de templates, outra melhoria que poderia ser explorado é a derivação de novos templates a partir de templates existentes, utilizando técnicas de Fusão de Sentenças.

Foi utilizada uma estratégia simples para a geração de expressões de referência, utilizando duas bases de dados, uma contendo referências a serem utilizadas para introduzir uma informação no texto, e outra com referências para utilizar com informações já previamente referenciadas. Apesar de essa estratégia funcionar para alguns casos, ela é pouco flexível, na medida em que exige que haja uma referência na base de dados para que uma informação seja referenciada, o que não deveria ser um impedimento para, por exemplo, referenciar entidades, que pode ser feita utilizando pronomes pessoais que podem ser determinados a partir do gênero e número da entidade.

Foram utilizados modelos N-grama como modelos de rankeamento em todos os módulos, mas diferente dos demais módulos, o módulo Seleção de Texto calcula escores para sequências de tamanhos distintos. Dado que o escore é um produto de probabilidades, há um viés a favor de sequências de tamanhos menores. Uma melhoria para o módulo Seleção de Texto pode vir com a adoção de uma penalidade inversamente proporcional ao tamanho das sequências avaliadas, como foi explorado com trabalhos de Tradução Automática como (WU *et al.*, 2016).

Apesar de a solução proposta superar a maior parte das soluções com as quais foi comparada, em relação aos escores retornados pelos métodos BLEU, METEOR e TER, não há uma interpretação direta do que significa ela ter obtido 0.95 pontos de BLEU a menos que a solução rankeada em primeiro lugar. Uma forma de melhor entender esses escores consiste em aplicar alterações controladas aos textos de referência, como remover aleatoriamente palavras, e avaliá-los utilizando esses métodos. Um trabalho futuro consiste em aplicar essa abordagem explorando novas formas de alterações dos textos, bem como outras bases de dados utilizadas para avaliação de soluções de GLN.

Para melhor avaliar a capacidade da arquitetura proposta de gerar textos para triplas, é interessante que ela seja testado fazendo uso de outras bases de dados com esse formato, como, por exemplo, a *KBGen* (BANIK; GARDENT; KOW, 2013) ou a E2E (NOVIKOVA; DUŠEK; RIESER, 2017), bem como seja realizada uma avaliação manual.

Finalmente, considerando aspectos de operação da arquitetura proposta, cabe ressaltar que a divisão do processo em módulos favorece o monitoramento da solução, permitindo acompanhar se há a necessidade de criar novos templates ou novas referências, bem como permitindo maior controle sobre as decisões tomadas ao longo do processo.

# REFERÊNCIAS


ARRIA. *Home | Arria NLG*. 2020. Disponível em: https://www.arria.com/. Acesso em: 20 jun. 2020.

BANAEE, H.; AHMED, M. U.; LOUTFI, A. Towards nlg for physiological data monitoring with body area networks. *In*: *14th European Workshop on Natural Language Generation, Sofia, Bulgaria, August 8-9, 2013*. [*S. l.*: *s. n.*], 2013. p. 193–197.

BANERJEE, S.; LAVIE, A. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. *In*: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. [*S. l.*: *s. n.*], 2005. p. 65–72.

BANIK, E.; GARDENT, C.; KOW, E. The KBGen challenge. *In*: *Proceedings of the 14th European Workshop on Natural Language Generation*. Sofia, Bulgaria: Association for Computational Linguistics, 2013. p. 94–97. Disponível em: https://www.aclweb.org/anthology/W13-2111.

BELZ, A. *et al.* The first surface realisation shared task: Overview and evaluation results. *In*: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 13th European workshop on natural language generation*. [*S. l.*], 2011. p. 217–226.

BLYTHE, J. An overview of planning under uncertainty. *In*: ____. *Artificial Intelligence Today: Recent Trends and Developments*. Berlin, Heidelberg: Springer-Verlag, 1999. p. 85–110. ISBN 3540664289.

BOUAYAD-AGHA, N.; CASAMAYOR, G.; WANNER, L. Natural language generation in the context of the semantic web. *Semantic Web*, IOS Press, v. 5, n. 6, p. 493–513, 2014.

BREITMAN, K.; CASANOVA, M. A.; TRUSZKOWSKI, W. *Semantic web*: concepts, technologies and applications. [*S. l.*]: Springer Science & Business Media, 2007.

CHEN, B.; CHERRY, C. A systematic comparison of smoothing techniques for sentence-level bleu. *In*: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. [*S. l.*: *s. n.*], 2014. p. 362–367.

CHEN, D.; MANNING, C. D. A fast and accurate dependency parser using neural networks. *In*: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [*S. l.*: *s. n.*], 2014. p. 740–750.

CHEN, S. F.; GOODMAN, J. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, Elsevier, [*S. l.*], v. 13, n. 4, 1999.

CHO, K. *et al.* Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

CIMIANO, P. *et al.* Exploiting ontology lexica for generating natural language texts from rdf data. *In*: *Proceedings of the 14th European Workshop on Natural Language Generation*. [*S. l.*: *s. n.*], 2013. p. 10–19.

COLIN, E. *et al.* The webnlg challenge: Generating text from dbpedia data. *In*: *Proceedings of the 9th International Natural Language Generation conference*. [*S. l.*: *s. n.*], 2016. p. 163–167.

CORONAREPORTER. *Corona Repórter COVID19 (@CoronaReporter) / Twitter*. 2020. Disponível em: https://twitter.com/CoronaReporter. Acesso em: 20 jun. 2020.

DBPEDIA. *About | DBpedia*. 2020. Disponível em: https://wiki.dbpedia.org/about. Acesso em: 5 mai. 2020.

DEEMTER, K. V.; THEUNE, M.; KRAHMER, E. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, MIT Press, v. 31, n. 1, p. 15–24, 2005.

DICTIONARY, O. E. *About | Oxford English Dictionary*. 2020. Disponível em: https://public.oed.com/about/. Acesso em: 9 mai. 2020.

DISTIAWAN, B. *et al.* Gtr-lstm: A triple encoder for sentence generation from rdf data. *In*: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [*S. l.*: *s. n.*], 2018. p. 1627–1637.

DUMA, D.; KLEIN, E. Generating natural language from linked data: Unsupervised template extraction. *In*: *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*. [*S. l.*: *s. n.*], 2013. p. 83–94.

DUŠEK, O.; HOWCROFT, D. M.; RIESER, V. Semantic noise matters for neural natural language generation. *arXiv preprint arXiv:1911.03905*, 2019.

FEDERICO, M.; CATTELAN, A.; TROMBETTI, M. Measuring user productivity in machine translation enhanced computer assisted translation. *In*: AMTA MADISON, WI. *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*. [*S. l.*], 2012. p. 44–56.

FERREIRA, T. C. *et al.* Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. *arXiv preprint arXiv:1908.09022*, 2019.

FERREIRA, T. C. *et al.* Enriching the webnlg corpus. *In*: *Proceedings of the 11th International Conference on Natural Language Generation*. [*S. l.*: *s. n.*], 2018. p. 171–176.

FLESCH, R. A new readability yardstick. *Journal of applied psychology*, American Psychological Association, [*S. l.*], v. 32, n. 3, p. 221, 1948.

GARDENT, C. *et al.* The WebNLG Challenge: Generating Text from RDF Data. *In*: *Proceedings of the 10th International Conference on Natural Language Generation*. [*S. l.*: *s. n.*], 2017. p. 124–133.

GATT, A.; KRAHMER, E. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, v. 61, p. 65–170, 2018.

GATTI, L.; LEE, C. van der; THEUNE, M. Template-based multilingual football reports generation using wikidata as a knowledge base. *In*: *Proceedings of the 11th International Conference on Natural Language Generation*. [*S. l.*: *s. n.*], 2018. p. 183–188.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [*S. l.*]: MIT press, 2016.

GREEN, S.; HEER, J.; MANNING, C. D. The efficacy of human post-editing for language translation. *In*: *Proceedings of the SIGCHI conference on human factors in computing systems*. [*S. l.*: *s. n.*], 2013. p. 439–448.

HARRIS, M. D. Building a large-scale commercial nlg system for an emr. *In*: *Proceedings of the Fifth International Natural Language Generation Conference*. [*S. l.*: *s. n.*], 2008. p. 157–160.

HEAFIELD, K. *et al.* Scalable modified Kneser-Ney language model estimation. *In*: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria: [*s. n.*], 2013. p. 690–696. Disponível em: https://kheafield.com/papers/edinburgh/estimate_paper.pdf.

HENDRICKX, I. *et al.* Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. *In*: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 5th International Workshop on Semantic Evaluation*. [*S. l.*], 2010. p. 33–38.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

III, H. D.; MARCU, D. Generic sentence fusion is an ill-defined summarization task. *In*: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004. p. 96–103. Disponível em: https://www.aclweb.org/anthology/W04-1016.

INSIGHTS, A. *Associated Press | Automated Insights*. 2020. Disponível em: https://automatedinsights.com/customer-stories/associated-press/. Acesso em: 20 jun. 2020.

INSIGHTS, A. *Automated Insights: Natural Language Generation*. 2020. Disponível em: https://automatedinsights.com/. Acesso em: 20 jun. 2020.

JAGFELD, G.; JENNE, S.; VU, N. T. Sequence-to-sequence models for data-to-text natural language generation: Word-vs. character-based processing and output diversity. *arXiv preprint arXiv:1810.04864*, 2018.

JURAFSKY, D.; MARTIN, J. H. Speech and language processing: An introduction to speech recognition, computational linguistics and natural language processing. *Upper Saddle River, NJ*: Prentice Hall, 2008.

KLEIN, G. *et al.* OpenNMT: Open-source toolkit for neural machine translation. *In*: *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 67–72. Disponível em: https://www.aclweb.org/anthology/P17-4012.

KNESER, R.; NEY, H. Improved backing-off for m-gram language modeling. *In*: IEEE. *1995 International Conference on Acoustics, Speech, and Signal Processing*. [*S. l.*], 1995. v. 1, p. 181–184.

KOEHN, P. *et al.* Moses: Open source toolkit for statistical machine translation. *In*: *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*. [*S. l.*: *s. n.*], 2007. p. 177–180.

LEE, C. van der; KRAHMER, E.; WUBBEN, S. Pass: A dutch data-to-text system for soccer, targeted towards specific audiences. *In*: *Proceedings of the 10th International Conference on Natural Language Generation*. [*S. l.*: *s. n.*], 2017. p. 95–104.

LEE, C. van der; KRAHMER, E.; WUBBEN, S. Automated learning of templates for data-to-text generation: comparing rule-based, statistical and neural methods. *In*: *Proceedings of the 11th International Conference on Natural Language Generation*. [*S. l.*: *s. n.*], 2018. p. 35–45.

LEHMANN, J. *et al.* Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, IOS Press, v. 6, n. 2, p. 167–195, 2015.

LIN, C.-Y. Rouge: A package for automatic evaluation of summaries. *In*: *Text summarization branches out*. [*S. l.*: *s. n.*], 2004. p. 74–81.

MARCHEGGIANI, D.; PEREZ-BELTRACHINI, L. Deep graph convolutional encoders for structured data to text generation. *arXiv preprint arXiv:1810.09995*, 2018.

MCCRAE, J.; SPOHR, D.; CIMIANO, P. Linking lexical resources and ontologies on the semantic web with lemon. *In*: SPRINGER. *Extended Semantic Web Conference*. [*S. l.*], 2011. p. 245–259.

MEI, H.; BANSAL, M.; WALTER, M. R. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*, 2015.

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

MILLE, S.; DASIOPOULOU, S. Forge at webnlg 2017. Universitat Pompeu Fabra, 2017.

MORYOSSEF, A.; GOLDBERG, Y.; DAGAN, I. Step-by-step: Separating planning from realization in neural data-to-text generation. *arXiv preprint arXiv:1904.03396*, 2019.

NOVIKOVA, J.; DUŠEK, O.; RIESER, V. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*, 2017.

ORABY, S. *et al.* Controlling personality-based stylistic variation with neural natural language generators. *arXiv preprint arXiv:1805.08352*, 2018.

PAPINENI, K. *et al.* Bleu: a method for automatic evaluation of machine translation. *In*: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 40th annual meeting on association for computational linguistics*. [*S. l.*], 2002. p. 311–318.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. *In*: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [*S. l.*: *s. n.*], 2014. p. 1532–1543.

PUZIKOV, Y.; GUREVYCH, I. E2e nlg challenge: Neural models vs. templates. *In*: *Proceedings of the 11th International Conference on Natural Language Generation*. [*S. l.*: *s. n.*], 2018. p. 463–471.

REITER, E. Pipelines and size constraints. *Computational Linguistics*, MIT Press, v. 26, n. 2, p. 251–259, 2000.

REITER, E.; DALE, R. Building applied natural language generation systems. *Nat. Lang. Eng.*, Cambridge University Press, USA, v. 3, n. 1, p. 57–87, mar. 1997. ISSN 1351-3249. Disponível em: https://doi.org/10.1017/S1351324997001502.

REITER, E.; ROBERTSON, R.; OSMAN, L. M. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, Elsevier, v. 144, n. 1-2, p. 41–58, 2003.

REITER, E. *et al.* Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, Elsevier, v. 167, n. 1-2, p. 137–169, 2005.

RIESER, V.; LEMON, O. Natural language generation as planning under uncertainty for spoken dialogue systems. *In*: *Empirical methods in natural language generation*. [*S. l.*]: Springer, 2009. p. 105–120.

SCIENCE, N. *Home | Narrative Science*. 2020. Disponível em: https://narrativescience.com/. Acesso em: 20 jun. 2020.

SHARMA, S. *et al.* Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*, 2017.

SNOVER, M. *et al.* A study of translation edit rate with targeted human annotation. *In*: *Proceedings of association for machine translation in the Americas*. [*S. l.*: *s. n.*], 2006. v. 200, n. 6.

SWARTOUT, W. R. Xplain: A system for creating and explaining expert consulting programs. *Artificial intelligence*, Elsevier, v. 21, n. 3, p. 285–325, 1983.

TIMES, L. A. *Quakebot - Los Angeles Times*. 2020. Disponível em: https://www.latimes.com/people/quakebot. Acesso em: 20 jun. 2020.

VASWANI, A. *et al.* Attention is all you need. *In*: *Advances in neural information processing systems*. [*S. l.*: *s. n.*], 2017. p. 5998–6008.

VEDANTAM, R.; ZITNICK, C. L.; PARIKH, D. Cider: Consensus-based image description evaluation. *In*: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [*S. l.*: *s. n.*], 2015. p. 4566–4575.

WISEMAN, S.; SHIEBER, S. M.; RUSH, A. M. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*, 2017.

WU, Y. *et al.* Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

XU, W.; CALLISON-BURCH, C.; NAPOLES, C. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, MIT Press, v. 3, p. 283–297, 2015.

ZANG, H.; WAN, X. A semi-supervised approach for low-resourced text generation. *arXiv preprint arXiv:1906.00584*, 2019.

ZHU, Y. *et al.* Triple-to-text: Converting rdf triples into high-quality natural languages via optimizing an inverse kl divergence. *In*: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. [*S. l.*: *s. n.*], 2019. p. 455–464.

# APÊNDICE A – INSPEÇÃO MANUAL DETALHADA

Abaixo seguem os casos de problemas identificados, com o resultado da análise feita:

- Os textos *"californian gemstone benitoite ."* e *"pacific grove in california ."* não são orações bem formadas. No primeiro caso, não há templates na base de templates para a tripla de entrada, $\langle California,\ gemstone,\ Benitoite\rangle$, de forma que o texto é gerado utilizando o template *fallback*. Já o segundo texto foi gerado utilizando o template de texto *"**slot_1** in **slot_2**."*, que foi gerado indevidamente a partir do texto de referência *"Albuquerque is in New Mexico, in the United States."*, porque a expressão de referência para a entidade *Albuquerque* foi indevidamente identificada como *"Albuquerque is"*. Ambos os problemas decorrem de falhas no processo de geração da base de templates, seja por não conter templates suficientes, seja por conter templates de baixa qualidade.
- O texto *"adare manor is located in adare , county limerick ."* informa uma hierarquia entre as localizações referenciadas por *"adare"* e *"county limerick"*, o que não está informado nos dados de entrada:

  $\{\langle Adare\_Manor,\ location,\ Adare\rangle,$
  $\langle Adare\_Manor,\ location,\ County\_Limerick\rangle\}$

  Como a modelagem de templates utilizada leva em consideração a ordem das triplas, caso esse mesmo conjunto de triplas fosse utilizado, mas na ordem contrária, seria gerado o texto *"adare manor is located in county limerick , adare ."*, o que não seria preciso, dado que a vila Adare está localizada dentro do condado de Limerick[1]. Um dos textos de referência para esse conjunto de triplas não informa o relacionamento entre *Adare* e *County_Limerick*: *"Adare Manor, Adare is in County Limerick."*. Isso faz parte de um problema mais amplo, que consiste em um texto verbalizar mais informações que as presentes nos dados de entrada. Como a base de templates utilizada na solução deriva de uma base de textos com esse tipo de problemas, é esperado que esse problema ocorra. Um processo de curadoria do corpus utilizado, identificando e solucionando esse tipo de problema resolveria também o problema dos templates dele extraídos.
- O trecho *"johns hopkins university is the parent company of johns hopkins university"*, verbalizando a tripla

[1] https://en.wikipedia.org/wiki/Adare_Manor. Acesso em: 20 jun. 2020

$\langle Johns\_Hopkins\_University\_Press, parentCompany, Johns\_Hopkins\_University \rangle$
fica confuso, dado que tanto o *sujeito* quanto o *objeto* da tripla foram referenciados utilizando a mesma expressão *"johns hopkins university"*. Nos dois textos de referência para esse caso o *sujeito Johns_Hopkins_University_Press* é verbalizado como *"Johns Hopkins University Press"*. É possível verificar na base de dados de referências que uma das referências registradas para a entidade *Johns_Hopkins_University_Press* é *"Johns Hopkins University"*. Essa referência foi adicionada à base de dados a partir do texto de referência *"The American Journal of Mathematics was published by Johns Hopkins University."*, referente à tripla
$\langle American\_Journal\_of\_Mathematics, publisher, Johns\_Hopkins\_University\_Press \rangle$.
Novamente, a qualidade dos textos gerados pela solução reflete a qualidade dos textos de referência, dado que esse texto não contém todas as informações presentes na tripla. Outros exemplos de referências que informam menos do que a informação que referenciam é *"peso"*, como referência a *Mexican_peso*, utilizada para gerar o texto *"bionico is a food found in mexico where the currency is the peso"*; ou *"english"*, como referência a *Philippine_English*, utilizada para gerar o texto *"binignit comes from the philippines where english is spoken ."*.

- O texto *"atlanta is from the united states where african americans are an ethnic group and the capital is washington d . c ."* fala em *"african americans"*, mas as triplas de entrada não fazem referência a essa informação:

  $\{\langle Atlanta, country, United\_States \rangle,$
  $\langle United\_States, ethnicGroup, Asian\_Americans \rangle$
  $\langle United\_States, capital, Washington, \_D.C. \rangle\}$

  É possível verificar na base de dados de referências que uma das referências registradas para a entidade *Asian_Americans* é *"african americans"*. Essa referência foi adicionada à base de dados a partir do texto de referência *"The leader of the United States is known as the president, and within the US there are many* ***African Americans****. Additionally, A severed Wasp is from the United States."*, referente às triplas

  $\{\langle United\_States, leaderTitle, President\_of\_the\_United\_States \rangle,$
  $\langle United\_States, ethnicGroup, Asian\_Americans \rangle$
  $\langle United\_States, country, United\_States \rangle\}$

Novamente um problema com os textos de referência.

- O texto *"massimo drago plays for s . s . chieti calcio . he currently plays for u . s castrovillari calcio and manages a . c . cesena ."* verbaliza as seguintes triplas:

  $\{\langle Massimo\_Drago,\ club,\ U.S.\_Castrovillari\_Calcio\rangle,$

  $\langle Massimo\_Drago,\ club,\ S.S.\_Chieti\_Calcio\rangle$

  $\langle A.C.\_Cesena,\ manager,\ Massimo\_Drago\rangle\}$

  As duas primeiras triplas, cujos *predicado* é *club* são verbalizadas utilizando o tempo presente, com *"plays for"*. Ocorre que, do conjunto de textos de referência para essa entrada, há textos que verbalizam o predicado *club* no passado, como no trecho *"he once also was a player for US Castrovillari Calcio"*. Isso traz a questão de que as triplas não especificam o período de ocorrência dos eventos que representam. A solução proposta também não leva em consideração a temporalidade da informação a ser verbalizada, assumindo que um template adequado irá verbalizar corretamente a informação. Uma possível solução para o caso de um *predicado* ter que ser verbalizado com tempos verbais diferentes seria criar especializações deles, como, no caso de *club*, usando *club_presente*, para indicar que se refere a um evento atual, e *club_passado*, para indicar que se refere a um evento passado.

- O texto *"john cowper powys died in blaenau ffestinlog . he was born in shirley , derbyshire . he wrote a glastonbury romance ."* verbaliza cada tripla com uma sentença, o que o torna menos fluente que, por exemplo, o texto de referência *"John Cowper Powys, author of A Glastonbury Romance, was born in Shirley, Derbyshire, and died in Blaenau Ffestiniog."*. Apesar de no processo de geração o texto *"john cowpher powys wrote a glastonbury romance . he was born in shirley , derbyshire and died in blaenau ffestinlog ."* ser gerado, contendo menos sentenças, ele recebe um escore de -18.485 pelo módulo Text Selection, enquanto o texto retornado recebe um escore de -15.214[2].

- O trecho *"serving the city of amsterdam , amsterdam airport schipol is - 3 . 3528 above sea level . "* verbaliza uma informação de altitude sem especificar a unidade de medida. A tripla que é verbalizada,
  $\langle Amsterdam\_Airport\_Schiphol,\ elevationAboveTheSeaLevel\_(in\_metres),\ -3.3528\rangle$,
  informa a unidade de medida no *predicado*, mas nenhum dos textos de referência verbaliza essa informação detalhando a unidade de medida:

[2] Os escores são negativos pois são o logaritmo da probabilidade estimada.

  - *"The city of Amsterdam is served by Amsterdam Schiphol Airport. This airport has an elevation of -3.3528 from sea level and the runway name 18L/36R 'Aalsmeerbaan. It also has the runway length of 2014.0 metres."*
  - *"Amsterdam Airport Schiphol is -3.3528 above sea level, has a runway name 18L/36R'Aalsmeerbaan which is 2014.0 in length and serves the city of Amsterdam."*
  - *"Amsterdam is served by Amsterdam Airport Schiphol which is -3.3528 above sea level and has a runway length of 2014.0 with the name of 18L/36R."*

  Apesar de o texto gerado perder em clareza ao não informar a unidade de medida, ele está de acordo com os textos utilizados como referência.

- O texto *"antioch , california time zone offset is - 7 . the population of antioch , california is 102372 . 925 are area codes for antioch , ca . the total area of antioch , california is 75 . 324 square km ."* faz referência ao *sujeito Antioch,_California* diversas vezes utilizando a mesma expressão de referência *"antioch , california"*, tornando o texto pouco fluente. Além disso, a sentença *"925 are area codes for antioch , ca ."* está com o verbo flexionado de forma errada, dado que o sujeito contém apenas um elemento. Em relação à repetição da expressão de referência, apesar de serem utilizadas duas bases de dados de referências, a primeira contendo referências para serem utilizadas para introduzir uma informação, e a segunda para se referir a uma informação que já foi introduzida, pode ocorrer, para uma determinada informação, de não haver referências na segunda base. É o caso da referência *"antioch , california"* para *Antioch,_California*. Uma melhoria poderia ser a utilização de um módulo responsável por gerar referências na forma de pronomes, de acordo com o gênero e número da informação a ser referenciada.
- O trecho *"buzz aldrin was born in glen ridge new jersey in 1930"* contém a informação representada na tripla $\langle Buzz\_Aldrin, birthDate, \text{1930-01-20}\rangle$, mas verbaliza a data *1930-01-20* informando apenas o ano *"1930"*. Ocorre que há tipos de dados, como datas e numerais, que tanto podem ser representados de diversas formas, como podem ser divididos em partes, como uma data contém dia, mês e ano, ou um numérico contém parte inteira e parte decimal. Não há informações na tripla que oriente como representar essa data ou que partes dela são relevantes. Os textos de referência para esse caso todos representam a data por completo, como *"20th January 1930"* ou *"January 20, 1930"*. Analisando as referências que há na base de referências para *1930-01-20* há desde referências completas,

como *“20 / 01 / 1930"* e *“january 20th , 1930"*, quanto outras referências que não informam a data completa, como *“the twentieth of january"*, e referências erradas como *“january 30th , 1930"*. Novamente, dado que a solução é baseada em dados, os problemas na base de dados utilizada no treinamento são refletidos nos textos gerados.