Programa de Pós-Graduação em

Computação Aplicada

Mestrado/Doutorado Acadêmico

Blanda Helena de Mello

Pythia NLG – Um modelo para integração de recursos voltados à geração de linguagem natural

São Leopoldo, 2019

UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

BLANDA HELENA DE MELLO

PYTHIA NLG – UM MODELO PARA A INTEGRAÇÃO DE RECURSOS VOLTADOS À GERAÇÃO DE LINGUAGEM NATURAL

São Leopoldo
2019

BLANDA HELENA DE MELLO

PYTHIA NLG – UM MODELO PARA A INTEGRAÇÃO DE RECURSOS VOLTADOS À GERAÇÃO DE LINGUAGEM NATURAL

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre, pelo Programa de Pós-Graduação em Computação Aplicada - PIPCA da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Prof. Dr. Sandro José Rigo

Coorietadora: Profª. Drª. Marta Rosecler Bez

São Leopoldo

2019

M525p Mello, Blanda Helena de.
Pythia NLG – um modelo para integração de recursos voltados à geração de linguagem natural / Blanda Helena de Mello – 2019.
140 f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2019.
"Orientador: Prof. Dr. Sandro José Rigo Coorientadora: Profª. Drª. Marta Rosecler Bez."

1. Geração de linguagem natural. 2. Pythia NLG. 3. Framework. 4. Orquestração. 5. Reuso. I. Título.

CDU 004

Dados Internacionais de Catalogação na Publicação (CIP)
(Bibliotecária: Amanda Schuster – CRB 10/2517)

Blanda Helena de Mello

Pythia NLG - Um modelo para integração de recursos voltados à geração de linguagem natural.

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 26/03/2019

BANCA EXAMINADORA

Prof. Dr. Silvio Cesar Cazella – UFCSPA

Prof. Dr. Jorge Luis Victória Barbosa – UNISINOS

Prof. Dr. Sandro José Rigo (Orientador)

Profª. Drª. Marta Rosecler Bez (Coorientadora)

São Leopoldo,

Prof. Dr. Rodrigo da Rosa Righi

Coordenador PPG em Computação Aplicada

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

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Dedico esta dissertação a minha família, por sua compreensão e apoio incondicional.

Aos meus pais, que me ensinaram a seguir em frente e nunca desistir...

Aos meus irmãos, pois sabemos que as nossas conquistas sempre serão de todos: Bethânia, Bárbara, Sebastião, Bernardo, Bruno, Betina, Júnior e João.

AGRADECIMENTOS

Agradeço especialmente ao meu professor Orientador Sandro José Rigo, por sua orientação ao longo desta pesquisa. Seus conselhos, apoio e paciência, sempre disposto a responder às dúvidas e incertezas em todos os momentos.

À professora Marta Rosecler Bez, que aceitou a tarefa de Coorietadora nesta caminhada. Pelas palavras de apoio, sua constante orientação e presença em mais esta conquista. Sua dedicação, companheirismo e amor ao que fazes são exemplos para a profissional e pessoa que desejo ser. Que me permita a vida ter sua energia para fazer o que fazes por nós, seus "preferidos".

A vocês, caros professores, que nos inspiram a ir além, meu sincero agradecimento. Sua confiança nos encoraja a voar mais alto para construir novos caminhos.

Às pessoas que tive a honra de conhecer, colegas, professores e grupo de pesquisa da Feevale e Unisinos, vocês tornaram as noites mais alegres e amenas para que os desafios nos fizessem crescer.

Ao Diego Pinheiro, de conhecido do grupo de pesquisa à colega e amigo. Grata pelo companheirismo, pelo constante apoio e bom humor. Juntos iremos mais longe, construindo através da educação.

A minha família, amigos e colegas, sua compreensão, carinho, e companheirismo foram a força para avançar. Ao Gustavo Cervi, pelo apoio e todos os conselhos para prosseguir com o mestrado.

Aos avaliadores membros da banca, por aceitar esta tarefa.

À Unisinos e colegas, por me proporcionarem um ambiente acolhedor para desenvolver esta pesquisa.

À CAPES, por continuar a investir em caminhos por meio da educação e proporcionar esta bolsa.

E, principalmente a Deus, pela graça da vida e suas realizações.

RESUMO

A geração de linguagem natural é um campo de estudos destinado à análise da estrutura da linguagem para construir textos em linguagem compreensível ao ser humano. A geração de linguagem natural possui diferentes nichos de aplicação, seja para gerar respostas a perguntas, gerar conteúdo criativo como narrativas e poesias ou para flexibilizar sentenças promovendo uma interação mais amigável com seus usuários, entre outros. Observa-se o crescimento da área a partir de pesquisas voltadas para estes diferentes nichos de uso. Existe, no entanto, uma carência de ferramentas para uso e exploração de tais aplicações em conjunto, o que poderia propiciar benefícios de reuso e comparação de desempenho. A análise de trabalhos recentes na área permite observar sistemas contendo técnicas empregadas em apenas uma aplicação ou finalidade, mas não se observa a existências de uma ferramenta que possa atender a diferentes aplicações ou promover a comparação de técnicas. Esta pesquisa tem como objetivo apresentar a proposta de um modelo para um ambiente que permita a integração de ferramentas já conhecidas no campo de geração de linguagem natural, e facilite a exploração e comparação de novas técnicas e algoritmos. A metodologia empregada é de cunho exploratório e buscou-se identificar soluções para um problema inicialmente apresentado. O desenvolvimento da pesquisa envolveu a realização de um estudo bibliográfico na área de geração de linguagem natural, a fim de identificar informações referentes à área de interesse. Por fim, fez-se uma análise das possibilidades para a construção de um modelo que contemple as necessidades evidenciadas pelo problema identificado, relacionado à integração e reutilização de recursos para geração de linguagem natural. Como contribuição, propõe-se um modelo denominado Pythia NLG, que promove uma abordagem aberta para permitir a integração contínua de recursos nesta área. Como forma de avançar no sentido de sua avaliação, são descritos cenários de uso que permitem verificar o seu potencial. O protótipo implementado Pythia NLG apresentou uma avaliação preliminar favorável, por meio de cenários de uso, para um ambiente que permita a integração e reutilização de algoritmos e técnicas em tarefas complexas de GLN. Conclui-se que o desenvolvimento do protótipo Pythia NLG propõe uma ferramenta com potencial para exploração de novos experimentos na área de geração de linguagem, facilitando a reutilização e compartilhamento de técnicas, bem como a possibilidade de integração de algoritmos de avaliação e comparação de resultados. O protótipo implementado Pythia NLG permite que acadêmicos e pesquisadores na área de GLN beneficiem-se em novas pesquisas com um ambiente para experimentações de tarefas GLN.

Palavras-Chave: Geração de Linguagem Natural, Pythia NLG, *Framework, Orquestração, Reuso.*

ABSTRACT

The natural language generation is a field of study designed to analyze the structure of language to finally construct text in a language understandable to the human being. The natural language generation has different application niches, either to generate answers to questions, creative content such as narratives and poetry or to refine sentences to achieve a friendlier interaction with its users, among others. It is observed the growth of the area, with research focused on these different niches of use. There is, however, a lack of tools to use and exploit such applications together, which could provide benefits of reuse and performance comparison. The analysis of recent works in the area allows observing techniques used in only one application or purpose, but, for the best of our knowledge, there is not a tool that can attend different applications or promote the comparison of techniques. This study aims to present proposal of a model for an environment that allows the integration of tools already known in the field of natural language generation and facilitates the exploration and comparison of new techniques and algorithms. The methodology used is exploratory, and we sought to identify solutions to a problem initially presented. The development of the research involved the accomplishment of a bibliographic study in the area of natural language generation, in order to identify information related to the area of interest. Finally, an analysis was made of the possibilities for the construction of a model that contemplates the needs evidenced by the problem identified, related to the integration and reuse of resources for the natural language generation (NLG). As a contribution, we propose a model nominated Pythia NLG, which promotes an open approach to allow the continuous integration of resources in this area. As a way of moving towards its evaluation, use scenarios are described that allow verifying its potential. The implemented Pythia NLG prototype presented a favorable evaluation, through use scenarios, for an environment that allows the integration and reuse of algorithms and techniques in complex NLG tasks. It is concluded that the development of the Pythia NLG prototype proposes a tool with potential for exploration of new experiments in the area of natural language generation, facilitating the reuse and sharing of techniques, as well as the possibility of integration of algorithms for evaluation and comparison of results. The implemented Pythia NLG prototype allows NLG scholars and researchers to benefit from new research with an environment for NLG task trials.

Keywords: Natural Language Generation, Pythia NLG, Framework, Orchestration, Reuse.

LISTA DE FIGURAS

Figura 1 Cadastro do protocolo na ferramenta StArt. 41
Figura 2 Árvore de artigos selecionados pela ferramenta ToS. 42
Figura 3 Gráfico de resultados obtidos nos motores de busca. 43
Figura 4 Interface para inserir as referências na ferramenta StArt. 44
Figura 5 Etapas internas da ferramenta StArt. 44
Figura 6 Gráfico resumo das fases de seleção. 46
Figura 7 Gráfico quantitativo dos artigos selecionados. 48
Figura 8 Gráfico comparativo de publicações por ano. 48
Figura 9 Gráfico das formas de validação apresentadas. 55
Figura 10 Relação de recursos léxicos e corpus. 56
Figura 11 Gráfico comparativo de linguagens mais utilizadas. 57
Figura 12 Gráfico comparativo de técnicas utilizadas. 57
Figura 13 Gráfico comparativo das metodologias adotadas. 58
Figura 14 Gráfico comparativo de modelos probabilísticos adotados. 59
Figura 15 Comparativo de bases de dados utilizadas. 59
Figura 16 Gráfico comparativo resumo da relação com o foco da pesquisa. 60
Figura 17 Visão geral modelo proposto 68
Figura 18 Exemplo de uma arquitetura pipeline clássica. 70
Figura 19 Exemplo do pacote para encapsular mensagens. 74
Figura 20 Formato de dados padrão para realizar solicitação de serviços. 75
Figura 21 Formato de dados padrão para comunicação – comunicação interna. 76
Figura 22 Especificação de um modelo de composição conforme definição. 78
Figura 23 Cenário 1: exemplo de requisição feita ao Web Service. 81
Figura 24 Cenário 1: exemplo do modelo de composição selecionado. 82
Figura 25 Cenário 1: exemplo de requisição para alocar o recurso de consulta Virtuoso. 83
Figura 26 Cenário 1: exemplo de requisição para alocar o recurso de seleção de tripla. 84
Figura 27 Cenário 1: exemplo de requisição para alocar o recurso de geração a partir de tripla. 84
Figura 28 Cenário 2: Fragmento de uma rede bayesiana de diagnóstico de diarreia e risco de diarreia construída para o projeto *Health Simulator*. 85
Figura 29 Cenário 2: exemplo de requisição padrão para o Web Service pelo projeto *Health Simulator*. 87
Figura 30 Cenário 2: modelo de composição para projeto *Health Simulator*. 87

Figura 31 Cenário 2: exemplo de requisição para alocar recurso e remover palavras irrelevantes. 88
Figura 32 Cenário 2: exemplo de requisição ao recurso responsável por selecionar as palavras de maior relevância na sentença. 89
Figura 33 Cenário 2: exemplo de requisição para o recurso responsável pela verificação de ambiguidade entre palavras de uma sentença. 89
Figura 34 Modelo proposto com destaque para os componentes implementados. 93
Figura 35 Implementação núcleo de processamento e orquestrador. 96
Figura 36 Cenário 4: Modelo de interface para catálogo e criação de composições. 99
Figura 37 Modelo da base de dados utilizada. 101
Figura 38 Cenário 1: Registro de um novo canal. 108
Figura 39 Cenário 1: Composição para geração de respostas a partir de URI. 109
Figura 40 Cenário 1: Solicitação de serviço para geração de resposta a partir de list-uri. 110
Figura 41 Cenário 1: resultado da solicitação para geração de resposta. 110
Figura 42 Cenário 1: Alterações necessárias para usar outra composição na solicitação. 111
Figura 43 Cenário 1: Alterações na solicitação - geração de perguntas a partir de list-uri. 112
Figura 44 Cenário 1: Resultado da solicitação para geração de pergunta. 112
Figura 45 Cenário 2: Registro de um novo canal para Educator Pro. 114
Figura 46 Cenário 2: Composição para sumarização de documento de texto. 114
Figura 47 Cenário 2: Solicitação de serviço para sumarização documento de texto. 114
Figura 48 Cenário 2: Resultado da solicitaçao para sumarização de texto. 115
Figura 49 Cenário 2: elementos que alteraram de uma composição de sumarização para outra. 116
Figura 50 Cenário 2: elementos que alteraram em na solicitação ao mudar a composição de sumarização. 116
Figura 51 Cenário 2: elementos que alteraram nos resultados recebidos ao mudar a composição de sumarização. 116
Figura 52 Cenário 3: composição para avaliação de sumarização automática. 118
Figura 53 Cenário 3: solicitação para avaliação de sumarização automática. 118
Figura 54 Cenário 3: resultado da avaliação da sumarização com composição para uso de métrica. 119
Figura 55 Cenário 4: primeira etapa para construção de uma composição de recursos. 120
Figura 56 Cenário 4: Criando nova composição para sumarização automática e aplicação de métrica. 120
Figura 57 Cenário 4: solicitação construída para a nova composição. 121
Figura 58 Cenário 4: resultado da composição construída - recurso de sumarização e avaliação. 122
Figura 59 Exemplo de workflow construído para projeto OKBQA em uma versão demo. 124

LISTA DE TABELAS

Tabela 1 Resultados obtidos nos motores de busca. 42
Tabela 2 Artigos selecionados para esta revisão. 46
Tabela 3 Relação de revistas das publicações selecionadas. 49
Tabela 4 Relação de resultados às perguntas foco da pesquisa. 50
Tabela 5 Comparativo entre trabalhos relacionados e aspectos do desenvolvimento. 65
Tabela 6 Definição dos tipos de dados. 73
Tabela 7 Formato padrão para encapsular mensagens. 74
Tabela 8 Descrição do formato de dados padrão para solicitação de serviço. 75
Tabela 9 Descrição do formato de dados padrão para comunicação interna. 76
Tabela 10 Descrição do formato de dados para definição de uma composição de recursos. 77
Tabela 11 Ações possíveis ao componente Cliente. 97
Tabela 12 Ações possíveis ao componente Produtor (orquestrador). 98
Tabela 13 Ações possíveis ao componente Consumidor. 98
Tabela 14 Relação de tabelas criadas no banco de dados MySQL. 101
Tabela 15 Parâmetros de entrada para o algoritmo de sumarização. 103
Tabela 16 Parâmetros de entrada para o algoritmo de avaliação ROUGE - MS e AS. 104
Tabela 17 Parâmetros de entrada para o algoritmo Athena!. 105
Tabela 18 Aspectos de implementação dos trabalhos relacionados 126

LISTA DE ABREVIATURAS

PLN	Processamento de Linguagem Natural
GLN	Geração de Linguagem Natural
RST	*Rhetorical Structure Theory*
LOD	*Linked Open Data*
URI	*Uniform Resourc Identifier*
WOS	*Web of Science*
WS	*Web Service*
RB	Rede Bayesiana
BI	*Business Intelligence*
T2T	*Text-to-text*
D2T	*Data-to-text*
POS	*Part-of-speech*
CHUNK	*Chunking*
NER	*Named Entity Recognition*
SRL	*Semantic Role Labeling*
CT	*Content Task*
ST	*Structure Task*
AMQP	*Advanced Message Queuing Protocol*
UUID	*Universally Unique IDentifier*
SA	Sumarização Automática
ROUGE	*Recall-Oriented Understudy for Gisting Evaluation*
ROUGE-L	*Recall-Oriented Understudy for Gisting Evaluation: Longest Common Subsequence (LCS)*

SUMÁRIO

1 INTRODUÇÃO 23
1.1 Contexto e motivação 23
1.2 Questão de pesquisa 25
1.3 Objetivos 25
1.4 Metodologia 26
1.5 Organização do documento 26

2 FUNDAMENTAÇÃO TEÓRICA 29
2.1 Processamento de linguagem natural 29
2.2 Geração de linguagem natural 30
2.2.1 Classificação de sistemas de geração de linguagem natural 31
2.2.2 Fases de um sistema de geração de linguagem natural 32
2.3 Comunicação 33
2.3.1 Protocolo AMQP 34
2.3.2 RabbitMQ – *Message Broker* 36
2.4 Considerações 36

3 TRABALHOS RELACIONADOS 37
3.1 Revisão sistemática de literatura sobre geração de linguagem natural 37
3.1.1 Protocolo 37
3.1.2 Seleção e definição dos estudos 39
3.1.3 Desenvolvimento da revisão sistemática 41
3.1.4 Fases de seleção 42
3.1.5 Fase 1 – validação ToS 43
3.1.6 Fase 2 – critérios de inclusão e exclusão 44
3.1.7 Fase 3 – leitura de título, palavras-chave e resumo 45
3.1.8 Fase 4 – leitura da introdução e conclusão 45
3.1.9 Fase 5 – leitura integral dos artigos 45
3.1.10 Resultados 46
3.1.11 Perguntas foco da pesquisa 50
3.1.12 Análise dos artigos 53
3.1.13 Considerações 60
3.2 Trabalhos relacionados com a oportunidade de pesquisa 62
3.2.1 Comparativo entre os artigos relacionados 62

4 MODELO PROPOSTO 67
4.1 Visão geral do modelo 67
4.1.1 Usuário ou sistema 68
4.1.2 Comunicação 68
4.1.3 Orquestrador 69
4.1.4 Núcleo de Processamento 69
4.1.5 Fontes de dados 71
4.1.6 Recursos 71
4.1.7 Bases Léxicas 71
4.1.8 Serviços 71
4.1.9 Query's 73
4.2 Definição dos formatos de dados 73
4.2.1 Padrão para encapsular pacotes 74
4.2.2 Solicitação de serviço 74
4.2.3 Comunicação entre os módulos 76
4.2.4 Composição de recursos 76
4.3 Algoritmos, recursos e dinâmica de funcionamento do modelo 78
4.4 Cenários de uso 80
4.4.1 Geração de sentenças a partir de triplas – Cenário 1 80
4.4.2 Geração de respostas a partir de perguntas pré-definidas – Cenário 2 85
4.5 Protótipo Implementado 92
4.5.1 Visão geral do protótipo 92
4.5.2 Componentes implementados e seu funcionamento 93
4.5.3 Fontes de dados 100

4.5.4 Serviços 102
4.6 Algoritmos 103
4.6.1 Algoritmos de Sumarização: *ConceptRank* e *LexRank* 103
4.6.2 Algoritmos de Avaliação: Métricas *Rouge* e *Bleu* 103
4.6.3 Algoritmo Athena! 104
4.7 Considerações sobre o protótipo 105

5 ASPECTOS DE AVALIAÇÃO 107
5.1 Cenários de uso sobre as funcionalidades do protótipo 107
5.1.1 Cenário 1 – Serviço para a geração de perguntas/respostas 107
5.1.2 Cenário 2 – Solicitação de serviço para a geração de sumarização de documento 113
5.1.3 Cenário 3 – Solicitação de serviço de avaliação – métrica para sumarização automática 117
5.1.4 Cenário 4 – Inclusão de uma nova composição no protótipo 119
5.2 Comparação com ferramentas existentes 123

6 CONCLUSÕES 127
6.1 Trabalhos futuros 128

7 REFERÊNCIAS 129

1 INTRODUÇÃO

Nesta seção serão apresentados o contexto e a motivação que nortearam o desenvolvimento desta pesquisa, bem como a definição da questão de pesquisa. Na sequência, define-se o objetivo geral e os objetivos específicos. Por fim, é descrita a metodologia adotada, seguida da organização do documento.

1.1 Contexto e motivação

Nos últimos anos, pesquisas dedicadas ao estudo da área de geração de linguagem natural (GLN) têm crescido exponencialmente, observando-se avanços consideráveis quanto ao desenvolvimento de novos algoritmos e quanto à descoberta de novos focos para a aplicação de geração de linguagem (GATT; KRAHMER, 2018). De acordo com (REITER; DALE, 1997, 2000), a GLN é um subcampo da inteligência artificial e linguística computacional. De forma geral, a GLN pode ser compreendida como um processo de representação de informações da linguagem natural em estruturas que permitam analisar e simular a linguagem humana.

O processo de produção de linguagem é uma atividade social inerente ao ser humano, que envolve a resolução de problemas em contextos espaciais, temporais e situacionais (BATEMAN; ZOCK, 2012). Um sistema de GLN aborda o tratamento da informação de forma similar à forma com que seres humanas a tratam. Ou seja, é necessário um planejamento do uso da informação disponível para identificar o conteúdo e o formato do que será transformado em linguagem natural, como etapa preliminar à uma possível etapa de construção de frases ou de textos, que proporcionará a construção final de um discurso em linguagem natural.

Esta escolha para a construção do discurso varia de acordo com a fonte de conhecimento a ser utilizada. As fontes de dados para sistemas de GLN podem ser fontes linguísticas ou não linguísticas. As abordagens que utilizam fontes linguísticas são conhecidas como abordagens "text-to-text" e geralmente utilizam como fonte os textos em forma oral ou escrita. As abordagens denominadas não-linguísticas, também denominadas de "data-to-text" partem de dados brutos, tais como bases de dados estruturados, planilhas, dados numéricos, dados de sensores. Além disso, podem usar representações semânticas, tais como bases de dados abertos e ligados – *Linked Open Data (LOD)*[1] e ontologias. Esta transformação de um conjunto de dados e informações para trechos de textos em linguagem natural representa um dos pontos complexos de um sistema destinado à geração de linguagem (BATEMAN; ZOCK, 2012).

O planejamento de discursos é frequentemente aplicado à sistemas de GLN por meio de tarefas bem definidas. As tarefas de GLN referem-se às escolhas envolvidas: estas incluem determinação e estruturação de conteúdos, por exemplo, a escolha da mensagem; a estrutura retórica em vários níveis, texto, parágrafo e frases; a escolha das palavras que melhor se adequam às estruturas sintáticas e, por fim, a determinação dos componentes do texto, destacando o seu início, meio e final (BATEMAN; ZOCK, 2012). Este cenário pode ser

[1] http://linkeddata.org/

facilmente observado na tradicional rotina de tarefas empregadas em sistemas de GLN, e envolve as etapas de determinação de conteúdo, estruturação de texto, agregação de sentenças, lexicalização e, por fim, geração das expressões (GATT; KRAHMER, 2018).

Em sistemas de geração de linguagem, não há necessidade de discutir o que um sistema de GLN deve entregar como produto final. Parece ser de comum acordo entre a comunidade que a saída de um sistema GLN sempre seja texto. Por outro lado, de acordo com (GATT; KRAHMER, 2018), as fronteiras entre as diferentes abordagens para sistemas de GLN turvam os critérios e meios de classificação destes sistemas quanto a suas entradas. Para a comunidade, esta classificação de sistemas não é clara. A classificação de um sistema para sumarização de texto, que é frequentemente caracterizada como um sistema *text-to-text,* utilizando tarefas de extração de informações (EI). Sistemas com objetivo de EI aplicam tarefas distintas de sistemas GLN, pois seu principal objetivo está na extração de pontos chave dos conteúdos. Por outro lado, eventualmente sistemas para a sumarização de texto podem utilizar-se de tarefas de GLN após extrair as características do texto de entrada, com o objetivo de gerar conteúdos criativos a partir destes, e não limitar-se à extração da informação. É, portanto, classificado como um sistema de GLN. Com este cenário, (GATT; KRAHMER, 2018) propõe, em sua pesquisa, considerar como sistemas de GLN apenas aqueles que apresentam como entrada um conjunto de dados não linguísticos, sendo esta uma decisão com o intuito de direcionar sua pesquisa.

Esta diversidade de aplicações das tarefas GLN têm motivado a busca por soluções que possam reutilizar técnicas existentes. Alguns estudos apresentam como objetivo a construção de arquiteturas genéricas para viabilizar a reutilização de componentes e suportar o compartilhamento (MELLISH; EVANS, 2004; MELLISH et al., 2006). Estes trabalhos não são recentes, porém demonstram a identificação precoce desta necessidade de reuso e integração, enfatizando a importância de recursos e dados reutilizáveis e a definição formal de arquiteturas modulares. Com a popularização e diversificação destes sistemas de GLN, esta necessidade tornou-se mais evidente e compartilhada por um maior número de pesquisas.

Portanto, esta pesquisa apresenta o estudo de diversos cenários de geração de linguagem natural, área responsável por atender a objetivos comunicativos específicos, através do processo de construção automática de um texto em linguagem natural. Mediante o desenvolvimento de uma revisão sistemática de literatura, identificou-se a existência de uma tendência ao crescimento desta área, em especial na diversificação e aprimoramento de técnicas, com consequente diversificação de formas de utilização. Tendo em vista a convergência de novas tecnologias promovendo a interação com usuários e máquinas, contexto que contribui para a proposição de novas técnicas, ferramentas e mesmo modelos para implementação, pode-se observar trabalhos de aplicações variadas, como evidenciado em estudos desenvolvidos na área (VICENTE et al., 2015; PERERA; NAND, 2017; GATT; KRAHMER, 2018).

A procura por ferramentas que atendam a esta demanda, com a intenção de explorar as aplicações de GLN e enriquecer trabalhos com a comunicação em linguagem natural, tem fomentado experimentos que estão apoiados no desenvolvimento de ferramentas dedicadas, para atender à uma necessidade (ARAUJO; HENTGES; RIGO, 2018; SILVA et al., 2018; MELLO et al., 2018). Todavia, este cenário conduz para uma situação que culmina em retrabalho, o que ocorre quando uma solução requer mais de um tipo de resultado na geração de linguagem. Neste caso, é necessário o desenvolvimento e adequação das diferentes técnicas para cada novo projeto, envolvendo também a adaptação do material de apoio, estrutura e comunicação entre as novas ferramentas.

Para atender a esta necessidade de reuso e integração de recursos de GLN, algumas pesquisas têm focado no desenvolvimento de recursos para permitir o uso integrado de diversas ferramentas, indispensáveis para realizar determinado serviço. Alguns trabalhos já atuam nesta direção (KIM DBCLS et al., 2017; SINGH et al., 2018; WALKOWIAK, 2018), promovendo o desenvolvimento de ferramentas que incorporam atividades de geração de linguagem natural, porém, ainda sem abstrair serviços para mais de uma forma de aplicação, mas sim fornecendo um contexto mais flexível para um tipo definido de aplicação.

Após os estudos realizados na área de geração de linguagem, pode-se observar a necessidade de integrar, reutilizar e compartilhar técnicas e algoritmos. No entanto, ainda não foram estabelecidas soluções categóricas para resolver este problema. Esta lacuna foi identificada e adotada como motivação para esta pesquisa.

Os diferenciais a serem destacados nesta pesquisa são a integração de algoritmos, técnicas e recursos voltados à área de GLN, que pode ser observada no modelo proposto – Pythia NLG. Este beneficia a reutilização e compartilhamento de recursos, facilitando a realização de experimentos de integração de técnicas. Destaca-se também que a presente pesquisa fortalece a possibilidade de avaliação e comparação de resultados para recursos já implementados. Outra contribuição que pode ser destacada é que o modelo permite integrar algoritmos que tratam de dados com entradas diversas. Uma consequente contribuição é a definição de um formato de dados para comunicação entre os recursos e componentes do modelo.

1.2 Questão de pesquisa

Durante o estudo realizado, observou-se que trabalhos abordam a integração de algoritmos e recursos existentes, uma oportunidade de pesquisa evidenciada pela ampla variedade de algoritmos exclusivos para a geração de linguagem natural, bem como, uma crescente variedade de recursos. Apesar de existirem aplicações com finalidades relacionadas, em geral, estas são variações otimizadas dos mesmos algoritmos ou recursos. No estudo realizado não foi encontrado um ambiente de integração e compartilhamento consolidado.

Desta forma, esse trabalho visa responder a seguinte questão de pesquisa: Quais os componentes necessários para um modelo com o objetivo de integração de recursos e algoritmos para geração de linguagem natural?

1.3 Objetivos

O objetivo geral desta dissertação é definir um modelo para reuso e compartilhamento de algoritmos e recursos para geração de linguagem natural. Justifica-se o trabalho ao observar a variedade de técnicas e algoritmos desenvolvidos e exaustivamente validados no campo da geração de linguagem, bem como a necessidade de avaliação conjunta e integração de componentes.

Para suportar o objetivo geral descrito, os seguintes objetivos específicos são indicados:

a) Estudar as características e aspectos que um modelo de compartilhamento e integração de algoritmos e recursos para a geração de linguagem natural apresenta, com base na literatura da área;

b) Propor um modelo para integração e compartilhamento de algoritmos e recursos de geração automática de linguagem natural;

c) Desenvolver o protótipo para validar experimentalmente o modelo proposto e os mecanismos desenvolvidos;

d) Intermediar a avaliação dos experimentos desenvolvidos, por meio de recursos integrados e implementados.

1.4 Metodologia

Os aspectos metodológicos considerados no desenvolvimento desta pesquisa iniciam com a realização de pesquisa bibliográfica para colher informações sobre o tema em análise. Segue-se um estudo aprofundado dos trabalhos encontrados, relacionados ao tema de interesse, a fim de identificar aspectos e características. O segundo passo envolveu uma análise de bases de dados abertos e conectados, para colher detalhes de interesse do trabalho. No terceiro passo foi especificada a solução a ser desenvolvida, junto com a escolha e definição de tecnologias para o desenvolvimento de um protótipo e seus aspectos de avaliação.

Foi considerada a abordagem descrita por (KÖCHE, 2011), segundo a qual o presente trabalho pode ser caracterizado como uma pesquisa de caráter exploratório, pois a mesma busca inicialmente descrever em detalhes um determinado cenário no qual se identifica o problema tratado. Com base neste ponto inicial, busca-se demonstrar possibilidades para o encaminhamento de uma solução, que neste sentido foi descrita e validada com base no estudo de um caso específico.

Foi adotado por (WAZLAWICK, 2014) para definir o seguinte método de trabalho: a) realização de revisão bibliográfica, estudo documental e de diferentes técnicas para geração de linguagem natural; b) realização de estudo e análise de trabalhos relacionados com as áreas tratadas; c) proposta de um modelo e suas definições; d) descrição da proposta de protótipo e abordagem para avaliação; e) realização de estudo de cenários para avaliar os resultados possíveis; f) descrição e documentação da pesquisa.

1.5 Organização do documento

No Capítulo 2, inicialmente é apresentada uma visão geral do campo de processamento de linguagem natural, em seguida apresenta-se os conceitos fundamentais da geração de linguagem natural e estágios para desenvolvimento, seguida da definição e detalhes da tecnologia adotada para a camada de comunicação do protótipo implementado. Uma revisão sistemática sobre a área de pesquisa foi desenvolvida para identificação de trabalhos correlatos, e pode ser conferida no Capítulo 3. No Capítulo 4 é descrito o modelo proposto, sua construção e definição dos componentes, bem como a descrição do modelo de controle. No Capítulo 5 é apresentado o protótipo desenvolvido, seguido do Capítulo 6, onde

detalha-se a avaliação do protótipo proposto por meio de cenários. Por fim, no Capítulo 7, são apresentadas as conclusões, problema de pesquisa, contribuições e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os temas utilizados para o desenvolvimento desta pesquisa. Define-se o conceito de Processamento de Linguagem Natural na Seção 2.1, com uma breve análise das áreas de aplicação a ela destinadas. Na Seção 2.2 é definida a conceituação de Geração de Linguagem Natural, as arquiteturas comumentemente utilizadas para sistemas nesta linha, bem como as principais técnicas aplicadas às soluções nesta área de pesquisa. Por fim, na Seção 2.3 descreve-se o modelo de comunicação adotado no desenvolvimento do protótipo e características desta tecnologia.

2.1 Processamento de linguagem natural

A Linguagem Natural (LN) pode ser compreendida como a linguagem utilizada no cotidiano, tanto o ato de fala quanto a escrita, semelhante ao vocabulário normal, com variações para padrões informais (LOPES, 2002). O epíteto "natural" tem a função de distinguir das manifestações da comunicação natural – como a escrita e a fala – as linguagens mais formais, como as notações matemáticas ou lógicas, ou ainda as linguagens de programação como Java, LISP, C++, C#, Python e inúmeras outras linguagens (JACKSON; MOULINIER, 2007).

O Processamento de Linguagem Natural (PLN) é compreendido como uma tarefa destinada a analisar a linguagem, falada ou escrita, e torná-la compreensível às máquinas. As tarefas de PLN ainda permanecem distantes do processo de compreensão humana, onde as palavras observadas, ativam conceitos semanticamente relacionados, episódios relevantes e experiências sensoriais, realizando uma espécie de tarefa extremamente complexa para PLN (CAMBRIA; WHITE, 2014). Um sistema que deseja adquirir conhecimento precisa entender a linguagem que os seres humanos usam (RUSSEL; NORVIG, 2013). A área de pesquisa dedicada a compreensão de linguagem com foco nesta característica humana, denomina-se *Natural Language Understanding* (NLU), e almeja de fato alcançar a compreensão da linguagem pela máquina, da mesma forma que faz o ser humano (JACKSON; MOULINIER, 2007). No entanto, o processo de compreensão da linguagem, no PLN, normalmente segue as etapas tradicionais da análise linguística: fonologia, morfologia, sintaxe, semântica, pragmática/discurso; gradualmente, seguindo do texto para suas intenções (INDURKHYA; DAMERAU, 2010).

Algumas atividades e técnicas no PLN são consideradas como etapas básicas, que eventualmente são aplicadas antes de quaisquer outras tarefas mais complexas, para tratar o texto em linguagem natural, através de classificações diversas, em formas e níveis diferentes, e posteriormente outras técnicas mais sofisticadas podem ser usadas. Em maior ou menor nível, estão presentes em todos os contextos de aplicações que tenham a finalidade de trabalhar com linguagem natural, uma vez que são os primeiros passos do processamento (COLLOBERT et al., 2011; HIRSCHBERG; MANNING, 2015; SCHLÜNZ; DLAMINI; KRUGER, 2016). Estas etapas são descritas brevemente a seguir:

a) *Part-of-speech Tagging (POS)*[2]: tarefa que objetiva categorizar e etiquetar palavra por palavra na sentença analisada, indicando sua regra sintática correspondente, como por exemplo, classificando os termos em adjetivos, substantivos, verbos e

[2] https://nlp.stanford.edu/software/tagger.html

advérbios (SCHLÜNZ; DLAMINI; KRUGER, 2016). É uma das tarefas fundamentais do PLN, pois as atividades subsequentes beneficiam-se desta marcação das partes do discurso, identificando focos de ambiguidade e outras análises, por meio da classe nomeada, que define o seu papel. Uma análise desta e outras tarefas similares pode ser conferida em mais detalhes no trabalho elaborado por (KHIN; AUNG, 2016).

b) *Chunking (CHUNK)*[3]*:* frequentemente denominado como a análise superficial, nesta etapa é necessário compreender como funciona a sintaxe e a estrutura da linguagem em estudo. Eventualmente CHUNK é a segunda etapa de PLN, pois aproveita a categorização feita em POS, para identificar a estrutura do discurso, as dependências na estrutura gramatical e as palavras que a formam (SCHLÜNZ; DLAMINI; KRUGER, 2016). Tem o objetivo de rotular segmentos do discurso, o que significa agrupar os termos por sua relação estrutural, usualmente os resultados são apresentados em uma estrutura de arvore.

c) *Named Entity Recognition (NER)*[4]*:* também conhecido como a classificação semântica de entidades, tem o papel de definir os elementos atômicos nas sentenças, categorizando-as por seu, por exemplo, "pessoa", "localização", "expressão do tempo", "organização" ou "quantidades e medidas" (KHIN; AUNG, 2016; SCHLÜNZ; DLAMINI; KRUGER, 2016). Tem o objetivo de rotular os termos e dar nome as "coisas", identificar os objetos do discurso.

d) *Semantic Role Labeling (SRL)*[5]*:* tem a responsabilidade de definir um papel semântico aos termos categorizados sintaticamente e agrupados por sua relação. De acordo com (SCHLÜNZ; DLAMINI; KRUGER, 2016), o estado da arte para sistemas SRL consiste em uma série de etapas, e somente então pode-se chegar ao SRL. Estas etapas são, inicialmente produzir uma árvore com a análise da sentença, para identificar seus nós, que representam argumentos para um determinado verbo e, finalmente, classificar os nós para identificar o SRL dos termos agrupados.

O processamento de linguagem natural é, portanto, o primeiro passo ao se trabalhar com linguagem natural, vista como uma etapa de pré-processamento da linguagem. A seguir, na Seção 2.2, será descrita em detalhes a geração de linguagem natural.

2.2 Geração de linguagem natural

A Geração de Linguagem Natural (GLN) é um subcampo da inteligência artificial e linguística computacional (REITER; DALE, 2000). A Linguística Computacional (LC) atua como ponto de confluência para áreas como a linguística aplicada, a informática e inteligência artificial, também interessadas nas formas de interação entre computadores e humanos (VICENTE et al., 2015). Este interesse converge nas diferentes tarefas conhecidas do PLN, assim como a GLN, que atua como campo complementar ao PLN, e envolve-se não apenas no processamento da linguagem e suas etapas para análise do discurso, como a geração efetiva de

[3] http://www.nltk.org/howto/chunk.html

[4] https://nlp.stanford.edu/software/CRF-NER.html

[5] http://www.nltk.org/howto/propbank.html

linguagem natural por um computador (CAGAN, 2016), sob a forma e estrutura da linguagem natural humana.

Não obstante, este estudo tem como foco de pesquisa o estudo dos sistemas GLN, suas aplicações e técnicas. Neste sentido, é imprescindível identificar formas de classificar estes sistemas. Estudou-se os critérios a serem considerados para classificação. Para esta pesquisa optou-se por seguir a abordagem popularmente usada, sugerida por (VICENTE et al., 2015; GATT; KRAHMER, 2018) para este tipo de classificação, que espelha-se na entrada de dados que o sistema está apto a trabalhar, considerando seu produto final, empregando tarefas GLN no processo.

2.2.1 Classificação de sistemas de geração de linguagem natural

Sistemas para geração de linguagem podem ser classificados por uma infinidade de critérios, desde as técnicas que são empregadas, objetivos do sistema até sua entrada, relacionados ao formato de dados que o sistema recebe como entrada. A seguir é detalhada a classificação de sistemas de GLN por entrada de dados.

A classificação por meio do tipo de informações de entrada em um sistema GLN não se limita a textos ou discursos. Popularmente é empregada uma nomenclatura para a classificação de sistemas de geração de linguagem, denominados: dados-para-texto (*data-to-text: D2T)* e texto-para-texto (*text-to-text: T2T)* (VICENTE et al., 2015). Sistemas com enfoque na geração de linguagem D2T referem-se a quaisquer sistemas que têm o objetivo de gerar linguagem natural a partir de entrada de dados não linguísticos, os quais contemplam desde imagens, vídeos, dados numéricos, informações estatísticas, bancos de dados, arquivos de log, corpus rotulados e demais bases de conhecimento estruturado (GATT; KRAHMER, 2018).

Trabalhos nesta área vêm, cada vez mais, explorando diferentes informações de entrada não linguísticas. A seguir alguns trabalhos para exemplificar situações, não se limitando a estes para mapear as possibilidades de entradas:

- geração de relatórios acessíveis e resumos técnicos a partir de dados médicos (GATT et al., 2009; HUNTER et al., 2011, 2012);
- geração de trechos e descrições para vídeos (KRISHNAMOORTHY et al., 2013; KHAN; AL HARBI; GOTOH, 2015);
- geração de descrições a partir de dados de sensores (PEREIRA; TEIXEIRA, 2015; MOLINA; SANCHEZ-SORIANO; CORCHO, 2015; AIRES; SOBRAL NUNES, 2016);
- geração textual automática a partir de dados meteorológicos (SRIPADA et al., 2005; RAMOS-SOTO et al., 2015).

Por outro lado, sistemas que recebem como entrada textos e orações, são denominados T2T, e possuem atividades especialmente moldadas para lidar com o conteúdo que entrou em linguagem natural, e construir a saída de acordo com o objetivo do sistema. É comum ver aplicações para este cenário de dados voltados para a sumarização de textos, simplificação e/ou condensação de informações, assim como a presença dos sistemas de diálogo (VICENTE et al., 2015).

2.2.2 Fases de um sistema de geração de linguagem natural

Conseguir gerar linguagem natural com qualidade a partir de sistemas não é uma tarefa fácil, e requer especial atenção para que o resultado produzido esteja coerente, e tenha os elementos adequados, de acordo com as regras do idioma. A coesão e coerência são itens essenciais para que a informação seja comunicada (VICENTE et al., 2015). Sistemas de GLN frequentemente seguem fases para essa classificação e análise da entrada, decomposta em 3 estágios que tradicionalmente são executados em 6 etapas.

Dentre as etapas de classificação da linguagem, há tarefas destinadas ao tratamento do conteúdo (*Content Task: CT*) e tarefas destinadas à trabalhar sobretudo com a estrutura da língua (*Structure Task*: ST), visando a saída em linguagem natural (REITER; DALE, 1997; CURRY et al., 2013; VICENTE et al., 2015; CAGAN, 2016; PERERA; NAND, 2017).

a) *Macro-planning* ou *Text Planning:* trata-se do estágio inicial do processo de geração, neste momento será selecionado, dentre o conteúdo de entrada (D2T ou T2T) o que será utilizado no processo de geração. O sistema deverá decidir as informações que serão incluídas na construção da linguagem. Geralmente, as informações de entrada excedem o que se deseja expressar, com mais detalhes e pouco compreensíveis. É necessário, neste caso, fazer a seleção do conteúdo que é relevante para o contexto de geração, o ato de falar sobre – o que dizer? (*what to say?*):

   - *Content Determination* (estágio CT)*:* responsável principalmente pela seleção, o planejamento do conteúdo que será empregado na geração da linguagem natural. Esta seleção claramente é relacionada ao público-alvo, o texto que será gerado pode ser destinado a profissionais de uma área específica, alunos, crianças, adultos, idosos e órgãos ou fontes consumidoras. São algumas das variáveis que precisam ser levadas em consideração ao extrair os dados para realizar uma comunicação efetiva;

   - Document *Structuring* (estágio ST): responsável pela estruturação do conteúdo que foi selecionado entre os dados de entrada. O conteúdo que será transmitido nas mensagens em linguagem natural foi extraído do conjunto de informações. Agora é necessário estruturá-las em um formato que seja compreensível ao leitor; uma estrutura que respeite a ordem temporal, espacial e situacional das informações e faça sentido aos conceitos extraídos, respeitando as regras da língua (BATEMAN; ZOCK, 2012). Esta etapa pode ser denominada por autores como estruturação de texto, do discurso ou planejamento do documento (*structure text, structure discourse* ou *document plan*).

b) *Micro-planning* ou *Sentence Planning:* recebe o planejamento do documento com as informações extraídas, e suas relações com cada mensagem selecionada para a geração do texto, bem como a estrutura que deve apresentar ao final do processo. A responsabilidade do micro planejamento recai sobre as tarefas linguísticas, o texto que será gerado deverá estar completamente caracterizado às regras da língua. A forma de saída será em uma árvore de palavras, representando suas ligações sintáticas, limites da oração e suas relações de correferência. Este estágio compreende as etapas da agregação, geração de frases e expressões de referência e lexicalização. Contempla a estruturação primordial das sentenças – como dizer? (*how to say?*):

- *Lexicalization* (estágio CT)*:* responsável por trabalhar nos termos e conceitos que serão incluídos no texto. Até o momento as informações que devem constar no texto foram selecionadas e reunidas em mensagens, agrupadas pelo estágio *document plan*. Agora é o momento em que a mensagem será convertida em linguagem natural. A complexidade desta etapa pode ser em maior ou menor nível, depende da implementação definida. Caso a intenção seja trabalhar com mensagens alternadas, optando-se por selecionar uma mensagem aleatoriamente, dentre as opções inicialmente formadas, para resultar textos diferentes para entradas idênticas, assim como outras abordagens possíveis;
- *Referring Expression Generation* (estágio CT)*:* extensivamente estudado pela linguística computacional, REG está preocupado com as relações que as entidades devem assumir no texto (KRAHMER; VAN DEEMTER, 2012). Em suma, é responsável por eliminar a redundância nas sentenças. Deve selecionar termos e partes da sentença para identificar as entidades redundantes, e relaciona-las corretamente, para remover e substituir por referências sem perder a coerência do texto;
- *Aggregation* (estágio ST)*:* responsável por agrupar as palavras recebidas após o *document plan*, e constituir pequenos grupos coerentes de mensagens. Uma característica da comunicação humana é comunicar-se através de discursos que podem ser diretos ou compostos de mais de uma intenção comunicativa. Eventualmente, ao se comunicar com uma pessoa, o diálogo alterna entre uma frase que contêm apenas um assunto, ou contêm uma ideia que conclui um assunto iniciado anteriormente. Outra responsabilidade desta tarefa é a agregação de sentença (*sentence aggregation*). Tem a finalidade de dar às mensagens uma característica da comunicação humana, informar através de uma única sentença, mais de uma intenção comunicativa através de mensagens e termos agrupados em expressões.

c) *Realization:*

- *Linguistic Realisation* (estágio CT) e *Surface Realization* (estágio ST): responsável pelo processo de constituir as sentenças ou frases, de acordo com a estrutura da sentença e mecânica da linguagem adotada para a saída. Por fim as palavras e frases relevantes foram decididas, os termos redundantes foram removidos, e as relações foram feitas corretamente. Agora as partes da sentença precisam ser combinadas, gerar as adequações das sentenças às regras sintática, morfológica e ortográfica da língua, incluindo as conjugações de verbos e concordância, preposições e demais termos auxiliares.

Nesta seção foram apresentadas as principais características envolvidas para a GLN, bem como as tarefas comumentemente presentes a estes sistemas. A seguir será detalhado o modelo de comunicação adotado para desenvolvimento do protótipo Pythia NLG.

2.3 Comunicação

O modelo Pythia NLG demanda uma estratégia de comunicação eficiente e flexível entre os seus componentes, que coincide com os aspectos observados nas abordagens de

orquestração. Para tanto, foi realizado um estudo a fim de identificar ferramentas que pudessem contribuir com o objetivo de orquestração de recursos na camada de comunicação do modelo. A seguir é explicado em detalhes o protocolo AMQP, um protocolo destinado a troca de mensagens.

2.3.1 Protocolo AMQP

O *Advanced Message Queuing Protocol*[6] *(AMQP)* é um protocolo de padrão aberto. Este propõe o uso de comunicação por meio de mensagens para conectar aplicativos, sistemas e processos em diferentes linguagens, plataformas e localizações. O AMQP é uma especificação que define a semântica de um protocolo de mensagens interoperável. O uso de mensagens ou enfileiramento de mensagens é um estilo de comunicação, que permite uma arquitetura fracamente acoplada.

O AMQP permite que os recursos que o utilizem sejam independentes. A vantagem de sua utilização reflete na possibilidade de uma interoperabilidade completa, pois o protocolo AMQP define tanto o protocolo de rede quanto a semântica dos serviços no servidor. Assim, tem-se um conjunto de recursos de mensagens denominados *Advanced Message Queuing Protocol Model (AMQ Model)* (VINOSKI, 2006).

A partir do *AMQ Model,* propõe-se os componentes com a capacidade de rotear e armazenar as mensagens trocadas, assim como as regras necessárias para conectá-los. O componente ***exchange*** tem a função de receber as mensagens de todos os utilizadores e encaminhar para os componentes ***message queues***, de acordo com as configurações definidas nas mensagens. O componente ***message queue*** recepciona as mensagens encaminhadas, armazenando-as até o momento em que possam ser processadas pelo consumidor. Por fim, o componente ***binding*** tem a finalidade de estabelecer o relacionamento entre um ***message queue*** e as configurações e critérios de roteamento definidos na mensagem (VINOSKI, 2006; AIYAGARI et al., 2008).

Uma camada de comunicação que possibilite este cenário independente, permite a sistemas um bom desempenho no funcionamento autônomo. Com a possibilidade de conectar aplicativos, plataformas e diferentes linguagens utilizando um padrão aberto completo (FERNANDES et al., 2013), bem como atribuir controle total das regras de filas de mensagens e trocas ao arquiteto, não ao código.

Uma das principais vantagens de adotar a abordagem de mensagens para sistemas é que estes são fracamente acoplados. Os componentes do sistema não têm a necessidade de saber exatamente onde estão localizados, assim como não há necessidade de saber a localização dos demais, apenas saber um nome (***exchange***) e o intermediador (***broker***). Os sistemas podem, assim, ser desenvolvidos de maneira independente. A terminologia usual, segundo a especificação do protocolo AMQP, para os componentes recorrentes – apenas os principais – são descritas a seguir.

a) ***Broker:*** refere-se ao papel semelhante a um servidor. No AMQP os termos empregados são cliente e servidor. Recebe as mensagens produzidas pelos produtores e entrega-as aos consumidores.

[6] https://www.amqp.org/

b) ***Connection:*** o AMQP está no nível *application* do protocolo TCP para realizar as entregas. As conexões são tipicamente de longa duração (*long-lived*). As conexões podem ser autenticadas utilizando o padrão TLS-SSL. É possível encerrar uma conexão utilizando o método *Connection.Close*.

c) ***Channel:*** conexões leves, multiplexadas em uma mesma conexão TCP. Esta característica evita a necessidade de manter múltiplas conexões TCP abertas em simultâneo. Estes permitem isolamento da interação entre clientes e intermediador.

d) ***Exchange***: é o ponto de entrada de todas as mensagens, pois é responsável por aplicar as regras, determinando os destinos das mensagens. As regras são compostas pelo tipo de *exchange* definido (*direct, fanout, topic*) e chaves de roteamento definidas no *binding*.

- ***direct*** **(ponto-a-ponto):** um *exchange* do tipo *direct* permite que as mensagens sejam enviadas diretamente, baseando-se na chave de roteamento definida. A mensagem que é publicada em um *exchange direct* será roteada a apenas uma fila. Esta abordagem pode ser interessante, quando há interesse em um balanceamento de carga, onde todas as mensagens estarão em uma única fila, e haverá N clientes consumidores para processá-las, retirando-as à medida que estejam disponíveis.
- ***fanout (multicast)*****:** um *exchange* do tipo *fanout* ignora as chaves de roteamento nas mensagens. Simplesmente a mensagem será roteada a todas as filas encontradas. Se *N* filas estiverem conectadas, todas receberão uma cópia da mensagem recebida. Um *exchange fantout* é tipicamente incorporado a cenários onde um sistema precisa de um canal *broadcast* para seus clientes.
- ***topic*** **(publicar-assinar):** um *exchange* do tipo *topic* faz o roteamento de mensagens de uma para muitas filas baseando-se nas definições de *routing keys* e *binding queues*. Este cenário prevê situações em que consumidores esperam receber diferentes tipos de mensagens, portanto trabalha no cenário *publish/subscribe*.

e) ***Queue*****:** trata-se do destino final das mensagens, prontas a serem consumidas. Uma mensagem pode ser replicada a várias filas, se esta definição for feita na regra de roteamento.

f) ***Consumer***: a utilização de uma camada de comunicação com protocolo AMQP permite enfileirar mensagens a um destino. No entanto, apenas armazenar mensagens não é útil, é necessário que estas sejam direcionadas a quem as possa processar, um consumidor. No AMQP, um consumidor tem duas maneiras de realizar esta ação. Pode assinar uma determinada fila, e passará a receber todas as mensagens que entrarem a esta fila, ou consumi-las conforme necessário.

g) ***Binding:*** são regras utilizadas pelos *exchanges* criados, para rotear as mensagens às filas. Um *exchange* e uma fila são declaradas, e um cliente AMQP normalmente vincula-os, definindo um *binding*. Se necessário, é possível associar uma chave de roteamento (*routing key*) ao *binding*, adicionando uma regra ao roteamento.

Atualmente, existem diversas ferramentas que implementam o protocolo AMQP. Uma delas é a ferramenta RabbitMQ. A seguir serão detalhadas características da ferramenta.

2.3.2 RabbitMQ – *Message Broker*

A ferramenta RabbitMQ é uma implementação *open source* e completa do *broker* AMQP. Atualmente com a versão 0.9.1 da especificação AMPQ. O RabbitMQ *broker* é um servidor de mensageria, escrito em *Erlang*, e implementa, além do protocolo AMQP, *gateways* para protocolos HTTP, STOMP e MQTT (FERNANDES et al., 2013), assim como ser utilizado no balanceamento de carga, ou no suporte a recursos avançados, como *clustering* (DOSSOT et al., 2014).

Segundo (FERNANDES et al., 2013), ao avaliar comparativamente a performance de serviços *RESTful* e o protocolo AMQP, os melhores resultados foram obtidos com uso do AMQP. O critério de estudo baseou-se na média de trocas de mensagens por um período de tempo. Neste cenário, para situações em que haja um grande volume de troca de mensagens, o RabbitMQ apresenta melhores resultados.

RabbitMQ é uma ferramenta de mensageria popularmente empregada, e para tanto, a apresenta interfaces para diferentes linguagens, e uma variedade de plugins disponíveis, assim como um ambiente para gerenciamento, através de uma *interface web*. Algumas das linguagens as quais permite implementação, são: Java, Python, C#, PHP, Ruby entre outras [7].

2.4 Considerações

Neste capítulo foram detalhados os aspectos conceituais das áreas de processamento de linguagem natural e geração de linguagem natural. Bem como apresentação e discussão de tecnologias para camada de comunicação AMQP e RabbitMQ.

O estudo aprofundando em PLN e GLN justifica-se pelo objetivo da presente pesquisa, em propor um modelo que contemple a integração de soluções já desenvolvidas e consagradas nestes campos de estudo. E, por fim, ser possível atender tarefas complexas de GLN e PLN sem a preocupação com atividades desenvolvidas para domínios específicos, aplicando reuso de tarefas já construídas.

O estudo destas áreas contribuiu para a definição do modelo, a ser apresentado no Capítulo 4, para integrar diferentes tarefas e atividades de PLN e GLN. Estes possuem distintas tecnologias, aplicações, plataformas e linguagens empregadas, assim como diferentes tipos e fontes de dado. O estudo realizado proporcionou o reconhecimento das características comuns presentes em tarefas, algoritmos e técnicas comumentemente presentes nesta área.

Com o objetivo de desenvolver um modelo que proporcione independência entre os componentes envolvidos foram estudados recursos de comunicação e optou-se pelo uso do RabbitMQ. A escolha de um *message broker* para camada de comunicação atribui ao modelo a característica de baixo acoplamento, comunicação assíncrona entre os componentes, assim como a integração facilitada entre componentes de diferentes linguagens e plataformas, necessário para ambientes de atividades distintas.

[7] http://www.rabbitmq.com/getstarted.html

3 TRABALHOS RELACIONADOS

Visando identificar as técnicas, ferramentas e tecnologias comumentemente adotadas para a geração de linguagem natural, com especial atenção para aquelas aplicadas à área da saúde, foi desenvolvida uma revisão sistemática sobre a GLN, a qual abrange o período do ano de 2011 até a data de sua elaboração em agosto de 2017. Buscou-se, com este intervalo de datas os últimos 7 anos de publicações. Este estudo está descrito na Seção 3.1 a seguir. Além desta revisão sistemática, foi realizada uma segunda pesquisa, desta vez com foco específico no estudo de trabalhos que tratem a problema de reuso e integração de algoritmos e recursos para GLN. Esta segunda pesquisa está descrita na Seção 3.2 deste capítulo.

3.1 Revisão sistemática de literatura sobre geração de linguagem natural

A revisão sistemática sobre geração de linguagem natural foi desenvolvida com base no protocolo criado por (MEDEIROS, 2016), uma mescla entre dois protocolos de áreas distintas, o protocolo da pesquisadora Elisabete Kitchenham (KITCHENHAM; CHARTERS, 2007), referente à área da computação, e o protocolo de Recomendação PRISMA (MOHER et al., 2010), direcionado para a área da saúde.

3.1.1 Protocolo

Esta revisão sistemática foi desenvolvida baseando-se nos protocolos apresentados anteriormente, seguindo o seu planejamento com:

a) Título

Revisão sistemática sobre geração de linguagem natural.

b) Resumo

Esta revisão sistemática utilizará como guia o protocolo criado por (MEDEIROS, 2016), tendo como tema desta revisão a utilização de geração de linguagem natural e identificação de técnicas híbridas na área de interesse já citada, bem como a busca de tais técnicas e demais abordagens aplicadas à saúde e simuladores. O protocolo foi desenvolvido com a orientação de especialistas da área da computação (ligados à saúde e inteligência artificial) e, caso faça-se necessário, posteriormente serão realizados ajustes, sendo estes sempre justificados após os mesmos.

c) Objetivo – PICOC

Para o auxílio na criação da *string* de busca, (KITCHENHAM; CHARTERS, 2007) recomendam a utilização do PICOC (População, Intervenção, Comparação, Resultados e Contexto). A formulação da pesquisa é apresentada a seguir.

Formulação da pesquisa

a) Foco da questão:

A presente revisão sistemática tem por objetivo o levantamento dos trabalhos publicados na área de inteligência artificial, com foco em geração de linguagem natural, aplicados à simuladores e/ou sistemas de interação simultânea com o usuário, bem como a geração de diálogos em linguagem natural, privilegiando técnicas e validações já aplicadas à área da saúde.

b) Questões de interesse:

- Técnicas utilizadas;
- Abordagens;
- Estudos já realizados;
- Aplicações recorrentes;
- Resultados positivos.

c) Palavras-chaves:

Natural Language Processing e Natural Language Generation.

d) Intervenção:

Verificar e relacionar as características, técnicas e aplicações de geração de linguagem natural e identificar pesquisas que tenham interesse na área da saúde.

e) Controle:

Não será utilizado nesta revisão.

f) Efeito:

Identificar as oportunidades de pesquisa e aplicação de inteligência artificial, com foco em geração de linguagem, para identificar técnicas e soluções aplicadas à área da saúde.

g) Medida do Resultado:

Gerar o embasamento necessário ao marco teórico da dissertação, bem como a publicação de artigos científicos nas áreas de estudo aqui relacionadas.

h) População de interesse:

Pesquisadores, desenvolvedores, profissionais da área da computação, saúde e demais interessados em geração de linguagem natural.

i) Aplicação:

Esta revisão sistemática tem como foco o uso por pesquisadores e desenvolvedores da área da computação, pois viabiliza um aporte teórico e técnico das principais ferramentas e abordagens no uso da tecnologia, auxiliando novos trabalhos de investigação e futuras aplicações.

j) Desenho do experimento:

Não será desenvolvido nesta revisão sistemática.

k) Financiamento:

CAPES

Seleção das bases de dados

a) Definição dos critérios de seleção das bases:

Após análise das bases de maior interesse, foram definidas 4 bases a serem utilizadas nesta revisão. Em vistas da interdisciplinaridade, buscou-se artigos em bases com foco na área de computação, assim como saúde.

Desta forma, na área da computação, serão utilizadas as bases de dados: IEEE*xplore*, que fornece acesso para mais de três milhões de documentos de algumas das publicações mais citadas no mundo em Engenharia Elétrica, Ciência da Computação e Eletrônica (IEEEXPLORE, 2017); ACM *Digital Library*, base de dados bibliográficos com foco na área da computação (ACM DIGITAL LIBRARY, 2017); assim como, utilizou-se a base Web of Science, composta de periódicos de assuntos diversos (WEB OF SCIENCE, [s.d.]). Para a área da saúde, será utilizada a base de dados PubMed. A mesma conta com citações da literatura biomédica do MEDLINE (PUBMED, 2017). Para acesso irrestrito às bases, utilizou-se o convênio da universidade.

b) Idiomas das Fontes de Dados:

Serão considerados somente os materiais bibliográficos no idioma em inglês.

c) *String* de busca:

Com base nas palavras-chave definidas anteriormente, gerou-se a *string*, onde as chaves obrigatórias são "natural language processing" e "natural language generation", as demais chaves de busca tem o objetivo de evidenciar nos resultados, áreas e aplicações. Esta *string* é aplicada nos motores de buscas definidos para esta revisão; adaptadas para refinar seu conteúdo, e possibilitar um resultado mais apurado à busca.

(("natural language generation" OR "NLG") AND ("natural language processing" OR "NLP"))

d) Artigos de controle:

A presente revisão integra duas áreas distintas, assim caracterizada como interdisciplinar, sendo elas computação e saúde. Desta forma, optou-se por não utilizar nenhum artigo de controle.

3.1.2 Seleção e definição dos estudos

a) Critérios para inclusão/exclusão dos estudos. Os artigos selecionados precisam obedecer aos seguintes critérios:
   - O artigo deve ter sua publicação entre os anos 2011 a 2017;
   - Ser um artigo científico publicado em congresso científico;
   - O artigo deve estar escrito no idioma inglês;
   - O artigo deve apresentar uma forma de validação;
   - O artigo deve estar disponível na íntegra na internet ou disponível através dos convênios fornecidos pela instituição de ensino;
   - Artigos da base *Web of Science* devem ser validados pela ferramenta *Tree of Science* (ToS).

b) Definição dos tipos de estudo:

- Serão selecionados estudos teóricos, qualitativos e/ou quantitativos referentes ao tema.

c) Procedimentos para seleção dos estudos

Após a definição da *string* de busca, a mesma será executada nos motores de buscas já definidos. As referências coletadas destas bases são registradas na ferramenta StArt (*State of the Art through Systematic Reviews*), desenvolvida pelo Laboratório de Pesquisa de Engenharia de *Software* (Lapès), sob supervisão do Departamento de Computação da Universidade Federal de São Carlos. A ferramenta tem como objetivo dar suporte à pesquisadores no processo de desenvolvimento de revisões sistemáticas. A mesma é utilizada por alunos de pós-graduação e possui um histórico positivo de *feedbacks* (FABBRI et al., 2016).

Para o processo de seleção, optou-se pela divisão do mesmo em duas grandes etapas. No primeiro momento, o pesquisador deverá ler o título dos artigos, suas palavras-chave e *abstract,* aplicando os critérios de seleção e qualidade já definidos no protocolo.

d) Fases da seleção de artigos:

- fase 1: validar as publicações da base *web of science* na ferramenta ToS;
- fase 2: aplicar os critérios de inclusão e exclusão;
- fase 3: leitura do título, palavras-chave e *abstract*;
- fase 4: leitura completa da introdução e conclusão;
- fase 5: leitura integral dos artigos restantes para responder às perguntas da presente pesquisa.

Posteriormente, os artigos selecionados na primeira etapa, terão sua introdução e conclusão lidos por completo, para, por fim, ler integralmente os artigos selecionados; objetiva-se, ao final, responder às perguntas foco da pesquisa.

e) Critérios de qualidade das fases da Revisão Sistemática:

- Q01. A solução proposta pelos artigos é aplicada à área da saúde?
- Q02. Quais foram os autores base na fundamentação teórica dos artigos?
- Q03. Os artigos apresentaram resultados positivos?
- Q04. Os artigos possuem alguma forma de validação?
- Q05. Qual formato/público de validação foi aplicado?
- Q06. Foi desenvolvido algum protótipo?
- Q07. Quais as ferramentas e recursos relevantes?
- Q08. Os artigos apresentaram recursos léxicos ou corpora utilizados?
- Q09. Quais as linguagens utilizadas?
- Q10. Quais as técnicas utilizadas?
- Q11. Quais foram os tipos de metodologias adotadas?
- Q12. Foi utilizado algum modelo estatístico auxiliar?
- Q13. Os artigos apresentaram a origem dos dados utilizados?

- RQ14. Qual o foco de ligação entre o tema dos artigos e o tema proposto nesta revisão?

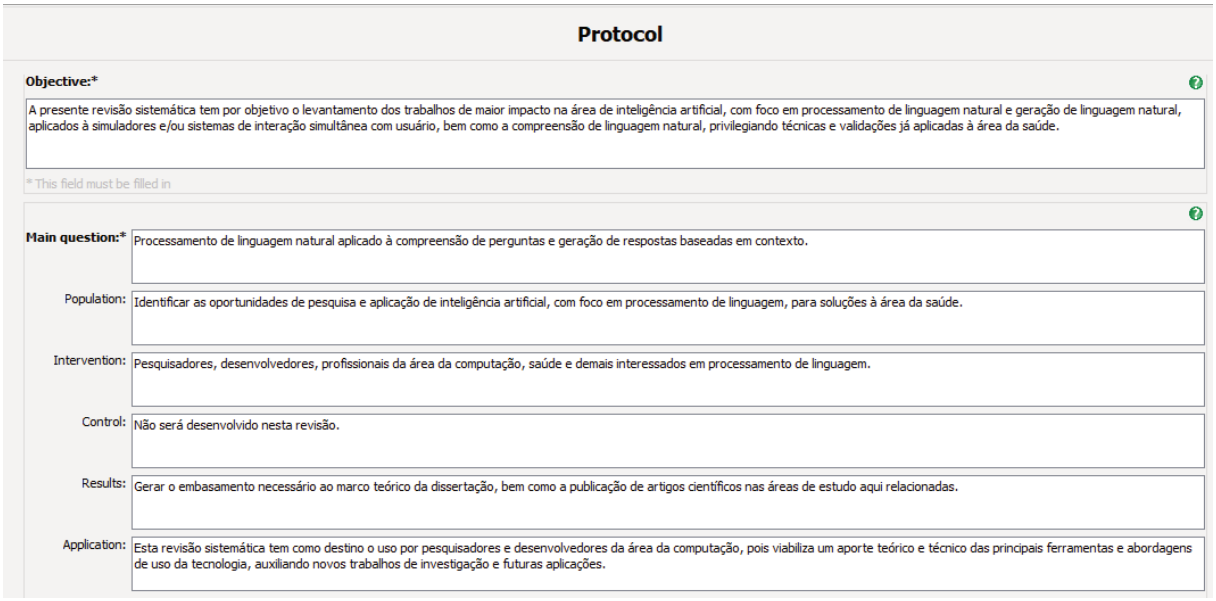
f) Análises adicionais (sensibilidade e precisão):

- Para esta revisão não foi considerada a análise de precisão e sensibilidade.

3.1.3 Desenvolvimento da revisão sistemática

Na sequência, iniciou-se a etapa de desenvolvimento da Revisão Sistemática, a qual compreende a busca dos artigos nos motores definidos no protocolo desta revisão, bem como o processo de extração dos resultados. A Figura 1 ilustra o formulário de cadastro do protocolo na ferramenta StArt.

Figura 1 Cadastro do protocolo na ferramenta StArt.



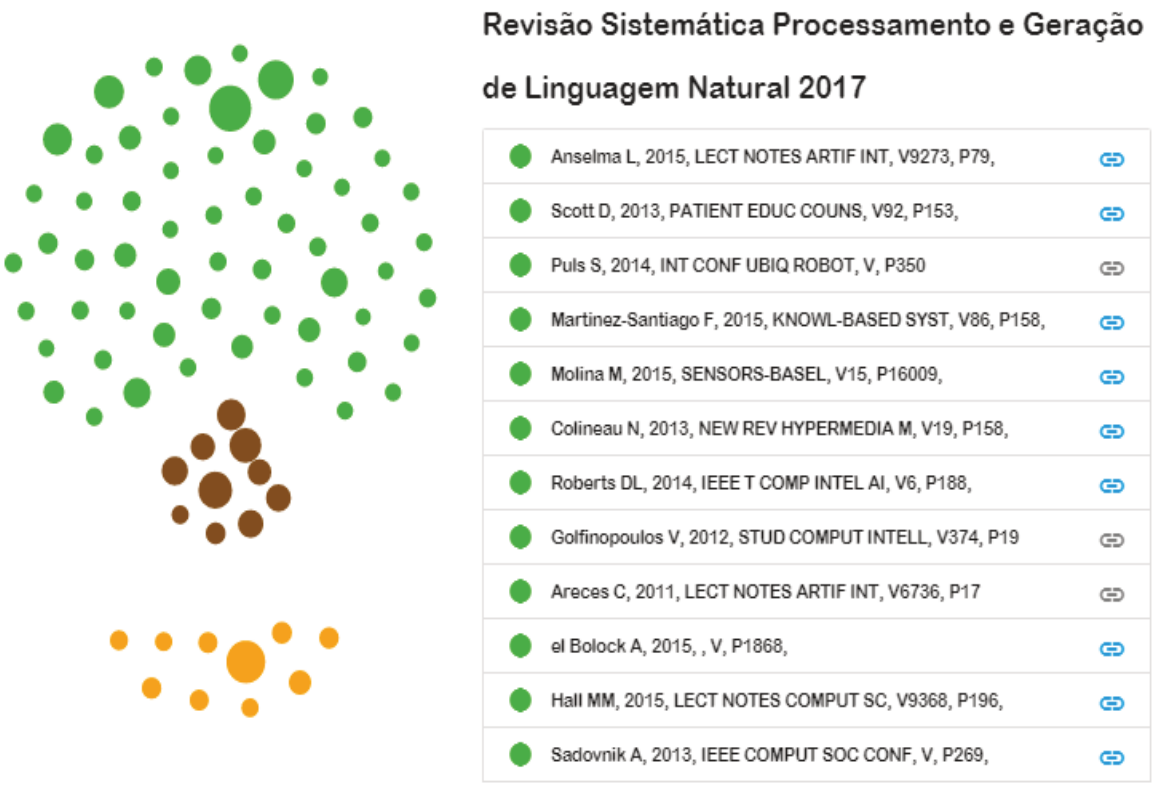
Fonte: Elaborado pela autora.

Como ferramenta de apoio a esta revisão, foi utilizado o software StArt para auxílio no processo de gerenciamento e catalogação, favorecendo uma análise ampla e criteriosa das bases selecionadas. Inicialmente a ferramenta solicita o cadastro do protocolo. Desta forma, é possível realizar a classificação dos artigos, aplicando os critérios definidos no protocolo no decorrer das etapas na própria ferramenta.

Os resultados obtidos na base *Web of Science* foram submetidos à ferramenta *Tree of Science* (Tos), desenvolvida no grupo de pesquisa GAIA, na Universidad Nacional de Colombia, Sede Manizales (GIRALDO; ZULUAGA; ESPINOSA, 2014). A ferramenta traz os resultados produzidos através de uma série de algoritmos, baseado na teoria de rede de grafos, calculando os graus de entrada e de saída para cada artigo, além de determinar a sua respectiva relevância. Para utilizá-la, as referências são extraídas em formato de texto como referência completa. O arquivo de referências gerado deve ser enviado ao ToS, realizando o *upload* do mesmo, para que seja processado. Os artigos serão separados em 3 camadas: *leaf*, *trunk* e *root* (folhas, tronco e raiz), como apresentado na Figura 2. As camadas fazem referência a seu ano de publicação e peso. Os artigos clássicos da área de investigação são

aqueles que têm maior grau de entrada na busca para a área de conhecimento (mais referenciados), portanto, são as raízes. Na sequência, a área denominada tronco, entrega uma lista de artigos que servem de suporte à investigação. Por fim, chega-se aos artigos atuais, aqueles de maior pontuação dentre as referências – publicações – mais recentes. Ao finalizar o processo da análise na ferramenta, ToS entrega, por padrão da ferramenta, aproximadamente 80 artigos, onde estes são aceitos na primeira fase de análise do protocolo.

Figura 2 Árvore de artigos selecionados pela ferramenta ToS.



Fonte: Elaborado pela autora.

3.1.4 Fases de seleção

Com a definição do protocolo finalizada e concluídas as configurações para utilização das ferramentas, inicia-se a fase de seleção dos artigos. A consulta nos motores de busca foi realizada no dia 17 de agosto de 2017, totalizando 717 artigos com foco em processamento e geração de linguagem natural, sendo posteriormente acrescentados 2 artigos pela autora, de assunto relevante para a pesquisa. A Tabela 1 demonstra os resultados para cada base consultada.

Tabela 1 Resultados obtidos nos motores de busca.

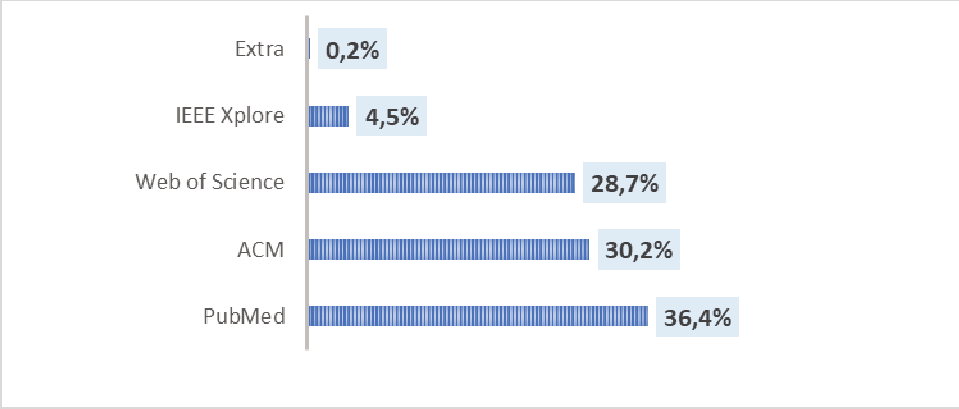
BASE	TOTAL
WEB OF SCIENCE	206
ACM	217
PUBMED	262

IEEE	32
Extras	2

Fonte: Elaborado pela autora.

A seguir, na Figura 3, pode ser observada a representatividade dos motores de busca através de um gráfico dos resultados obtidos. Na sequência, serão apresentados os resultados para cada fase de seleção.

Figura 3 Gráfico de resultados obtidos nos motores de busca.



Fonte: Elaborado pela autora.

3.1.5 Fase 1 – validação ToS

Na primeira fase de seleção, os resultados da base Web of Science foram extraídos em formato "*plain text*" e em seguida submetidos à ferramenta ToS para validação dos artigos; na ferramenta, o arquivo exportado da base deve ser enviado via *upload*. Como resultado, dentre os 206 artigos encontrados, a validação selecionou 80 artigos, os quais estão separados entre folha, tronco e raiz. Deste modo, 126 artigos foram rejeitados, aplicando um de nossos critérios de exclusão "Não validado pelo Software ToS".

Conforme a proposta da ferramenta, os artigos que se apresentam como base para a fundamentação teórica, referem-se àqueles denominados clássicos, no software ToS apresentando-se como *root*. As publicações avaliadas para esta categoria não foram acrescentados à esta revisão, pois apresentaram ano de publicação fora do período proposto nesta pesquisa; os artigos que representam suporte à pesquisa, denominados *trunk*, são demonstrados a seguir: (BANAEE; AHMED; LOUTFI, 2013); (GAROUFI; KOLLER, 2013); (MORENO-GARCIA et al., 2014); (DEEMTER et al., 2012); (PARABONI; VAN DEEMTER, 2013); (HUNTER et al., 2012); (DOS SANTOS SILVA; PARABONI, 2015) e (ANDROUTSOPOULOS; LAMPOURAS; GALANIS, 2013).

Os artigos classificados como folhas, ou seja, que representam as pesquisas atuais e tendências, são apresentados a seguir – somente aqueles com assunto relevante à processamento e geração de linguagem natural foram selecionados – desta forma, dos 80 artigos validados pela ferramenta, selecionou-se: (ARGUELLO et al., 2011), (LEMON, 2011), (PAIVA et al., 2011), (WALKER; LIN; SAWYER, [s.d.]), (RUBIOLO et al., 2012),

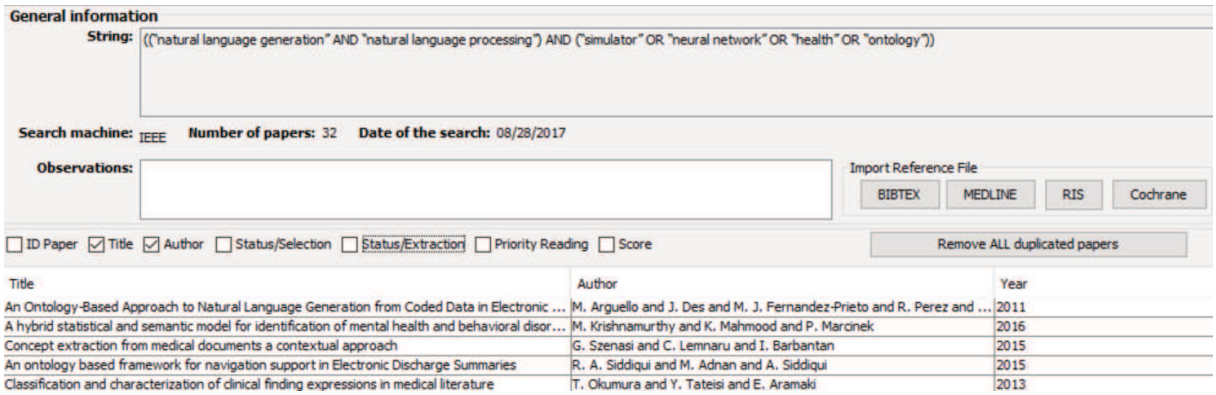
(LÓPEZ SALAZAR et al., 2012), (DANNÉLLS; GRŪZĪTIS, 2014), (MAIRESSE; YOUNG, 2014), (RAMOS-SOTO et al., 2015) e (GRUZITIS; DANNÉLLS, 2017).

Devido à incompatibilidade do Software ToS com as bases restantes, IEEExplorer, ACM Digital Library e PubMed, suas publicações iniciam na fase 2 desta seleção, pulando esta primeira etapa.

3.1.6 Fase 2 – critérios de inclusão e exclusão

A seguir, realiza-se o processo de seleção dos artigos restantes aplicando os critérios de inclusão e exclusão do protocolo. Como os motores de busca possibilitam a utilização de filtros avançados, o critério "Artigos publicados entre 2011 a 2017" pôde ser aplicado anteriormente, ao realizar a consulta, e desta forma não será validado novamente. Os resultados obtidos foram exportados em formato *BibTex* e inseridos na ferramenta StArt, conforme mostra a Figura 4.

Figura 4 Interface para inserir as referências na ferramenta StArt.

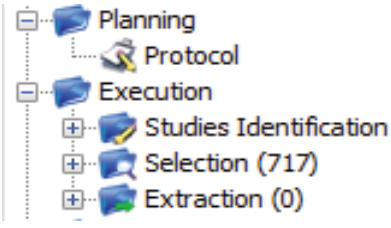


Fonte: Elaborado pela autora.

Devido a utilização de diferentes bases, eventualmente há publicações duplicadas, assim, foi realizada uma análise preliminar, para remoção de artigos duplicados. A seguir, aplicou-se os demais critérios de inclusão e exclusão. Este processo é facilitado ao trabalhar-se com a ferramenta StArt, que disponibiliza em sua metodologia um processo interno divido em etapas.

Na Figura 5 pode-se observar o processo de seleção interno que a ferramenta proporciona. Na seção chamada "*Execution*": a camada "*Studies Identification*" compreende o cadastro das bases; na camada "*Selection*" faz-se a primeira seleção, aplicando-se os critérios definidos e, por fim, a camada "*Extraction*", a seleção final dos artigos que serão lidos integralmente, após todos os critérios de inclusão/exclusão já aplicados.

Figura 5 Etapas internas da ferramenta StArt.



Fonte: Elaborado pela autora.

Ao aplicar os critérios de inclusão e exclusão, houve grande dificuldade em encontrar artigos que apresentassem validação, devido à essa característica, foi necessário realizar uma alteração no critério pré-definido "O artigo deve apresentar uma forma de validação". Desta forma, não foi considerada sua obrigatoriedade nos artigos selecionados, mas sim a evidência do desenvolvimento de protótipo ou implementação com experimentos, objetivando uma análise de técnicas e resultados.

A seleção e validação dos critérios foram realizadas nas demais bases – IEEExpore, ACM e PubMed – obtendo-se como resultado 30 publicações rejeitadas quando confrontadas com os critérios definidos e, 689 aceitas para a etapa seguinte, a fase 3.

3.1.7 Fase 3 – leitura de título, palavras-chave e resumo

Nesta fase, foram lidos o título, as palavras-chave e o *abstract*, buscando validar o foco de pesquisa presente nas publicações selecionadas. Após finalizar a fase 2, de análise dos critérios, percebeu-se um extenso volume de artigos aceitos, e que os mesmos apresentavam uma distanciação com o tema foco da pesquisa. Em vista da necessidade de uma extração adequada, e conforme a definição do protocolo, onde – os assuntos devem apresentar relevância e consistência com a proposta desta revisão – um novo critério de exclusão foi somado ao protocolo, para que os resultados apresentem forte relação com a proposta de pesquisa. Para tanto, o seguinte critério de exclusão foi acrescentado:

Os artigos devem apresentar o tema proposto como foco da revisão – processamento e geração de linguagem e/ou análise de técnicas com mesmo propósito; após a leitura de título, palavras-chave e *abstract*, e validação do critério recém acrescentado, obteve-se um resultado qualificado para a fase seguinte, onde 91 publicações foram aceitas e 598 rejeitadas após análise dos temas.

3.1.8 Fase 4 – leitura da introdução e conclusão

Nesta fase, realizou-se a leitura completa da introdução e conclusão dos artigos aceitos na última fase, visando encontrar forte relação com o tema de pesquisa, assim como clareza na descrição das técnicas aplicadas e qualidade na apresentação dos resultados. Após a leitura e análise dos artigos, foram aceitos para a fase final e leitura integral, 27 artigos.

3.1.9 Fase 5 – leitura integral dos artigos

Como última etapa de seleção, os artigos aceitos foram lidos integralmente, visando responder às perguntas definidas no protocolo desta revisão, a fim de elucidar questões conceituais e técnicas, bem como tendências no desenvolvimento de aplicações para geração de linguagem natural. Neste momento as fases de seleção dos artigos estão concluídas.

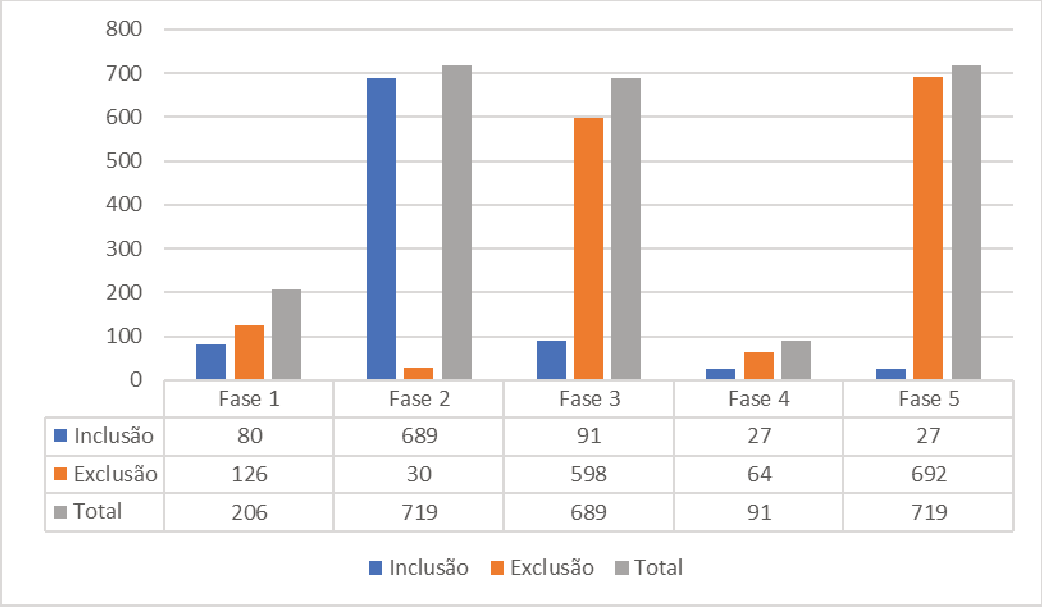
3.1.10 Resultados

Para a elaboração desta revisão, definiu-se um processo composto por 2 grandes etapas distintas, onde os artigos selecionados nos motores de busca foram confrontados com os critérios de inclusão e exclusão pré-definidos no protocolo, bem como, lidos título, palavras-chave, *abstract*, introdução e conclusão dos mesmos. Posteriormente, os artigos selecionados na validação inicial foram lidos integralmente, com o intuito de analisar e extrair destes as questões de interesse.

A leitura dos artigos foi finalizada, sendo possível a identificação das questões de interesse desta pesquisa. Para a Fase 1 foi realizada a validação das publicações da base *Web of Science* no software ToS, obtendo-se os artigos relevantes. A Fase 2 é aplicada às demais bases, exceto às publicações da base WOS, passadas diretamente à fase seguinte. Na Fase 3 são analisados título, palavras-chave e *abstract* das publicações. Para trabalhar com a ferramenta StArt, todas as referências da base WOS são inseridas e, posteriormente, é informado em seu processo interno, os artigos validados na ferramenta ToS como "Aceitos", através do critério de inclusão definido "Validado pelo software ToS".

A seguir, para facilitar a compreensão do processo, a Figura 6 ilustra um gráfico que evidencia um resumo dos resultados no decorrer das 5 etapas de seleção, com os respectivos dados de inclusão e exclusão a cada fase.

Figura 6 Gráfico resumo das fases de seleção.



Fonte: Elaborado pela autora.

Após finalizada a seleção dos artigos, com ambas as fases concluídas, obteve-se 27 artigos considerados relevantes. Os artigos aceitos e seus respectivos anos de publicação e título podem ser conferidos na Tabela 2.

Tabela 2 Artigos selecionados para esta revisão.

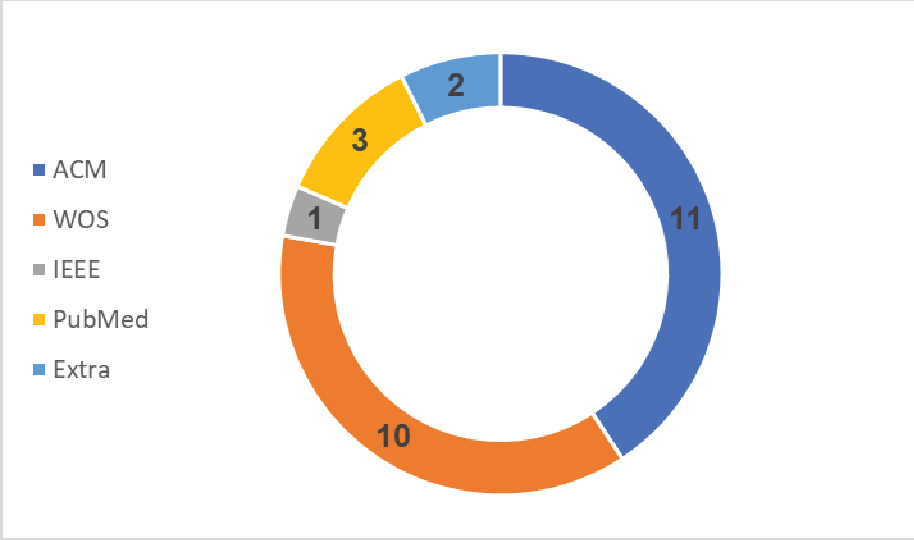
Ano	Titulo
2011	*Automatic Question Generation Using Discourse Cues*

Ano	Titulo
2011	*An Ontology-Based Approach to Natural Language Generation from Coded Data in Electronic Health Records*
2011	*The Bremen System for the GIVE-2.5 Challenge*
2011	*Combining Hierarchical Reinforcement Learning and Bayesian Networks for Natural Language Generation in Situated Dialogue*
2011	*Hierarchical Reinforcement Learning and Hidden Markov Models for Task-oriented Natural Language Generation*
2011	*Combining Symbolic and Corpus-based Approaches for the Generation of Successful Referring Expressions*
2011	*If It May Have Happened Before, It Happened, but Not Necessarily Before*
2011	*Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation*
2011	*Precision: a Guided-Based System for Semantic Validation and Personalized Natural Language Generation of Queries*
2011	*Generation of Formal and Informal Sentences*
2012	*A case-based reasoning model for multilingual language generation in dialogues*
2012	*Aligning Predicates Across Monolingual Comparable Texts Using Graph-based Clustering*
2012	*Comparing HMMs and Bayesian Networks for Surface Realisation*
2012	*An Annotated Corpus of Film Dialogue for Learning and Characterizing Character Style*
2012	*Knowledge discovery through ontology matching: An approach based on an Artificial Neural Network model*
2014	*A grammar-based semantic similarity algorithm for natural language sentences*
2014	*Extracting a bilingual semantic grammar from FrameNet-annotated corpora*
2014	*Stochastic Language Generation in Dialogue Using Factored Language Models*
2015	*Linguistic Descriptions for Automatic Generation of Textual Short-Term Weather Forecasts on Real Prediction Data*
2015	*Multilingual part-of-speech tagging with weightless neural networks*
2015	*A framework for ontology-based question answering with application to parasit immunology*
2016	*Skipping Word: A Character-Sequential Representation Based Framework for Question Answering*
2016	*Lexicalizing Linked Data towards a Human Friendly Web of Data*
2016	*Ask Me Anything: Dynamic Memory Networks for Natural Language Processing*
2017	*A multilingual FrameNet-based grammar and lexicon for controlled natural language*
2017	*An Effective Framework for Question Answering over Freebase via Reconstructing Natural Sequences*
2017	*Generative and Discriminative Text Classification with Recurrent Neural Networks*

Fonte: Elaborado pela autora.

Para uma análise mais apurada dos resultados, a seguir será apresentada uma série de gráficos ilustrativos, evidenciando contrastes, relações e características entre os dados extraídos. Na Figura 7, é ilustrado um gráfico quantitativo relacionando as publicações consideradas relevantes para esta pesquisa. Pode-se observar que as bases Web of Science (WOS) bem como ACM apresentaram material rico quanto ao tema proposto, sobrepondo-se às demais bases em número de publicações aceitas. Em contrapartida, as bases IEEEXplore (IEEE) e PubMed não apresentaram artigos de grande relevância neste estudo onde, respectivamente, a primeira resultou em apenas uma publicação aceita, e o segundo em três. Para futuras revisões, pode-se discutir a estratégia de utilizar outras bases em seu lugar, no caso de um mesmo tema.

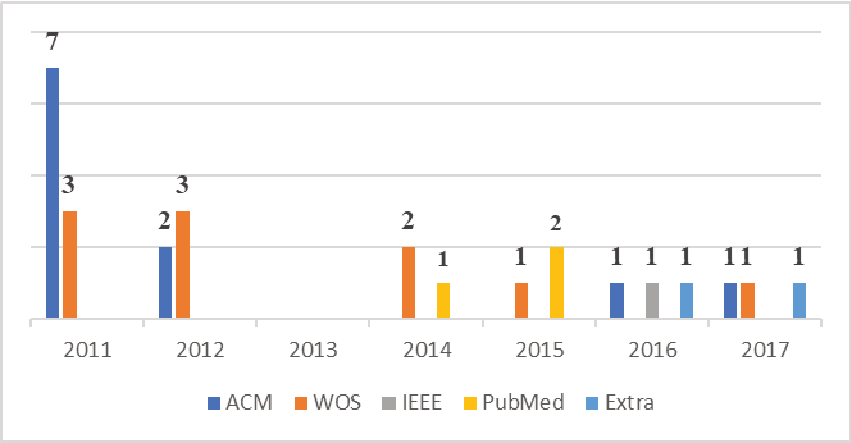
Figura 7 Gráfico quantitativo dos artigos selecionados.



Fonte: Elaborado pela autora.

Para o desenvolvimento desta revisão, definiu-se como período para seleção, os artigos publicados entre os anos 2011 a 2017; a Figura 7 demonstra um gráfico para comparativo, os artigos aceitos, suas bases e respectivos anos de publicação. Ao analisarmos a Figura 8, pode-se perceber a ausência de publicações relevantes no decorrer do ano de 2013, assim como deve-se recordar que para o ano de 2017, foram considerados os artigos publicados até a data início desta revisão, portanto, os artigos publicados após agosto de 2017 não foram selecionados e analisados nesta pesquisa. A seguir serão apresentadas as perguntas foco desta pesquisa.

Figura 8 Gráfico comparativo de publicações por ano.



Fonte: Elaborado pela autora.

Dentre as publicações aceitas para esta revisão e, vislumbrando futuramente realizar publicações na área de pesquisa aqui proposta, buscou-se identificar revistas relevantes para a área de estudo; na Tabela 3 pode-se verificar as revistas identificadas como potenciais objetivos para publicações futuras.

Tabela 3 Relação de revistas das publicações selecionadas.

Revistas
ACM International on Conference on Information and Knowledge Management
Annual Meeting of the Association for Computational Linguistics (ACL)
Computational Linguistics
Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)
European Workshop on Natural Language Generation (ENLG)
Expert Systems with Applications
IEEE International Conference on Consumer Electronics (ICCE)
IEEE International European Symposium on Computer Modeling and Simulation (SEM)
IEEE Transactions on Fuzzy Systems
Information Sciences
Innovative Use of NLP for Building Educational Applications (IUNLPBEA)
International Conference on Language Resources and Evaluation (LREC)
International Conference on Tools with Artificial Intelligence (ICTAI)
International Conference on World Wide Web Companion
International Speech Communication Association (ISCA)
Journal of biomedical semantics
Neural networks: the official journal of the International Neural Network
North American Chapter of the Association for Computational Linguistics
The Scientific World Journal

Fonte: Elaborado pela autora.

3.1.11 Perguntas foco da pesquisa

A presente revisão, através dos resultados obtidos, buscou evidenciar as seguintes perguntas definidas no protocolo, a fim de serem respondidas durante a leitura integral dos artigos. Em seguida, na Tabela 4, são apresentadas as perguntas foco de pesquisa com os resultados extraídos durante a revisão de literatura.

Tabela 4 Relação de resultados às perguntas foco da pesquisa.

Q01. A solução proposta pelos artigos é aplicada à área da saúde?	
Sim	3 artigos
Não	24 artigos

Q02. Quais foram os autores base na fundamentação teórica dos artigos?	
(ANGELI; LIANG; KLEIN, 2010)	1 artigo
(BARZILAY; LAPATA, 2008)	1 artigo
(BELZ; GATT, 2008)	1 artigo
(BILMES; KIRCHHOFF, 2003)	1 artigo
(CARIDAKIS et al., 2012)	1 artigo
(CAVAZZA; CHARLES, 2005)	1 artigo
(DETHLEFS; CUAYÁHUITL, 2011a)	1 artigo
(DETHLEFS; CUAYÁHUITL, 2012)	3 artigos
(DOAN et al., 2004)	1 artigo
(FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996)	1 artigo
(FILLMORE, 1985)	1 artigo
(FOLTZ; KINTSCH; LANDAUER, 1998)	1 artigo
(GRUZITIS; PAIKENS; BARZDINS, 2012)	1 artigo
(HENDLER; JAMES, 2001)	1 artigo
(JOHNSON et al., 2008)	1 artigo
(KAUFMANN; BERNSTEIN; FISCHER, 2007)	1 artigo
(KIM et al., 2015)	1 artigo
(KIMURA; KITAMURA, 2006)	1 artigo
(KLABUNDE; RALF, 2007)	1 artigo
(KOLLER; STONE, 2007)	1 artigo
(LOPEZ et al., 2007)	1 artigo
(MAIRESSE et al., 2010)	1 artigo
(MANNEM et al., 2010)	1 artigo
(MARQUES et al., 2007)	1 artigo
(NG; JORDAN, 2002)	1 artigo
(PAIVA; RAMOS-CABRER; GIL-SOLLA, 2010)	1 artigo

(PERERA; NAND, 2015); (PERERA, 2015)	3 artigos
(REITER; DALE, 2000)	1 artigo
(SEVERYN; MOSCHITTI, 2015)	1 artigo
(SHADBOLT; BERNERS-LEE; HALL, 2006)	1 artigo
(WALKER et al., 1997)	2 artigos
(WESTON et al., 2015)	1 artigo
(YONGJIAN FU, 1997)	1 artigo

Q03. Os artigos apresentaram resultados positivos?

Comparativo de resultados	1 artigo
Sim	26 artigos
Não	0

Q04. Os artigos possuem alguma forma de validação?

Sim	10 artigos
Não	17 artigos

Q05. Qual formato/público de validação foi aplicado?

Estudantes	3 artigos
Usuários	2 artigos
Questionário	2 artigos
Avaliadores	1 artigo
Especialistas	2 artigos
Não aplicado	17 artigos

Q06. Foi desenvolvido algum protótipo?

Sim	27 artigos
Não	0

Q07. Quais as ferramentas e recursos relevantes?

AskCuebee	1 artigo
Framework Ollie	1 artigo
Framework Paradise	3 artigos
Mallet (Machine Learning for Language Tooltik)	1 artigo
Mate Tools	1 artigo
Não consta	16 artigos
OntoNLQA	1 artigo
ProtoPropp Software	1 artigo
SimpleNLG	1 artigo
ASPIC+	1 artigo

Q08. Os arquivos apresentarão recursos léxicos e corpora utilizados?

FrameNet	4 artigos
LIWC (The Linguistic Inquiry Word Count)	1 artigo
Não apresenta	16 artigos
NPS Chat Corpus 1.0	1 artigo
SentiWordnet	1 artigo
Stanford Sentiment Treebank	1 artigo
Wordnet	3 artigos

Q09. Quais as linguagens utilizadas?

AIML	1 artigo
C#	1 artigo
JAVA	1 artigo
Não informado	19 artigos
OWL	2 artigos
Python	1 artigo
RDF	3 artigos
SPARQL	2 artigos

Q10, Quais as técnicas utilizadas?

Análise e/ou comparativo de técnicas	1 artigo
Geração de conteúdo utilizando ontologias	5 artigos
Geração de expressões utilizando análise de conectivos no discurso	1 artigo
Geração de expressões utilizando árvores de decisão	1 artigo
Geração de expressões utilizando *hierarchical reinforcement learning*	3 artigos
Geração de expressões utilizando *linked open data*	1 artigo
Geração de expressões utilizando lógica *fuzzy*	2 artigos
Geração de expressões utilizando modelos de entropia (MaxEnt)	1 artigo
Geração de expressões utilizando modelos fatoriais	1 artigo
Geração de expressões utilizando semântica de quadros	2 artigos
Geração de expressões utilizando similaridade em pares semânticos	2 artigos
Processamento de linguagem utilizando redes neurais artificiais	6 artigos

Q11. Quais foram os tipos de metodologias adotadas?

Análise de técnicas existentes	8 artigos
Extração de conteúdo para sistemas de agentes inteligentes	1 artigo
Geração de expressões a partir de documentos web	2 artigos
Geração de expressões em ambiente virtual	6 artigos
Geração de expressões para sistemas de consulta	2 artigos
Geração de expressões para sistemas de diálogo	3 artigos
Geração de expressões para sistemas de perguntas/respostas	1 artigo
Geração de expressões/relatórios a partir de dados brutos	2 artigos

Geração de relatórios climáticos	1 artigo
Geração de relatórios clínicos	1 artigo

Q12. Foi utilizado algum modelo estatístico auxiliar?

Bayesian Network	2 artigos
Hidden Markov Model	2 artigos
Markov Decision Process	1 artigo
Não consta	22 artigos

13. Os artigos apresentaram a origem dos dados utilizados?

Não informado	8 artigos
AGNews	1 artigo
BD relacional	1 artigo
DBPedia	3 artigos
Facebook bAbi dataset	1 artigo
Freebase	1 artigo
GeneBank	1 artigo
IMSDB Ontology	1 artigo
Ontologias	5 artigos
TrecQA	1 artigo
TriTrypDB	1 artigo
Wikipédia	1 artigo
WikiQA	1 artigo
Yahoo	1 artigo

Q14. Qual o foco de ligação entre o tema dos artigos e o tema proposto nesta revisão?

Análise de dados e processamento de conteúdo.	2 artigos
Compreensão semântica de sentenças.	1 artigo
Extração de conteúdo.	1 artigo
Geração de expressões e descrições curtas.	10 artigos
Geração de frases e sentenças.	11 artigos
Geração de relatórios/sumarização de conteúdo.	2 artigos

Fonte: Elaborado pela autora:

3.1.12 Análise dos artigos

Neste momento, concluiu-se a fase de levantamento dos dados, assim como a devida formulação das respostas às perguntas foco desta revisão. A seguir serão discutidos, em detalhes, os resultados das questões de pesquisa (RQ), passando item a item dos artigos selecionados.

a) Aplicação na área da saúde (RQ01)

Dentre os artigos selecionados para a fase final, apenas 3 apresentaram aplicação na área da saúde. Um número pequeno ao considerar-se o todo, no entanto, é importante ressaltar que ambos, mesmo que aplicados à questão de interesse – processamento e geração de linguagem natural – não exploram a proposta de geração de linguagem para diálogo, mas em contextos como a geração de relatórios e sumarização de conteúdo, respectivamente (ARGUELLO et al., 2011) e (ASIAEE et al., 2015); ou na geração de pequenas descrições para cenários virtuais (LÓPEZ SALAZAR et al., 2012). Em vista disto, o objetivo de identificar aplicações de geração de linguagem na área da saúde, não obteve muitos resultados; em contrapartida, o objetivo de identificar técnicas comumentemente utilizadas para geração de linguagem pôde-se verificar um resultado positivo e animador, pois uma variedade de técnicas e versatilidade de bases de dados foram verificadas durante a leitura e análise dos artigos.

b) Autores relevantes na pesquisa (RQ02)

Durante o processo de análise, identificou-se 32 autores como base para fundamentação teórica de processamento e geração de linguagem natural. Dentre estes, 3 autores destacam-se por relevância em seus estudos, onde os mesmos tiveram suas pesquisas apontadas como temas norteadores para trabalhos futuros, sendo citados em mais de um dos artigos selecionados, como o caso de "*PARADISE: A Framework for Evaluating Spoken Dialogue Agents*" (WALKER et al., 1997); igualmente relevante "*Comparing HMMs and Bayesian Networks for Surface Realisation Nina*"(DETHLEFS; CUAYÁHUITL, 2011b); e "*A Multi-strategy Approach for Lexicalizing Linked Open Data*" (PERERA; NAND, 2015).

É importante ressaltar que os autores aqui destacados como relevantes foram definidos de acordo com o número de citações dentre as diferentes publicações, não necessariamente com o foco descrito na proposta evidenciada ao final desta pesquisa.

c) Resultados positivos (RQ03)

A análise dos resultados não é aplicada correlacionando o objetivo desta revisão, mas considerando e relacionando os objetivos propostos nos artigos; se, ao final destes, através de seus experimentos, avaliações e validações aplicadas, foi possível apresentar resultados positivos. Em todos os artigos foram apresentadas análises positivas ou alcance de seus pontos de interesse e objetivo, tais como melhora na escalabilidade para processamento, velocidade, efetividade dos algoritmos propostos e evolução no grau de naturalidade nos diálogos gerados em relação aos algoritmos tradicionais.

d) Validação (RQ04)

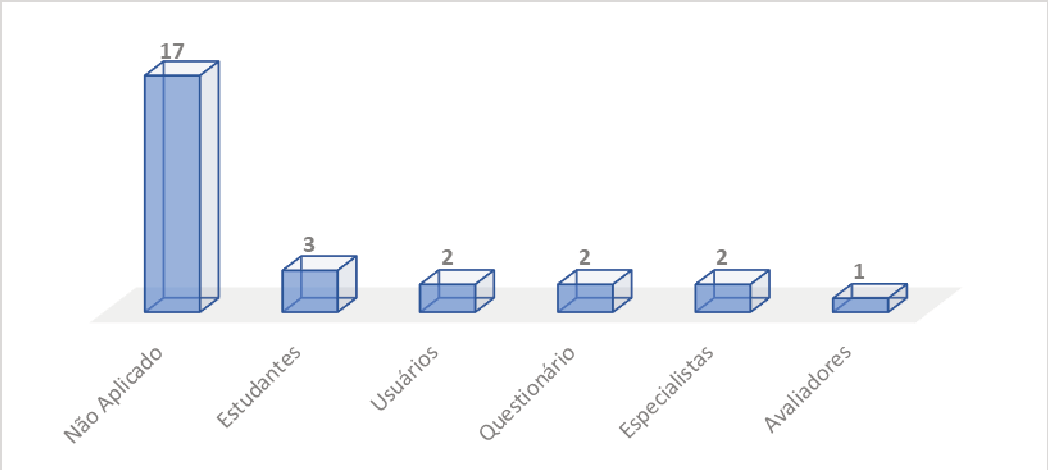
Conforme a alteração prevista durante o processo de seleção na fase 2, a não obrigatoriedade de os artigos apresentarem validação, desde que os mesmos apresentem uma implementação em forma de protótipo, foi aplicada. Com tal mudança, dentre os artigos selecionados, apenas 10 apresentaram alguma forma de validação, das quais os detalhes serão abordados no item a seguir.

e) Formas de validação (RQ05)

Como abordado anteriormente, ao analisar de forma geral, fica evidente a ausência de validação em grande parte dos artigos selecionados. Tal situação apresentou-se como uma dificuldade à esta pesquisa, pois esperava-se encontrar um modelo de validação que estivesse consolidado, e que pudesse ser aplicado a esta pesquisa. Para tanto, foi necessária a alteração nos critérios de inclusão e exclusão no protocolo desta revisão, para a sua não obrigatoriedade. Na Figura 9, o gráfico demonstra as formas de validação aplicadas. Pode-se perceber que apenas 10 artigos realizaram alguma forma de validação, dentre estes, as formas

variam, prevalecendo a validação realizada com estudantes; em segundo momento prevalecem as validações via questionário, com usuários e especialistas, por fim, apenas uma publicação apresentou validação através de um avaliador.

Figura 9 Gráfico das formas de validação apresentadas.



Fonte: Elaborado pela autora.

f) Desenvolvimento de protótipo (RQ06)

Todos os artigos apresentaram algum formato de experimento e/ou protótipo em sua proposta, sendo estes aplicados através de algoritmos de otimização para métodos tradicionais, novos métodos explorados e seus respectivos resultados, ou mesmo a validação de técnicas conjuntas, para obter-se resultados diferenciados. Portanto, juntamente às informações obtidas na análise dos protótipos desenvolvidos, a questão "RQ07 Ferramentas e recursos relevantes", ambos trazem um resultado positivo a esta pesquisa, ao também observar quais ferramentas podem ser integradas posteriormente ao modelo.

g) Ferramentas e recursos relevantes (RQ07)

Nesta questão, buscou-se identificar ferramentas e algoritmos utilizados na construção de experimentos ou protótipos que tenham sido desenvolvidos. As ferramentas listadas durante a revisão não representam uma aplicação por artigo, pois a revisão é de cunho exploratório. Com este objetivo, pôde-se encontrar mais de uma ferramenta aplicada ao mesmo problema, incluso material para referencial teórico e análise de qualidade dos resultados. Desta forma, as ferramentas apresentadas não irão coincidir com o número de artigos selecionados.

Dentre as publicações, foram identificadas ferramentas com maior utilização, ferramentas e recursos que poderiam ser úteis para o tema de pesquisa, e portanto, foram acrescidos à lista. O framework Paradise se destacou, sendo utilizado em diversos artigos, com aplicação no PLN.

Assim como a ferramenta *Protoprop Software* apresentou uma abordagem interessante, e de utilidade ao tema de pesquisa: um gerador de narrativas em língua inglesa. O software pode ser um guia para o presente estudo, concomitante à possibilidade de torná-lo um serviço disponível no modelo.

Em contrapartida, quatro artigos não informaram a utilização de ferramentas, apenas relatando resultados e comparações com algoritmos tradicionais. E outras ferramentas, algoritmos e bibliotecas foram verificadas durante a leitura e, posteriormente podem ser

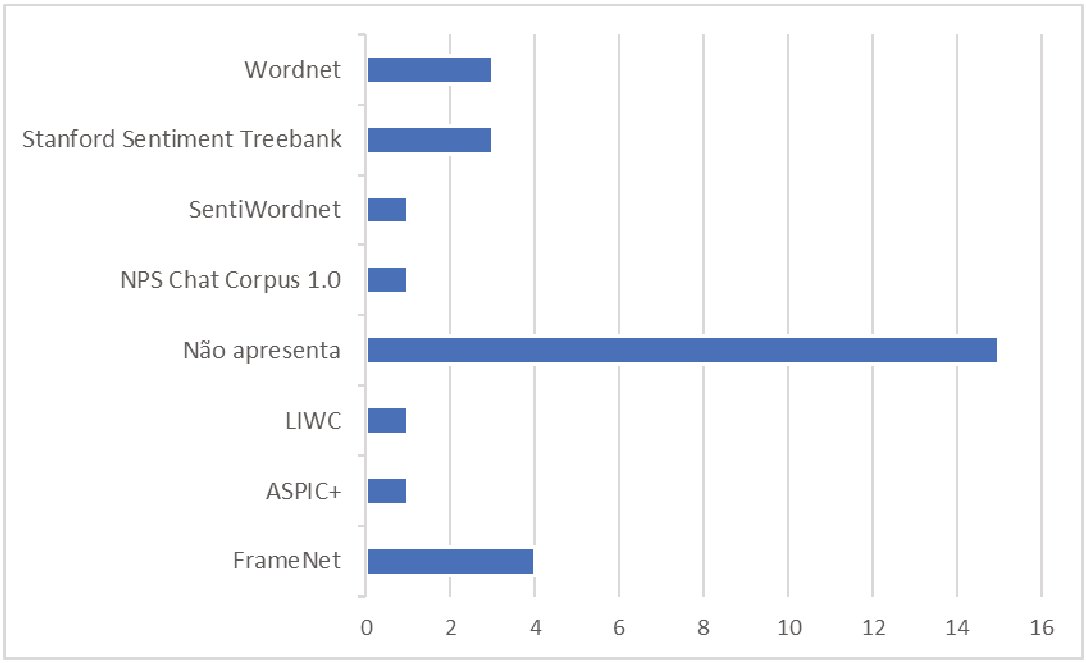
melhor exploradas, no entanto, para responder as perguntas, ateve-se às ferramentas e recursos que apresentavam possibilidade de aplicabilidade ao tema de pesquisa. As ferramentas não citadas foram utilizadas isoladamente em artigos e justificadas em suas metodologias, não sendo aplicadas em outras situações, portanto, seguem listadas para conferência da autora, mas irrelevantes à esta pesquisa.

h) Recursos léxicos e corpus utilizados (RQ08)

Ao realizar as primeiras leituras, pôde-se identificar os recursos léxicos e corpus adotados, assim como indicados por alguns autores, tanto para a língua inglesa quanto outras. Mesmo que não seja possível utilizar tais recursos em aplicações de língua portuguesa, com base nas metodologias adotadas, foi considerado relevante confrontar tais dados, a fim de estudar alternativas disponíveis para a língua portuguesa e, possivelmente, aplicá-las à futuros experimentos.

Dentre os recursos utilizados, destaca-se o *frameNet*, apresentado como recurso em 4 artigos. Apresenta implementações e disponibilidade em inglês e demais línguas. É um banco de dados lexical que implementa a teoria da semântica de quadros, o mesmo possui uma implementação para português. Em segundo lugar, destacado em 3 publicações, está o *WordNet* que, da mesma forma, trata-se de um banco de dados lexical, diferentemente do *FrameNet*, o *WordNet* traz um banco de sinônimos, e relações que estes têm com outros sinônimos (*synsets).* Pode ser usado para obter descrições curtas sobre determinado grupo de palavras e exemplos de utilização, ideal para desambiguação de sentido. A ferramenta pode ser utilizada em português, através de uma *Application Programming Interface* (API). Também interessante, a ferramenta *Stanford Sentiment Treebank* apresenta um recurso para análise de sentimentos em *feedbacks* de filmes. Traz uma proposta diferenciada, pois em contraste às aplicações comuns, onde analisa-se em separado cada palavra, pontuando-a e, posteriormente, define-se sua polaridade. O *Sentiment Treebank* trabalha com um conjunto de palavras, com intuito de não ser facilmente enganado por falsos negativos ou falsos positivos. Outros recursos também foram identificados, no entanto, foram utilizados isoladamente nos artigos, e não detalhados em separado. A Figura 10 traz uma relação completa.

Figura 10 Relação de recursos léxicos e corpus.

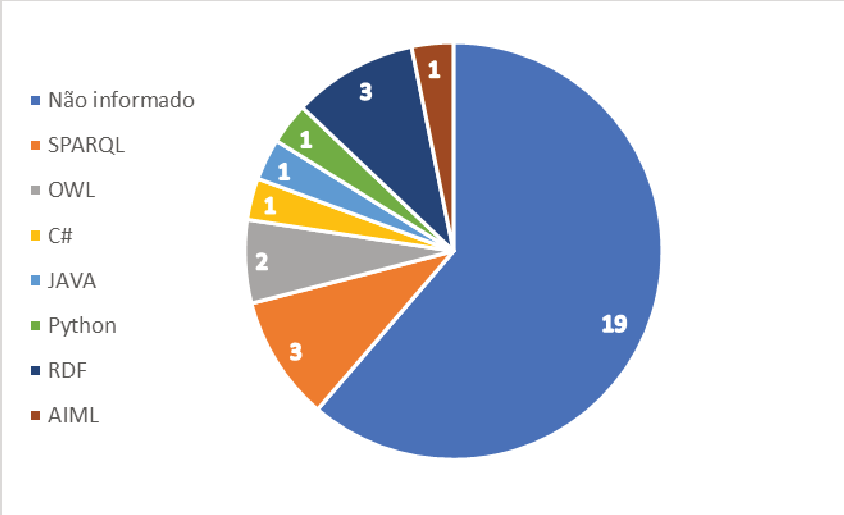


Fonte: Elaborado pela autora.

i) Linguagens utilizadas (RQ09)

De acordo com a definição dos critérios de inclusão e exclusão do protocolo, a evidência de protótipo ou implementação com experimentos é obrigatória para os artigos selecionados. Com isso, buscou-se encontrar as linguagens de programação utilizadas. Conforme a Figura 11, o gráfico comparativo demonstra que grande parte das publicações não costumam informar detalhes técnicos quanto a linguagem de implementação, geralmente privilegiando uma análise aprofundada de experimentos comparativos com aplicações tradicionais, posteriormente relatando os resultados promissores, assim como técnicas aplicadas que obtiveram êxito. Essa relação evidencia que a linguagem torna-se uma decisão de segundo plano, comparado à correta seleção de ferramentas e uma base de dados sólida para apoio, bem como modelos e técnicas existentes.

Figura 11 Gráfico comparativo de linguagens mais utilizadas.

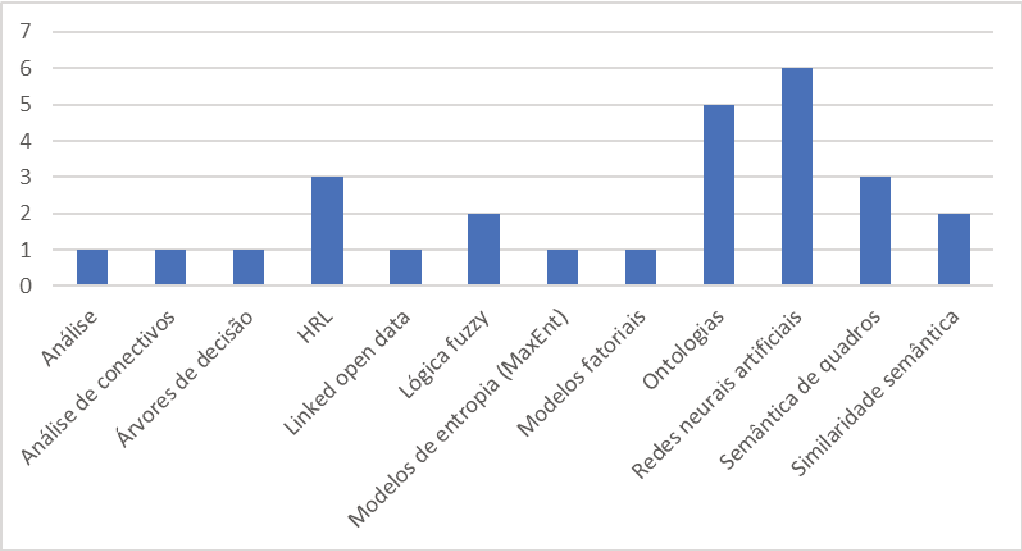


Fonte: Elaborado pela autora.

j) Técnicas utilizadas (RQ10)

Com o intuito de validar técnicas a serem utilizadas posteriormente, foram analisadas diferentes abordagens para geração de linguagem, em alguns casos, as mesmas apresentavam implementações mistas. Um exemplo pode ser visto na Figura 12, técnicas aplicando redes neurais sobressaem-se quando comparados as demais, no entanto, deve-se atentar que em muitos casos foi utilizada uma técnica híbrida, como redes neurais aplicadas em ontologias e em modelos de markov, assim como combinadas à técnica de *Hierachical Reinforcement Learning* (HRL). Assim sendo, esta pergunta busca evidenciar as diferentes técnicas, definindo-se uma extração baseada em características que fogem ao comumentemente aplicado.

Figura 12 Gráfico comparativo de técnicas utilizadas

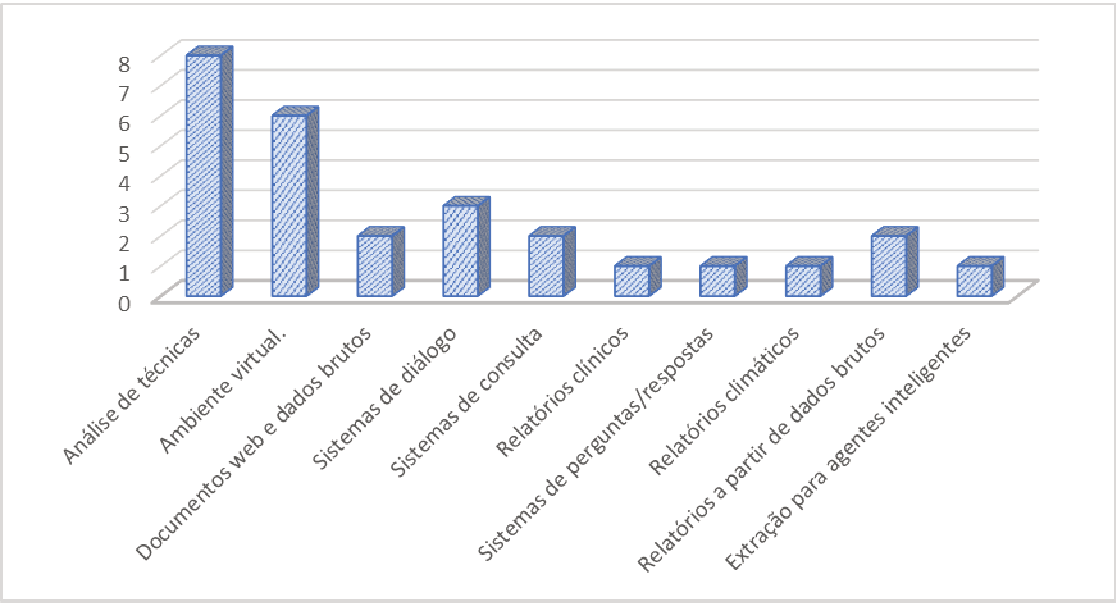


Fonte: Elaborado pela autora.

k) Metodologias adotadas (RQ11)

Com o intuito de identificar as distintas áreas de aplicação para a geração de linguagem natural, buscou-se analisar a existência de tendências quanto às metodologias adotadas. Na Figura 13, o gráfico ilustra uma forte preferência nas publicações em propor melhorias de técnicas existentes, e evidenciar sua evolução através de comparativos com a aplicação tradicional. Essa metodologia apresenta-se fortemente relacionada a questão de interesse, pois possibilita a seleção de métodos com experimentos onde obteve-se êxito nos resultados.

Figura 13 Gráfico comparativo das metodologias adotadas.



Fonte: Elaborado pela autora.

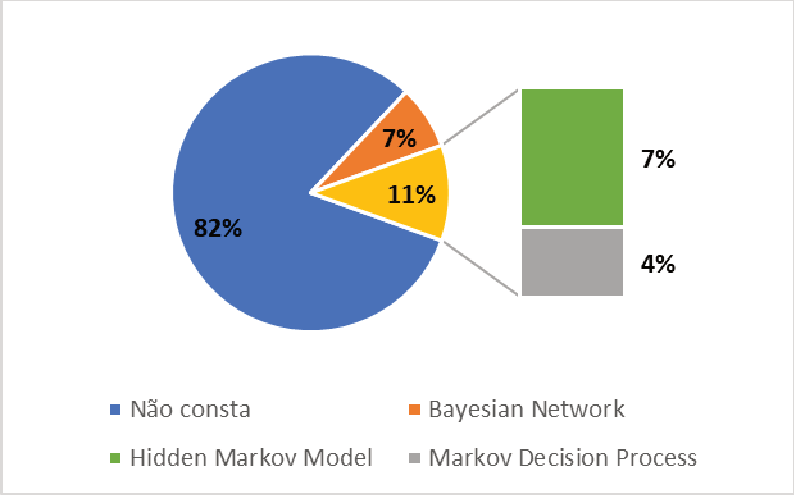
Da mesma forma, é possível verificar uma das questões de interesse para a realização desta revisão, como demonstrado na Figura 13, há uma inclinação de pesquisas já realizadas

para aplicação em ambientes virtuais, estes com foco em descrições curtas e orientações em jogos digitais. Dentre as metodologias identificadas, ambientes virtuais aproximam-se do foco de pesquisa, desta forma, buscou-se evidenciar nestas publicações, quais eram voltadas para os profissionais ou estudantes da área da saúde, onde foi confirmado apenas um trabalho (LÓPEZ SALAZAR et al., 2012).

l) Adoção de modelo estatístico (RQ12)

No decorrer desta revisão, observou-se que a aplicação de métodos estatísticos recebe grande aplicabilidade em geração de linguagem natural, sendo utilizados como auxiliares às técnicas adotadas. Um exemplo de aplicação que utiliza um modelo probabilístico como apoio pode ser verificado em (DETHLEFS; CUAYÁHUITL, 2011b), onde *Bayesian Networks* são utilizadas para flexibilização de expressões de superfície. Aplicações parecidas ocorrem com os demais modelos, servindo de suporte às técnicas adotadas. Na Figura 14, um gráfico ilustra a presença destes modelos de forma auxiliar.

Figura 14 Gráfico comparativo de modelos probabilísticos adotados.

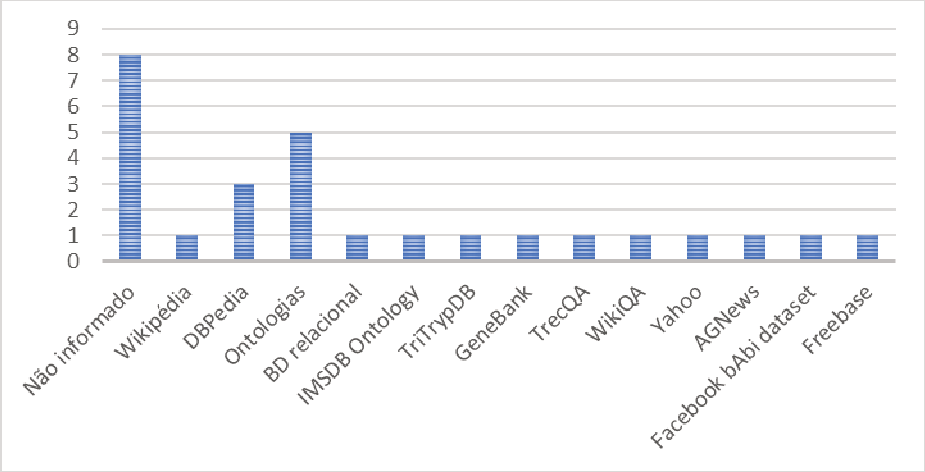


Fonte: Elaborado pela autora.

m) Origem dos dados utilizados (RQ13)

Durante o processo de análise, identificou-se uma preferência por utilização de bases disponíveis na web, tanto por fornecer uma alta disponibilidade para uso em sistemas interligados, como pela possibilidade de reaproveitamento e volume de dados para treinamento e testes. Foram identificados 16 diferentes tipos de bases, que podem ser verificadas na Figura 15. Pode-se notar uma tendência na utilização de ontologias próprias, necessitando primordialmente a geração manual; assim como uma preferência na adoção de uma base de dados web, DBPedia, como fonte de dados, como pode ser verificado nos artigos de (PERERA; NAND, 2016; YUE et al., 2017; YOGATAMA et al., 2017).

Figura 15 Comparativo de bases de dados utilizadas.

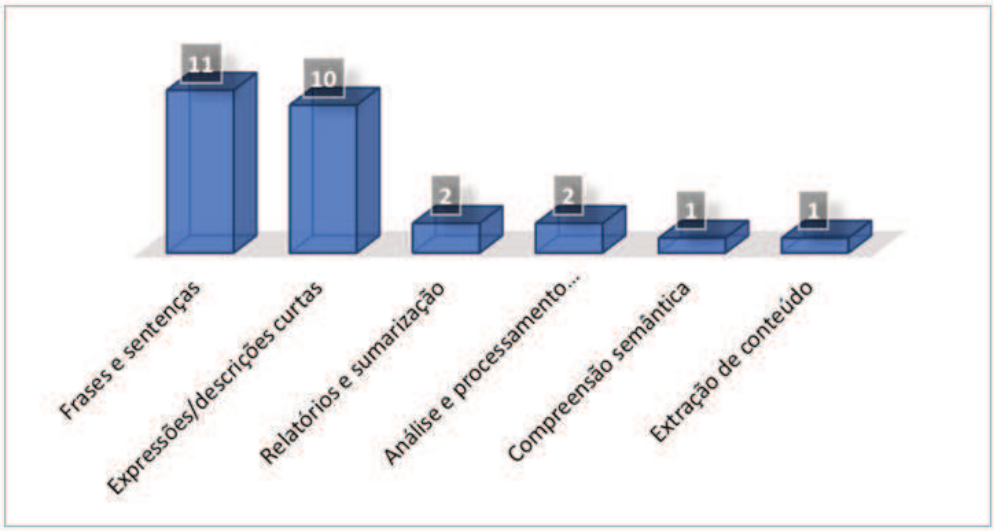


Fonte: Elaborado pela autora

n) Foco de ligação entre o tema proposto (RQ14)

Para esta revisão, a questão de interesse tratou de extrair os cenários mais frequentes na geração de linguagem natural e as técnicas comumentemente utilizadas, objetivando identificar suas eventuais aplicações e resultados positivos. Foram identificados 6 temas, dos quais observou-se um volume de trabalhos voltados à "geração de frases completas e sentenças", como ilustra a Figura 16. Esta abordagem emprega uma abordagem mais complexa, visto a preocupação envolvendo a contextualização das respostas.

Figura 16 Gráfico comparativo resumo da relação com o foco da pesquisa.



Fonte: Elaborado pela autora.

3.1.13 Considerações

A revisão de literatura apresentada (Seção 3.1) foi realizada com o objetivo de compreender a abrangente área de geração de linguagem natural, a fim de identificar oportunidades de pesquisa a serem exploradas. Esta metodologia foi adotada para identificar lacunas de pesquisa, levantar temas a serem explorados futuramente, bem como evidenciar

algoritmos, ferramentas e recursos que apresentassem relevância e contribuições a pesquisas subsequentes na área de GLN.

No decorrer das fases de seleção, pôde-se perceber uma tendência na linha dos trabalhos, destinados primordialmente à otimização de algoritmos já consagrados, focados muitas vezes em análises de resultados melhorados, assim como comparações com novas abordagens destes algoritmos. Portanto, identificou-se como uma prática recorrente e focou-se principalmente em artigos com uma abordagem de aplicação, com evidência de protótipo e/ou validação dos resultados.

Durante o desenvolvimento, pode-se identificar uma variedade de trabalhos com o propósito para "geração de expressões e/ou descrições curtas", abordagem que, em alguns artigos, demonstrou servir de apoio à propostas de *games*, onde os mesmos seriam como o "guia" do jogador, informando-lhe as direções que deveria tomar de acordo com o cenário (DETHLEFS; CUAYÁHUITL, 2011b; DETHLEFS; NINA, 2011).

Outras aplicações relacionaram GLN como ferramenta para interação com bases estruturadas e de grande volume. A utilização de linguagem natural aliada ao uso de ontologias para apresentar vocabulário expressivo e especializado para consulta, como suporte destinado a atender peritos em determinados temas. Traz vantagens quando requer uma busca expressiva e aprofundada que necessita não apenas uma simples busca por palavras-chave, mas um conjunto de termos que resulte em uma consulta intuitiva e recuperação de respostas mais precisa e assertiva (ASIAEE et al., 2015).

As publicações que apresentaram "sumarização e geração de relatórios", bem como "extração de conteúdo" foram consideradas relevantes, pois são uma técnica comumentemente empregada em sistemas GLN em apoio a situações de análise e tratamento de dados não estruturado. Caracteriza-se pelo uso de dados brutos ou massivamente gerados, como dados de sensores em ambientes hospitalares e unidades intensivas, dados de meteorologia ou mesmo jogos esportivos.

Estes cenários apresentam extenso material de análise, no entanto, requer qualidade apurada, pois geralmente o período para análise e processamento é curto, e espera-se um parecer resumido da situação que considere o contexto. Ao relacionar a geração de linguagem à cenários do tipo D2T (dados para texto), pode-se identificar uma oportunidade de pesquisa a ser explorada neste campo. Dados médicos ainda apresentam um alto grau de dificuldade visto sua realidade em volume, características não estruturadas e dificuldade em correlacionar contexto aos relatórios médicos.

O campo de GLN tornou-se uma área de pesquisa com crescimento exponencial, inicialmente nasceu como um subcampo no processamento de linguagem natural, por fim transcendendo à uma área de pesquisa independente (PERERA; NAND, 2017). A geração de linguagem assume hoje uma extensa gama de pesquisas destinadas a suprir o mercado carente de ferramentas efetivas sem requerer investimento de um extenuante trabalho manual dos interessados.

O exponente crescimento oportuniza uma riqueza de ferramentas, recursos e resultados otimistas à disposição. Estes recursos, já avidamente validados, podem ser vistos como um amplo catálogo a ser aplicado em diferentes áreas, para solucionar, auxiliar, atender e remediar as mais diversas carências reconhecidas e mitigadas através do uso de linguagem natural.

Com isto em mente, e dispondo desta vasta lista de artefatos constante e copiosamente guarnecida de novos, é possível identificar uma oportunidade de pesquisa. Um cenário que se

possa prezar pela facilidade de acesso aos algoritmos e técnicas presentes na comunidade acadêmica. Portanto, observa-se uma oportunidade em desenvolver novos experimentos que usufruam deste catálogo de uma forma abstraída, por meio de uma interface. E nesta interface possa ser feita a interação com os componentes de GLN e PLN sem a necessidade de aprofundar-se na área fora do produto de aplicação.

Para analisar esta oportunidade de pesquisa, foi necessária uma nova busca, focada em identificar trabalhos relacionados ao reaproveitamento, reuso e disponibilidade de algoritmos, técnicas e demais ferramentas de GLN já estabelecidos. A seguir serão apresentados alguns trabalhos identificados com a oportunidade de pesquisa, relacionadas à "*frameworks*", "reaproveitamento" ou "reuso" de tarefas e recursos, sempre alinhadas à área de interesse GLN e PLN.

3.2 Trabalhos relacionados com a oportunidade de pesquisa

A partir da revisão de literatura realizada, descrita na Seção 3.1, a análise dos artigos selecionados proporcionou a identificação de uma lacuna quanto à integração de recursos na área de geração de linguagem natural. A partir desta constatação foram realizados novos estudos com a finalidade de identificar trabalhos com exemplos e estratégias para a integração de recursos. Em especial foram estudados trabalhos que tivessem como tema geral a integração mais ampla de recursos, de forma a diferenciar esta iniciativa de outras já observadas em alguns trabalhos selecionados na revisão, que apresentavam frameworks especializados, porém que atendiam efetivamente a um domínio, tais como o trabalho de título "*Skipping Word: A Character-Sequential Representation based Framework for Question Answering*" (MENG et al., 2016) e o trabalho "*An Effective Framework for Question Answering over Freebase via Reconstructing Natural Sequences*" (YUE et al., 2017). Portanto, buscou-se trabalhos que apresentassem integração ampla visando o reuso e a integração.

A partir dos resultados já previamente levantados na revisão de literatura, foram identificados artigos adicionais que seguem uma linha semelhante ao tema de pesquisa acima comentado. Os artigos foram lidos e, a seguir, apresenta-se um comparativo de suas contribuições e dificuldades, bem como as principais características consideradas relevantes para implementação das ferramentas.

3.2.1 Comparativo entre os artigos relacionados

Na sequência serão avaliados trabalhos que relacionam-se com a pesquisa realizada. Buscou-se evidenciar aspectos comuns e tecnologias adotadas para a solução dos cenários apresentados. Neste sentido, faz-se uma comparação de suas contribuições, dificuldades enfrentadas e detalhes que puderam ser adotados como base no desenvolvimento do modelo proposto.

O trabalho de (WALKOWIAK, 2018) possui como título "*Language Processing Modelling Notation – Orchestration of NLP Microservices*". Ele define uma linguagem formal para orquestração de ferramentas, denominada *Language Processing Modelling Notation* (LPMN). O objetivo no desenvolvimento desta linguagem visa a solução de um cenário de difícil integração de ferramentas de mineração de texto. Seu cenário conta com

diversas ferramentas que já foram implementadas, em linguagens de programação diferentes, e estão configuradas em ambientes próprios, mas que precisam trabalhar para um único experimento, resultando em um difícil cenário de busca de compatibilidade.

Nesta situação, foi construída a linguagem LPMN, definida com base na notação formal para descrever fluxos complexos para micros-serviços, gramática estabelecida para ANTLR [8] , com elementos simples. Não baseia-se em XML e pode ser facilmente compreendida por humanos. Esta linguagem resolve o problema de fazer as ferramentas cooperarem entre si, sem a necessidade de adaptar todas as ferramentas a uma linguagem comum. Define-se o *workflow* de ferramentas LPMN, bem como as configurações para cada uma destas.

Por fim, para que este *workflow* seja executado e o experimento realizado, foi implementado o *LPMN engine,* que carrega as configurações estabelecidas na linguagem LPMN. A arquitetura definida para o *LPMN engine* complementa a solução para a difícil integração entre ferramentas com a implementação de um *message-oriented middleware*, utilizando o protocolo *AMQP e RabbitMQ broker.* Com esta definição, o *LPMN engine* possui uma estrutura para trocas de mensagens, tornando cada ferramenta independente, pois comunicam-se apenas por envio e recebimento entre os canais estabelecidos.

O trabalho desenvolvido por (SINGH et al., 2018), denominado Frankenstein, propõe um Framework para atender a tarefas de perguntas e respostas. O sistema integra componentes especializados em tarefas específicas necessárias para o *pipeline* de tarefas de sistemas de Perguntas e Resposta, a fim de construir *pipelines* otimizados. Sua motivação surgiu da variedade de *softwares* que implementam diferentes estratégias para cada tarefa. O maior desafio evidenciado na pesquisa foi como conseguir combinar e selecionar os melhores componentes para uma determinada atividade que considere o tipo de entrada (a questão) e o tipo de saída (a resposta objetivo), nesta seleção.

O problema proposto, portanto, refere-se à otimização de *pipelines QA*. Para a pesquisa foi desenvolvido o *Frankenstein Framework ,* que se utiliza de um algoritmo de classificação para montar *pipelines* otimizados. O algoritmo de classificação implementa modelos de aprendizagem para gerar novas análises automaticamente. As características são extraídas das diferentes questões de entrada, e tem o objetivo de selecionar e nutrir o algoritmo classificador, para este selecionar os QA componentes que melhor se adequem ao cenário. No *Frankenstein Framework* , para novos formatos de questões, um modelo precisa ser treinado para classificar adequadamente os componentes com melhor performance para compor o *pipeline.* Com esta abordagem, o processo de construir os *pipelines* é totalmente automatizado baseando-se nas características determinadas para entrada e objetivo de saída. O modelo completo tem uma arquitetura modular, composta por 7 componentes dedicados a resolver tarefas QA: *Feature Extraction, QA Componente, QA Component Classifiers, QA Pipeline Optimiser, COmponent Selector, Pipeline Generator, Pipeline Executor.* Cada componente resolve uma única tarefa. O *Frankenstein Framework* conta com 29 componentes implementados para atender cenários de perguntas e respostas, dos quais estes atendem 5 tarefas QA especializadas: *Named Entity Recognition (NER), Named Entity Disambiguation (NED), Relation Linking (RL), Class Linking (CL) e Query Building (QB).* Para cada tarefa, existem diferentes componentes que são avaliados pelo algoritmo classificador, e podem ser selecionados para o *pipeline*.

[8] https://www.antlr2.org/

O trabalho desenvolvido por (KIM DBCLS et al., 2017) denominado OKBQA Framework tem o objetivo de facilitar o desenvolvimento colaborativo de sistemas QA através de contribuições distribuídas. O Framework propõe identificar e definir módulos para atender a tarefas de *natural language* (NL) *question-answering* (QA), bem como desenvolver e manter um serviço público. Para tornar isto possível, foi desenvolvido o OKBQA Framework, onde podem ser construídos *workflows* de recursos QA e estes posteriormente executados. As definições de cada recurso são disponibilizadas em uma interface web[9], com configurações de entrada e saída definidas. O projeto, desenvolvido para intermediar o uso dos recursos desenvolvidos em diferentes grupos, possui uma arquitetura[10] modular composta por: *Template Generation Module (TGM), Disambiguation Module IDM), Query Generation Module (QGM), Answer Generation Module (AGM)*. Por fim, para gerenciar a execução dos *workflows* e fazer a conexão entre os módulos, existe o *control module (CM).*

Como regra, para manter um padrão no desenvolvimento de novos recursos para o Framework, é necessário que estes sejam disponibilizados via serviço REST API, e obrigatoriamente tenha entradas e saídas em formato JSON. Essa abordagem permite que um módulo possa ser implementado em qualquer linguagem de programação. Dessa forma, quaisquer *workflows* definidos são compostos por uma sequência de serviços REST API, de entrada e saída em formato JSON. A motivação para o desenvolvimento desta pesquisa trabalho propunha-se a atender a necessidade da comunidade OKBQA (Open Knowledge Base and Question-Answering), que tem o objetivo de avançar no desenvolvimento de sistemas de bases de conhecimento abertas e de perguntas a respostas. Portanto, para promover esse intercâmbio de recursos entre a comunidade distribuída, assim como organizar as diferentes soluções que têm sido desenvolvidas nos diferentes grupos, propôs-se o framework OKBQA.

A partir da análise detalhada destes trabalhos, foram realizados estudos comparativos e uma análise crítica de seu contexto, abordagens e vantagens. O estudo e análise destes trabalhos auxiliou na definição e construção do modelo proposto, Pythia NLG. A Tabela 5 apresenta um comparativo das diferentes características dos trabalhos, bem como aspectos importantes que serviram para dar sequência ao desenvolvimento e posterior implementação do protótipo.

Para cada trabalho analisado na Tabela 5 foram sumarizadas informações sobre os seguintes itens:

a) objetivo do trabalho – buscando identificar claramente o foco e direcionamento do trabalho;

b) algoritmos empregados na integração – analisa o tipo de algoritmo de GLN disponibilizado na solução;

c) aspectos de reutilização existentes – indica funcionalidades para reuso dos componentes instalados;

d) incorporação de novos componentes – analisa possibilidades e facilidades para inclusão de componentes externos adicionais;

e) flexibilidade quanto à diferentes fontes de dados – avalia quais as facilidades para indicação de fontes de dados diversas.

[9] http://repository.okbqa.org/

[10] http://doc.okbqa.org/overview/v1/

Tabela 5 Comparativo entre trabalhos relacionados e aspectos do desenvolvimento.

Questão	**LPMN** (WALKOWIAK, 2018)	**FRANENSTEIN** (SINGH et al., 2018)	**OKBQA** (KIM DBCLS et al., 2017)
Objetivo	Definição de uma linguagem formal para orquestrações de tarefas de mineração de texto.	Construção de *Pipelines* otimizados para QA System.	Repositório para desenvolvimento colaborativo de tarefas QA System.
Algoritmos empregados	*ANTLR Parser, NLP* e *Machine Learning.*	*Greedy algorithm (GA),* Tarefas *Q&A.*	Algoritmos da comunidade OKBQA.
Reutilização	Combinações variadas entre tarefas.	Combinações variadas dentre as 5 categorias de tarefas QA[11]	Combinações variadas respeitando as entradas de cada tarefa.
Incorporação de novos componentes	Permite por meio da arquitetura de mensageria adotada.	Permite, porém, apresenta restrições de performance.	Permite, por meio de cadastro na interface web, definindo *input* e *output* JSON.
Diferentes fontes de dados.	Documentos variados (doc, docx, rtf, pdf).	*Dataset* para treinamento e sentenças (questões de entrada).	Sentenças (questões de entrada).
Área de aplicação.	*Text mining.*	*QA System.*	*QA System.*

Fonte: Elaborado pela autora.

Um aspecto interessante do trabalho LPMN[12] foi a decisão de utilizar um *message broker* para intermediar a comunicação dos componentes. Sua proposta apresenta o uso do *RabbitMQ broker* para troca de mensagens em sua rede local, pois possui diferentes soluções implementadas em máquinas virtuais, já configuradas e funcionando. Desta forma, sua solução utiliza-as por meio de chamadas RPC. Outro aspecto a ser ressaltado foi sua opção por utilizar um *parser* para ajudar na formalização da linguagem de orquestração.

As características apresentadas pela arquitetura do projeto *OKBQA Framework* [13] propõe ideias significativas. O projeto apresenta o objetivo de um executor de orquestração para REST APIs, assim como uma interface que permita um desenvolvimento colaborativo. Divididos em grupos de desenvolvimento distintos, diferentes desenvolvedores contribuem com novas tarefas implementadas no *Framework* através de serviços REST API.

Outra abordagem interessante foi proposta pelo projeto *Frankenstein Framework*[14]. O projeto identificou um problema na integração das diferentes soluções existentes para sistemas de Perguntas e Resposta. A dificuldade em integrar componentes, e que estes sejam adequados ao conteúdo de entrada. Neste cenário, propôs uma solução por meio do estudo de otimizações. O problema é confrontado otimizando a seleção dos componentes que serão utilizados na construção de um *pipeline*. O *pipeline* é construído com os componentes que melhor se adequam as características de entrada e objetivo esperado para os resultados.

Após esta revisão de literatura pôde-se verificar uma crescente tendência no desenvolvimento de tarefas especializadas para sistemas GLN e PLN. Sendo assim, é possível encontrar um grande volume de técnicas, algoritmos e modelos diferenciados para suprir distintas aplicações nesta área. Para compreender como este cenário pode ser explorado, foram estudados artigos que propunham diferentes abordagens e aplicações, interessados em

---

[11] *Named Entity Recognition (NER), Named Entity Disambiguation (NED), Relation Linking (RL), Class Linking (CL) e Query Building (QB).*

[12] *Language Processing Modelling Notation – Orchestration of NLP Microservices*

[13] *OKBQA Framework for collaboration on developing natural language question answering systems.*

[14] *Why Reinvent the Wheel – Let`s Build Question Answering System Together.*

proporcionar reuso e reaproveitamento de tarefas existentes. De modo geral, os artigos destacados acima propõem a exploração deste rico cenário de recursos para GLN e PLN. Por meio de tecnologias distribuídas e arquiteturas modularizadas, foi possível construir sistemas e incorporar diferentes implementações existentes. Os trabalhos têm como objetivo comum integrar recursos a fim de elaborar soluções que permitam executar cada vez mais tarefas complexas de GLN e PLN.

4 MODELO PROPOSTO

A proposta de um modelo para GLN foi concebida após uma análise detalhada dos resultados evidenciados na revisão de literatura, aliada ao estudo dos artigos relacionados ao tema proposto (Seção 3.2), focados na reutilização de recursos em GLN. Pôde-se verificar aplicações e técnicas para geração de linguagem. Com isto, observou-se uma tendência nas aplicações em oferecer serviços com funcionalidades exclusivas, como por exemplo, aplicações para a geração de resumos, relatórios, narrativas, perguntas e respostas ou a geração de descrições curtas para cenários. A oferta de produtos ocorre sempre na perspectiva de atender a apenas um serviço.

Em contraponto a esta gama de ferramentas com aplicações de nicho, a presente pesquisa visa munir a comunidade acadêmica de um modelo que traga facilidades para a experimentação de novas técnicas, disponibilizando variados serviços de GLN a ser prontamente consumidos. Exemplos destas facilidades são: utilização de etapas básicas de processamento de linguagem, canal que possibilite um padrão uniforme de dados, entre outras.

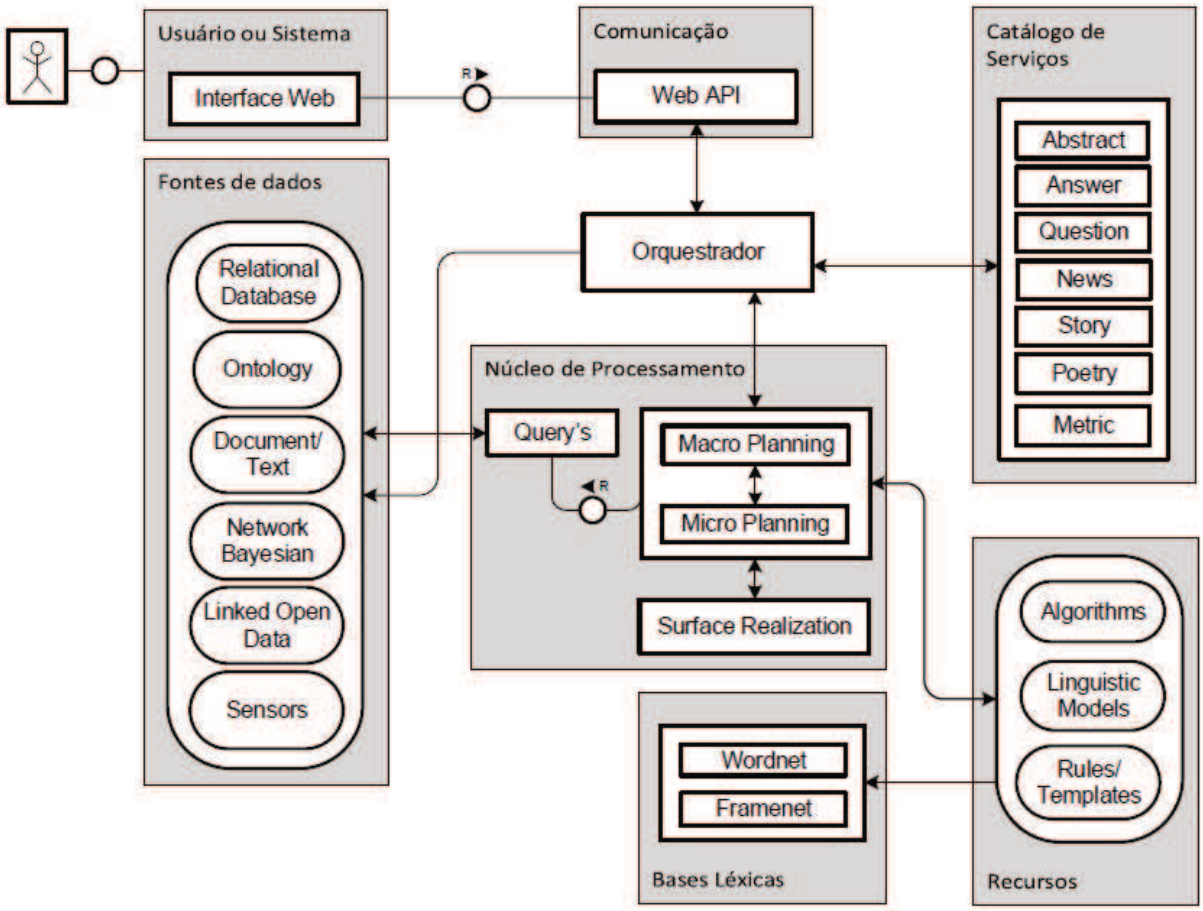
Um modelo singular que possa ser adequado à diferentes contextos e suprir suas necessidades através de técnicas de GLN; proporcionar uma forma facilitada para incorporar bases de dados independentes; atender contextos diversos como simuladores, geradores de notícias, resumos e sumarização de conteúdo, geração automatizada de perguntas ou respostas por meio de um modelo de recursos independentes. Desta forma, pode-se fomentar a maior disponibilidade de GLN e sua aplicabilidade a cenários variados e mesmo multidisciplinares. A definição geral e o detalhamento dos componentes do modelo estão descritos a seguir.

4.1 Visão geral do modelo

A Figura 17 ilustra a estrutura proposta para o modelo, desenvolvida utilizando-se a notação TAM (*Technical Architecture Modeling*) (SAP AG, 2007), um conhecido padrão para representar a definição de arquiteturas. Esta incorpora o padrão FCM (*Fundamental Modeling Concepts*) a nível conceitual, bem como a notação UML para definições a nível design (KNÖPFEL, 2007).

O modelo consiste em componentes que permitem a sua configuração e o acesso aos seus recursos, gerando uma interface para uso. Internamente, diversos tipos de técnicas e recursos podem ser cadastrados e descritos quanto ao seu formato de utilização, em descritores internos. Estes descritores internos são utilizados pelo componente orquestrador do modelo, que possui como função atender às solicitações recebidas, de acordo com o descritor de atendimento para cada tipo de solicitação.

Figura 17 Visão geral modelo proposto



Fonte: Elaborado pela autora.

A seguir, apresenta-se, em detalhes, a definição dos módulos do modelo.

4.1.1 Usuário ou sistema

O módulo de interação com o usuário ou sistema prevê uma interface amigável para acesso rápido às funcionalidades propostas no modelo. A interface acessada pelo usuário, via *browser*, será apenas para verificação e conhecimento de todos os serviços disponíveis, onde alguns recursos poderão ser experimentados. No entanto, o uso por completo dos serviços só poderá ser consumido utilizando RabbitMQ, protocolo AMQP, onde será definido o serviço, formalizada a forma de acesso à base de dados e demais configurações necessárias para a GLN.

4.1.2 Comunicação

O módulo de comunicação prevê recursos para integração dos componentes do modelo e sua interação com o ambiente externo. É a camada de comunicação que tem o objetivo de intermediar a troca de informações entre os componentes do modelo. O modelo

prevê que está camada seja flexível, a fim de facilitar a integração entre os diversos componentes e recursos internos e externos.

A partir da análise de configuração e requisitos do modelo, está previsto que a comunicação será feita através do software de mensageria RabbitMQ, tecnologia de código aberto, desenvolvido na linguagem de programação Erlang[15], amplamente utilizado para intermediar a comunicação entre aplicativos, aplicações e serviços. Um protocolo de mensageria ou fila de mensagens é um estilo de comunicação pensada para incorporar modelos fracamente acopladas entre os diferentes tipos de serviço (DOSSOT et al., 2014). O software implementa o *Advanced Message Queuing Protocol* (AMQP), o qual define a semântica do protocolo interoperável por meio de mensagens.

4.1.3 Orquestrador

O papel do orquestrador é fundamental para o funcionamento do modelo proposto, é o módulo que garante a característica independente, através do conceito de orquestração de recursos. São tarefas exclusivamente destinadas ao orquestrador garantir o recebimento das solicitações de novos serviços, onde este mantém um canal aberto, mesmo que os recursos ou módulos do núcleo de processamento estejam indisponíveis. O orquestrador garante a disponibilidade e permanece disponível para receber solicitações, quando restabelecido o estado dos módulos, estes retomam as tarefas.

Da mesma forma, a responsabilidade de gerenciar o *pipeline* é destinada ao orquestrador. Este recebe a solicitação, encaminha ao módulo responsável pelo processamento e recebe o *pipeline* definido, para que inicie a execução das atividades, delegando as tarefas aos recursos e, por fim, o produto ao cliente.

De modo geral, o processo do modelo é coordenado de acordo com a composição que foi selecionada. Uma composição define o conjunto de recursos necessários para o serviço solicitado. A comunicação que envolve a execução deste é feita exclusivamente pelo orquestrador. As etapas descritas representam as funções do orquestrador, que se mantêm na camada de comunicação. Esta camada engloba todo o processamento presente no modelo, atendo-se apenas a tarefa de comunicar, atua como o intermediador entre os diferentes módulos. Cumpre com o objetivo de isolar a camada de processamento.

Primordialmente, o orquestrador trabalha como o canal de comunicação centralizado, que informa os recursos, recebe o retorno independente destes para dar sequência às atividades, e comunica-se com o cliente quando preciso. O *core* é implementado no módulo núcleo de processamento, que apenas delega ao orquestrador as tarefas de comunicar. Sendo assim, todas as funções acima definidas, tem seu *back-end* implementado no módulo descrito a seguir.

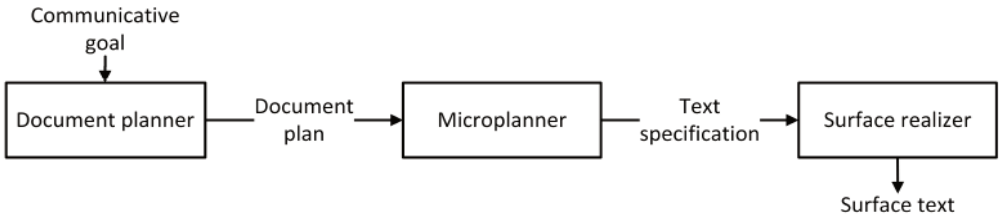
4.1.4 Núcleo de Processamento

[15] Uma linguagem de programação usada para construir sistemas massivamente escalonáveis, com requisitos de alta disponibilidade para sistemas em tempo real.

Os dados recebidos pelo orquestrador são processados em sua totalidade pelo módulo central, detalhado a seguir. Para identificar as características necessárias a este modelo como um todo, e definir cada uma de suas funções, foram verificadas diversas arquiteturas comumente empregadas em sistemas GLN, a fim de reconhecer nestas as particularidades comuns. A análise justifica-se pela necessidade de compreender o contexto geral aos algoritmos, técnicas e ferramentas auxiliares normalmente empregadas a estes sistemas. Este processo traduz o objetivo de integração, para que os recursos sejam utilizados e reutilizados facilmente.

Como estratégia para a definição dos requisitos do módulo núcleo de processamento, decidiu-se adotar uma arquitetura clássica de sistemas GLN como abstração, como princípio norteador para o modelo. Com o objetivo de orientar as definições quanto a organização e segmentação de funções para os componentes internos. Este módulo, portanto, explora esta arquitetura clássica em *pipeline* como uma meta-arquitetura para o modelo. O módulo prevê o processo de decompor os dados recebidos; selecionar o conteúdo relevante recebido pelo orquestrador; aplicar as regras e algoritmos definidos; recompor as instruções necessárias conforme as regras gramaticais da linguagem, por fim, resultando no conteúdo em linguagem natural. Este processo é usualmente aplicado à muitos sistemas GLN, por meio da arquitetura clássica pipeline (PERERA; NAND, 2017). Após a leitura da composição de recursos pelo orquestrador, este módulo central, núcleo do processamento, define de fato o *pipeline*. Uma estratégia dinâmica, que verifica a disponibilidade dos recursos presentes na composição, e delega ao orquestrador que atribua a execução dos mesmos. A Figura 18 ilustra a arquitetura clássica de pipeline.

Figura 18 Exemplo de uma arquitetura pipeline clássica.



Fonte: (PERERA; NAND, 2017).

Com isto, optou-se por utilizar uma abordagem *pipeline* para incorporar o processo de execução dos recursos, assim como incutir seu processo clássico, uma abordagem comumentemente empregada divide o processo de geração de linguagem natural em 3 estágios, assim denominados:

- *Macro-planning* ou *Text Planning*: o ato de falar sobre – o que dizer? Define o escopo de informações e expressões que serão utilizadas na construção da linguagem natural;
- *Micro-planning* ou *Sentence Planning:* etapa de agregação, geração de expressões de referência e lexicalização, estruturação primordial das sentenças – como dizer? A partir dos termos definidos no *Macro-planning* será necessária a estruturação da sentença a ser gerada, seguindo os padrões da linguagem objetivo, português do Brasil;
- *Linguistic:* estágio responsável por definir a estrutura da sentença e mecanismos gerais da língua utilizada. Para a presente pesquisa, aplicam-se as regras do português do Brasil.

4.1.5 Fontes de dados

O modelo proposto beneficia a independência de fontes de dados, pois o modelo pode ser adequado à novos recursos, e estes, por sua vez, realizam a tarefa de acesso às bases distintas. Desta forma, um serviço pode requerer consultas em documentos de texto, ontologias, consultas a bancos de dados, bases de dados abertos e ligados (LOD), e demais bases de dados estruturas, assim como passível de trabalhar com formatos de dados gerados por sensores. Enquanto é desenvolvido o recurso para comunicação com a tarefa desejada, o restante dos recursos disponíveis poderá ser consumido normalmente.

4.1.6 Recursos

O módulo **recursos** representa todos os serviços disponibilizados no modelo, implementados ou acessados externamente através de *APIs*, assim como algoritmos, modelos linguísticos, *templates* e regras. Todos estes recursos formam uma extensa biblioteca de possibilidades para as mais diversas tarefas requeridas ao trabalhar-se com a GLN. É visando esta riqueza de possibilidades para experimentação e aplicação que propõem-se este modelo, que permita a implementação contínua de novos recursos e serviços, bem como a melhora constante de recursos já implementados, isto sem afetar o funcionamento do modelo como um todo.

4.1.7 Bases Léxicas

Bases de dados léxicas são utilizadas em apoio ao processamento e GLN. Este módulo é disponibilizado para que os recursos internos possam acessá-lo de forma facilitada, sem necessidade de um pré-processamento em ferramentas externas. Desta forma, o orquestrador tem este acesso à sua disposição, sem a necessidade de delegar o conteúdo recebido a um posterior recurso, pois o módulo respeita o formato de dados padronizado no modelo.

4.1.8 Serviços

O módulo de serviços serve como catálogo de produtos GLN disponíveis, ao consumir um produto, o usuário solicita-o a partir dos serviços disponíveis. A requisição de um serviço é feita a partir de um formato de dados padrão estabelecido, definido a seguir (Seção 4.2). No campo da GLN, existem diferentes atividades que podem ser exploradas, algumas destas foram analisadas para que fossem disponibilizadas no modelo Pythia NLG. A seguir apresenta-se os tipos de serviços inicialmente previstos ao modelo.

a) *Question e Answer*

No campo da geração de linguagem natural, a geração de respostas é uma atividade comumentemente vinculada à sistemas de perguntas, denominados *Question Answering Systems*, frequentemente possuem as atividades vinculadas, por esta razão, mesmo sendo

serviços disponibilizados de forma independente no modelo, é oportuno fazer sua caracterização em conjunto. Este serviço visa receber uma informação de entrada e, com base em um modelo de conhecimento para consulta, deve identificar o núcleo da questão dada como entrada e, por fim, construir a respectiva resposta. No entanto, para o serviço que visa a geração de perguntas, será recebida a informação de entrada, e desta deve-se gerar questionamentos a partir do núcleo da informação, não necessitando a consulta prévia a uma base de conhecimento para construção de respostas ao contexto.

b) *News*

A geração de notícias é uma área da GLN que visa a construção de uma discussão em torno de um tema proposto. Uma abordagem para este serviço é a utilização de *templates* pré-definidos, onde os mesmos tenham "introdução", "discussão central", "análise crítica" e "conclusão", uma estrutura básica definida para uma notícia. O tema de entrada pode ser, desde um documento contendo apenas dados quantitativos, e a saída esperada seja uma notícia explorando estes dados, ou mesmo a entrada como uma informação e/ou dado único, do qual deve-se desenvolver a notícia. (AIRES; SOBRAL NUNES, 2016)

c) *Story*

A geração de histórias, ou comumentemente denominado como narrativas, consiste na aplicação de técnicas de geração de linguagem para a construção de narrativas criativas a partir de um tema central e/ou parâmetros de partida. A geração de histórias ou narrativas sempre esteve entre os objetivos da área de GLN, especialmente quando aplicada a áreas comerciais, como programas de TVs ou videogames. Certamente este seguimento poderia se beneficiar de um modelo para a automação de processos criativos, prototipagem rápida de narrativas, geração de um determinado conjunto de circunstâncias, como a proposta baseada em modelos é aplicada em (PABLO GERVÁS; BELÉN DÍAZ AGUDO; FEDERICO PEINADO; RAQUEL HERVÁS, 2005) para a geração de histórias.

d) *Summarization*

A sumarização de conteúdo é um campo da geração de linguagem natural usualmente utilizado em áreas que apresentam a geração de um massivo volume de dados, e faz-se necessária uma análise em caráter geral, sendo frequentemente aplicado a unidades clínicas de tratamento intensivo UTI. Um exemplo deste uso é apresentado por (GATT et al., 2009); (HUNTER et al., 2012); assim como aplicações comumentemente apresentadas em sumarização de conteúdos esportivos, notícias e reportagens extensas (MARIA; RAMOS, 2016). Este recurso é habilmente explorado para identificar informações de relevância no contexto geral de um documento, e priorizar estes no produto gerado como resumo.

e) *Poetry*

A geração de poesia compreende a área de construção de narrativas criativas, no entanto, é caracterizado pelo respeito à aspectos da linguagem para a estrutura de poesias. Busca ressignificar a aplicação da geração de linguagem à campos idealmente distantes de produtos empresariais, como perguntas e respostas ou mesmo a geração de relatórios. Não somente representa um cenário inovador para aplicação, como explora características de linguagem informal eventualmente descartadas para modelos mais formais, que se fazem necessários para outras aplicações. Trabalho que serviu de inspiração para exploração desta aplicação neste modelo (EL BOLOCK; ABDENNADHER, 2015).

4.1.9 Query's

Representa o módulo encarregado de intermediar consultas a diferentes modelos de dados aceitos, através de recursos desenvolvidos para esta finalidade. Cada algoritmo, técnica, ou abordagem disponível pode conter um tipo de base como ferramenta auxiliar, ou até mesmo como fonte de dados para a GLN, e nestes casos, é viável configurar um recurso independente para que faça uso desta base, durante o *pipeline.*

Este intermédio de consultas deve ser previamente configurado no módulo ***query's***, e as informações de acesso às fontes de dados são informados no momento da requisição do serviço. A utilização deste módulo se assemelha muito ao uso que se dá ao módulo **bases léxicas**, disponibilizar um recurso de forma integrada para acesso a uma base frequentemente consumida pode favorecer a agilidade com que se entrega o produto GLN.

4.2 Definição dos formatos de dados

A diversidade de serviços a serem oferecidos pelo modelo Pythia NLG é diretamente relacionado à independência entre os recursos, serviços e módulos. Portanto, foi desenvolvido um padrão para encapsular quaisquer comunicações, respeitando um protocolo interno. Foram definidos também outros três formatos de dados destinados a comunicação e troca de informações necessárias às tarefas NLG. Os formatos foram desenvolvidos a fim de promover a interoperabilidade com os serviços e recursos disponíveis.

A definição destes formatos torna o modelo flexível, uma vez que o desenvolvedor se preocupa em definir os dados que irá fornecer para o processamento e geração de linguagem, sem alteração da estrutura de sua aplicação. Definir corretamente as informações que serão enviadas na solicitação de um novo serviço ou no momento de atribuir nova tarefa a um recurso, garante o sucesso na execução das tarefas de GLN. Por esta razão, e compreendendo que toda comunicação requer uma padronização, e que a definição de um protocolo para formalizar um vocabulário e fazer-se compreensível as partes é imprescindível. Para todos os formatos que serão abaixo definidos, utiliza-se a seguinte formalização de dados, conforme indica a Tabela 6, a seguir:

Tabela 6 Definição dos tipos de dados.

Nome	Tipo de informação prevista
Symbol	['A'-'Z''a'-'z''0'-'9'].*{5..32}
Object	string | list | dictionary
Number	['0'-'9']
Character	string

A definição de um elemento do tipo **symbol**, para efeitos de padronização, sugere-se criar um UUID[16] versão 4.

[16] Um identificador universalmente exclusivo, popularmente empregado no mundo da computação. É um número de 128 bits representado por 32 dígitos hexadecimais. A versão 4 indica que é gerado a partir de um número pseudoaleatório.

Com base na formalização indicada na Tabela 6, os formatos de dados definidos são apresentados em detalhes na sequência. Essa definição facilita a compreensão de dados que podem ser utilizados e/ou enviados nas trocas realizadas.

4.2.1 Padrão para encapsular pacotes

Todos os formatos definidos a seguir devem ser encapsulados por um pacote padrão, definido em apenas 3 elementos básicos. A Tabela 7 demonstra em detalhes os tipos de dados para cada elemento definido:

Tabela 7 Formato padrão para encapsular mensagens.

Nome do elemento	Tipo do elemento
action	"register" | "listClients" | "listHashes" | “response" | "lexicalDb" | "request" | "info" | "error" | "task" | "post"
data	Object
fromExchange	Symbol

a) O elemento **action** representa a ação que será comunicada ao módulo destino.

b) O elemento **data** comunica os dados a serem processados pelos diferentes recursos.

c) O elemento **fromExchange** comunica o nome de quem enviou o pacote.

Como esta definição representa apenas o padrão para encapsular as mensagens, este também é feito no formato JSON. Um exemplo do pacote pode ser conferido na Figura 19, a seguir:

Figura 19 Exemplo do pacote para encapsular mensagens.

```
{
    "action": ACTION,
    "data": DATA,
    "fromExchange": EXCHANGE
}
```

Fonte: Elaborado pela autora.

4.2.2 Solicitação de serviço

Buscou-se, durante a definição do modelo, priorizar a facilidade de integrar novos recursos e, neste caso, a possibilidade de adições de novos elementos aos formatos de dados, caso necessário. Primeiramente, adequa-se os dados ao padrão, depois este é encapsulado pelo pacote padrão. A seguir, na Tabela 8, é apresentado o formato de dados para definição de uma solicitação de um novo serviço.

Tabela 8 Descrição do formato de dados padrão para solicitação de serviço.

Nome do elemento	Tipo do elemento	Descrição
data	Object	Dados e configurações necessárias para realizar o serviço solicitado.
composition	Symbol	Nome da composição selecionada para o serviço.
hash	Symbol	Código identificador do usuário (cliente).
name	Character	Nome do serviço solicitado.
description	Character	Descrição da solicitação realizada.
pipelineHash	Symbol	Código único da solicitação.
pipeline	Character	De uso exclusivo do orquestrador.
data	Object	Pacote de dados a ser processado no pipeline.

Fonte: Elaborado pela autora.

A seguir são comentados maiores detalhes de alguns dos elementos da Tabela 8:

a) O elemento **composition** é previamente cadastrado no modelo, e somente então poderá ser indicado em uma solicitação de serviço, isto se deve ao fato de ser um elemento identificador para selecionar uma composição de recursos.

b) O elemento **hash** é o identificador único para o cliente. Todo usuário que deseja consumir serviços no modelo deve solicitar previamente um código único para autenticar-se antes de utilizar o serviço.

c) O elemento **pipelineHash** é gerado pelo usuário (cliente) e não pode ser repetido, é o identificador único para que o cliente, ao receber o produto, saiba a qual solicitação refere-se.

d) O elemento **pipeline** é de uso exclusivo do orquestrador, reservado apenas para configurações internas. Se enviado com conteúdo na solicitação, o mesmo é ignorado.

A Figura 20 apresenta o formato de dados desenvolvido para solicitação de novos serviços, já envolto no pacote padrão:

Figura 20 Formato de dados padrão para realizar solicitação de serviços.

```
{
    "action":ACTION,
    "data": {
        "composition":COMPOSITION,
        "hash":HASH,
        "name":NAME,
        "description":DESCRIPTION,
        "pipelineHash":{...},
        "pipeline":{...},
        "data": {...}
    },
    "fromExchange": FROMENXCHANGE
}
```

Fonte: Elaborado pela autora.

4.2.3 Comunicação entre os módulos

Com o intuito de maior independência entre os módulos, o orquestrador tem a responsabilidade de delegar, alocar e comunicar ao cliente a conclusão dos serviços. Com este objetivo, foi definido um formato de dados padrão para comunicar-se internamente, para comunicação direta entre orquestrador – recurso; recurso – orquestrador. O formato adotado para comunicação interna foi JSON. A seguir, na Tabela 9, explica-se em detalhes os elementos do formato definido.

Tabela 9 Descrição do formato de dados padrão para comunicação interna.

Nome do elemento	Tipo do elemento	Descrição
data	Object	Pacote de dados e configurações.
hash	Symbol	Código único do recurso destino.
pipelineHash	Symbol	Código único da solicitação.
pipeline	Object	Uso exclusivo do orquestrador.
data	Object	Contém o tipo de serviço, e o pacote de dados a ser processado no pipeline.

Fonte: Elaborado pela autora.

O formato definido segue o mesmo padrão já estabelecido, com alguns elementos acrescidos, para atender às demandas do orquestrador. A seguir, a Figura 21 ilustra o formato de dados definido.

Figura 21 Formato de dados padrão para comunicação – comunicação interna.

```
{
    "action":ACTION,
    "data": {
        "hash":HASH,
        "pipelineHash":PIPELINEHASH,
        "pipeline":{...},
        "data":{...}
        },
    "fromExchange": FROMEXCHANGE
}
```

Fonte: Elaborado pela autora.

4.2.4 Composição de recursos

O modelo proposto visa empregar técnicas e algoritmos para GLN, independentemente de arquiteturas adotadas pelos recursos integrados. Esta diversidade implica em uma estruturação de módulos autônomos, tanto a nível de processamento, quanto ações e estratégias de funcionamento. Desta forma, pode-se empregar o uso de técnicas ou algoritmos disponíveis de forma desvinculada a um processo monólito, onde é preciso definir em uma composição, quais os recursos que deseja empregar para atender sua necessidade. Esta característica do modelo proporciona ao usuário, explorar os serviços e empregar tarefas GLN em diferentes aplicações, reutilizando as técnicas e algoritmos disponíveis, pois a dependência está no formato de dados definido como entrada e saída de dados.

Portanto, neste cenário, cada módulo é completamente autônomo, subordinado a um único componente, o orquestrador. O componente orquestrador interage com todos os

recursos através do formato de dados, para isto, adotou-se uma abordagem de modelos de composição. Um modelo de composição é um *template* pré-definido de recursos, algoritmos e outros componentes necessários para a geração de um serviço, definidos em um formato padrão. A especificação de um modelo de composição é feita a partir dos recursos já cadastrados e validados para uso. Estes recebem uma chave de verificação, para que seja liberada a atribuição de tarefas a sua fila. Todos os elementos presentes no formato definido possuem um detalhamento para identificar seu uso na Tabela 10, onde define-se cada elemento, o tipo de dados que este pode conter, por fim, descreve-se rapidamente sua utilização.

Tabela 10 Descrição do formato de dados para definição de uma composição de recursos.

Elemento	Tipo	Descrição
tiposervico	"pergunta" | "resposta" | "resumo" | "notícia" | "narrativa" | "metrica"	Tipo de serviço solicitado.
chave	symbol	Identificador único definido para a composição.
tipoentrada	uri-list" | "uri" | "text-text" | "data-text"	Tipo de entrada esperada para a composição.
tiposaida	"text-text" | "uri" | "metric-data" | “full-data”	Tipo de saída esperada para a composição.
composicao	object	Pacote com uma lista dos recursos.
hash	symbol	Código de identificação do recurso.
nome	symbol	Nome do recurso.
ordem	number	Ordem do recurso no pipeline.
data	object	Pacote de dados a ser processado no pipeline.

A seguir são comentados maiores detalhes de alguns dos elementos da Tabela 10:

a) O elemento **tiposervico** prevê uma série de opções, a cada composição apenas uma é informada. Outras opções podem ser disponibilizadas futuramente.

b) O elemento do tipo **chave** não pode repetir-se dentre as composições existentes, pois trata-se do identificador único de cada composição.

c) O elemento **composicao** prevê como dados uma lista e/ou pacote de dados, informando os recursos selecionados para a composição. Os dados a serem fornecidos são previamente cadastrados e posteriormente fornecidos para montar as composições.

d) O elemento **hash**, somente será válido se os recursos nominados no modelo estiverem dentre os publicados na interface. Para solicitar a incorporação de novas técnicas ou algoritmos, é necessário adaptá-las ao modelo em forma de recursos independentes, e posteriormente este receberá um código identificador.

Assim sendo, para cada tipo de serviço solicitado, na requisição é informada a composição que será utilizada pelo orquestrador. A seguir, a Figura 22 ilustra o formato de dados desenvolvidos para o modelo de composição de recursos, em formato JSON.

Figura 22 Especificação de um modelo de composição conforme definição.

```
{
    "tipoServico":TIPOSERVICO,
    "chave":CHAVE,
    "tipoEntrada": TIPOENTRADA,
    "tipoSaida":TIPOSAIDA,
    "composicao": [
        {
            "nome": NOME,
            "hash": HASH,
            "ordem": ORDEM,
            "data": DATA
        },
        {...}
    ]
}
```

Fonte: Elaborado pela autora.

Foram apresentados os formatos de dados definidos para troca de informações entre módulos e usuários (clientes). Estes permitem que a execução do modelo seja realizada e o consumo de serviços aconteça, através da reutilização de algoritmos e técnicas disponíveis. A seguir serão apontados alguns dos recursos previstos para a implementação do protótipo e sua organização como componentes.

4.3 Algoritmos, recursos e dinâmica de funcionamento do modelo

Nesta seção são comentados estudos preliminares sobre algoritmos e recursos e a possibilidade de seu uso no modelo. Esta etapa foi importante para avaliar preliminarmente as necessidades do modelo e atestar a sua viabilidade. Além disso, são detalhados aspectos da dinâmica geral do modelo proposto.

A proposta de modelo visa apresentar os recursos, disponibilizados via *pipeline*, de duas formas: implementações internas e implementações externas. Com essa abordagem, alguns algoritmos, técnicas ou etapas do processamento de recursos podem ser implementados no modelo como componentes internos. Assim como, quando necessário, podem ser consumidos como recursos externos, não integralmente implementados no modelo, mas apenas acessados por meio de um módulo intermediador.

Ao adotar esta forma de implementação, mantem-se a independência entre recursos, mesmo que seu processamento seja parte como recursos internos, e parte como recursos externos – fornecido por terceiros. A seguir serão apresentados alguns recursos identificados para funcionamento do modelo e suas características para implementação.

Alguns componentes internos identificados são relacionados nos itens abaixo:

a) Bases léxicas para consultas: o uso de bases lexicais em tarefas GLN é recorrente e por esta razão a decisão adotada foi de manter como um recurso interno, disponibilizado através de componentes e abstrações.

b) Recursos básicos para pré-processamento:

   – remoção de palavras de parada (*stopwords)*;

- tokenização (*tokenize*);
- lematização (*stemmer*);
- categorização de termos (*speech tagger*).

c) Recursos de sumarização automática: recurso destinado à sumarização de documentos em texto. As implementações foram realizadas a partir do trabalho realizado por (MARIA; RAMOS, 2016).

d) Métricas para avaliação: para verificar a qualidade dos textos gerados pela Sumarização Automática, existem algoritmos especializados neste tipo de análise. Inicialmente foram selecionadas duas métricas que serão implementadas e disponibilizadas para avaliar o produto gerado nos algoritmos. Foram selecionadas a métrica Bleu e a métrica Rouge.

Alguns componentes externos identificados são relacionados nos itens abaixo:

a) Consumir os recursos do projeto Athena: recurso intermediador, que permite acesso ao projeto Athena! (SILVA et al., 2018), este é distribuído através de *Web Api* desenvolvida pelo autor.

b) Consumir recurso para compreensão de linguagem: recurso intermediador, que permite acesso ao projeto ENSEPRO (ARAUJO; HENTGES; RIGO, 2018), recebe uma afirmação e/ou pergunta, e devolve uma resposta em forma de URI.

c) Consumir recurso de inferência no modelo bayesiano: recurso intermediador, que permite o acesso ao projeto *Health Simulator* (BEZ et al., 2018) para realizar inferências no modelo bayesiano projetado para o simulador, a fim de recuperar perguntas e respostas pré-definidas.

Nas seções preliminares foram definidos os componentes do modelo, os formatos para troca de dados, assim como o *template* para criar composições de recursos no modelo, bem como alguns exemplos de algoritmos e recursos previstos para a implementação no protótipo, divididos entre recursos internos e externos. Este contexto descreve o que é necessário para efetuar as tarefas GLN previstas no modelo e atender as particularidades e exigências ao aplicar os algoritmos, via *pipeline* de recursos, independentemente de suas arquiteturas. Deste modo, é possível iniciar a análise de como funcionará o fluxo de processamento no modelo, descrevendo como de fato deverá ocorrer a troca de informações e a relação dos módulos do modelo.

O ponto de partida para esta análise inicia na seleção de um modelo de composição desejado. Com o modelo de composição, o orquestrador realiza a leitura, a fim de definir o pipeline de recursos, daqueles que estejam disponíveis – online. Em seguida perpassará as tarefas definidas no *pipeline*, e alocará os recursos na ordem definida. A atribuição de tarefas é realizada pelo orquestrador, por meio da comunicação orquestrador – recurso e recurso – orquestrador; essa abordagem permite que os recursos mantenham a característica de módulo autônomo.

O papel do orquestrador compreende a rotina de atribuir tarefas aos recursos – ou seja, delegar tarefas e atividades essencialmente comunicativas. Esta rotina prevê algumas etapas,

como: analisar o modelo de *pipeline*; identificar os recursos especificados; seguir a ordem de acordo com a ordem estabelecida na composição; alocar os módulos ao enviar os dados; por fim, receber o produto do recurso e enviar ao usuário solicitante (cliente).

Para exemplificar as situações comentadas acima, e proporcionar elementos de avaliação futura, fez-se um processo de análise por cenários de diferentes casos de uso do modelo. Estes cenários são detalhados a seguir.

4.4 Cenários de uso

Nesta seção serão descritos dois cenários de uso do modelo, com aplicações que permitem a visualização e compreensão das funcionalidades previstas para o protótipo que será desenvolvido. Com estes cenários, propõe-se validar a flexibilidade idealizada durante a definição do modelo. Os cenários pertencem a contextos distintos, sendo um para a geração de sentenças a partir de uma ou mais URI fornecidas pelo usuário, solicitante do serviço. O outro cenário descreve possibilidades para geração de respostas a partir de perguntas pré-definidas em um simulador de casos clínicos.

4.4.1 Geração de sentenças a partir de triplas – Cenário 1

No cenário em estudo optou-se por trabalhar com os resultados disponibilizados pelo projeto ENSEPRO (ARAUJO; HENTGES; RIGO, 2018). Este faz uma análise e posterior compreensão de sentenças dadas como entrada. Com o auxílio do modelo proposto – Pythia NLG, sugere-se empregar o serviço de geração de sentenças aos diferentes termos de entrada inseridos, questões de pergunta, produto disponibilizado pelo projeto ENSEPRO.

Para compreensão do cenário é necessário verificar inicialmente o funcionamento do projeto ENSEPRO. Em sua definição recebe uma sentença, da qual deve compreender o núcleo da informação a ser transmitida. Posteriormente, a sentença será analisada e classificada entre seus tipos internos, tais como: consulta sobre informações como qual; quem; quando; quanto; onde; o que. Com estas informações, uma consulta SPARQL é construída pelo ENSEPRO. Esta consulta é executada na base de dados abertos e ligados DBPedia, para obter os conceitos daquilo que foi questionado na sentença de entrada. Após a consulta, os resultados são apresentados em formato de triplas, recebendo uma lista de *URIs*, das quais seleciona-se as que são relevantes.

Neste contexto, é oportuno aplicar a solução prevista pelo projeto Athena!, desenvolvido por (SILVA et al., 2018), que trata da geração de frases curtas com perguntas a partir de bases de dados ligados, utilizando triplas *URI* como dados de entrada. Foi empregada uma abordagem para geração de linguagem baseada em *templates,* um formato pré-definido para nortear a geração de novas sentenças. Dada a situação do algoritmo ser formalmente desenvolvido para trabalhar com *templates* de geração de perguntas, há a necessidade de adaptá-los para a geração de respostas, e assim viabilizar o serviço requerido. Os *templates* desenvolvidos são disponibilizados no módulo de recursos da Pythia NLG.

O projeto ENSEPRO materializa sua necessidade por meio da requisição de um serviço via Web API. O orquestrador, por sua vez, recebe a requisição com os dados necessários para a definição do serviço. A Figura 23 demonstra um exemplo de requisição

para a sentença "**Liste todos os musicais com músicas de Elton Jonh.**", sentença enviada pelo projeto ENSEPRO, disponibilizando como resultado principal uma consulta SPARQL.

Figura 23 Cenário 1: exemplo de requisição feita ao Web Service.

```
{
  "version":                    "1.0",
  "service":{
    "name":                     "answer"

},
  "parameters": [{
    "dataType":                 "text",
    "dataEncode":               "utf-8",
      "dataSource": [{
        "type":                 "LOD",
        "name":                 "dbpedia",
        "queryType":            "sparql",
        "dataAcess": [{
          "host":               "http://dbpedia.org/sparql/",
          "databaseName":       [],
          "port":               [],
          "user":               [],
          "password":           []
        }]
      }]
  }],
  "contents": [{
    "dataInput":                "Liste todos os musicais com músicas de Elton John.",
    "dataQuery":[               "PREFIX dbo: http:////dbpedia.org/ontology//",
                                "PREFIX res: <http:////dbpedia.org/resource/>",
                                "PREFIX rdf: <http:////www.w3.org//1999//02//22-rdf-syntax-ns#>",
                                "SELECT DISTINCT ?uri",
                                "WHERE { ",
                                        "?uri rdf:type dbo:Musical . ",
                                        "?uri dbo:musicBy res:Elton_John .} "
    ]
  }]
}
```

Fonte: Elaborado pela autora.

As informações de entrada para a Web API, bem como suas respectivas especificidades são definidas neste momento. O serviço requerido foi o tipo *"answer"*, e seus parâmetros definidores informam o tipo de entrada em formato de texto, com codificação em *"utf-8"*.

As propriedades dividem-se em duas grandes categorias, a primeira refere-se exclusivamente a base de dados e acesso a mesma, a segunda ao conteúdo que dará contextualização as sentenças geradas.

Desta forma, extrai-se da propriedade *"parameters"* as informações necessárias para apoio e construção da sentença em linguagem natural. O documento JSON contém desde os dados de acesso, tais como *host*, usuário, senha e nome da base, bem como, a linguagem e o tipo de consulta que esta aceita. A possibilidade de trabalhar com bases diferentes torna estes dados necessários. Para a requisição, foi definida uma base de dados abertos e conectados, a DBPEDIA, que disponibiliza uma interface para que consultas sejam realizadas nos dados da *Wikipedia*. A primeira categoria de dados norteará o processo de análise do orquestrador para definir uma estratégia de ação, pois a partir desta pode-se saber qual será a entrada de dados, seu formato, e qual será a base utilizada para consultas. A outra grande categoria, refere-se a propriedade "*contents*", traz todo o material para contextualização, o conteúdo para geração de linguagem. Suas informações são enviadas em duas sub propriedades: "*dataInput*", que

recebe o conteúdo de entrada, e "*dataQuery*", traz a consulta construída pelo projeto ENSEPRO, a qual será executada na base de dados definida.

De modo geral, ao analisar a requisição feita pelo projeto, este deseja consumir o serviço do tipo "*answer*" – geração de resposta ou sentença. Como seu produto será a entrada, sua requisição envia uma consulta em linguagem SPARQL, definindo a DBPedia como base de dados para consulta. A requisição, por fim, é analisada por completo pelo orquestrador e, conforme a proposta do modelo Pythia NLG, uma estratégia de ação será definida. A definição de uma estratégia dá-se a partir da análise de todos os parâmetros e configurações dadas como entrada, selecionando-se um modelo de composição de recursos. Desta forma, atribui-se ao orquestrador delegar aos módulos independentes suas tarefas, pois este interpreta o modelo de composição. A possibilidade de módulos autônomos dita uma série de produtos em geração de linguagem, diversificados por meio da combinação destes recursos. A Figura 24 ilustra o *template* selecionado como modelo de composição de recursos para o cenário em estudo.

Figura 24 Cenário 1: exemplo do modelo de composição selecionado.

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="answer" >
    <resources>
        <resource name="algorithm" ordem="10">consult_sparql_virtuoso</resource>
        <resource name="algorithm" ordem="8">select_best_triple</resource>
        <resource name="algorithm" ordem="7">triple_to_text_augusto</resource>
    </resources>
    <datasources>
        <datasources name='dbpedia'/>
    </datasources>
    <parameters>
        <parameter name="">  </parameter>
    </parameters>
</service>
```

Fonte: Elaborado pela autora.

Com o modelo de composição selecionado, o orquestrador perpassará as tarefas definidas, delegando-as na ordem pré-estabelecida na composição. A ordem atribui ao modelo de composição um *template* para controle dos módulos, e por meio desta comunicação direta: orquestrador – recurso; recurso – orquestrador. A abordagem permite tornar cada módulo autônomo, o recurso do modelo, e o usuário (cliente) do modelo.

O papel do orquestrador, nesta etapa, compreende a rotina de alocação de recursos – delegar tarefas. Esta rotina segue os seguintes passos: analisar o modelo de composição; identificar os recursos especificados; selecionar o de maior ordem; alocar o módulo selecionado e enviar os dados; por fim, receber o produto do recurso. Após a leitura do modelo, define-se, portanto, o *pipeline* do serviço requerido. Este processo de análise do valor de ordem é realizado para cada recurso ou algoritmo especificado no modelo de composição, até que o *pipeline* seja esgotado. A Figura 25 ilustra um exemplo do formato de dados para alocação do recurso de consulta à DBPEDIA, através da API Virtuoso.

Figura 25 Cenário 1: exemplo de requisição para alocar o recurso de consulta Virtuoso.

```
{
    "service":                ["answer", "data-text"],
    "resource":{
        "algorithm":          ["consult_sparql_virtuoso", 10]
    },
    "parameters":[{
        "type":               ["LOD"],
        "name":               ["DBPEDIA"],
        "consult":            ["SPARQL"]
    }],
    "contents": {
        "dataInput": [{
            "dataType":       ["SPARQL"],
            "data":           ["PREFIX dbo: http:////dbpedia.org/ontology//
                               PREFIX res: http:////dbpedia.org/resource/
                               PREFIX rdf: <http:////www.w3.org//1999//02//22-rdf-syntax-ns#>
                               SELECT DISTINCT ?uri
                               WHERE {
                                      ?uri rdf:type dbo:Musical .
                                      ?uri dbo:musicBy res:Elton_John .
                               }"]
        }],
        "dataOutput":[{
            "dataType":       ["uri-list"],
            "data":           ["http://dbpedia.org/resource/Billy_Elliot_the_Musical",
                               "http://dbpedia.org/resource/The_Lion_King_(musical)",
                               "http://dbpedia.org/resource/Aida_(musical)",
                               "http://dbpedia.org/resource/Lestat_(musical)",
                               "http://dbpedia.org/resource/The_Lion_King_Jr_(musical)"],
        }]
    }
}
```

Fonte: Elaborado pela autora.

Os dados informados são repassados pelo orquestrado, ao recurso alocado. No cenário em estudo, o recurso "*consult_sparql_virtuoso*" é inicialmente alocado, pois seu valor de ordem é "10", evidência de sua prioridade na execução. Através do recurso, são acessados os dados da consulta em linguagem SPARQL, e este executa-a na base. Como informado anteriormente, este recurso utiliza uma base com o padrão *LOD*, portanto, o retorno esperado será uma lista de *URIs* que satisfaçam o conceito da consulta. Os parâmetros que foram enviados definem o tipo de base em que serão realizadas as consultas e contextualizam o tipo de busca que é aceita. O retorno será entregue novamente ao Orquestrador, que definirá a próxima etapa do *pipeline*.

O orquestrador deverá selecionar no modelo de composição o próximo recurso de maior ordem e alocá-lo. Novamente, no papel de regente do *pipeline*, o orquestrador analisará o modelo de composição, neste cenário, o modelo de composição traz como segundo recurso de maior ordem, o responsável pela seleção de melhor tripla *URI*.

O projeto Athena! requer uma entrada de dados de apenas uma *URI*, e que corresponda ao *template* pré-determinado. Portanto, será necessário validar as triplas encontradas, e selecionar dentre estas, a que melhor se encaixe no *template*. Vale lembrar que um recurso nunca irá comunicar-se diretamente a outro recurso, esta responsabilidade recai apenas ao orquestrador. O orquestrador deverá alocar aquele que agora apresenta o número de maior ordem, *"select_best_triple"*. A Figura 26 ilustra os dados necessários para alocar o recurso, onde o valor de entrada, propriedade *"dataInput"*, representa o produto recebido pelo orquestrador do recurso imediatamente anterior.

Figura 26 Cenário 1: exemplo de requisição para alocar o recurso de seleção de tripla.

```
{
    "service":              ["answer", "data-text"],
    "resource":{
        "algorithm":        ["select_best_triple", 8]
    },
    "parameters":[{
        "type":             ["uri-list"],
        "name":             [],
        "consult":          []
    }],
    "contents": {
        "dataInput": [{
            "dataType":     ["uri-list"],
            "data":         ["http://dbpedia.org/resource/Billy_Elliot_the_Musical",
                             "http://dbpedia.org/resource/The_Lion_King_(musical)",
                             "http://dbpedia.org/resource/Aida_(musical)",
                             "http://dbpedia.org/resource/Lestat_(musical)",
                             "http://dbpedia.org/resource/The_Lion_King_Jr_(musical)"]
        }],
        "dataOutput":[{
            "dataType":     ["uri"],
            "data":         ["http://dbpedia.org/resource/The_Lion_King_(musical)"]
        }]
    }
}
```

Fonte: Elaborado pela autora.

O recurso para seleção da melhor tripla receberá a lista de *URIs* a serem analisadas, esta lista é recebida no formato de dados definido, e confrontada aos *templates* linguísticos disponíveis para o recurso alocado. Após selecionada a tripla correspondente, esta é devolvida ao orquestrador, para que o mesmo possa alocar o recurso subsequente.

O orquestrador segue perpassando o *pipeline*, e analisa novamente o modelo de composição, onde o recurso de maior ordem passa a ser *"tiple_to_text_augusto"*, e aloca-o para uso. A Figura 27 demonstra a chamada ao recurso que irá gerar a sentença a partir da tripla selecionada.

Figura 27 Cenário 1: exemplo de requisição para alocar o recurso de geração a partir de tripla.

```
{
    "service":              ["answer", "data-text"],
    "resource":{
        "algorithm":        ["triple_to_text_augusto", 7]
    },
    "parameters":[{
        "type":             ["uri"],
        "name":             [],
        "consult":          []
    }],
    "contents": {
        "dataInput": [{
            "dataType":     ["uri"],
            "data":         ["http://dbpedia.org/resource/The_Lion_King_(musical)"]
        }],
        "dataOutput":[{
            "dataType":     [],
            "data":         []
        }]
    }
}
```

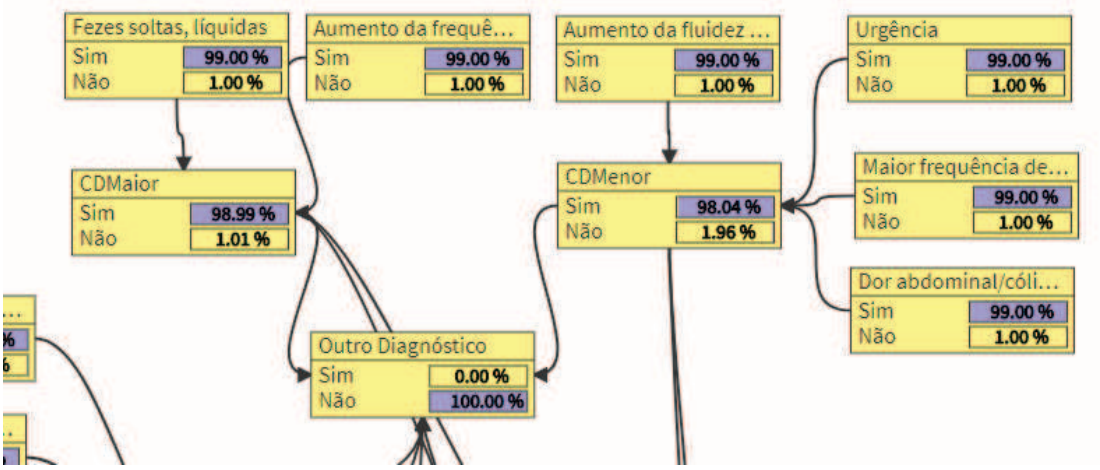
Fonte: Elaborado pela autora.

4.4.2 Geração de respostas a partir de perguntas pré-definidas – Cenário 2

Para este cenário, optou-se por trabalhar com uma proposta diferenciada, com foco na área da saúde, a partir de um simulador de casos clínicos denominado *Health Simulator*. O projeto visa proporcionar ao professor da área da saúde uma ferramenta de apoio ao ensino e ao aluno um ambiente tolerante a falhas, com simulações de cenários reais, preparando-o para a rotina de um profissional da saúde (BEZ et al., 2018). O simulador utiliza um modelo estatístico para representação do conhecimento, uma rede bayesiana. Esta é uma reconhecida abordagem para a tomada de decisões na área da medicina, por se tratar de uma representação do conhecimento incerto (LIMA et al., 2015). Através de inferências no modelo bayesiano, o simulador tem acesso ao conhecimento do especialista, materializado na rede.

Para melhor compreender a proposta de aplicação da Pythia NLG ao *Health Simulator*, faz-se necessário compreender o funcionamento do simulador. O enredo do simulador (no formato de um jogo) inicialmente é construído por um especialista, que irá cadastrar previamente o caso clínico pela interface administrativa, inserindo informações que são recorrentes na atividade e rotina de profissionais da área da saúde, tais como: anamnese, exames físicos disponíveis, hipóteses diagnósticas, exames, diagnósticos, tratamento e o desfecho do caso clínico. Estas informações são disponibilizadas para o profissional da saúde – aluno em simulação – consultar durante sua partida. O caso clínico é vinculado a uma rede bayesiana. Esta possui uma série de perguntas e respostas atreladas a seus nodos. Posteriormente, são definidos os nodos ativos para o caso clínico, bem como seus percentuais de possíveis ocorrências em sinais, sintomas, diagnósticos e tratamentos. Essa interação ocorre através de uma integração da interface administrativa com o modelo de conhecimento, a rede bayesiana. A Figura 28 ilustra um fragmento de uma das redes bayesianas desenvolvidas para serem utilizadas no projeto *Health Simulator*.

Figura 28 Cenário 2: Fragmento de uma rede bayesiana de diagnóstico de diarreia e risco de diarreia construída para o projeto *Health Simulator*.



Fonte: (ROCKENBACK et al., 2017).

Para cada nodo da rede são cadastradas perguntas e suas respectivas respostas. Ao ser iniciada, a simulação apresenta ao jogador, aluno da área da saúde, o ambiente selecionado pelo especialista para o caso clínico. O projeto *Health Simulator* traz, dentre os cenários disponíveis, consultórios ou hospitais das classes A, B ou C, assim como opcionalmente o

cenário que representa um hospital do Sistema Único de Saúde (SUS) (MELLO; STAHNKE; BEZ, 2015).

No momento em que o aluno, no papel de personagem profissional da saúde, atende o paciente virtual, o jogo lhe apresenta as perguntas pré-definidas e vinculadas à rede, para que o mesmo possa inquirir o paciente sobre sua condição atual. As perguntas feitas pelo aluno tem a finalidade de constatar sinais, sintomas e características que possam direcionar seu diagnóstico e tomada de decisão para o desfecho proposto no caso clínico. A este processo de entrevista sobre a condição do paciente, dá-se o nome de anamnese, e será o momento em que o jogador – aluno da área da saúde – selecionará as perguntas que melhor lhe satisfaçam no caso apresentado. Este cenário, onde há a seleção de perguntas pré-definidas, pode vir a induzir o aluno a selecionar perguntas desnecessárias ao caso em questão, assim como irrelevantes à real condição do paciente, impossibilitando-o de concluir corretamente seu raciocínio, e impedindo-o de formular seus próprios questionamentos, pois as perguntas já estarão formuladas.

Diante de tal cenário, propõe-se uma abordagem onde o aluno possa escrever suas perguntas em linguagem natural no simulador, e receba deste uma resposta em linguagem natural, através do uso do modelo Pythia NLG para recebimento das perguntas e formulação das respostas. Com este fim, foi realizada uma análise dos recursos necessários para atender ao projeto *Health Simulator* por meio de um cenário de controle. Esta análise considera reaproveitar recursos já implementados, mas principalmente, contempla a etapa para identificar quais recursos seriam necessários para atender ao cenário simulado.

O processo de análise, adaptações, e posterior implementação são etapas intrínsecas para que um novo tipo de serviço seja disponibilizado. Inicialmente, o cenário em análise requer um serviço que possa trabalhar com uma base de conhecimento diferenciada, voltada para a área da saúde, com um módulo especializado em inferências para modelos bayesianos e entradas de texto em linguagem natural. Outra importante etapa levantada durante o processo de análise, é a formulação e contextualização das respostas, em que o recurso precisa estar apto a receber os percentuais de referência, estes vindos da inferência no modelo de bayesiano. Os percentuais são o conhecimento do especialista a ser insuflado na resposta, para que estas direcionem o aluno, estudante da área da saúde, assim como suas decisões, ao desfecho esperado para o caso clínico.

Uma vantagem inerente ao modelo pode ser melhor observado quando se trata da adequação de recursos já existentes, estes são prontamente adaptados e facilmente consumidos pelo novo serviço, sem necessidade de outras modificações. Os recursos que precisam ser desenvolvidos exclusivamente para o projeto *Health Simulator,* e adaptados ao do modelo, igualmente realçam a proposta de flexibilidade para novos serviços. A seguir serão apresentados em detalhes os recursos já disponíveis, e na sequência, sem particularidades da implementação, uma visão prévia dos recursos em desenvolvimento para satisfazer o projeto como usuário (cliente).

Para exercitar a análise realizada, fez-se essa simulação do cenário na execução do *pipeline*, detalhes do seu processamento, onde o projeto *Health Simulator* materializa sua necessidade por meio da requisição de um serviço via Web API. O orquestrador receberá esta requisição com os dados necessários para definição do serviço.

A Figura 29 demonstra um exemplo de requisição prevista para a entrada de perguntas em linguagem natural feita durante a simulação **"Você sente dores abdominais com alguma frequência durante o dia?"**. Em conjunto, alguns dados serão enviados, principalmente, visando a contextualização da pergunta com a rede e o nodo em questão. Isto porque o projeto

Health Simulator deve informar sob qual modelo de conhecimento está efetuando a simulação, o que possibilita a formulação de uma resposta adequada ao caso, correspondente à base de conhecimento e, por fim, permite o entendimento por parte do usuário, aluno da área da saúde.

Figura 29 Cenário 2: exemplo de requisição padrão para o Web Service pelo projeto *Health Simulator*.

```
{
    "version":                  ["1.0"],
    "service": {
        "name":                 ["answer"],
    },
    "parameters": [{
        "dataType":             "text",
        "dataEncode":           "utf-8",
        "dataSource": [{
            "type":             ["SQL"],
            "name":             ["SQLSERVER"],
            "queryType":        ["SQL"],
            "dataAcess": [{
                "host":         ["http://ca.feevale.br/healthsimulator/query/"],
                "databaseName": ["healthsimulator"],
                "port":         [],
                "user":         ["0000#.$@"],
                "password":     ["r2d2"]
            }]
        }]
    }],
    "contents": [{
        "dataInput":            ["Você sente dores abdominais com alguma frequência durante o dia?"],
        "dataQuery":[],
        "data":                 ["dor abdominal","cólicas"]
    }]
}
```

Fonte: Elaborado pela autora.

Ao receber a requisição, o regente do *pipeline,* no papel do Orquestrador, deverá definir uma estratégia de ação com base nos parâmetros de entrada. Os dados iniciais referem-se ao serviço que foi requerido, do tipo *"answer",* esta informação é um dado chave para nortear a seleção do modelo de composição que será adotado. A seguir, os parâmetros para acesso a base de dados, que servem de apoio e fonte de contextualização para a GLN. Conforme as especificações para o modelo Pythia NLG, a codificação dos dados deve ser em *"utf-8"*, tanto para dados de entrada, quanto dados de bases independentes.

No papel do orquestrador, este gerencia o processo de auditoria. Análise realizada perpassando o modelo de dados, com o intuito de verificar a consistência das informações enviadas. Posteriormente, deve correlacionar e validar os pré-requisitos dos modelos de composição existentes, a fim de selecionar o modelo de composição que melhor atenda ao serviço requerido. Esta etapa foi assim definida para possibilitar uma facilidade na contínua construção e adaptação de novos serviços. Depois de selecionado o modelo de composição, e feita a sua leitura pelo Orquestrador, será iniciada a execução do *pipeline*, alocando recurso a recurso através de seus níveis de ordem definidos no modelo. A Figura 30 ilustra o modelo de composição construído para atender o projeto *Health Simulator*:

Figura 30 Cenário 2: modelo de composição para projeto *Health Simulator*.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<service name="answer">
    <resources>
        <resource name="algorithm" implement="internal" ordem="10">remove_stopwords</resource>
        <resource name="algorithm" implement="internal" ordem="9">verify_redundancy</resource>
        <resource name="algorithm" implement="internal" ordem="2">build_answer_bayesian_percent</resource>
        <resource name="algorithm" implement="internal" ordem="8">select_keywords</resource>
        <resource name="algorithm" implement="internal" ordem="6">sqlserver_consult</resource>
        <resource name="algorithm" implement="internal" ordem="7">sqlserver_builder</resource>
        <resource name="algorithm" implement="internal" ordem="5">search_similar_word</resource>
        <resource name="algorithm" implement="external" ordem="3">inference_node_bayesian_network</resource>
    </resources>
    <datasources>
        <datasource name="sqlserver"/>
        <datasource name="bayesian_network"/>
    </datasources>
    <parameters>
        <parameter name=""> ... </parameter>
    </parameters>
</service>
```

Fonte: Elaborado pela autora.

Conforme o modelo de composição, o orquestrador atribui a primeira tarefa ao recurso de maior ordem, alocando, por sua vez as atividades do recurso *"remove_stopwords"*. A remoção de palavras irrelevantes é uma das medidas mais básicas no PLN, é utilizada para remover termos com papel irrelevante na sentença, que não apresentem importância ao significado geral e contexto. A Figura 31 ilustra os dados enviados para o recurso através do formato definido.

Figura 31 Cenário 2: exemplo de requisição para alocar recurso e remover palavras irrelevantes.

```json
{
    "service":              ["answer", "text"],
    "resource":{
        "algorithm":        ["remove_stopwords", 10]
    },
    "parameters":[{
        "type":             [],
        "name":             [],
        "consult":          []
    }],
    "contents": {
        "dataInput": [{
            "dataType":     [],
            "data":         ["Você sente dores abdominais com alguma frequência durante o dia?"],
            "dataObs":      []
        }],
        "dataOutput":[{
            "dataType":     ["text"],
            "data":         ["dores", "abdominais", "frequência", "dia"]
        }]
    }
}
```

Fonte: Elaborado pela autora.

O recurso a ser alocado, denominado *"select_keywords"*, *é* responsável por analisar a sentença e etiquetar as palavras em seus significados e papéis sintáticos. Apenas as palavras de papéis relevantes devem ser inclusas na consulta que será posteriormente executada na base de dados. Os papéis sintáticos de maior relevância em uma sentença são: verbos no infinitivo, adjetivos e substantivos, respectivamente nesta ordem, conforme análise de resultados preliminares apontados em experimentos para este cenário (MELLO et al., 2018). Assim sendo, adotou-se esta abordagem para identificar e selecionar palavras-chave por nível de relevância, a fim de encontrar o núcleo do discurso na sentença; as palavras-chave servirão de ponto de contextualização para construir a consulta. Igualmente importante para a

construção da consulta, é limitar o número de palavras-chave que serão utilizadas, para não tornar a consulta demasiadamente ampla pela variedade de termos, isto poderia ocasionar o retorno de perguntas indevidas e fora de contexto. A Figura 32 ilustra a requisição enviada pelo Orquestrador ao recurso para seleção das palavras de relevância na sentença **"Você sente dores abdominais com alguma frequência durante o dia?"**.

Figura 32 Cenário 2: exemplo de requisição ao recurso responsável por selecionar as palavras de maior relevância na sentença.

```
{
    "service":              ["answer", "text"],
    "resource":{
        "algorithm":        ["select_keywords", 8]
    },
    "parameters":[{
        "type":             [],
        "name":             [],
        "consult":          []
    }],
    "contents": {
        "dataInput": [{
            "dataType":     [],
            "data":         ["Você sente dores abdominais com alguma frequência durante o dia?"],
            "dataObs":      []
        }],
        "dataOutput":[{
            "dataType":     ["text"],
            "data":         ["dores", "abdominais", "frequência"]
        }]
    }
}
```

Fonte: Elaborado pela autora.

O recurso é responsável pela análise e seleção das palavras-chave de maior relevância, limitando-as a somente três para o retorno. A existência de um número excedente de palavras-chave não representa um indicativo de que a pergunta seja facilmente respondida, da mesma forma, a resposta encontrada pode não responder diretamente à pergunta realizada. Por esta razão, juntamente à requisição, é enviada a informação do nodo ao qual o jogador, aluno da área da saúde, está inquirindo no momento, pois cada pergunta deve relacionar-se diretamente a um nodo.

Removidos os termos irrelevantes ao contexto, e definidos os termos relevantes, a sentença pré-processada é devolvida ao Orquestrador, que seleciona o próximo recurso de maior ordem. Conforme o modelo, é alocado o recurso *"verify_redundancy"*, este deve analisar as características lexicais da sentença e remover termos de significado redundante. Esta tarefa não é intrínseca ao PLN, no entanto, dada a abordagem para identificar o conceito da pergunta de entrada, bem como a necessidade de localizar sua respectiva resposta na rede, não serão necessários conceitos redundantes. A Figura 33 ilustra um exemplo da requisição realizada ao recurso.

Figura 33 Cenário 2: exemplo de requisição para o recurso responsável pela verificação de ambiguidade entre palavras de uma sentença.

```
{
    "service":              ["answer", "text"],
    "resource":{
        "algorithm":        ["verify_redundancy", 9]
    },
    "parameters":[{
        "type":             [],
        "name":             [],
        "consult":          []
    }],
    "contents": {
        "dataInput": [{
            "dataType":     [],
            "data":         ["Você sente dores abdominais com alguma frequência durante o dia?"],
            "dataObs":      []
        }],
        "dataOutput":[{
            "dataType":     ["text"],
            "data":         ["dores", "abdominais", "frequência", "dia"]
        }]
    }
}
```

Fonte: Elaborado pela autora.

Os termos resultantes devem ser enviados ao Orquestrador, que atribuirá uma tarefa ao próximo recurso de maior ordem, responsável pela construção da consulta SQL utilizando as palavras-chave e nodo, estes com a função de contextualizar a busca. O recurso em desenvolvimento *"sqlserver_builder"* será responsável por esta tarefa.

O projeto *Health Simulator* utiliza o banco de dados *Sql Server Express* fornecido pela *Microsoft*. Ao concluir a construção da consulta, o recurso a devolverá ao Orquestrador, e este deverá alocar o recurso subsequente, o qual será responsável por efetivamente executar as consultas na base de dados. Quaisquer interações com o *Health Simulator* devem ser feitas através de sua Web API, sejam elas apenas consultas referentes ao jogo, para receber e retornar perguntas ou respostas, bem como acessos à sua base de dados.

Com este propósito, e após feita a análise inicial, foi solicitado o desenvolvimento de dois métodos para adequação ao modelo Pythia NLG: um para executar consultas SQL enviadas diretamente, cumprindo os requisitos de autenticação necessários; outro para realizar inferências no modelo bayesiano. Com os métodos prontos, será possível utilizar os recursos para adequação do modelo, de modo que estes possam comunicar-se com a interface para enviar e receber os resultados da consulta.

Se efetuada com sucesso, a consulta à base de dados deverá retornar uma lista das perguntas relacionadas ao nodo/rede e as palavras-chave enviadas. Distintas abordagens para tratamento das perguntas e suas respectivas respostas têm sido discutidas. Em vista do objetivo, em primeiro momento, ser a possibilidade de aceitar perguntas em linguagem natural, ao atingir este, o segundo objetivo é flexibilizar as respostas pré-definidas às perguntas realizadas pelo jogador. A proposta consiste em solucionar a identificação do formato de pergunta realizada. Como apresentado na análise do cenário 1, com uso conjunto do projeto ENSEPRO, a forma como a pergunta é construída impacta na formatação de sua resposta. Fica clara essa constatação no exemplo a seguir, a pergunta "**Com que frequência você sente dores abdominais fortes?**", poderia ser construída **"Quando você sente dores abdominais fortes?**". Em essência, ao analisar as sentenças, suas palavras-chave seriam idealmente as mesmas, mas a construção das respostas precisa estar de acordo ao que a pergunta questiona de fato, e sua resposta deve obedecer aos termos "com que' e "quando", que definirão o formato. Ignorar tais termos pode desestimular o jogador, aluno, a arriscar perguntas diferenciadas, assim como prejudicar sua compreensão das respostas.

A resposta formatada será entregue novamente ao orquestrador, que atribuirá ao recurso subsequente a tarefa de flexibilizar a resposta por meio de uma análise em seus termos formadores. O recurso a ser alocado é denominado "*search_similar_word*", e receberá a sentença completa para análise dos termos, confrontando-os a um dicionário de termos. A proposta visa realocar e substituir termos da resposta por sinônimos que não alterem seu contexto. A substituição de termos em uma sentença pode apresentar uma boa estratégia em situações onde a pergunta e sua resposta são conhecidas, e pretende-se apenas flexibilizar sua apresentação. Ao fim, a sentença construída e flexibilizada deverá receber o conhecimento provindo do modelo bayesiano. Para tanto, a sentença retorna novamente ao Orquestrador que, respeitando o modelo de composição estabelecido, alocará o recurso "*inference_node_bayesian_network*", o qual consumirá um dos métodos disponibilizados pelo projeto *Health Simulator* para realizar inferências na rede bayesiana que esteja servindo de base para a simulação em andamento.

O método disponibilizado via *Web Api* deverá retornar um percentual de incerteza, e posteriormente deverá ser traduzido por meio de uma tabela de conversão auxiliar, construída com a finalidade de tornar os resultados da rede palpáveis ao simulador. Importante ressaltar que todas as respostas são pensadas de forma a manter uma variável de referência, identificada por meio da expressão "&&tabela". Esta variável deve ser substituída pelas opções de respostas do paciente, conforme a tabela de conversão sugerida por (BEZ, 2013) ao se trabalhar com simuladores e bases de conhecimento incerto: "nunca", "quase nunca", "raramente", "poucas vezes", "algumas vezes", "a maioria das vezes", "boa parte das vezes", "sempre", "quase sempre", "não" e "sim". Experimentos foram realizados com esta abordagem em um cenário similar, aplicado a ontologias, com resultados satisfatórios ao problema apresentado (MELLO et al., 2017). Este processo de substituição é realizado pelo último recurso definido no modelo de composição, denominado "*build_answer_bayesian_percent*", responsável pela leitura da tabela de conversão e sobreposição das variáveis na resposta pelos termos convertidos. Por fim, o recurso deve retornar o resultado de sua tarefa ao Orquestrador, e este ao *Health Simulator*, que requisitou os serviços da Pythia NLG.

Por fim, a proposição deste cenário visa apresentar uma abordagem diferenciada ao processo do projeto do simulador, incorporando sua proposta de apoio à construção do raciocínio clínico, através da simulação de situações reais, a técnicas de GLN. Juntamente às simulações de cenários, será possível formular e inserir as perguntas em linguagem natural e, por meio do modelo proposto, construir a resposta à pergunta realizada.

4.5 Protótipo Implementado

O protótipo Pythia NLG foi implementado para avaliação e validação de funcionalidades. O desenvolvimento foi realizado utilizando a linguagem de programação python 3.7. Como tecnologia de comunicação optou-se por empregar o uso do protocolo AMPQ, utilizando o RabbitMQ broker – um reconhecido servidor de mensageria (Seção 2.3.2). Para comportar o cadastro de novas composições e demais configurações de recursos, optou-se pelo banco de dados relacional MySQL. Por fim, para acesso facilitado às bases léxicas *Wordnet* e *Framenet*, trabalha-se com a biblioteca *Natural Language Toolkit* (NLTK) para linguagem python, que traz uma interface para acesso às bases.

A seguir são expostos em detalhes os componentes implementados para o protótipo, bem como as etapas do desenvolvimento. Na sequência, apresenta-se a estrutura da base de dados e suas definições. Em seguida, explica-se os algoritmos integrados - externos e internos, e suas respectivas aplicações nos serviços disponíveis. Por fim, discute-se os detalhes levantados para seleção das tecnologias adotadas e as vantagens na utilização das mesmas.

4.5.1 Visão geral do protótipo

Ao iniciar o desenvolvimento, algumas decisões tomadas como foco, principalmente em pontos bem definidos quanto à comunicação entre os componentes, foram questões consideradas chave na implementação. A seguir os principais pontos considerados:

a) priorizar a possibilidade de reuso dos recursos integrados em diferentes modelos de composição, a fim de trazer uma riqueza na construção de diferentes *pipelines*;

b) proporcionar autonomia ao orquestrador frente às diferentes tarefas executadas pelos recursos;

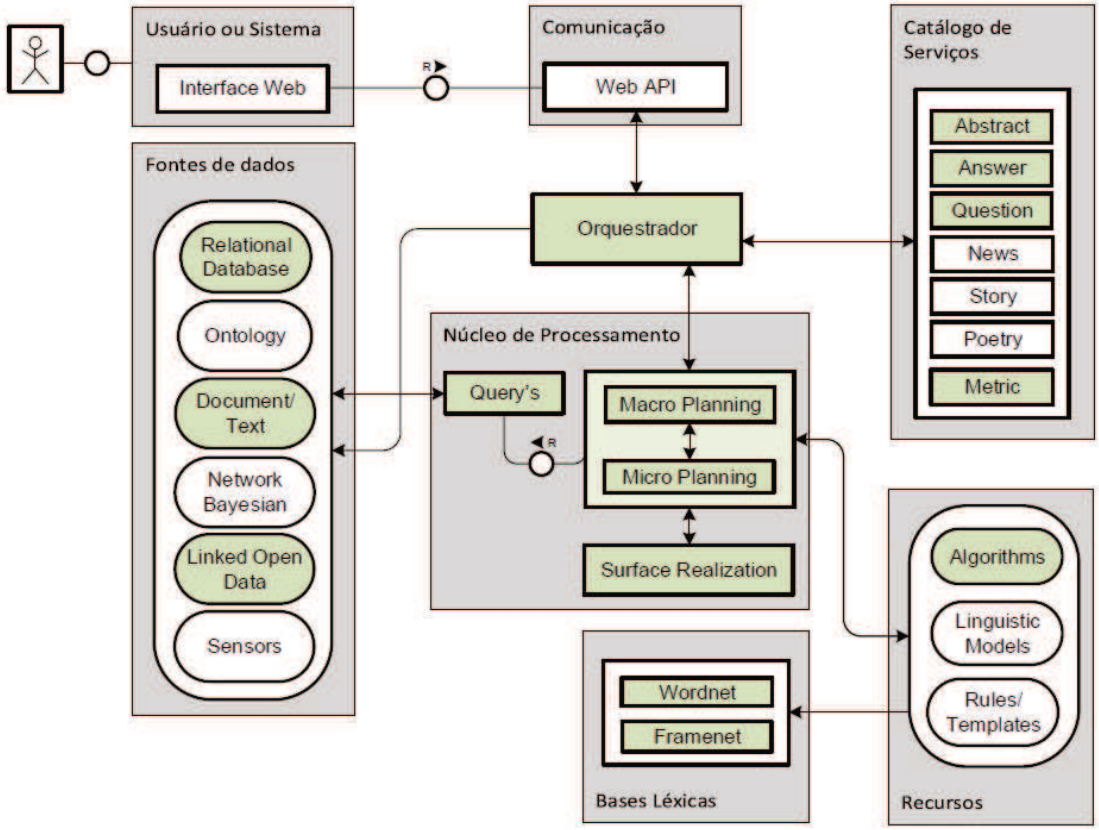
c) dar independência aos recursos perante o cliente que envia a solicitação ao orquestrador;

d) prezar pela disponibilidade do canal de solicitação de novos serviços.

O desenvolvimento do protótipo envolveu em todas as suas etapas um acompanhamento dos itens previstos no modelo - construído de forma ampla e subdividido em módulos, de acordo com as funções exercidas por estes. Desta forma, foi possível identificar a necessidade de realizar modificações ou adaptações, ou se o planejamento inicial seguia atendido na implementação.

Portanto, a seguir faz-se uma análise dos componentes implementados e suas responsabilidades, relacionando-os com o modelo proposto. Com isto, a leitura se dá de forma natural e as funcionalidades do protótipo podem ser destacadas. Na Figura 34, traz-se o modelo conceitual apresentado e em destaque estão os módulos que tiveram os componentes implementados no protótipo. Os módulos implementados foram escolhidos devido à necessidade de garantir que as funcionalidades principais do modelo estivessem representadas, de modo que pudessem ser também avaliadas as contribuições trazidas pelo modelo.

Figura 34 Modelo proposto com destaque para os componentes implementados.



Fonte: Elaborado pela autora.

4.5.2 Componentes implementados e seu funcionamento

Para fins de validação e avaliação, foram escolhidos os componentes principais para serem implementados no protótipo. Estes foram escolhidos atendo-se aos algoritmos e técnicas previamente estudados e selecionados, conforme a Seção 4.3 descreve.

O componente responsável pela recepção e atribuição das tarefas – **orquestrador**, irá, juntamente ao componente **núcleo de processamento**, processar as solicitações recebidas. O primeiro recebe a solicitação realizada pelo cliente; o segundo processa adequadamente o que foi solicitado e retorna ao **orquestrador**, para que este comunique ao destino. O destino pode ser um recurso que recebe sua tarefa, ou o usuário que solicita o serviço e recebe seu resultado.

O componente denominado **orquestrador** tem funções exclusivamente voltadas à orquestração e comunicação, ou seja, não envolve-se em processamento de solicitações, essencialmente comunica-as. Este componente implementa uma extensa lista de atividades, sobretudo comunicativas, que serão descritas a seguir:

a) **Comunicar o registro de recursos/clientes:** recebe e comunica o sucesso de registro, este feito tanto por recursos quando usuários solicitantes (clientes).

b) **Comunicar status da lista de recursos disponíveis (online):** comunica constantemente a ocorrência de recurso indisponível e/ou usuário irregular.

c) **Comunicar sucesso na leitura de composição ao usuário:** após recebida uma nova solicitação de serviço, o orquestrador informa que a mesma foi aceita com sucesso ou erro. Esta etapa significa que a leitura da composição foi realizada assim como as etapas seguintes de validação.

d) **Comunicar indisponibilidade de recursos selecionados:** a composição lida apontou irregularidades, ou a verificação nos recursos selecionados apresentou recursos indisponíveis, sem canal ativo (offline).

e) **Atribuição de tarefas aos recursos:** a execução do *pipeline* inicia a etapa de atribuição de tarefas aos recursos. Esta etapa não sobrecarrega o orquestrador, visto que este atribui uma tarefa, o recurso que foi alocado aplica seu processamento isoladamente e, durante a execução da tarefa, o orquestrador não fica no aguardo. Com esta característica, o segundo ponto chave levantado para a implementação pode ser atendido (Seção 5.1 item b). Somente ao concluir a tarefa, o recurso devolve para o canal do orquestrador, que somente então passará ao próximo recurso do *pipeline*.

f) **Comunicar o produto ao solicitante:** quando a última tarefa do *pipeline* é concluída com sucesso e o produto final retorna ao orquestrador, este comunica ao usuário solicitante o produto, e informa o fim deste serviço.

g) **Comunicar enceramento do pipeline:** após o retorno do produto final ser comunicado, o orquestrador informa o encerramento do *pipeline* ao componente núcleo de processamento.

O componente denominado **núcleo de processamento** possui diferentes etapas destinadas a atender o objetivo de integrar recursos. Este componente trabalha como a camada abaixo do **orquestrador**, onde faz-se todo o processamento necessário para definir as mensagens a serem enviadas. A seguir são descritos os detalhes das atividades implementadas e suas funções nesta camada *back-end*:

a) **Realizar o registro dos canais:** recursos e usuários (clientes) precisam realizar prévio registro quando aptos, disponíveis e/ou clientes ativos para consumo. Como usuários solicitantes de novos serviços (clientes), ou recursos, este prévio registro formaliza o canal de comunicação. A autenticação é feita através de um registro, informando um código único de identificação para o canal. Caso já esteja em uso, uma mensagem é informada imediatamente.

b) **Manter lista de recursos disponíveis (*online*):** para que sejam consumidos os recursos definidos nas composições, estes devem estar listados como disponíveis (*online*). A lista é mantida, atualizada e consultada pelo módulo central, e pode resultar de uma ação de registro ou mesmo de uma alteração de status do próprio recurso, comunicando ao estar fora.

c) **Processar a solicitação de serviço:** a solicitação é feita no padrão estabelecido (Seção 4.2.2), o orquestrador recebe a mensagem e imediatamente encaminha os dados ao componente central. Este avalia os elementos de definição padrão na

solicitação, apenas após poderá passar a etapa de leitura da composição. A comunicação decorrente é feita somente através do orquestrador.

d) **Realizar a leitura da composição de recursos:** a leitura da composição apenas será realizada após as verificações iniciais. Em seguida, é realizada a validação através da chave e código identificador da composição, por fim, o componente central faz a leitura.

e) **Verificar recursos selecionados:** a leitura da composição é realizada, portanto inicia-se o processo de analisar a disponibilidade dos recursos envolvidos. Esta etapa consiste em relacionar cada recurso presente na composição, com a lista de canais online. É possível realizar a verificação dos recursos selecionados, pois todos devem estar listados no catálogo, no momento de registro. Esta validação encerra o processo da solicitação, caso um recurso esteja indisponível – sem canal de comunicação.

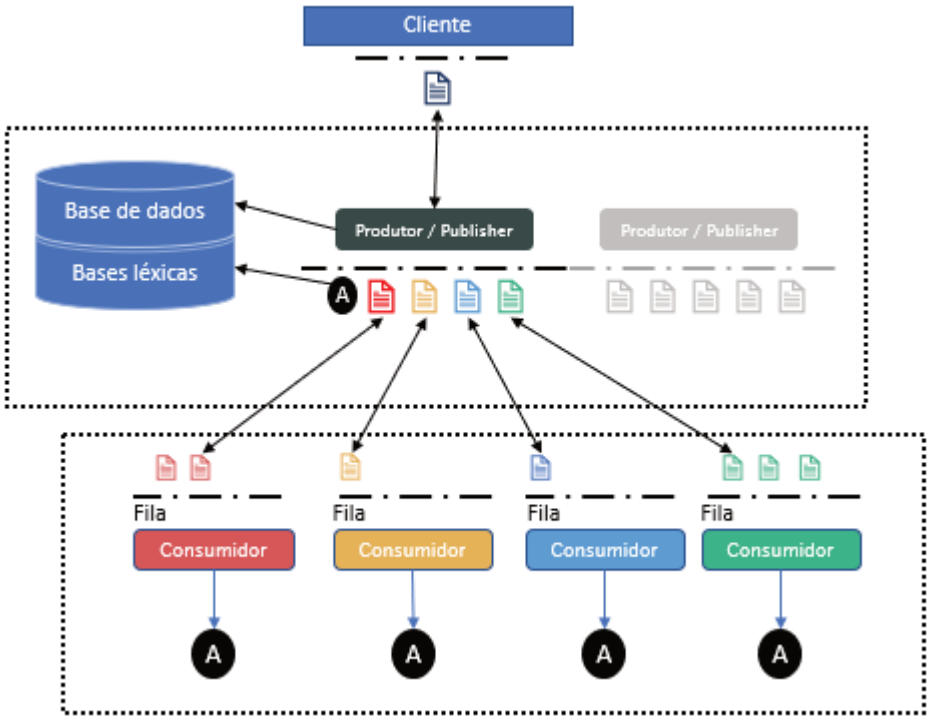
f) **Definir o pipeline:** com as verificações da composição concluídas, será definido o *pipeline* de recursos, de acordo com a composição e as configurações nela presentes. Na sequência, será informado ao orquestrador o *pipeline*, para que faça a execução e gerencie o mesmo.

g) **Encerrar o pipeline:** é recebida a comunicação de que o orquestrador finalizou o *pipeline*, e este pode ser encerrado. Após o retorno, os identificados da solicitação são armazenados e a solicitação encerrada de fato.

h) **Avaliar erros de processamento:** caso ocorra erro durante o processamento do *pipeline,* este é repassado ao **orquestrador**, que encaminha para o **núcleo de processamento**, para que seja avaliado. Caso seja problemas com a composição, o *pipeline* é encerrado.

A partir do protótipo pode-se abstrair os papéis de cada componente e relacionar adequadamente no modelo. Para facilitar a análise, definiu-se uma nomenclatura que permite compreender a implementação, conforme ilustra a Figura 35, a seguir:

Figura 35 Implementação núcleo de processamento e orquestrador.



Fonte: elaborado pela autora.

O componente que recebe a denominação **cliente** representa o **usuário** (Seção 4.1.1). Ele não consiste de uma implementação dentro do Pythia NLG, mas sim representa o **usuário solicitante.** É caracterizado pelo papel de usufruir dos serviços integrados no protótipo Pythia NLG. Esta responsabilidade lhe atribui a função de enviar a solicitação de um novo serviço no padrão estabelecido (Seção 4.2.2) e assim consumir os serviços via composição de recursos (Seção 4.2.4). A solicitação de um serviço pelo cliente é realizada diretamente via protocolo AMQP, utilizando-se da ferramenta RabbitMQ.

O componente denominado **produtor** representa o **orquestrador** (Seção 4.1.3) e **núcleo de processamento**, aqui representados como componente único, pois suas funções têm estreita relação. É importante ressaltar que o orquestrador representa a camada de comunicação, enquanto que o componente central representa a camada de processamento, o *back-end*.

O orquestrador recebe as solicitações dos clientes, o componente central possui o papel de processar os dados recebidos, para então liberar a execução e orquestração dos recursos - previamente cadastrados – para o **orquestrador**. A atribuição de tarefas às respectivas filas dos recursos é papel do orquestrador, o gerenciamento do *pipeline*. Ao solicitante (cliente) é comunicada a resposta processada quando as tarefas são concluídas conforme a composição.

Esta é uma das características que justifica o uso de um *message broker* no protótipo: a possibilidade de manter um cenário, onde o único que precisa saber origens e destinos será o componente responsável pela comunicação, os demais mantem-se auto contidos em suas tarefas. As filas servem tanto para receber como enviar dados, e o objetivo da mensagem é comunicado na própria mensagem, em forma de ação.

A responsabilidade de receber e distribuir as tarefas recai sobre o **orquestrador**, para que seja possível desacoplar os recursos e proporcionar reuso. O **núcleo de processamento** não é responsável por comunicar, apenas realizando o processamento e delegando ao orquestrador, que irá comunicar e atribuir. Deste modo, o primeiro ponto levantado como questão de implementação na (Seção 5.1 item a) pode ser atendido.

A característica de cada recurso criar, manter e subscrever sua própria fila, proporciona ao recurso independência, pois não precisa conhecer os canais de onde são feitas as diferentes solicitações. Sendo assim, sua única tarefa consiste em resolver problemas relacionados a GLN. O único componente que precisa saber origens e destinos é o **orquestrador**, tornando cada recurso independente e autônomo em sua especialidade. Neste cenário, o terceiro ponto chave levantado (Seção 5.1 item c) para a implementação pode ser atingido.

Cada componente tem ações pré-definidas que podem ser utilizadas. Na Tabela 11 são apresentadas as ações que podem ser utilizadas pelo **cliente** - **usuário solicitante** para comunicar-se com o componente **orquestrador**.

Tabela 11 Ações possíveis ao componente Cliente.

Origem	Ação	Destino	Descrição
Cliente	*register*	Produtor	Registrar canal.
Cliente	*request*	Produtor	Solicitar novo serviço.

Fonte: Elaborado pela autora.

O **cliente** tem a possibilidade de comunicar-se por duas ações com o **produtor** (orquestrador) para consumir um serviço no Pythia NLG. Para solicitar um serviço, o cliente deverá efetuar um registro prévio. A ação inicial, *"register"*, visa comunicar a abertura de um novo canal de solicitação. O registro consiste do cliente criar uma fila utilizando RabbitMQ *broker*, e definir a ela um identificador único. Na sequência, o cliente deve tornar-se um subscritor desta fila, tornando-se consumidor. Esta característica garante que, quaisquer mensagens enviadas ao canal, serão consumidas pelo cliente. Por meio deste canal criado, a fila, envia-se a mensagem inicial de registro.

Após efetuar o registro, poderá enviar apenas um tipo de ação, do tipo "*request*", trata-se efetivamente da solicitação de um serviço, nenhuma outra ação é destinada ao cliente. A fila pode ser mantida indefinidamente, enviando solicitações de novos serviços e recebendo os respectivos produtos, ou pode ser encerrada a cada serviço concluído, pois cada solicitação deve comunicar um identificador.

O componente denominado **fila**, citado anteriormente, representa o **canal de comunicação** entre os componentes. Esta é possível por meio do uso do RabbitMQ *broker,* que recebe as mensagens em seu canal, diretamente, como **produtor** (orquestrador) e deste irá direcionar para os demais componentes. Com essa característica implementada no protótipo, cada recurso, para estar disponível, deve iniciar sua fila, subscrevê-la e informar ao orquestrador uma ação de registro, somente assim será registrado na lista de recursos *online*.

O componente denominado **consumidor** representa o **recurso**, a implementação com especialidade em GLN, com características e responsabilidades aplicadas *in loco*. Os recursos podem ser implementações externas ou internas (Seção 4.3), algoritmos que disponibilizam suas funcionalidades ao protótipo. Estes por sua vez são consumidores de suas próprias filas. Com esta abordagem, é possível empilhar novas tarefas para um recurso, e o mesmo irá

consumindo-as à medida que concluir e liberar de suas tarefas. A cada conclusão, irá informar ao **produtor** (orquestrador) com o devido resultado, passando à tarefa seguinte.

Todas as interações serão mantidas essencialmente com o componente **produtor**, por esta razão possui mais opções de ação. A seu encargo fica a responsabilidade de veicular as tarefas. As atividades do produtor compreendem primordialmente a camada de comunicação, como intermediador. Entre as atividades, irá receber as solicitações de novos serviços, os produtos de cada recurso, resultados das tarefas, e gerenciamento do *pipeline* – atribuição das tarefas. Na Tabela 12 são apresentadas as ações que o **produtor** pode efetuar.

Tabela 12 Ações possíveis ao componente Produtor (orquestrador).

Origem	Ação	Destino	Descrição
Produtor	*task*	Consumidor	Atribuir tarefa ao recurso.
Produtor	*post*	Cliente	Enviar produto do serviço solicitado ao cliente.
Produtor	*Info*	Consumidor	Enviar mensagens comunicativas.
Produtor	*error*	Consumidor	Comunicar ocorrência de erros.
Produtor	*error*	Cliente	Comunicar ocorrência de erros.

Fonte: Elaborado pela autora.

A ação *"task"* trata-se da interação do **orquestrador** com o recurso indicado pelo pipeline – o recurso indicado na composição. Esta ação consiste na função do orquestrador atribuir a próxima tarefa a um recurso, conforme dita o pipeline. O recurso recebe sua tarefa, será momentaneamente alocado, e devolve o resultado ao canal do **produtor** (orquestrador).

Em nenhum momento o **produtor** fica no aguardo de respostas, apenas atribui a tarefa do *pipeline,* e segue a sua próxima atividade. Caso este recurso tenha necessidade de maior tempo de processamento por sua especialidade, o **produtor** pode trabalhar com outras solicitações e novas tarefas que entraram em sua fila. Apenas retomará contato com este recurso, quando houver retorno pelo canal.

A ação *"info"* é destinada a quaisquer mensagens comunicativas referentes à estados, sem efetivamente impactar em processamento ou atribuição de tarefas, restringindo-se a atividades de consulta, informativo de conclusão ou resposta à recepção de solicitação. A ação *"post",* por sua, vez destina-se essencialmente a comunicar o produto de uma solicitação ao **cliente**.

Um recurso tem suas atividades restritas à sua especialidade, portanto suas ações possíveis são fundamentalmente destinadas a receber uma tarefa, executá-la e responder com o produto. Assim como um **cliente**, todo **consumidor** (recurso) comunica a abertura do seu canal através da ação *"register"*, e torna-se subscritor desta. Por fim estará disponível para recebimento e processamento das tarefas. As demais ações que o consumidor pode utilizar são apresentadas a seguir, na Tabela 13.

Tabela 13 Ações possíveis ao componente Consumidor.

Origem	Ação	Destino	Descrição
Consumidor	*register*	Produtor	Registrar canal.
Consumidor	*listClients*	Produtor	Consultar os consumidores *online*.
Consumidor	*response*	Produtor	Informar conclusão de tarefa ao produtor junto com o resultado.

Consumidor	*lexicalDb*	Produtor	Consultar bases léxicas disponíveis.
Consumidor	*error*	Produtor	Comunicar ocorrência de erros.

Fonte: Elaborado pela autora.

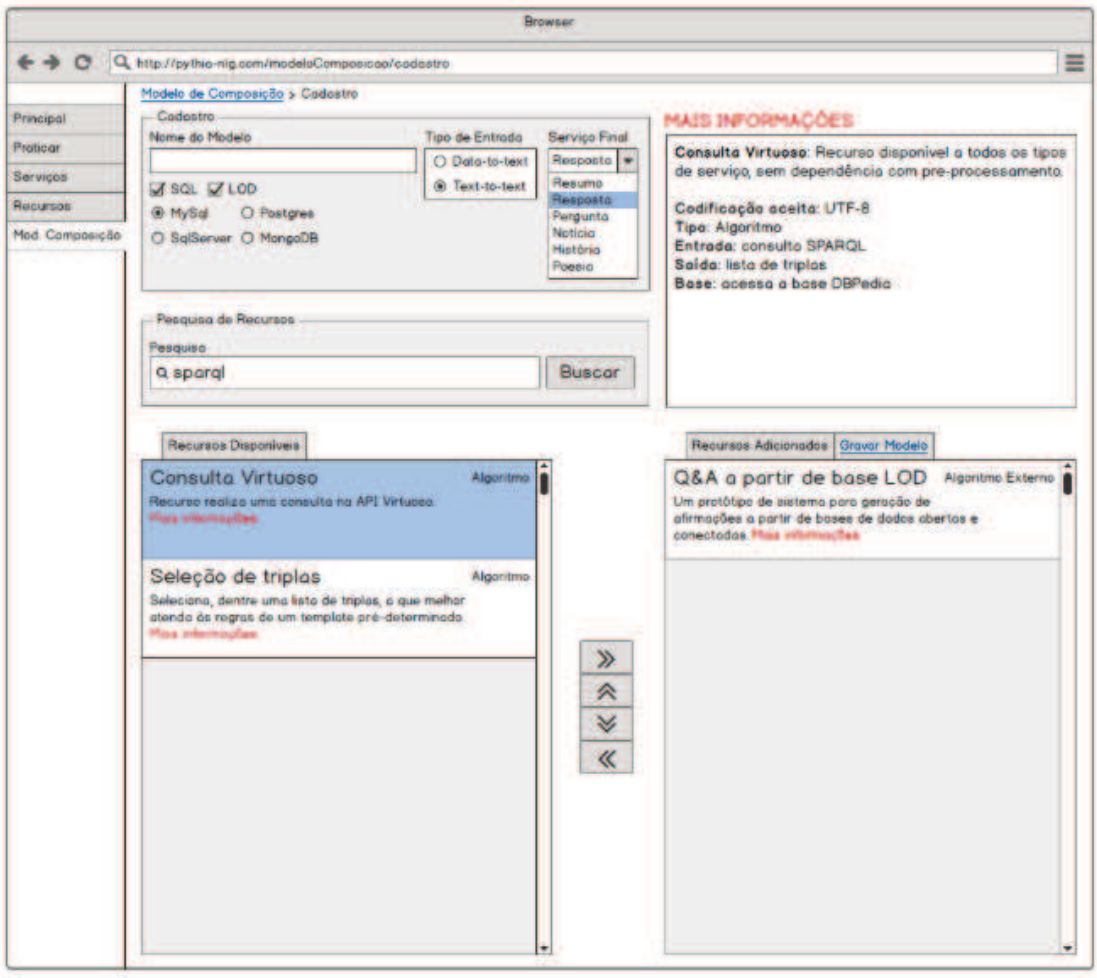
Contrário ao cliente, um recurso tem a possibilidade de uma comunicação mais completa com o **produtor**, onde poderá informar estágios de início e conclusão da tarefa. Da mesma forma, pode comunicar a ocorrência de erros durante sua execução. Esta mensagem será comunicada por meio da ação *"error"*. Por fim, a solicitação de uma consulta aos recursos léxicos é possível através da ação *"lexicalDb"*. Esta ação comunica uma solicitação de consulta diretamente ao produtor, que atribui ao recurso de consulta lexical. O recurso léxico realiza a tarefa e devolve ao **produtor**, e este comunica o produto da consulta ao recurso.

Dentre as ações possíveis do **consumidor** (recurso), está prevista a ação de envio do resultado, ou seja, a tarefa foi concluída, sem ocorrência de erros, retorna uma mensagem com ação *"response"*. Esta ação encerra a atividade recebida, e o recurso novamente está livre, e pode passar a uma nova tarefa.

Como o protótipo tem o objetivo de demonstrar as principais funcionalidades do modelo, futuramente foi prevista uma interface para lidar com o catálogo de recursos e composições disponíveis. Portanto, para este protótipo e os testes de execução realizados, algumas composições foram desenvolvidas com base nos recursos disponíveis, arquivos JSON.

Este processo foi pensado para ser realizado por usuários, pois tem-se o interesse em possibilitar a geração de composições diversas. Uma interface permitiria a geração sem necessidade de contato prévio com um desenvolver. Com isto, o usuário poderia identificar quais recursos tem interesse, e adicionar a uma nova composição, de forma interativa. Estes dados seriam salvos na base de dados do Pythia NLG, e estariam disponíveis para uso na sequência. Uma espécie de catálogo, para visualizar os detalhes de cada recurso, suas restrições e diretivas de uso descritas, assim como configurações dos tipos de entrada e saída para cada recurso. Assim como verificar as composições existentes, a Figura 36 ilustra o modelo de interface previsto para desenvolvimento futuro.

Figura 36 Cenário 4: Modelo de interface para catálogo e criação de composições.



Fonte: Elaborado pela autora.

Em seguida serão apresentadas as bases léxicas disponíveis para consulta no protótipo Pythia NLG.

4.5.3 Fontes de dados

Com a proposta de integração de algoritmos e técnicas com aplicações distintas apresenta uma diversidade de formatos a serem tratados. O protótipo Pythia NLG intermedia a comunicação entre diferentes fontes de dados, e sua construção independente entre os componentes privilegia esta interação. O Pythia NLG não foi limitado a uma especificação única para entrada, isto é feito pelos recursos disponibilizados. Portanto, a diversidade de fontes aceitas é diretamente relacionada à variedade de recursos que podem ser gradativamente inseridos no protótipo.

Para os recursos baseados na entrada URI, o protótipo trabalha com dados provenientes de fontes *Linked Open Data*. Por sua vez, os algoritmos de sumarização automática provêm diretamente de documentos de texto. Para armazenamento das configurações do protótipo, bem como a realização dos cadastros necessários, é utilizado o banco de dados relacional *MySql*.

Estas diferentes fontes de dados foram selecionadas dentre as indicadas no modelo proposto (Capítulo 4). E representam os diferentes formatos aos quais o protótipo pode trabalhar, permitindo-se independência de dados de entrada para realizar seu produto. A seguir serão abordadas as fontes de dados que o protótipo Pythia NLG está apto a intermediar. Um dos formatos é o *Linked Open Data*. Os projetos Athena! (SILVA et al., 2018) e Ensepro (ARAUJO; HENTGES; RIGO, 2018) têm em comum a fonte de dados que utilizam, o formato de URIs. Trata-se do conteúdo estruturado extraído das informações da *Wikipedia* por meio da *DBPedia*. O projeto *DbPedia* é uma popular iniciativa resultante do projeto *Linked Open Data*, que representa uma forma padronizada de publicar os dados, a fim de que estes sejam estruturados e possam ser interligados semanticamente. Um formato mais simples é o *Document Text*. A implementação dos algoritmos de sumarização automática, traz ao Pythia NLG a intermediação de documentos de texto. Estes podem ser de origens variadas, seja de revistas e jornais online, livros, sites de notícias, bem como material autoral do próprio usuário (cliente).

Para o desenvolvimento do protótipo optou-se por trabalhar com o banco de dados relacional *MySql*. Este contém as configurações necessárias ao Pythia NLG. A Tabela 14 apresenta a definição inicial como estrutura para o banco de dados:

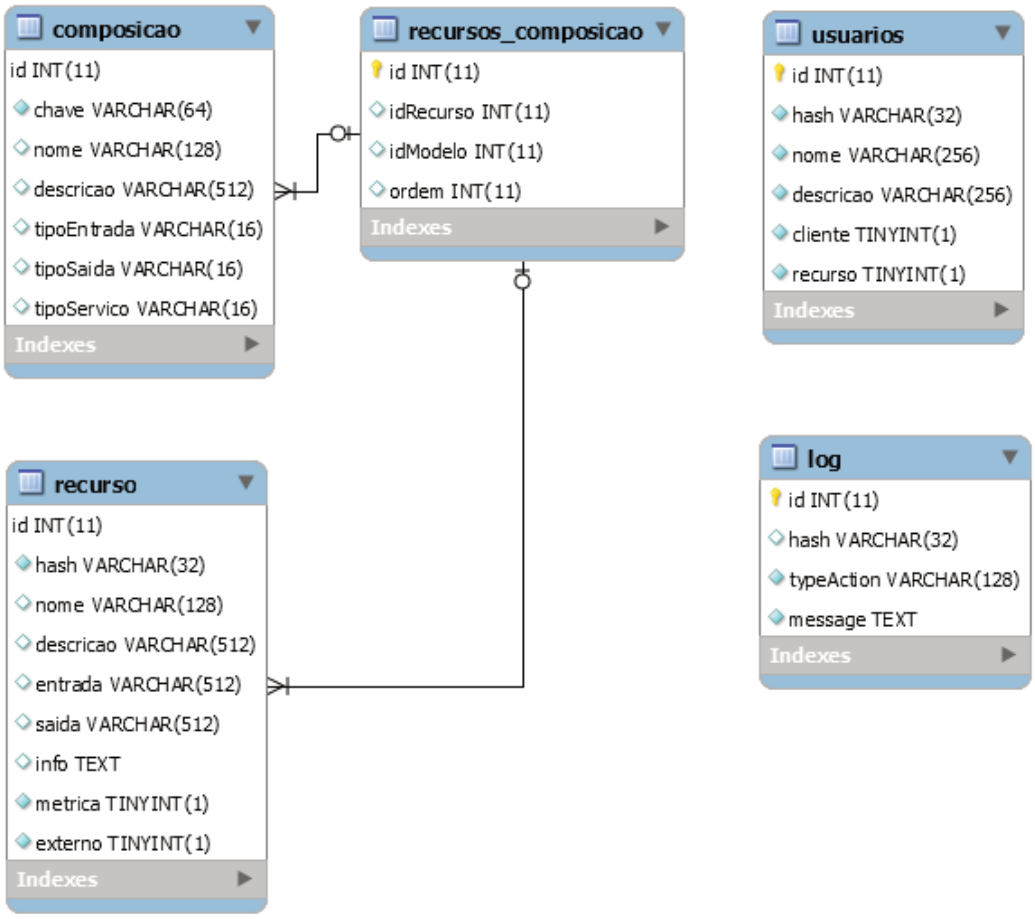
Tabela 14 Relação de tabelas criadas no banco de dados MySQL.

Nome	**Descrição**
recurso	Onde realiza-se o cadastro dos recursos e seus respectivos identificadores para posterior autenticação no RabbitMQ *broker*.
composicao	Onde realiza-se o cadastro de uma nova composição, com suas configurações de dados de entrada, saída e tipo de serviço. O cadastro completo da composição envolve a tabela **recursos_composicao.**
recursos_composicao	Onde são correlacionados os recursos às respectivas composições criadas. Uma tabela N-N para **recurso** e **composicao**.
usuarios	Onde mantem-se o registro dos identificadores para cada novo usuário (cliente ou recurso).
log	Onde registra-se o histórico de solicitações, pois deve ser o identificador único. Assim como é interessante, pois pode-se verificar os serviços frequentemente consumidos.

Fonte: Elaborado pela autora.

A Figura 37 ilustra o modelo ER do banco de dados, elaborado para atender ao Pythia NLG nas configurações necessárias ao protótipo.

Figura 37 Modelo da base de dados utilizada.



Fonte: Elaborado pela autora.

Com está análise das fontes, buscou-se demonstrar a preocupação em manter a flexibilidade de fontes de dados para os recursos. E tornar o protótipo do Pythia NLG aplicável a diferentes cenários e necessidades, independente da entrada que se dispõe para a GLN.

4.5.4 Serviços

No protótipo implementado, os serviços (Seção 4.1.8) referem-se ao produto GLN disponível para o usuário. Este pode consumir o Pythia NLG através de composições de recursos, e assim usufruir de algoritmos e técnicas diversificadas. Para a implementação foram selecionados quatro serviços, a possibilidade de gerar perguntas, respostas, sumarização automática e avaliação da qualidade por meio de métricas.

No Pythia NLG, podem existir uma infinidade de composições que apresentem o mesmo tipo de serviço, mas diverso pela relação de recursos que foram escolhidos para realiza-lo. Assim sendo, a seguir são detalhados os algoritmos e técnicas selecionados e implementados no protótipo.

4.6 Algoritmos

Os algoritmos possuem duas formas de implementação no protótipo, estes podem ser algoritmos internos ou externos (Seção 4.3). Esta condição não limita o uso do algoritmo, apenas requer uma intermediação ou uma interface para interação. Assim como a incorporação de um novo componente interno requer algumas adaptações, o mesmo ocorre para componentes externos.

Para a escolha dos algoritmos neste primeiro protótipo, buscou-se atender aos cenários definidos para avaliação (Capítulo 6). De forma a evidenciar as características do Pythia NLG, como ocorre o processo de uso por um usuário, e a interação deste com os componentes do modelo. Assim sendo, a seguir descreve-se os detalhes dos algoritmos implementados.

4.6.1 Algoritmos de Sumarização: *ConceptRank* e *LexRank*

Foram implementados no protótipo dois algoritmos para sumarização, ambos são implementações internas. O *conceptRank* trata-se de um algoritmo desenvolvido com base no trabalho produzido por (MARIA; RAMOS, 2016), e adaptado para que atenda as características do Pythia NLG. Já o *LexRank*, trata-se de uma implementação recorrente do algoritmo de sumarização automática, com uma interface disponível em diversas bibliotecas, e nesta pesquisa é utilizada a biblioteca *LexRank*[17] para linguagem python. Sendo estes componentes internos, pôde-se trabalhar os tipos de dados de entrada e flexibilizar a ambos, para combinar o mesmo formato.

Todos os recursos podem ser utilizados por intermédio de uma composição, onde define-se os dados de entrada e saída esperados para o serviço. No momento de construir uma solicitação para o Pythia NLG – recepcionada pelo orquestrador – todos os dados a serem enviados. Para este recurso, as configurações base que a ser informadas dentro do elemento **data,** na construção da composição, e podem ser conferidos na Tabela 15, a seguir:

Tabela 15 Parâmetros de entrada para o algoritmo de sumarização.

Parâmetro	Descrição
SENTENCES_COUNT	Define o número de parágrafos a sumarização.
CONTENT	Conteúdo na íntegra que será sumarizado.

Fonte: Elaborado pela autora.

4.6.2 Algoritmos de Avaliação: Métricas *Rouge* e *Bleu*

Algoritmos para avaliação automática foram disponibilizados no protótipo com o objetivo de propor recursos que forneçam métricas de qualidade. Para o protótipo foram selecionadas duas métricas tradicionalmente empregadas na avaliação de *Machine*

[17] A biblioteca *LexRank* utilizado está disponível no *PyPI (Pthon Package Index)* da linguagem de programação python. Foi desenvolvida pelo Ucraniano Luka Shostenko.

Translation (MT) ou *Automatic Summarisation* (AS): ROUGE (LIN, 2004) e BLEU (PAPINENI et al., 2002).

Os meios de avaliar uma sumarização automática envolvem, normalmente, o julgamento humano e diferentes métricas de qualidade (VICENTE et al., 2015). Portanto, no uso das métricas de qualidade implementadas, sempre haverá a presença de uma referência produzida por humanos, frequentemente denominado de padrão ouro *(gold standard).*

A métrica ROUGE-L[18] utiliza uma sentença de referência para realizar as análises da sumarização automática ou tradução. Esta avalia o número de *n-grams* da sumarização gerada por um humano contra a sumarização automática.

A métrica BLEU, por sua vez, é empregada para avaliação de MT e AS. Tem a característica de complementar a métrica ROUGE, pois avaliará o número de *n-grams* da MT ou AS contra a sumarização gerada por um humano. Assim sendo, as métricas implementadas no protótipo Pythia NLG, ROUGE e BLEU, a primeira avalia *recall*, a segunda avalia a *precision.*

Ambos os algoritmos de avaliação implementados utilizam dois parâmetros de entrada. Para a utilização destes é preciso informar no elemento **data,** quando construída a solicitação (Seção 4.2.2), os parâmetros definidos a seguir, na Tabela 16:

Tabela 16 Parâmetros de entrada para o algoritmo de avaliação ROUGE - MS e AS.

Parâmetro	Descrição
CONTENT	Conteúdo resultado da sumarização automática.
REFERENCE	Conteúdo gerado por humanos (gold standard).

Fonte: Elaborado pela autora.

4.6.3 Algoritmo Athena!

Representando um recurso externo no Pythia NLG, o projeto Athena!, desenvolvido por (SILVA et al., 2018), trata da geração de perguntas e repostas (afirmações) a partir de bases de dados ligados, utilizando *URI* como dados de entrada. A implementação utiliza uma abordagem baseada em *templates* pré-definidos para geração de novas sentenças.

Com a intenção de disponibilizar o projeto Athena! como um recurso no Pythia NLG, foi necessário implementar uma interface intermediadora, que trabalha como consumidora da Web API que o projeto propõe para de interação.

A decisão em aplicar uma interface de intermediação ocorre com o projeto Athena!, visto que ele não trabalha com a tecnologia de mensageria, RabbitMQ, e portanto não cria sua própria fila, assim como não subscreve esta fila para que possa receber e consumir as tarefas. Inviabilizando a intenção de possuir este algoritmo no catálogo de recursos do Pythia NLG.

A implementação no protótipo consiste em uma interface, que cria a fila e a subscreve – consome a fila, e esta interface aciona o uso do Athena!, recurso externo. Quando a interface receber uma tarefa, atribuída pelo **orquestrador**, enviará as requisições para a web

[18] Uma variação da métrica *ROUGE-N*, também denominada *ROUGE-L (LCS).* Não precisa de um comprimento *n-gram* predefinido, pois utiliza sempre a sequência mais longa automaticamente.

API do Athena!, para que este processe a tarefa recebida. Serve apenas como canal receptor e comunicador ao recurso específico, encaminhando os dados recebidos do **orquestrador** para **recurso** e do **recurso** para **orquestrador**.

O projeto Athena! possui configurações para a Web API disponibilizada. Para consumir via Pythia NLG, fez-se uma abstração dos tipos, perguntas ou respostas, para que seja definido na composição o produto (pergunta ou resposta). Já a informação da URI, será informada na solicitação. A Tabela 17 traz uma relação das configurações da Web API do projeto Athena!.

Tabela 17 Parâmetros de entrada para o algoritmo Athena!.

Endereço Web API	Descrição
URL (tipo)	A web API aceita a geração de perguntas ou respostas sendo, respectivamente /***question*** e /***sentence***.
QUANTITY	Quantidade de sentenças a serem geradas para uma URI.
ENTITY	Conteúdo para geração, a URI.

Fonte: Elaborado pela autora.

Para facilitar a inclusão de novos recursos ao Pythia NLG, bem como acesso a pesquisadores e desenvolvedores para realização de testes, foram desenvolvidos dois exemplos de código que poderão ser usados para implementar um cliente (como usuário solicitante) ou disponibilizando um algoritmo (recurso). Ambos os códigos fornecem um exemplo para criar o canal, subscrevê-lo, e as ações que podem ser enviadas ao Pythia NLG. As linguagens selecionadas foram Python e C#.

4.7 Considerações sobre o protótipo

Neste capítulo foi apresentado o protótipo implementado Pythia NLG. Apresentou-se a nomenclatura definida e motivação para a adoção de um *message broker* para a comunicação entre os componentes. Por meio da ferramenta RabbitMQ, cada componente possui um canal próprio, onde recebe e comunica suas mensagens.

Em seguida, foram detalhados os componentes implementados e suas funções no Pythia NLG. Bem como elencadas as possíveis ações que cada componente pode assumir e tem a responsabilidade de resolver. Foram descritos os componentes envolvidos e implementados, e como estes mantêm independência quanto aos canais de origem e destino das tarefas. O único componente responsável pela comunicação intermediadora é o componente **orquestrador**, na nomenclatura denominado **produtor**.

Uma característica importante ao adotar esta tecnologia de mensageria, é a possibilidade de tornar a aplicação um canal permanente. Neste cenário, cada componente cria sua própria fila (clientes, recursos e orquestrador). Este canal criado é mantido aberto para recepção de novas mensagens. Mesmo que a aplicação fique indisponível, o canal permanece aberto, assim sendo, ainda que o **orquestrador** esteja com problemas de processamento, seu canal permanecerá recebendo novas solicitações, e quando restabelecido, voltará a consumir sua fila. Da mesma forma ocorre aos recursos, quando indisponíveis, têm suas tarefas enfileiradas até o momento de retomar o funcionamento, ao normalizar voltará a consumir as tarefas e liberar a fila.

Na sequência, foram descritas as diferentes fontes de dados que o protótipo pode intermediar entre os recursos, evidenciando, não limitando-se a especificação de uma entrada fixa. A limitação de fontes de dados é diretamente relacionada à variedade de recursos que pode ser gradualmente acrescida ao protótipo. Por fim, apresentou-se os serviços selecionados para a implementação. Seguidos da descrição dos respetivos algoritmos e técnicas implementados para atender aos serviços. Para cada algoritmo selecionado, descreve-se os parâmetros de entrada aceitos para a construção da solicitação de um serviço.

A seguir, será apresentado em detalhes os aspectos para avaliação do protótipo implementado. Com o objetivo de evidenciar as características definidas, foram descritos cenários e as respectivas soluções executadas no protótipo Pythia NLG.

5 ASPECTOS DE AVALIAÇÃO

Neste capítulo serão descritas as iniciativas de avaliação do modelo proposto e analisados os seus resultados. A avaliação do modelo proposto foi realizada através do uso do protótipo em dois formatos diferentes. O primeiro destes formatos consiste em avaliar o protótipo sob uma perspectiva funcional, considerando cenários de uso com vistas a ilustrar as etapas e como se comporta a implementação, e como esta atenderá as diferentes solicitações de serviço. O segundo formato de avaliação implementa uma avaliação comparativa, na qual foram selecionados artigos relacionados ao tema de pesquisa e através de uma análise crítica, destacaram-se pontos significativos de melhora na implementação do protótipo em relação aos artigos, bem como, os objetivos esperados e atingidos.

5.1 Cenários de uso sobre as funcionalidades do protótipo

Nesta avaliação por cenários, são inicialmente descritas as situações práticas previstas pelo modelo proposto e que serão em seguida executadas no protótipo utilizado para a sua avaliação. As etapas de sua execução são comentadas e os pontos relevantes para a avaliação do protótipo quanto à satisfação dos objetivos propostos são comentadas. Empregar o uso de avaliação por cenários é uma prática frequentemente adotada em situações onde o sistema em desenvolvimento é resultado de interações entre atores externos, usuários ou outros sistemas (BERGMANN, 2002). Esta prática também foi adotada para avaliação de funcionalidades do protótipo no modelo de (SANTOS; DOS, 2016), usada para demonstrar objetivos atingidos, pontos positivos e possíveis adequações necessárias.

A seguir serão apresentados quatro cenários com diferentes composições de recursos envolvidos: a) um cenário para consumir composição para geração de perguntas e respostas; b) um cenário para consumir composição para geração de sumarização de documentos de texto; c) um cenário para consumir composição para avaliação de sumarização automática; por fim, d) um cenário que demonstra como se dá a criação de uma nova composição.

Para cada um dos cenários, serão apresentadas as solicitações realizadas, bem como os retornos recebidos e resultados finais de cada serviço. São demonstrados e comentados todos os passos do processo completo envolvendo os diferentes tipos de interação. Esta avaliação funcional objetiva demonstrar as vantagens ao utilizar o protótipo nas situações descritas em cada cenário. Além disso, busca avaliar o uso do protótipo e seus componentes no consumo de recursos para GLN.

5.1.1 Cenário 1 – Serviço para a geração de perguntas/respostas

Este cenário pretende demonstrar o funcionamento do protótipo quanto ao uso do serviço para geração de resposta a partir de uma entrada URI. Este cenário também pretende evidenciar como a seleção de uma composição é facilmente utilizada por um novo usuário sem necessidade de configurações complexas.

Descrição do cenário

Arthur, aluno de pós graduação, desenvolveu seu projeto de conclusão com aplicação na área de processamento e compreensão de linguagem natural. Seu projeto tem o objetivo de receber perguntas diversas em linguagem natural e identificar a que se referem – compreender seu contexto e significado da pergunta. Depois de identificar o que está sendo perguntado, procura responder através de consultas na DBPedia. Seu sistema cria as consultas SPARQL, como parte do protótipo e a resposta que tem é uma lista de URI. Ele possui bom conhecimento das tecnologias de web semântica, ontologias e PLN para realizar essas atividades, mas para complementar o resultado final, gostaria de conseguir construir uma resposta em linguagem natural, que envolve a área de GLN.

Para gerar as respostas em linguagem natural em seu projeto, Arthur decide usar uma aplicação pronta e terceirizar apenas essa tarefa. Seus dados para gerar as respostas provém de uma lista de URI, que pertence ao contexto da pergunta, mas não estão formatadas em linguagem natural para devolver ao usuário. E como resultado, espera uma resposta formatada adequadamente em linguagem natural.

Arthur decide que para atender essa demanda, utilizará o protótipo Pythia NLG, pois não gostaria de desenvolver e implementar uma solução especialmente para esta atividade.

Detalhes de execução do cenário

Para utilizar o Pythia NLG, Arthur faz uma análise das tecnologias que precisará configurar em seu ambiente para consumir os serviços. Nas especificações de uso, para comunicação, o protótipo utiliza o protocolo AMQP, através do *message broker* implementado via ferramenta RabbitMQ.

Após essa configuração inicial, Arthur precisa criar uma fila por meio do RabbitMQ, nada mais do que um canal, que se conecta ao componente **orquestrador** no protótipo Pythia NLG, e deverá tornar-se subscritor desta fila. Sendo subscritor de sua própria fila, Arthur se torna o **consumidor** do canal que acabou de criar, e assim, pode configurar para que, sempre que receber uma mensagem, automaticamente a consuma, seja uma mensagem de resultado ou outros estágios.

Sendo assim, após o canal ser criado, a primeira etapa para utilizar o Pythia NLG é comunicar uma mensagem de registro. O RabbitMQ não obriga este registro, o canal já está criado. Mas o orquestrador, numa posição de *menssage borker*, precisa conseguir identificar os canais que a ele se conectam. Portanto, nesta mensagem de registro vai comunicar a si como um canal cliente, e informar um código de identificação. O código é gerado pelo próprio Arthur, conforme a definição do formato (Seção 4.2.3). Este código único é utilizado pelo **orquestrador** para manter o controle dos canais e comunicar estágios do processamento de serviços. A solicitação de registro construída é ilustrada na Figura 38, a seguir:

Figura 38 Cenário 1: Registro de um novo canal.

```
{
    "action": "register",
    "data": {
        "hash": "consumidor",
        "name": "projeto ENSEPRO",
        "description": "serviços GLN a partir de URIs como dados de entrada",
        "data": ""
    },
    "fromExchange": "0328bb03aeb5417eacdd6f32214ac1a0"
}
```


Fonte: Elaborado pela autora.

A mensagem que Arthur comunicou, fez o registro do seu canal como cliente, e agora poderá fazer as solicitações para utilizar os serviços. O código identificador enviado como **hash** e **fromExchange** foram registrados. Com estas informações, o **orquestrador** pode comunicar os resultados e estágios que ocorram durante as solicitações ao canal correto – o cliente.

Com o registro confirmado pelo Pythia NLG, Arthur passa a procurar entre as composições disponíveis, uma que melhor se adeque a seu cenário de necessidade. Uma contém a configuração que parece lhe atender bem, pois aceita gerar respostas e afirmações a partir de uma lista de URI.

A seleção de uma composição envolve uma análise das configurações e formatos de dados que os recursos presentes na composição podem trabalhar. Estas configurações se limitam a identificar o tipo de serviço, a chave de identificação, o formato de entrada e o formato de saída. Para este cenário, Arthur precisava de uma composição com entrada de uma lista de URI, e como saída esperava formato texto. A composição que selecionou para sua solicitação é apresentada na Figura 39, a seguir:

Figura 39 Cenário 1: Composição para geração de respostas a partir de URI.


```
{
    "tipoServico":"resposta",
    "chave":"composicao.athena.respostas",
    "tipoEntrada":"uri-list",
    "tipoSaida":"text-text",
    "composicao": [
        {
            "nome": "strip list uri",
            "hash": "279e0e90fd1949368b5be3c30bf8d42e",
            "ordem": "1",
            "data": {}
        },
        {
            "nome": "athena! algorithm",
            "hash": "877cdb39a78a4fcb9194dd360fff3361",
            "ordem": "2",
            "data": {
                "tipoServico":"resposta"
            }
        }
    ]
}
```


Fonte: Elaborado pela autora.

Concluída a análise e escolha de uma composição, Arthur irá construir sua solicitação de serviço. O Pythia NLG possuía uma lista de ações possíveis para comunicação, essa varia de acordo com quem comunicará, e o que será comunicado na mensagem. Um cliente pode

comunicar dois tipos de ações, a de registro – *register*, e uma ação de solicitação de um serviço – *request*. Ambas as ações têm formato padrão para comunicar, a primeira, identificada na Figura 39, sucinta em sua forma. A segunda, uma solicitação de serviço, contém maiores configurações, pois comunicará um pedido e a composição que será utilizada para atender ao pedido.

Para construir a solicitação, Arthur já escolheu a composição que pretende usar, então o próximo passo será gerar um identificador único para esta solicitação, da mesma forma que o fez para o registro. Esta medida é para que o componente **núcleo de processamento**, que define os *pipelines*, consiga construí-los com o endereço dos canais dos recursos que receberão as tarefas. Este código também permite que mantenha o registro de solicitações realizadas e encerradas, e o mesmo possa ser feito pelo **orquestrador** ao finalizar cada um.

Este código de identificação é informado no elemento **pipelineHash**. Por fim, os demais dados da solicitação são informações de entrada preenchidos. Arthur preencheu com os dados que dispõe para gerar a resposta, e enviou a mensagem para o Pythia NLG. A solicitação construída pode ser conferida na Figura 40, a seguir:

Figura 40 Cenário 1: Solicitação de serviço para geração de resposta a partir de list-uri.


```
{
    "action": "request",
    "data": {
        "composition":"composicao.athena.respostas.json",
        "hash":"consumidor",
        "name":"projeto ensepro",
        "description":"Consumidor realiza requisicao de novo serviço.",
        "pipelineHash":"ea98e032730b49488ef612d9a8f63e04",
        "pipeline":"",
        "data": {
            "result": [
                {
                    "uri": {
                        "type": "uri",
                        "value": "http://dbpedia.org/resource/The_Lion_King_(musical)"
                    }
                },
                {
                    "uri": {
                        "type": "uri",
                        "value": "http://dbpedia.org/resource/Aida_(musical)"
                    }
                },
                {
                    "uri": {
                        "type": "uri",
                        "value": "http://dbpedia.org/resource/Mary_Todd_Lincoln"
                    }
                }
            ]
        }
    },
    "fromExchange": "0328bb03aeb5417eacdd6f32214ac1a0"
}
```


Fonte: Elaborado pela autora.

Ao enviar uma mensagem de solicitação de serviço para o Pythia NLG, Arthur recebeu uma mensagem de que a mesma foi aceita. Quando o Pythia NLG comunica este aceite de solicitação, significa que as verificações iniciais foram feitas com sucesso. Isso inclui a disponibilidade dos recursos presentes na composição, a validade do código de identificação enviado para a solicitação, bem como, o marco da definição do *pipeline*, que está concluído e comunicado ao **orquestrador**, para que gerencie sua execução. Na Figura 41, é ilustrado o resultado da solicitação enviada por Arthur:

Figura 41 Cenário 1: resultado da solicitação para geração de resposta.

```
{
    "action": "post",
    "data": {
        "hash": "consumidor",
        "pipelineHash": "ea98e032730b49488ef612d9a8f63e04",
        "pipeline": {
                "1": "279e0e90fd1949368b5be3c30bf8d42e",
                "2": "877cdb39a78a4fcb9194dd360fff3361",
                "999": "consumidor"
        },
        "name": "resposta",
        "description": "Produto da solicitação. Pipeline encerrado.",
        "data": [
            {
                "sentence": "Aida (musical)'s engine is David Henry Hwang.",
                "subject": "Aida (musical)"
            },
            {
                "sentence": "The Lion King (musical)'s engine is Irene Mecchi.",
                "subject": "The Lion King (musical)"
            },
            {
                "sentence": "Mary Todd Lincoln's death place is United States.",
                "subject": "Mary Todd Lincoln"
            }
        ]
    },
    "fromExchange": "server"
}
```


Fonte: Elaborado pela autora.

Conferindo as demais composições, Arthur verificou que existe uma que pode gerar perguntas a partir de URI, o que pode ser interessante, no seu contexto. Para solicitar um novo serviço com este resultado, ele precisa apenas trocar a composição utilizada, e o tipo de entrada permanece a mesma. A Figura 42 demonstra as diferenças ao fazer esta troca de composição utilizando o mesmo recurso:

Figura 42 Cenário 1: Alterações necessárias para usar outra composição na solicitação.


```
{
    "tipoServico":"pergunta",
    "chave":"composicao.athena.perguntas",
    "tipoEntrada":"uri-list",
    "tipoSaida":"text-text",
    "composicao": [
        {
            "nome": "strip list uri",
            "hash": "279e0e90fd1949368b5be3c30bf8d42e",
            "ordem": "1",
            "data": {}
        },
        {
            "nome": "athena! algorithm",
            "hash": "877cdb39a78a4fcb9194dd360fff3361",
            "ordem": "2",
            "data": {
                "tipoServico":"pergunta"
            }
        }
    ]
}
```


Fonte: Elaborado pela autora.

É importando ressaltar que essa alteração requer poucas modificações, pois envolve outra composição com o mesmo recurso. Assim sendo, as configurações para o tipo de entrada e o tipo de saída permanecem as mesmas da solicitação anterior. É preciso apenas modificar a composição que espera consumir e o código da solicitação, pois a intenção é utilizar o mesmo formato de entrada e manter o formato de saída. Na Figura 43 pode-se conferir quais foram as modificações na solicitação.

Figura 43 Cenário 1: Alterações na solicitação - geração de perguntas a partir de list-uri.


```
{
    "action": "request",
     "data": {
       "composition":"composicao.athena.perguntas.json",
        "hash":"consumidor",
          "name":"projeto ensepro",
          "description":"Consumidor realiza requisicao de novo serviço.",
          "pipelineHash":"687b7937a1854e799e90a1e331ee5a4c",
          "pipeline":"",
        "data": {
            "result": [
                {
                    "uri": {
                        "type": "uri",
                        "value": "http://dbpedia.org/resource/Bill_Gates"
                    }
                }
            ]
        }
    },
    "fromExchange": "0328bb03aeb5417eacdd6f32214ac1a0"
}
```


Fonte: Elaborado pela autora.

E por fim, Arthur recebe o resultado da outra composição utilizada, apenas com a diferença no tipo de sentença, agora uma pergunta. A Figura 44 ilustra o resultado, em destaque as diferenças do primeiro resultado – respostas, para o segundo – perguntas.

Figura 44 Cenário 1: Resultado da solicitação para geração de pergunta.


```
{
    "action": "post",
    "data": {
        "hash": "consumidor",
        "pipelineHash": "ea98e032730b49488ef612d9a8f63e04",
        "pipeline": {
                "1": "279e0e90fd1949368b5be3c30bf8d42e",
                "2": "877cdb39a78a4fcb9194dd360fff3361",
                "999": "consumidor"
        },
        "name": "pegunta",
        "description": "Produto da solicitação. Pipeline encerrado.",
        "data": [
            {
                "sentence": "What is the name of Bill Gates' birth place?",
                "subject": "Bill Gates"
            }
        ]
    },
    "fromExchange": "server"
}
```


Fonte: Elaborado pela autora.

Acima foram descritas as etapas para atender o cenário do Arthur, que utiliza o Pythia NLG para solicitar um serviço de geração de respostas. Neste cenário, foram utilizados

recursos interno e externo em uma mesma composição, evidenciando as características positivas que o uso de composições traz ao protótipo. Igualmente a oportunidade de apresentar como as comunicações por mensagem tornam o protótipo independente. Isso pode ser verificado no momento em que Arthur registra-se, e em seguida, envia uma ou mais solicitações pelo mesmo canal, sem precisar aguardar um resultado para realizar outra solicitação. Sendo consumidor de seu próprio canal, pode enviar diversas solicitações e saberá todos os resultados quando estes forem concluídos.

Este cenário também buscou evidenciar como as composições representam um componente positivo para este tipo de abordagem, de construção de *pipelines*, pois contém todas as definições e configurações auto contidas, sendo o único artefato necessário para indicar quais recursos serão utilizados. Por outro lado, a facilidade ao construir uma solicitação relaciona-se diretamente com as composições disponíveis. Caso não houvesse uma compatível com a necessidade de Arthur, seria necessário dedicar um tempo para sua construção.

5.1.2 Cenário 2 – Solicitação de serviço para a geração de sumarização de documento

Neste cenário será descrito o funcionamento quanto ao uso do serviço de sumarização automática de documentos (Seção 5.2.1). Este cenário tem o objetivo de evidenciar, através de uma descrição funcional, como o uso de composições facilita o consumo de diferentes recursos que realizam o mesmo serviço com qualidades diferentes.

Descrição do cenário

Bruno é gerente de uma equipe de desenvolvedores no setor de TI, e sua empresa trabalha com o desenvolvimento de ferramentas para fomentar a educação por tecnologias digitais, com público alvo infantil e juvenil. Recentemente recebeu um pedido para um novo projeto destinado para crianças e adolescentes. A ferramenta tem o objetivo de apoiar a construção da visão crítica do aluno quanto a leitura de manchetes, entrevistas, jornais e revistas, para que tenha a capacidade de ler e identificar os pontos de destaque, a fim de realizar relatórios resumidos.

No projeto, denominado Educator Pro, dentre as características, uma das funcionalidades da ferramenta será receber um documento qualquer de uma reportagem na íntegra, identificar e apontar, em um texto formatado como resumo, os pontos chave na discussão. Dado este contexto, Bruno, sabendo das diferentes habilidades de seu time, compreendeu que para este aplicativo seria necessário possuir conhecimento sobre a área de processamento de linguagem natural e geração de linguagem natural.

Na equipe de desenvolvimento atual, nenhum dos desenvolvedores supre este perfil. Portanto, para esta funcionalidade da ferramenta, a equipe decidiu utilizar o projeto Pythia NLG. Consumirá os recursos de sumarização automática (SA) na interface do aplicativo, e proporcionará uma ótima aprendizagem aos alunos, sem que seja preciso dedicar um laborioso estudo das diferentes técnicas de SA disponíveis.

Detalhes de execução do cenário

Inicialmente, Bruno fez as configurações para utilizar RabbitMQ. Com a ferramenta, cria sua fila, e a subscreve como consumidor. Esta comunicação é feita via protocolo AMQP,

o restante do registro consiste em comunicar a mensagem com os dados definidos para esta ação. A mensagem comunicará o código de identificação gerado. Este código único é utilizado pelo *message broker* para gerenciar os canais e comunicar estágios de processamento. Por fim, Bruno faz o envio da mensagem de registro do seu canal ao Pythia NLG. A Figura 45 ilustra como o formato de dados para registro é construído:

Figura 45 Cenário 2: Registro de um novo canal para Educator Pro.


```
{
    "action": "register",
    "data": {
        "hash": "87542c6801fb494c9987bd543c7a44d5",
        "name": "Educator Pro",
        "description": "serviços GLN a partir de documentos de texto como dados de entrada",
        "data": ""
    },
    "fromExchange": "8c3ebf6709d040ac89f32496d3d1bbf7"
}
```


Fonte: Elaborado pela autora.

Através do mesmo canal recém registrado é recebida a confirmação do registro. Portanto, agora o canal para utilizar as composições do Pythia NLG está liberado. Bruno pode fazer quaisquer solicitações de composições que tenha interesse. Para atender a sua necessidade, ele seleciona a composição para sumarização de documentos de texto. Analisando as configurações que esta composição utiliza, por exemplo, a entrada deve ser em texto, o que coincide com sua necessidade, pois as notícias podem ser de quaisquer veículos digitais, não atendo-se a um tipo ou formato de arquivo. Da mesma forma, a saída é configurada para texto, o texto sumarizado. Esta composição parece se encaixar bem em sua necessidade e expectativa de resultado. A composição que ele selecionou incorpora o uso do algoritmo *ConceptRank* para sumarização automática, e pode ser conferida na Figura 46, a seguir:

Figura 46 Cenário 2: Composição para sumarização de documento de texto.


```
{
    "tipoServico":"summarization",
    "chave":"composicao.summarization.conceptRank",
    "tipoEntrada":"text-text",
    "tipoSaida":"text-text",
    "composicao": [
        {
            "nome": "Text Summarization - Concept Rank",
            "hash": "e6823a7c08554e4484d4b8b732b7a40c",
            "ordem": "1",
            "data": {}
        }
    ]
}
```


Fonte: Elaborado pela autora.

Decidida a composição, Bruno parte para a construção da solicitação do serviço. As configurações não possuem muitos dados, limita-se a identificar a composição, a chave de identificação, um elemento *sentences_count* que define o tamanho, em parágrafos, para o resultado da sumarização, e o texto na íntegra. Na sequência, Bruno envia a solicitação ao Pythia NLG com as configurações finalizadas. A Figura 47 ilustra a solicitação construída.

Figura 47 Cenário 2: Solicitação de serviço para sumarização documento de texto.

```
{
    "action": "request",
    "data": {
        "composition":"composicao.summarization.conceptRank.json",
        "hash":"87542c6801fb494c9987bd543c7a44d5",
        "name":"Educator Pro",
        "description":"Usuário Educator Pro - solicitação de novo serviço.",
        "pipelineHash":"9f5619f41e31443093a7dfccc9b45652",
        "pipeline":"",
        "data": {
            "sentences_count":"2",
            "content": ["TEXTO NA ÍNTEGRA"],

        }
    },
    "fromExchange": "8c3ebf6709d040ac89f32496d3d1bbf7"
}
```


Fonte: elaborado pela autora.

E por fim, Bruno recebe a mensagem confirmando que sua solicitação foi aceita e será processada. Pouco tempo depois recebe o resultado do serviço pelo canal registrado. Assim como ele havia definido na solicitação, o número de parágrafos para o resultado da sumarização foram dois, e está conforme o recebido. A Figura 48 ilustra o resultado de sua solicitação utilizando a composição para gerar sumarização de texto.

Figura 48 Cenário 2: Resultado da solicitaçao para sumarização de texto.


```
{
    "action": "post",
    "data": {
        "hash": "87542c6801fb494c9987bd543c7a44d5",
        "pipelineHash": "1f5619f41e31442093a7dfccc9b45652",
        "pipeline": {
            "1": "e6823a7c08554e4484d4b8b732b7a40c",
            "999": "87542c6801fb494c9987bd543c7a44d5"
        },
        "name": "Text Summarization - Concept Rank",
        "description": "Sumarização de documentos de texto.",
        "data": ["
                    Brasil vive uma mudança pendular radical na
                presidência com a chegada de Jair Messias
                Bolsonaro, um militar da reserva, que toma posse no
                primeiro dia do novo ano.
                    Agora, Bolsonaro põe o Brasil na frente do
                espelho e da guinada direitista que marca a política
                internacional em alguns países.
            "]
    },
    "fromExchange": "server"
}
```


Fonte: Elaborado pela autora.

No entanto, a sumarização do texto apresentou alguns dados que não lhe interessavam, e percebeu que outros dados poderiam ter sido considerados. O algoritmo *ConceptRank* defende que seus resultados são mais adequados para entradas com menor volume. Como haviam mais composições com o mesmo tipo de serviço, sumarização automática, Bruno constrói outra solicitação para verificar esta outra possibilidade de resultado, utilizando o

algoritmo *BetaRank*. A composição possui as mesmas configurações, com exceção do recurso que realiza a tarefa de sumarização. A Figura 49 ilustra as diferenças da nova composição escolhida por Bruno.

Figura 49 Cenário 2: elementos que alteraram de uma composição de sumarização para outra.


```
{
    "tipoServico":"summarization",
    "chave":"composition.summarization.betaRank",
    "tipoEntrada":"text-text",
    "tipoSaida":"text-text",
    "composicao": [
        {
            "nome": "Text Summarization - Beta Rank",
            "hash": "6bf7fe8db81c487f8808be4dcac72edf",
            "ordem": "1",
            "data": {}
        }
    ]
}
```


Fonte: Elaborado pela autora.

Como as configurações de entrada e saída permanecem as mesmas para a nova composição escolhida, bem como o tipo de serviço, para modificar sua solicitação, Bruno fez apenas algumas alterações. A nova solicitação é ilustrada na Figura 50, a seguir:

Figura 50 Cenário 2: elementos que alteraram em na solicitação ao mudar a composição de sumarização.


```
{
    "action": "request",
    "data": {
        "composition":"composicao.summarization.betaRank.json",
        "hash":"87542c6801fb494c9987bd543c7a44d5",
        "name":"Educator Pro",
        "description":"Usuário Educator Pro - solicitação de novo serviço.",
        "pipelineHash":"699e1578c689488cbe7abcf27326ebfa",
        "pipeline":"",
        "data": {
            "sentences_count":"2",
            "content": ["TEXTO NA ÍNTEGRA"]
        }
    },
    "fromExchange": "8c3ebf6709d040ac89f32496d3d1bbf7"
}
```


Fonte: Elaborado pela autora.

Ao enviar a mensagem, novamente recebeu o retorno que a solicitação foi aceita para processamento e, em seguida, o resultado. Como esperado, houve algumas diferenças entre os resultados das composições, e Bruno decide-se por utilizar em sua ferramenta a composição com recurso *ConceptRank*. O resultado da solicitação para a composição que utiliza o recurso *BetaRank* é ilustrada na Figura 51, a seguir:

Figura 51 Cenário 2: elementos que alteraram nos resultados recebidos ao mudar a composição de sumarização.

```
{
    "action": "post",
    "data": {
        "composition": "composicao.summarization.betaRank.json",
        "hash": "consumidor",
        "name": "Text Summarization - Beta Rank",
        "description": "Sumariza\u00e7\u00e3o de documento.",
        "pipelineHash": "699e1578c689488cbe7abcf27326ebfa",
        "pipeline": {
            "2": "6bf7fe8db81c487f8808be4dcac72edf",
            "999": "consumidor"
        },
        "data": {
            "sentences_count": "2",
            "content": ["
                    Brasil vive uma mudança pendular radical na
                presidência com a chegada de Jair Messias Bolsonaro,
                um militar da reserva, que toma posse no primeiro dia
                do novo ano.
                    Um deles, o general da reserva Carlos Alberto
                Santos Cruz, vai ocupar o cargo de ministro da
                Secretaria de Governo, e dividir com outro ministro,
                Onyx Lorenzoni, um civil, o poder de articulação com
                o Congresso, o que representa um maior controle das
                negociações com os parlamentares.
            "],
            "reference": []
        }
    },
    "fromExchange": "server"
}
```


Fonte: Elaborado pela autora.

5.1.3 Cenário 3 – Solicitação de serviço de avaliação – métrica para sumarização automática

Neste cenário é apresentada a descrição de como faz-se uso dos recursos de avaliação de sumarização automática de documentos de texto. Este cenário visa demonstrar que a possibilidade de aplicar diversificadas métricas a um mesmo conteúdo é uma ferramenta positiva para análise de resultados, pois tem-se uma visão mais ampla da qualidade. Através de uma descrição funcional, será feito uso de composições que definem o consumo de diferentes recursos, que resultam em um mesmo serviço, a avaliação, no entanto, com características diferenciadas quanto aos itens analisados.

Descrição do cenário

Bruno enfim conseguiu implementar a comunicação da ferramenta Educator Pro com o Pythia NLG, e assim conseguiu fazer uso de sumarização como gostaria. Porém, antes de realizar a entrega do produto, decidiu fazer algumas avaliações com as sumarizações que estavam sendo geradas automaticamente, a fim de colocar como vantagem de seu software, e ter parâmetros reais para realizar análises.

Além de gerar a sumarização, é possível analisar qual a qualidade do produto que está sendo gerado, assim como, comparar a uma outra sumarização, feita por uma pessoa. Isso possibilita que, inclusive, no futuro, o Educator Pro possa dar pequenos *feedbacks* aos resumos feitos pelos alunos.

Para isso, Bruno procura uma composição que faça uso de métricas para avaliar produtos de sumarização. As configurações de entrada que possui hoje são o texto de referência, que ele mesmo fez, e o resultado de algumas consultas que fez ao Pythia NLG para gerar as sumarizações.

Detalhes de execução do cenário

Inicialmente, Bruno fez as configurações para mudar o uso da composição, e não será necessário realizar outro registro do canal, pois sendo o mesmo cliente, o **orquestrador** mantém o código identificador do canal até que este seja reiniciado.

Dentre as composições de métricas, Bruno identificou a que gostaria de utilizar, que faz a relação entre texto referência e texto produto de SM, que conta com a implementação do algoritmo de avaliação Rouge (Seção 5.2.2). A composição selecionada pode ser conferida na Figura 52, a seguir:

Figura 52 Cenário 3: composição para avaliação de sumarização automática.

```
{
    "tipoServico":"metric",
    "chave":"composicao.metric.rouge",
    "tipoEntrada":"text-text",
    "tipoSaida":"metric-data",
    "composicao": [
        {
            "nome": "Metric Summarization ROUGE",
            "hash": "f089cc74950840ca8cf494778cba9510",
            "ordem": "1",
            "data": {}
        }
    ]
}
```

Fonte: Elaborado pela autora.

No entanto, a construção da solicitação requer algumas informações, como a definição do texto que será referência para a análise de qualidade, no elemento **reference**. E para informar a sumarização que gerou através do Pythia NLG anteriormente, deve usar o elemento **content**. Por fim, Bruno faz o envio da mensagem para utilizar esta nova composição no Pythia NLG. A Figura 53 ilustra o formato final da solicitação:

Figura 53 Cenário 3: solicitação para avaliação de sumarização automática.

```
{
    "action": "request",
    "data": {
        "composition":"composicao.metric.rouge.json",
        "hash":"consumidor",
        "name":"Educator Pro - metrica de sumarizaçao",
        "description":"Usuário Educator Pro - solicitação de novo serviço.",
        "pipelineHash":"4cb7d049b0dc4f4d8b3819385406cf07",
        "pipeline":"",
        "data": {
            "content": ["TEXTO JÁ RESUMIDO POR OUTRA FERRAMENTA"],
            "reference":["TEXTO DE REFERÊNCIA PARA A METRICA"]
        }
    },
    "fromExchange": "8c3ebf6709d040ac89f32496d3d1bbf7"
}
```

Fonte: Elaborado pela autora.

O componente **orquestrador** recebe as solicitações, e este encaminha os dados recebidos para o componente central, **núcleo de processamento**, responsável por verificar se os recursos envolvidos na composição estão disponíveis no momento, e caso estejam, definir o *pipeline*, a partir dos identificadores dos canais para onde serão comunicadas as tarefas. Com o *pipeline* definido, o **orquestrador** recebe a confirmação do componente central, que pode assumir a execução, e passa a atribuir as diferentes tarefas aos recursos, na ordem pré-definida na composição, aos seus respectivos canais.

Quando o processamento é concluído, o último canal presente no *pipeline* é o destino, conhecido apenas pelo orquestrador, e para este é enviado o resultado da composição que foi processada. Este resultado é, portanto, enviado diretamente ao canal do cliente que fez a solicitação, não havendo a necessidade de o componente central intermediar, visto que as atividades de comunicação pertencem ao orquestrador.

Assim sendo, Bruno recebeu o primeiro resultado de sua avaliação para sumarização. A Figura 54 ilustra como é apresentado o resultado final para o serviço solicitado:

Figura 54 Cenário 3: resultado da avaliação da sumarização com composição para uso de métrica.


```
{
    "action": "post",
    "data": {
        "composition": "composicao.metric.rouge.json",
        "hash": "consumidor",
        "name": "Metric Summarization ROUGE",
        "description": "Metrica para avaliação de Sumarizaçao automática.",
        "pipelineHash": "4cb7d049b0dc4f4d8b38s9385406cf07",
        "pipeline": {
            "1": "f089cc74950840ca8cf494778cba9510",
            "999": "consumidor"
        },
        "data": {
            "rouge_1_recall": 0.2122,
            "rouge_1_precision": 1.0,
            "rouge_1_f_score": 0.35011,
            "rouge_2_recall": 0.20328,
            "rouge_2_precision": 0.96121,
            "rouge_2_f_score": 0.33559,
            "rouge_3_recall": 0.16788,
            "rouge_3_precision": 0.79654,
            "rouge_3_f_score": 0.27731,
            "rouge_4_recall": 0.14064,
            "rouge_4_precision": 0.66957,
            "rouge_4_f_score": 0.23245,
            "rouge_su4_recall": 0.20876,
            "rouge_su4_precision": 0.99276,
            "rouge_su4_f_score": 0.34498
        }
    },
    "fromExchange": "server"
}
```


5.1.4 Cenário 4 – Inclusão de uma nova composição no protótipo

Este cenário tem o objetivo de demonstrar como funciona o processo de criar uma nova composição de recursos para utilizar no Pythia NLG. Para isto, será apresentada a

descrição de um cenário hipotético, fazendo-se uso dos recursos de avaliação e sumarização automática de documentos já implementados.

Descrição do Cenário

Bruno, antigo usuário do Pythia NLG, desenvolveu o projeto Educator Pro, o qual gera pequenas sumarizações de documentos de texto, para ajudar no ensino de alunos. Agora que o projeto foi entregue, ele tem interesse em explorar melhores formas de usar a recém descoberta de composições de avaliação para outras funcionalidades do projeto. Pretende usar a avaliação de uma sumarização, mas sem a necessidade de fazer isso em duas solicitações diferentes. Com as composições disponíveis, primeiro é necessário fazer uma solicitação com o uso da composição para sumarização automática, para somente então realizar uma segunda solicitação e utilizar a composição para avaliação do texto sumarizado.

Com isto, Bruno recordou que poderia construir uma composição que tenha esse formato, aceite na configuração de entrada o texto na íntegra (texto completo), possa gerar o texto com o recurso de sumarização e, logo em seguida, aplicar o recurso de avaliação. Assim, o processo seria completo, podendo otimizar seu tempo na construção das solicitações em que precise avaliar textos dos alunos.

Detalhes de execução do cenário

A construção de uma composição é composta por duas etapas, primeiro é necessário definir quais os tipos de entrada e saída são esperados. A etapa seguinte consiste em avaliar os recursos cadastrados no Pythia NLG, e suas definições de entrada e saída. Essas definições precisam coincidir com as trocas entre orquestrador – recurso, e recurso – orquestrador, mas principalmente, precisam ser compatíveis entre recurso – recurso na ordem prevista na composição. A Figura 55 ilustra como os tipos de cada recurso devem ser vistos ao construir uma composição.

Figura 55 Cenário 4: primeira etapa para construção de uma composição de recursos.

Fonte: Elaborado pela autora.

Para visualizar estas informações, que são necessárias para a criação de uma nova composição, está previsto o desenvolvimento de uma interface, onde serão publicados os diferentes recursos disponíveis e composições desenvolvidas. No momento, a construção de uma composição baseia-se apenas nos dados que tem disponível na base de dados, sem uma interface de visualização. Bruno tem as informações de cada recurso, definições e parâmetros que são necessários, assim como as configurações de entrada e saída. Observando estes detalhes dos recursos, Bruno selecionou os que pretende usar na composição, atentando a ordem que definirá na composição. Com estes detalhes em mãos, Bruno elaborou sua composição, ilustrada na Figura 56 a seguir:

Figura 56 Cenário 4: Criando nova composição para sumarização automática e aplicação de métrica.

```
{
    "tipoServico":"metric",
    "chave":"composicao.metric.rouge.summarization",
    "tipoEntrada":"text-text",
    "tipoSaida":"metric-data",
    "composicao": [
        {
            "nome": "Text Summarization - Concept Rank",
            "hash": "e6823a7c08554e4484d4b8b732b7a40c",
            "ordem": "1",
            "data": {}
        },
        {
            "nome": "Metric Summarization ROUGE L",
            "hash": "f089cc74950840ca8cf494778cba9510",
            "ordem": "2",
            "data": {}
        }
    ]
}
```


Fonte: Elaborado pela autora.

Com a composição finalizada, Bruno atem-se a elaborar a solicitação que utilize esta nova configuração. O processo de criação da solicitação é relacionado às informações que inseriu na sua composição, pois os dados que enviará para processamento devem combinar com aqueles recursos que selecionou. A Figura 57 ilustra este modelo de solicitação desenvolvido:

Figura 57 Cenário 4: solicitação construída para a nova composição.


```
{
    "action": "request",
    "data": {
        "composition":"composicao.metric.rouge.summarization.json",
        "hash":"consumidor",
        "name":"Educator Pro - metrica de sumarizaçao",
        "description":"Usuário Educator Pro - solicitação de novo serviço.",
        "pipelineHash":"9d035f94a65c4dc6a00664c6325a7728",
        "pipeline":"",
        "data": {
            "sentences_count":"2",
            "content": ["TEXTO NA ÍNTEGRA "],
            "reference":["TEXTO DE REFERÊNCIA PARA A METRICA"]
        }
    },
    "fromExchange": "8c3ebf6709d040ac89f32496d3d1bbf7"
}
```


Fonte: Elaborado pela autora.

É importante ressaltar que para esta solicitação, Bruno precisou enviar, junto na solicitação, um elemento **data**, onde o **content** refere-se ao documento de texto na íntegra, aquele que pretende gerar a sumarização, e logo abaixo, no elemento **reference**, informou o texto sumarizado que ele gerou para avaliar. Assim sendo, o recurso de sumarização, em sua definição apenas requer que seja informado o elemento **content,** mas na ocasião do recurso seguinte da composição definir que precisa do elemento **reference**. Essa informação precisa ser enviada pelo usuário, visto que ambos os textos fazem parte das informações de entrada requeridas pelos recursos envolvidos.

Para poder realizar o envio, Bruno precisa novamente criar sua fila, e registrar seu canal junto ao **orquestrador**, pois havia encerrado a fila quando finalizou. Agora seu canal possui outro código identificador, portanto, isso deve ser passado à solicitação antes de enviar a mensagem. Caso não faça novamente o registro, receberá do **orquestrador** uma mensagem informando que a ação de solicitação é inválida, pois o canal informado ainda não foi registrado para receber mensagens.

Após realizar o registro, e receber a confirmação, envia a solicitação com a nova composição. Bruno recebe a comunicação de que sua solicitação fora aceita, o que significa que os recursos que selecionou estão disponíveis, e o pipeline não apresentou problemas com as configurações iniciais. O orquestrador, por fim, devolve o resultado da solicitação, como previsto, o formato corresponde ao resultado da composição que utiliza o recurso para a métrica.

Portanto, nesta composição, o **orquestrador** recebeu a solicitação, encaminhou a tarefa ao componente **núcleo de processamento**, que verificou e avaliou os recursos envolvidos, liberando ao orquestrador que iniciasse a execução do *pipeline* definido. Primeiramente, o **orquestrador** atribuiu a tarefa ao **recurso** de sumarização, e este resultado lhe foi devolvido. Este resultado foi dado como entrada para o recurso seguinte da composição, definido como a métrica, que recebeu e gerou seu próprio resultado.

Neste cenário, a sua composição fez como esperado, ao enviar um texto na íntegra, o Pythia NLG perpassou o *pipeline* definido, e pôde gerar a avaliação do texto sumarizado e lhe devolver o resultado da métrica. A Figura 58 apresenta o resultado após a execução da composição desenvolvida:

Figura 58 Cenário 4: resultado da composição construída - recurso de sumarização e avaliação.


```
{
    "action": "post",
    "data": {
        "composition": "composicao.metric.rouge.summarization.json",
        "hash": "consumidor",
        "name": "Metric Summarization ROUGE",
        "description": "Metrica de Avaliação de Sumarização automá1tica.",
        "pipelineHash": "9d035f94a65c4dc6a00664c6325a7728",
        "pipeline": {
            "1": "e6823a7c08554e4484d4b8b732b7a40c",
            "2": "f089cc74950840ca8cf494778cba9510",
            "999": "consumidor"
        },
        "data": {
            "rouge_1_recall": 0.2122,
            "rouge_1_precision": 1.0,
            "rouge_1_f_score": 0.35011,
            "rouge_2_recall": 0.20328,
            "rouge_2_precision": 0.96121,
            "rouge_2_f_score": 0.33559,
            "rouge_3_recall": 0.16788,
            "rouge_3_precision": 0.79654,
            "rouge_3_f_score": 0.27731,
            "rouge_4_recall": 0.14064,
            "rouge_4_precision": 0.66957,
            "rouge_4_f_score": 0.23245,
            "rouge_su4_recall": 0.20876,
            "rouge_su4_precision": 0.99276,
            "rouge_su4_f_score": 0.34498
        }
    },
    "fromExchange": "server"
}
```


Fonte: Elaborado pela autora.

Por meio do uso de cenários é possível apresentar e avaliar as características funcionais do protótipo Pythia NLG. Foram apresentadas as estratégias adotadas para intermediar a comunicação independente entre os componentes implementados, bem como o formato de dados definido para a composição de recursos. Evidenciou-se, por meio das descrições da execução, aspectos do funcionamento do protótipo quanto a adaptação a novas tarefas, criação de novas composições e flexibilidade para registro e solicitação de serviços.

O uso do protótipo Pythia NLG proporciona uma ferramenta com potencial para exploração de novos experimentos para a área de GLN. Da mesma forma, o protótipo desenvolvido consiste em um ambiente que viabiliza um cenário para implementação e disponibilização de novos algoritmos e técnicas de forma a reutilizar e compartilhar tarefas GLN.

A seguir serão detalhadas algumas características do protótipo implementado frente às propostas semelhantes e trabalhos analisados como base para o desenvolvimento.

5.2 Comparação com ferramentas existentes

O protótipo implementado, Pythia NLG, propõe integrar diferentes tarefas de geração de linguagem natural. Com o objetivo de proporcionar uma solução para integração e reuso das tarefas, algoritmos e técnicas e, por fim, facilitar a experimentação de novas aplicações GLN e PLN complexas. Com este cenário, a fim de avaliar as contribuições do protótipo implementado, faz-se a seguir uma análise comparativa quanto aos aspectos de implementação adotados pelos trabalhos relacionados (Seção 3.2) com a oportunidade de pesquisa e o protótipo Pythia NLG.

A implementação do protótipo Pythia NLG, como tecnologia de comunicação, adotou-se a ferramenta RabbitMQ *broker*, a fim de focar-se primordialmente na possibilidade de haver um estilo unidirecional de interação. As diferentes tarefas, recursos e aplicativos podem interagir entre si, de maneira assíncrona, por meio da troca de mensagens, onde há a presença de um *broker*, no papel de intermediário nesta comunicação.

A proposta de utilizar este tipo de comunicação possui como objetivo permitir o controle total ao arquiteto e aquele que utiliza, distanciando-se de configurações via código. Sendo assim, o canal para servidor, *message broker*, é aberto. Um novo canal é aberto e registrado por cada novo cliente junto ao servidor, a fim de solicitar um serviço (Seção 4.1.8). Da mesma forma, o registro de um novo recurso GLN é disponibilizado em um cadastro prévio, este receberá um código de autenticação destinado às composições que serão cadastradas futuramente. Com este registro prévio finalizado, o próprio recurso (tarefa GLN) cria seu canal, e registra-se como recurso disponível. As configurações de entrada e saída são definidas na composição. Portanto, os canais abertos de clientes e recursos disponíveis, a interação dá-se inteiramente por meio do intermediador – *broker,* denominado **orquestrador** (Seção 4.1.3). A camada de comunicação que permite uma interação assíncrona e com baixo acoplamento é composta pela implementação de AMQP e RabbitMQ – *message broker*. A escolha por estas tecnologias garante a interoperabilidade perante as diferentes linguagens de implementação das tarefas GLN, ao admitir uma interação exclusivamente por meio da troca de mensagens.

Análogo a este formato, o projeto LPMN[19] (Seção 3.2.1) desenvolveu uma linguagem formal para orquestração para definição de um *workflow* de recursos. Como tecnologia para sua *engine* responsável pela orquestração, utiliza a linguagem de programação python, em conjunto ao AMQP e *RabbitMQ* broker para comunicação. Este, no entanto, em sua implementação da mesma ferramenta, possui características diferenciadas. A decisão para definir sua camada de comunicação envolveu as necessidades de tarefas de mineração de texto. Em seu cenário, já possuía ambientes virtuais distintos com as tarefas configuradas e funcionando separadamente. Suas tarefas apresentavam implementações em linguagens e plataformas distintas, o que dificulta e justifica sua necessidade de integração. Para solucionar este cenário, na rede onde estão configurados os ambientes virtuais, optou-se pela interação através de chamadas RPC (disponíveis na ferramenta RabbitMQ) por meio da qual o LPMN *engine* executa a orquestração dos recursos na sequência formalizada na linguagem LPMN. A abordagem adotada permite que todas as tarefas sejam integralmente utilizadas, por meio do LPMN *engine* e linguagem para formalizada do *workflow*. Diferentemente do protótipo implementado Pythia NLG, onde optou-se pelo uso integral de canais e filas, a fim dos clientes (usuários) e recursos (tarefas) tornarem-se subscritores de suas próprias filas de mensagens, fazendo-os independentes do **orquestrador**.

Em contrapartida, para definição das composições no Pythia NLG, definiu-se o uso de um modelo de composição, em formato JSON. O formato permite determinar a sequência, bem como o tipo de serviço, a entrada e saída dos recursos são previamente cadastradas aos recursos. Quando estes são consumidos, aguardam o formato de entrada pré-estabelecido. O resultado é retornado ao orquestrador, que encaminha ao próximo recurso, ou entrada ao cliente, também em formato JSON. O Protótipo desenvolvido atua diferentemente da proposta do projeto LPMN, na qual existe uma linguagem formal destinada à orquestração e na linguagem define-se todas as configurações. As diferenças vão desde o tipo de entrada, processo de tratamento dos dados, assim como local onde encontra-se as informações de entrada e destino do produto. Sendo que o LPMN trata de tarefas de mineração de texto, a entrada de dados assim como saída caracteriza-se pelo extenso volume, apresentando, portanto, justificativa para esta decisão para *input* e *output*.

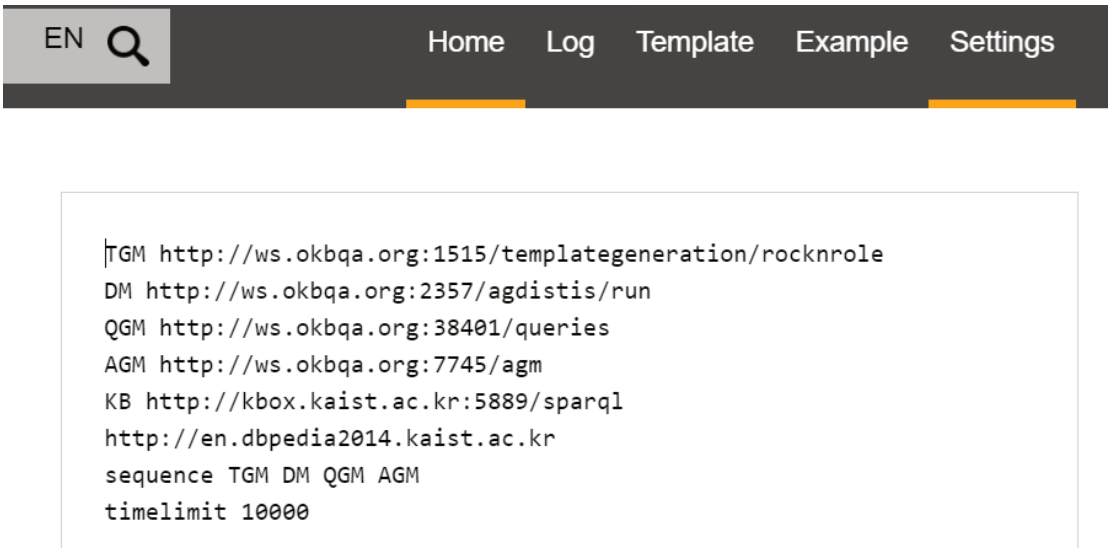
Por outro lado, o projeto OKBQA[20], apresenta uma abordagem diferenciada para integração de técnicas. Utiliza uma alternativa com a implementação de REST API. Propõe a integração de tarefas de Perguntas e Resposta da comunidade OKBQA por meio de *worflows* executados por chamadas REST API, que são previamente cadastrados no repositório *online*[21] do projeto. Neste repositório, cada desenvolvedor é responsável por detalhar as informações de entrada e saída de seu algoritmo. Isto permite que o utilizador do módulo de execução possa construir seu *workflow* adequadamente. Por tratar-se de tarefas de domínio Q&A, o modelo de *workflow* é inicialmente definido dentre as cinco diferentes categorias de tarefas, TGM, DM, QGM, AGM e CM. São prefixos para o modelo, seguidos a este são informados os URIs literais para chamada das tarefas. Por fim, é executado o *workflow* de tarefas, por meio do uso de REST API. A Figura 59 apresenta o modelo demo de *worflow* que define-se no OKBQA.

Figura 59 Exemplo de workflow construído para projeto OKBQA em uma versão demo.

[19] *Language Processing Modelling Notation – Orchestration of NLP Microservices.*

[20] *OKBQA Framework for collaboration on developing natural language question answering systems.*

[21] http://repository.okbqa.org/



Fonte: Adaptado a partir de http://ws.okbqa.org/wui-2016.

Análogo ao protótipo Pythia NLG, o projeto OKBQA visa integrar tarefas, no entanto, seu modelo de domínio é especializado em tarefas de sistemas de Perguntas e Respostas, com o exclusivo objetivo de atender à comunidade a fim de permitir a interoperabilidade no desenvolvimento pela comunidade e testá-las, caso necessário.

Em contraste a estas propostas, o Framework Frankenstein[22] propõe a criação de *pipelines* otimizados destinados a integração de tarefas de Perguntas e Resposta (Q&A System). Como proposta, é utilizado um algoritmo para classificação das tarefas em um *ranking* de pontuação quanto à qualidade. Os algoritmos são avaliados em *datasets* de perguntas para treinamento, onde são classificados por seus resultados, de acordo com o tipo de sentença de entrada. Conforme o estudo, a proposta é compor *pipelines* otimizados quanto ao tipo de entrada que foi recebida, bem como o resultado esperado. Adverso aos cenários anteriores, prezando pela independência dos recursos, este projeto mantém forte relação entre os componentes. Todos os componentes são classificados por meio de um *dataset* com questões para treinamento, onde tem-se os parâmetros para *ranking*. O treinamento e classificação prévia é feita para que seja possível ter os valores das métricas para avaliação.

O novo *pipeline*, antes de executado, precisa ser classificado. Portanto, inicialmente será recebida uma questão, o componente responsável pela classificação da sentença a recebe, e inicia a extração das características da sentença de entrada. Após, avalia-se, por meio do *Greedy Algorithmo*, o classificador, quais dos componentes apresentam melhor adequação quanto ao formato da questão de entrada. Este processo, como informado no estudo, é oneroso e afeta negativamente a execução do *pipeline.* Para uma aplicação nas "configurações do mundo real", planeja-se substituir o algoritmo classificador, para composição de serviços web, implementando métricas de custo (*precision, memory consumption, runtime).*

De modo geral, os projetos analisados propõem soluções diferenciadas para atender à necessidade de integração de tarefas. Em contraponto, percebe-se uma forte tendência no desenvolvimento de tarefas de domínio, como a mineração de texto, no projeto LPMN, ou sistemas de Perguntas e Respostas como os projetos OKBQA e *Framework* Frankenstein. Ambos contêm uma arquitetura modular exclusivamente vinculada aos preceitos de domínio,

[22] *Why Reinvent the Wheel – Let`s Build Question Answering System Together.*

como as cinco categorias definidas para o *workflow* OKBQA (TGM, DM, QGM, AGM e CM.), assim como as cinco atividades previstas para as tarefas do projeto Frankenstein (*NER, NED, RL, CL e QB).* A seguir, a Tabela 18 traz um resumo da das tecnologias empregadas nos trabalhos relacionados ao tema de pesquisa (Seção 3.2.1).

Tabela 18 Aspectos de implementação dos trabalhos relacionados

Trabalho	Tecnologias	Tarefas de domínio
LPMN	RabbitMQ (RPC)	*Text mining*
OKBQA	REST API	*Q&A System*
Frankenstein	Framework Qanary	*Q&A System*

Fonte: Elaborado pela autora.

A pesquisa apresentada pelos trabalhos acima citados serviu de base para identificar contribuições que pudessem ser aplicadas no cenário de implementação do Pythia NLG. Desta forma, para a implementação do protótipo, buscou-se propor aspectos complementares às soluções existentes, e que pudessem maximizar a possibilidade de reuso dos algoritmos, técnicas e recursos. A fim de alcançar este objetivo, o protótipo traz um cenário de implementação capaz de ampliar as áreas de aplicação. Para tanto, desvincula-se o Pythia NLG de serviços de domínio, para trabalhar com a possibilidade de atender a uma diversidade de serviços. E com isto, foi possível apresentar um cenário com algoritmos, técnicas e recursos compartilhados e disponíveis para utilização.

Outro aspecto relevante na implementação do protótipo, trata-se da busca por flexibilizar as fontes de dados. Permite-se que a fonte de dados seja definida pelo recurso e não pelo Pythia NLG. Cada recurso, em seu processamento isolado, pode flexibilizar os tipos de fontes de dados e estes têm como necessidade mínima de controle, informar suas definições nas composições. O uso de composições, por sua vez, oportuniza um cenário que prioriza a independência. Sendo possível favorecer a construção e configuração de novos pipelines, adequando-os, se necessário, a diferentes situações.

6 CONCLUSÕES

Esta pesquisa apresentou o Pythia NLG, um modelo para a integração e compartilhamento de recursos com o objetivo de viabilizar a reutilização de algoritmos e técnicas para a geração de linguagem natural. O protótipo implementado contou com o desenvolvimento de componentes definidos previamente no modelo, e posteriormente validados na implementação. Assim como uma camada de comunicação definida para atender a necessidade de integração e independência entre os módulos e componentes, por meio de uma comunicação por mensageria. A avaliação do modelo foi realizada através de cenários, onde o cenário apresentava as necessidades específicas, e seguia-se apresentando as características funcionais do protótipo, para solucionar as necessidades detalhadas.

Através de uma revisão sistemática de literatura pode-se verificar a existência de uma forte tendência de pesquisa nesta área, destinada à elaboração e aprimoramento de técnicas para geração de linguagem natural. Em contrapartida, identificou-se que os trabalhos em desenvolvimento têm características de aplicação específica, retratando um cenário com técnicas consolidadas para geração de linguagem, contudo, empregadas a cenários moderados, limitados ao escopo do projeto e domínio. Lacuna esta que foi adotada como motivação para este estudo. Alguns trabalhos já atuam nesta direção (KIM DBCLS et al., 2017; SINGH et al., 2018; WALKOWIAK, 2018) com a proposta de integração e reuso, porém de forma limitada. Estes apresentam uma ferramenta que incorpora e atende tarefas de geração de linguagem, porém, ainda não propõem uma abstração de serviços para variados cenários da geração de linguagem.

Com a finalidade de atender a esta lacuna, foi desenvolvido um modelo, denominado Pythia NLG, bem como, foram descritos os seus componentes, cenários de uso e modelos de controle. Posteriormente, foram desenvolvidos e detalhados os formatos de dados para compartilhamento e troca de dados entre os componentes do modelo, assim como os necessários para comunicação externa. Por fim, foi construído o protótipo para avaliar o modelo e os componentes idealizados para atender a tarefa de integração.

Os diferenciais a destacar para o modelo proposto estão relacionados com a integração de algoritmos e recursos, assim como a possibilidade de incorporação de algoritmos já disponíveis e consolidados para a geração de linguagem. Consequentemente, destaca-se a possibilidade de avaliação e comparação dos resultados em linguagem natural gerados. Uma contribuição adicional está localizada na especificação de um formato de dados padrão a ser adotado para a comunicação interna entre os recursos e demais módulos do modelo proposto, garantindo a interação entre componentes, independentemente da plataforma, linguagem ou localização destes.

As contribuições desta dissertação podem ser verificadas ao responder à questão de pesquisa definida, tendo sido construído um modelo conceitual para identificar os componentes necessários a um sistema com a finalidade de integração e reutilização de recursos. O modelo conceitual foi implementado no protótipo Pythia NLG, que define os componentes necessários para intermediar, processar e produzir o resultado a uma tarefa GLN. Assim como as contribuições com formatos de dados para intermediar a comunicação entre os componentes, e a camada de comunicação com o solicitante exterior. Por fim, é possível também avaliar comparativamente os resultados gerados pela possibilidade integrativa de recursos de avaliação automática.

A presente pesquisa procurou atender a um problema de integração e reutilização de recursos na área de GLN, propondo um modelo e os componentes necessários para intermediar esta necessidade. O modelo proposto foi implementado, onde o protótipo denominado Pythia NLG satisfez as necessidades estabelecidas, cumprindo com a tarefa de integração entre algoritmos e recursos, flexibilização de dados de entrada e reutilização de recursos em diferentes tarefas GLN.

Como contribuições desta pesquisa, destaca-se a publicação aberta do protótipo desenvolvido, fomentando o seu uso pela comunidade da área. Além disso, um artigo sobre a revisão sistemática, e outro sobre o protótipo implementado e a avaliação realizada estão em andamento. Também foram realizados estudos para identificar as tarefas necessárias para atender ao projeto Health Simulator. Já existem trabalhos (MELLO et al., 2018) identificando características e recursos que seriam necessários para inserir a geração de linguagem natural no projeto.

6.1 Trabalhos futuros

Em aspectos gerais, espera-se implementar e disponibilizar novos algoritmos e integrar novos recursos para a geração de linguagem natural, assim como publicar a versão completa do Pythia NLG para que a comunidade acadêmica possa usufruir de suas características e proporcionar um ambiente facilitado para experimentos na área.

Como próximo cenário de uso, está prevista a implementação das tarefas para que o Pythia NLG possa atender às necessidades do projeto Health Simulator. O simulador de casos clínicos prevê uma interação em linguagem natural entre o jogo e seus jogadores (alunos da área da saúde) no momento em que estes inquirem o paciente quanto a sua condição. Neste formato, o Pythia NLG intermediaria a recepção das perguntas, para elaborar respostas em linguagem natural.

Como trabalhos futuros, está previsto o desenvolvimento de uma interface web, permitindo que o usuário possa registrar novos recursos interativamente, bem como criar suas próprias composições, realizando testes diretamente na interface. Bem como a disponibilização do canal para solicitações diretas ao servidor Pythia NLG, a fim de intermediar solicitações múltiplas por sistemas, se necessário. Com esta implementação, será possível dispor de um ambiente para experimentos variados, assim como identificar novas necessidades diretamente com os usuários.

7 REFERÊNCIAS

ACM DIGITAL LIBRARY. **ACM Digital Library**. 2017. Disponível em: <https://dl.acm.org/>. Acesso em: 18 out. 2017.

AIRES, João Pinto Barbosa Machado; SOBRAL NUNES, Sérgio. **Automatic Generation of Sports News**. 2016. Faculdade de Engenharia da Universidade do Porto, [s. l.], 2016.

AIYAGARI, Sanjay et al. **AMQP Advanced Message Queuing Protocol Protocol Specification A General-Purpose Messaging Standard Technical Contributors Envoy Technologies Rafael Schloming Red Hat Matthias Radestock Rabbit Technologies**. [s.l: s.n.]. Disponível em: <http://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf>. Acesso em: 15 dez. 2018.

ANDROUTSOPOULOS, Ion; LAMPOURAS, Gerasimos; GALANIS, Dimitrios. Generating natural language descriptions from OWL ontologies: The natural OWL system. **Journal of Artificial Intelligence Research**, [s. l.], v. 48, p. 671–715, 2013. Disponível em: <https://www.jair.org/media/4017/live-4017-7471-jair.pdf>. Acesso em: 21 out. 2017.

ANGELI, Gabor; LIANG, Percy; KLEIN, Dan. **A simple domain-independent probabilistic approach to generationProceedings of the 2010 Conference on Empirical Methods in Natural Language Processing**Association for Computational Linguistics, , 2010. Disponível em: <https://dl.acm.org/citation.cfm?id=1870707>. Acesso em: 20 out. 2017.

ARAUJO, Denis De; HENTGES, Alencar; RIGO, Sandro. Uma abordagem linguística para sistemas de Perguntas e Respostas Curtas. **In XIV Simpósio Brasileiro de Sistemas de Informação SBSI**, [s. l.], 2018.

ARGUELLO, M. et al. An Ontology-Based Approach to Natural Language Generation from Coded Data in Electronic Health Records. In: 2011 UKSIM 5TH EUROPEAN SYMPOSIUM ON COMPUTER MODELING AND SIMULATION 2011, **Anais**... : IEEE, 2011. Disponível em: <http://ieeexplore.ieee.org/document/6131239/>. Acesso em: 18 set. 2017.

ASIAEE, Amir H. et al. A framework for ontology-based question answering with application to parasite immunology. **Journal of Biomedical Semantics**, [s. l.], v. 6, n. 1, p. 31, 2015. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/26185615>. Acesso em: 11 set. 2017.

BANAEE, Hadi; AHMED, Mobyen Uddin; LOUTFI, Amy. A framework for automatic text generation of trends in physiological time series data. In: PROCEEDINGS - 2013 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS, SMC 2013 2013, **Anais**... : IEEE, 2013. Disponível em: <http://ieeexplore.ieee.org/document/6722414/>. Acesso em: 21 out. 2017.

BARZILAY, Regina; LAPATA, Mirella. Modeling Local Coherence: An Entity-Based Approach. **Computational Linguistics**, [s. l.], v. 34, n. 1, p. 1–34, 2008. Disponível em: <http://www.mitpressjournals.org/doi/10.1162/coli.2008.34.1.1>. Acesso em: 20 out. 2017.

BATEMAN, John; ZOCK, Michael. Natural Language Generation (Cognitive, Linguistic, and Social Dimensions). **The Oxford Handbook of Computational Linguistics**, [s. l.], v. 9780199276, n. January, 2012.

BELZ, Anja; GATT, Albert. The Attribute Selection for GRE Challenge: Overview and Evaluation Results. [s. l.], 2008. Disponível em: <https://pdfs.semanticscholar.org/dd81/aab2d438f5dab82dfc8993a8c8f7db538921.pdf>. Acesso em: 18 out. 2017.

BERGMANN, ULF. **Evolução de cenários através de um mecanismo de rastreamento baseado em transformações**. 2002. PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, Rio de Janeiro, Brazil, 2002. Disponível em: <http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=3859@1>. Acesso em: 24 jan. 2019.

BEZ, Marta. **Construção de um modelo para o uso de simuladores na implementação de métodos ativos de aprendizagem nas escolas de medicina**. 2013. [s. l.], 2013. Disponível em: <http://www.lume.ufrgs.br/handle/10183/70612>. Acesso em: 17 jun. 2018.

BEZ, Marta et al. HEALTH SIMULATOR: um simulador de casos de estudo para a área da saúde. **Revista Observatório**, [s. l.], v. 4, n. 3, p. 283, 2018. Disponível em: <https://sistemas.uft.edu.br/periodicos/index.php/observatorio/article/view/4147>. Acesso em: 31 maio. 2018.

BILMES, Jeff A.; KIRCHHOFF, Katrin. Factored Language Models and Generalized Parallel Backoff. [s. l.], 2003. Disponível em: <http://aclweb.org/anthology/N03-2002>. Acesso em: 20 out. 2017.

CAGAN, Tomer. Opinionated Natural Language Generation. [s. l.], p. 96, 2016. Disponível em: <https://www.idc.ac.il/en/schools/cs/research/Documents/tomer-kagan-thesis.pdf>. Acesso em: 20 mar. 2018.

CAMBRIA, Erik; WHITE, Bebo. Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]. **IEEE Computational Intelligence Magazine**, [s. l.], v. 9, n. 2, p. 48–57, 2014. Disponível em: <http://ieeexplore.ieee.org/document/6786458/>. Acesso em: 25 jun. 2018.

CARIDAKIS, G. et al. Non parametric, self organizing, scalable modeling of spatiotemporal inputs: The sign language paradigm. **Neural Networks**, [s. l.], v. 36, p. 157–166, 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608012002626?via%3Dihub>. Acesso em: 20 out. 2017.

CAVAZZA, Marc; CHARLES, Fred. **Dialogue generation in character-based interactive storytellingProceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment**AAAI Press, , 2005. Disponível em: <https://dl.acm.org/citation.cfm?id=3022478>. Acesso em: 20 out. 2017.

COLLOBERT, Ronan et al. Natural Language Processing (almost) from Scratch. [s. l.], 2011. Disponível em: <https://arxiv.org/pdf/1103.0398v1.pdf>. Acesso em: 25 jun. 2018.

CURRY, James et al. Using a natural language generation approach to document simulation results. In: 2013 WINTER SIMULATIONS CONFERENCE (WSC) 2013,

Anais... : IEEE, 2013. Disponível em: <http://ieeexplore.ieee.org/document/6721589/>. Acesso em: 15 set. 2017.

DANNÉLLS, Dana; GRŪZĪTIS, Normunds. Extracting a bilingual semantic grammar from FrameNet-annotated corpora. [s. l.], 2014. Disponível em: <https://arxiv.org/abs/1404.2071>. Acesso em: 6 out. 2017.

DEEMTER, Kees Van et al. Generation of Referring Expressions: Assessing the Incremental Algorithm. **Cognitive Science**, [s. l.], v. 36, n. 5, p. 799–836, 2012. Disponível em: <http://doi.wiley.com/10.1111/j.1551-6709.2011.01205.x>. Acesso em: 21 out. 2017.

DETHLEFS, Nina;; CUAYÁHUITL, Dr. Heriberto. Optimising Natural Language Generation Decision Making For Situated Dialogue. In: PROCEEDINGS OF THE ANNUAL MEETING OF THE SPECIAL INTEREST GROUP ON DISCOURSE AND DIALOGUE (SIGDIAL). 2011a, **Anais**... : Association for Computational Linguistics, 2011. Disponível em: <https://dl.acm.org/citation.cfm?id=2132901>. Acesso em: 17 out. 2017.

DETHLEFS, Nina;; CUAYÁHUITL, Heriberto. **Proceedings of the 2012 Conference of the North American Chapter Of.** [s.l.] : Association for Computational Linguistics, 2012. Disponível em: <http://dl.acm.org/citation.cfm?id=2382029.2382135>. Acesso em: 11 set. 2017.

DETHLEFS; NINA. **The Bremen system for the GIVE-2.5 challengeProceedings of the 13th European Workshop on Natural Language Generation**Association for Computational Linguistics, , 2011. Disponível em: <http://dl.acm.org/citation.cfm?id=2187734>. Acesso em: 6 set. 2017.

DETHLEFS, Nina; CUAYÁHUITL, Heriberto. **Combining hierarchical reinforcement learning and Bayesian networks for natural language generation in situated dialogueProceedings of the 13th European Workshop on Natural Language Generation**Association for Computational Linguistics, , 2011. b. Disponível em: <http://dl.acm.org/citation.cfm?id=2187681.2187699>. Acesso em: 12 set. 2017.

DOAN, AnHai et al. Ontology Matching: A Machine Learning Approach. In: **Handbook on Ontologies**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 385–403.

DOS SANTOS SILVA, Diego; PARABONI, Ivandré. Generating Spatial Referring Expressions in Interactive 3D Worlds. **Spatial Cognition and Computation**, [s. l.], v. 15, n. 3, p. 186–225, 2015. Disponível em: <http://www.tandfonline.com/doi/full/10.1080/13875868.2015.1039166>. Acesso em: 21 out. 2017.

DOSSOT, David et al. **RabbitMQ Essentials Hop straight into developing your own messaging applications by learning how to utilize RabbitMQ RabbitMQ Essentials**. [s.l: s.n.]. Disponível em: <www.it-ebooks.info>. Acesso em: 2 dez. 2018.

EL BOLOCK, Alia; ABDENNADHER, Slim. Towards automatic poetry generation using constraint handling rules. In: PROCEEDINGS OF THE 30TH ANNUAL ACM SYMPOSIUM ON APPLIED COMPUTING - SAC '15 2015, New York, New York, USA. **Anais**... New York, New York, USA: ACM Press, 2015. Disponível em: <http://dl.acm.org/citation.cfm?doid=2695664.2695742>. Acesso em: 21 ago. 2017.

FABBRI, Sandra et al. Improvements in the StArt tool to better support the systematic review process. In: PROCEEDINGS OF THE 20TH INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING - EASE '16 2016, New York, New York, USA. **Anais**... New York, New York, USA: ACM Press, 2016. Disponível em: <http://dl.acm.org/citation.cfm?doid=2915970.2916013>. Acesso em: 15 abr. 2018.

FAYYAD, Usama M.; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. **Advances in knowledge discovery and data mining**. [s.l.] : AAAI Press, 1996. Disponível em: <https://dl.acm.org/citation.cfm?id=257942>. Acesso em: 20 out. 2017.

FERNANDES, Joel L. et al. Performance evaluation of RESTful web services and AMQP protocol. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS AND FUTURE NETWORKS, ICUFN 2013, **Anais**... : IEEE, 2013. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6614932>. Acesso em: 14 jan. 2019.

FILLMORE, Charles. Frames and the semantics of understanding. **Quaderni di Semantica**, [s. l.], v. 6, 1985. Disponível em: <https://framenet.icsi.berkeley.edu/fndrupal/node/5309>. Acesso em: 20 out. 2017.

FOLTZ, Peter W.; KINTSCH, Walter; LANDAUER, Thomas K. The measurement of textual coherence with latent semantic analysis. **Discourse Processes**, [s. l.], v. 25, n. 2–3, p. 285–307, 1998. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/01638539809545029>. Acesso em: 20 out. 2017.

GAROUFI, Konstantina; KOLLER, Alexander. Generation of effective referring expressions in situated context. **Language, Cognition and Neuroscience**, [s. l.], v. 29, n. 8, p. 986–1001, 2013. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/01690965.2013.847190>. Acesso em: 21 out. 2017.

GATT, A. et al. From data to text in the neonatal intensive care Unit: Using NLG technology for decision support and information management. **AI Communications**, [s. l.], v. 22, n. 3, p. 153–186, 2009. Disponível em: <https://dl.acm.org/citation.cfm?id=1605276>. Acesso em: 16 maio. 2018.

GATT, Albert; KRAHMER, Emiel. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. **Journal of Artificial Intelligence Research**, [s. l.], v. 61, p. 65–170, 2018. Disponível em: <http://www.jair.org/media/5477/live-5477-10398-jair.pdf>. Acesso em: 19 fev. 2018.

GIRALDO, Sebastián Robledo; ZULUAGA, German Augusto Osorio; ESPINOSA, Carolina López. Networking en pequeña empresa: una revisión bibliográfica utilizando la teoria de grafos. [s. l.], p. 10, 2014.

GRUZITIS, Normunds; DANNÉLLS, Dana. A multilingual FrameNet-based grammar and lexicon for controlled natural language. **Language Resources and Evaluation**, [s. l.], v. 51, n. 1, p. 37–66, 2017. Disponível em: <http://link.springer.com/10.1007/s10579-015-9321-8>. Acesso em: 11 set. 2017.

GRUZITIS, Normunds; PAIKENS, Peteris; BARZDINS, Guntis. FrameNet Resource Grammar Library for GF. [s. l.], p. 121–137, 2012. Disponível em:

<https://link.springer.com/chapter/10.1007/978-3-642-32612-7_9>. Acesso em: 20 out. 2017.

HENDLER, J.; JAMES. Agents and the Semantic Web. **IEEE Intelligent Systems**, [s. l.], v. 16, n. 2, p. 30–37, 2001. Disponível em: <http://ieeexplore.ieee.org/document/920597/>. Acesso em: 20 out. 2017.

HIRSCHBERG, Julia; MANNING, Christopher D. Advances in natural language processing. **Science (New York, N.Y.)**, [s. l.], v. 349, n. 6245, p. 261–6, 2015. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/26185244>. Acesso em: 25 jun. 2018.

HUNTER, James et al. BT-Nurse: computer generation of natural language shift summaries from complex heterogeneous medical data. **Journal of the American Medical Informatics Association : JAMIA**, [s. l.], v. 18, n. 5, p. 621–4, 2011. Disponível em: <https://academic.oup.com/jamia/article-lookup/doi/10.1136/amiajnl-2011-000193>. Acesso em: 10 jun. 2018.

HUNTER, James et al. Automatic generation of natural language nursing shift summaries in neonatal intensive care: BT-Nurse. **Artificial Intelligence in Medicine**, [s. l.], v. 56, n. 3, p. 157–172, 2012. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0933365712001170>. Acesso em: 21 out. 2017.

IEEEXPLORE. **IEEE Xplore Digital Library**. 2017. Disponível em: <http://ieeexplore.ieee.org/Xplore/home.jsp?reload=true>. Acesso em: 18 out. 2017.

INDURKHYA, Nitin.; DAMERAU, Frederick J. (Frederick Jacob). **Handbook of natural language processing**. [s.l.] : Chapman & Hall/CRC, 2010.

JACKSON, Peter; MOULINIER, Isabelle. **Natural language processing for online applications : text retrieval, extraction and categorization**. [s.l.] : John Benjamins Pub, 2007.

JOHNSON, Stephen B. et al. An electronic health record based on structured narrative. **Journal of the American Medical Informatics Association : JAMIA**, [s. l.], v. 15, n. 1, p. 54–64, 2008. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/17947628>. Acesso em: 17 out. 2017.

KAUFMANN, Esther; BERNSTEIN, Abraham; FISCHER, Lorenz. NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies. **4th European Semantic Web Conference ESWC 2007**, [s. l.], p. 1–2, 2007.

KHAN, Muhammad Usman Ghani; AL HARBI, Nouf; GOTOH, Yoshihiko. A framework for creating natural language descriptions of video streams. **Information Sciences**, [s. l.], v. 303, p. 61–82, 2015. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025514011761>. Acesso em: 28 jun. 2018.

KHIN, Nyein Pyae Pyae; AUNG, Than Nwe. Analyzing Tagging Accuracy of Part-of-Speech Taggers. In: [s.l.] : Springer, Cham, 2016. p. 347–354.

KIM DBCLS, Jin-Dong et al. OKBQA Framework for collaboration on developing natural language question answering systems Prototype System Demonstration Young-gyun Hahm. [s. l.], p. 2, 2017. Disponível em:

<http://sigir2017.okbqa.org/papers/OKBQA2017_paper_9.pdf>. Acesso em: 24 jun. 2018.

KIM, Yoon et al. Character-Aware Neural Language Models. [s. l.], 2015. Disponível em: <http://arxiv.org/abs/1508.06615>. Acesso em: 20 out. 2017.

KIMURA, Mikako; KITAMURA, Yasuhiko. Embodied Conversational Agent Based on Semantic Web. In: **Proceedings of the 9th Pacific Rim international conference on Agent Computing and Multi-Agent Systems**. [s.l.] : Springer-Verlag, 2006. p. 734–741.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3**Durham, UK, 2007. Disponível em: <https://pdfs.semanticscholar.org/e62d/bbbbe70cabcde3335765009e94ed2b9883d5.pdf>. Acesso em: 17 set. 2017.

KLABUNDE; RALF. **Lexical choice of modal expressionsProceedings of the Eleventh European Workshop on Natural Language Generation**Association for Computational Linguistics, , 2007. Disponível em: <https://dl.acm.org/citation.cfm?id=1610176>. Acesso em: 18 out. 2017.

KNÖPFEL, Andreas. Fundamental Modeling Concepts. [s. l.], 2007. Disponível em: <http://www.fmc-modeling.org/download/quick-intro/FMC-QuickIntroduction.pdf>. Acesso em: 19 mar. 2018.

KÖCHE, José Carlos. **Fundamentos de metodologia científica: teoria da ciência e iniciação à pesquisa**. 29. ed. [s.l.] : Editora Vozes, 2011.

KOLLER, Alexander; STONE, Matthew. Sentence Generation as a Planning Problem. **doi.org**, [s. l.], 2007. Disponível em: <https://academiccommons.columbia.edu/catalog/ac:163287>. Acesso em: 19 out. 2017.

KRAHMER, Emiel; VAN DEEMTER, Kees. Computational Generation of Referring Expressions: A Survey. **Computational Linguistics**, [s. l.], v. 38, n. 1, p. 173–218, 2012. Disponível em: <http://www.mitpressjournals.org/doi/10.1162/COLI_a_00088>. Acesso em: 30 jun. 2018.

KRISHNAMOORTHY, Niveda et al. Generating Natural-Language Video Descriptions Using Text-Mined Knowledge. **Association for the Advancement of Artificial Intelligence**, [s. l.], p. 1–7, 2013. Disponível em: <http://www.cs.utexas.edu/~ml/papers/krishnamoorthy.aaai13.pdf>. Acesso em: 28 jun. 2018.

LEMON, Oliver. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. **Computer Speech and Language**, [s. l.], v. 25, n. 2, p. 210–221, 2011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0885230810000367?via%3Dihub>. Acesso em: 21 out. 2017.

LIMA, Alessandro et al. Projeto para desenvolvimento do Simulador Health Simulator. **Anais do Computer on the Beach**, [s. l.], v. 0, n. 0, p. 279–288, 2015. Disponível em: <https://siaiap32.univali.br//seer/index.php/acotb/article/view/7043>. Acesso em: 20 ago. 2017.

LIN, C. Y. Rouge: A package for automatic evaluation of summaries. In: PROCEEDINGS OF THE WORKSHOP ON TEXT SUMMARIZATION BRANCHES OUT (WAS 2004) 2004, **Anais**... [s.l: s.n.] Disponível em: <http://www.aclweb.org/anthology/W04-1013>. Acesso em: 31 jan. 2019.

LOPES, Ilza Leite. Uso das linguagens controlada e natural em bases de dados: revisão da literatura. **Ciência da Informação**, [s. l.], v. 31, n. 1, p. 41–52, 2002. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652002000100005&lng=pt&tlng=pt>. Acesso em: 25 jun. 2018.

LÓPEZ SALAZAR, Víctor et al. A case based reasoning model for multilingual language generation in dialogues. **Expert Systems with Applications**, [s. l.], v. 39, n. 8, p. 7330–7337, 2012. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S095741741200098X>. Acesso em: 11 set. 2017.

LOPEZ, Vanessa et al. AquaLog: An ontology-driven question answering system for organizational semantic intranets. **Web Semantics**, [s. l.], v. 5, n. 2, p. 72–105, 2007. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1570826807000145>. Acesso em: 20 out. 2017.

MAIRESSE, François et al. **Phrase-based statistical language generation using graphical models and active learningProceedings of the 48th Annual Meeting of the Association for Computational Linguistics**Association for Computational Linguistics, , 2010. Disponível em: <https://dl.acm.org/citation.cfm?id=1858838>. Acesso em: 20 out. 2017.

MAIRESSE, François; YOUNG, Steve. Stochastic language generation in dialogue using factored language models. **Computational Linguistics**, [s. l.], v. 40, n. 4, p. 763–799, 2014. Disponível em: <http://dl.acm.org/citation.cfm?id=2730000>. Acesso em: 17 set. 2017.

MANNEM, Prashanth et al. Question Generation from Paragraphs at UPenn: QGSTEC System Description. [s. l.], 2010. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.169.3975>. Acesso em: 17 out. 2017.

MARIA, Ana; RAMOS, Schwendler. **ConceptRank: Extractive summarization based on graph conceptual centrality as salience**, 2016. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/153316/001014183.pdf?sequence=1>. Acesso em: 20 abr. 2018.

MARQUES, Nuno C. et al. Neuro-Symbolic Word Tagging. In: PROCEEDINGS OF 13TH PORTUGUESE CONFERENCE ON ARTIFICIAL INTELLIGENCE 2007, **Anais**... [s.l: s.n.] Disponível em: <https://pdfs.semanticscholar.org/d2c6/5997baded5dce2bc912a3c622c71923d25f5.pdf>. Acesso em: 20 out. 2017.

MEDEIROS, Edna Ramos De. **REVISÃO SISTEMÁTICA SOBRE OS DISPOSITIVOS VESTÍVEIS NA ÁREA DA SAÚDE** Novo Hamburgo - RS, Brasil, 2016. Disponível em: <https://tconline.feevale.br/NOVO/tc/files/0002_4240.pdf>

MELLISH, Chris et al. **A reference architecture for natural language generation**

systems. [s.l: s.n.]. v. 12

MELLISH, Chris; EVANS, Roger. Implementation architectures for natural language generation. [s. l.], v. 10, n. Power 2000, p. 261–282, 2004.

MELLO, Blanda et al. **PROPOSTA DE USO DE ONTOLOGIAS E PROCESSAMENTO DE LINGUAGEM NATURAL PARA RECUPERAÇÃO DE RESPOSTAS CURTAS NO HEALTH SIMULATOR**Novo Hamburgo - RS, BrasilUniversidade Feevale, , 2017.

MELLO, Blanda et al. Exploração do uso de bases de conhecimento e processamento de linguagem natural em um simulador de casos clínicos. **Anais do Computer on the Beach**, [s. l.], v. 0, n. 0, p. 641–650, 2018. Disponível em: <https://siaiap32.univali.br/seer/index.php/acotb/article/view/12824>. Acesso em: 31 maio. 2018.

MELLO, Blanda; STAHNKE, Fernando; BEZ, Marta. Projeto para desenvolvimento do Simulador Health Simulator. [s. l.], 2015. Disponível em: <http://www.feevale.br/Comum/midias/b97f6e7e-76af-47e6-a535-2bb2b3a5e0ad/Projeto para desenvolvimento do Simulador.pdf>. Acesso em: 1 jun. 2018.

MENG, Lingxun et al. Skipping Word: A Character-Sequential Representation based Framework for Question Answering. **Proceedings of the 25th ACM International on Conference on Information and Knowledge Management - CIKM '16**, New York, New York, USA, p. 1869–1872, 2016. Disponível em: <http://dl.acm.org/citation.cfm?doid=2983323.2983861>. Acesso em: 10 set. 2017.

MOHER, David et al. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. **International Journal of Surgery**, [s. l.], v. 8, n. 5, p. 336–341, 2010. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1743919110000403>. Acesso em: 17 set. 2017.

MOLINA, Martin; SANCHEZ-SORIANO, Javier; CORCHO, Oscar. Using Open Geographic Data to Generate Natural Language Descriptions for Hydrological Sensor Networks. **Sensors (Basel, Switzerland)**, [s. l.], v. 15, n. 7, p. 16009–26, 2015. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/26151211>. Acesso em: 15 abr. 2018.

MORENO-GARCIA, Juan et al. The generation of qualitative descriptions of multivariate time series using fuzzy logic. **Applied Soft Computing**, [s. l.], v. 23, p. 546–555, 2014. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494614002488?via%3Dihub>. Acesso em: 21 out. 2017.

NG, A.; JORDAN, M. On Discriminative vs Generative Classifiers: A comparison of logistic regression and Naive Bayes. **Advances in Neural Information Processing Systems (NIPS)**, [s. l.], p. 1–33, 2002. Disponível em: <https://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes>. Acesso em: 20 out. 2017.

PABLO GERVÁS; BELÉN DÍAZ AGUDO; FEDERICO PEINADO; RAQUEL HERVÁS. Story plot generation based on CBR. **Knowledge-Based Systems**, [s. l.], v. 18, n. 4–5, p. 235–242, 2005. Disponível em:

<http://www.sciencedirect.com/science/article/pii/S0950705105000407>. Acesso em: 17 set. 2017.

PAIVA, S. et al. Precision: A guided-based system for semantic validation and personalized natural language generation of queries. In: 2011 IEEE INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS (ICCE) 2011, **Anais**... : IEEE, 2011. Disponível em: <http://ieeexplore.ieee.org/document/5722707/>. Acesso em: 11 set. 2017.

PAIVA, Sara; RAMOS-CABRER, Manuel; GIL-SOLLA, Alberto. Automatic Query Generation in Guided Systems: Natural Language Generation from Graphically Built Query. In: 2010 11TH ACIS INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ARTIFICIAL INTELLIGENCE, NETWORKING AND PARALLEL/DISTRIBUTED COMPUTING 2010, **Anais**... : IEEE, 2010. Disponível em: <http://ieeexplore.ieee.org/document/5521519/>. Acesso em: 19 out. 2017.

PAPINENI, Kishore et al. BLEU: a Method for Automatic Evaluation of Machine Translation. **ACL**, [s. l.], n. July, p. 311, 2002. Disponível em: <https://www.aclweb.org/anthology/P02-1040.pdf>. Acesso em: 31 jan. 2019.

PARABONI, Ivandré; VAN DEEMTER, Kees. Reference and the facilitation of search in spatial domains. **Language, Cognition and Neuroscience**, [s. l.], v. 29, n. 8, p. 1002–1017, 2013. Disponível em: <http://www.tandfonline.com/doi/abs/10.1080/01690965.2013.805796>. Acesso em: 21 out. 2017.

PEREIRA, José Casimiro; TEIXEIRA, António. Geração de Linguagem Natural para Conversão de Dados em Texto - Aplicação a um Assistente de Medicação para o Português. In: **LinguaMatica**. [s.l: s.n.]. v. 7 n. 1p. 3–21.

PERERA, Rivindu;; NAND, Parma. A multi-strategy approach for lexicalizing linked open data. In: LECTURE NOTES IN COMPUTER SCIENCE (INCLUDING SUBSERIES LECTURE NOTES IN ARTIFICIAL INTELLIGENCE AND LECTURE NOTES IN BIOINFORMATICS) 2015, **Anais**... : Springer, Cham, 2015. Disponível em: <http://link.springer.com/10.1007/978-3-319-18117-2_26>. Acesso em: 20 out. 2017.

PERERA, Rivindu. **RealTextlex: A Lexicalization Framework for Linked Open Data**, 2015. Disponível em: <http://www.rivinduperera.com/publications/iswc2015.html>. Acesso em: 20 out. 2017.

PERERA, Rivindu; NAND, Parma. Lexicalizing Linked Data towards a Human Friendly Web of Data. In: 2016 IEEE 28TH INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE (ICTAI) 2016, **Anais**... : IEEE, 2016. Disponível em: <http://ieeexplore.ieee.org/document/7814695/>. Acesso em: 10 out. 2017.

PERERA, Rivindu; NAND, Parma. Recent advances in natural language generation: A survey and classification of the empirical literature. **Computing and Informatics**, [s. l.], v. 36, n. 1, p. 1–32, 2017.

PUBMED. **PubMed**. 2017. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/>. Acesso em: 18 out. 2017.

RAMOS-SOTO, Alejandro et al. Linguistic descriptions for automatic generation of

textual short-term weather forecasts on real prediction data. **IEEE Transactions on Fuzzy Systems**, [s. l.], v. 23, n. 1, p. 44–57, 2015. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6824839>. Acesso em: 21 out. 2017.

REITER, Ehud; DALE, Robert. Building applied natural language generation systems. **Natural Language Engineering**, [s. l.], v. 3, n. 1, p. 57–87, 1997. Disponível em: <http://www.journals.cambridge.org/abstract_S1351324997001502>. Acesso em: 19 mar. 2018.

REITER, Ehud; DALE, Robert. **Building Natural Language Generation Systems**. [s.l.] : Cambridge University Press, 2000. Disponível em: <https://dl.acm.org/citation.cfm?id=331955>. Acesso em: 19 out. 2017.

ROCKENBACK, L. D. S. .. et al. Uso de Redes Bayesianas no Diagnóstico de Enfermagem de Diarreia. In: (Universidade, Feevale, Eds.)V SEMINÁRIO DE ENFERMAGEM: ATUALIDADES EM TERAPIA INTENSIVA 2017, Novo Hamburgo - RS, Brasil. **Anais**... Novo Hamburgo - RS, Brasil Disponível em: <https://www.feevale.br/Comum/midias/48881b2f-7e21-4dd2-b5a8-c8d8f1d62354/V Seminário de Enfermagem 2017.pdf>

RUBIOLO, M. et al. Knowledge discovery through ontology matching: An approach based on an Artificial Neural Network model. **Information Sciences**, [s. l.], v. 194, p. 107–119, 2012. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S002002551100421X>. Acesso em: 15 set. 2017.

RUSSEL, Stuart; NORVIG, Peter. **Inteligência Artificial**. 3ª Ed ed. [s.l: s.n.]. v. 3

SANTOS, Fabio Rodrigues Dos; DOS, Fabio Rodrigues. Um modelo semântico para integração automática de conteúdo com um agente conversacional. [s. l.], 2016. Disponível em: <http://www.repositorio.jesuita.org.br/handle/UNISINOS/5243>. Acesso em: 24 jan. 2019.

SAP AG. **Standardized Technical Architecture Modeling Conceptual and Design Level**, 2007. Disponível em: <http://www.fmc-modeling.org/download/fmc-and-tam/SAP-TAM_Standard.pdf>. Acesso em: 19 mar. 2018.

SCHLÜNZ, Georg I.; DLAMINI, Nkosikhona; KRUGER, Rynhardt P. Part-of-Speech Tagging and Chunking in Text-to-Speech Synthesis for South African Languages. [s. l.], 2016. Disponível em: <https://www.isca-speech.org/archive/Interspeech_2016/pdfs/1040.PDF>. Acesso em: 27 jun. 2018.

SEVERYN, Aliaksei; MOSCHITTI, Alessandro. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In: PROCEEDINGS OF THE 38TH INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL - SIGIR '15 2015, New York, New York, USA. **Anais**... New York, New York, USA: ACM Press, 2015. Disponível em: <http://dl.acm.org/citation.cfm?doid=2766462.2767738>. Acesso em: 21 out. 2017.

SHADBOLT, N.; BERNERS-LEE, T.; HALL, W. The Semantic Web Revisited. **IEEE Intelligent Systems**, [s. l.], v. 21, n. 3, p. 96–101, 2006. Disponível em: <http://ieeexplore.ieee.org/document/1637364/>. Acesso em: 20 out. 2017.

SILVA, A. L. et al. Athena!: um protótipo de um sistema de perguntas e respostas a

partir de base de dados abertas e conectadas. **Computer in the Beach**, [s. l.], 2018.

SINGH, Kuldeep et al. Why Reinvent the Wheel – Let`s Build Question Answering System Together. In: PROCEEDINGS OF THE 2018 WORLD WIDE WEB CONFERENCE ON WORLD WIDE WEB - WWW '18 2018, New York, New York, USA. **Anais**... New York, New York, USA: ACM Press, 2018. Disponível em: <http://dl.acm.org/citation.cfm?doid=3178876.3186023>. Acesso em: 24 jun. 2018.

SRIPADA, Somayajulu et al. Choosing words in computer-generated weather forecasts. **Artificial Intelligence**, [s. l.], v. 167, n. 1–2, p. 137–169, 2005. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0004370205000998?via%3Dihub>. Acesso em: 30 jun. 2018.

VICENTE, Marta Esther et al. La generacion de lenguaje natural: análisis del estado actual. **Computación y Sistemas**, [s. l.], v. 19, n. 4, p. 721–756, 2015. Disponível em: <www.scielo.org.mx/pdf/cys/v19n4/1405-5546-cys-19-04-00721.pdf>. Acesso em: 26 jan. 2018.

VINOSKI, Steve. Advanced message queuing protocol. **IEEE Internet Computing**, [s. l.], v. 10, n. 6, p. 87–89, 2006. Disponível em: <http://ieeexplore.ieee.org/document/4012603/>. Acesso em: 10 fev. 2019.

WALKER, Marilyn A. et al. PARADISE. In: PROCEEDINGS OF THE EIGHTH CONFERENCE ON EUROPEAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS - 1997, Morristown, NJ, USA. **Anais**... Morristown, NJ, USA: Association for Computational Linguistics, 1997. Disponível em: <http://portal.acm.org/citation.cfm?doid=979617.979652>. Acesso em: 18 out. 2017.

WALKER, Marilyn A.; LIN, Grace I.; SAWYER, Jennifer E. An Annotated Corpus of Film Dialogue for Learning and Characterizing Character Style. [s. l.], [s.d.]. Disponível em: <http://www.lrec-conf.org/proceedings/lrec2012/pdf/1114_Paper.pdf>. Acesso em: 19 nov. 2017.

WALKOWIAK, Tomasz. Language processing modelling notation – Orchestration of NLP microservices. In: ADVANCES IN INTELLIGENT SYSTEMS AND COMPUTING 2018, **Anais**... : Springer, Cham, 2018. Disponível em: <http://link.springer.com/10.1007/978-3-319-59415-6_44>

WAZLAWICK, Raul Sidnei. **Metodologia de pesquisa para ciência da computação**. 2. ed. [s.l: s.n.].

WEB OF SCIENCE. **Web of Science**. [s.d.]. Disponível em: <https://www.webofknowledge.com/>. Acesso em: 18 out. 2017.

WESTON, Jason et al. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. [s. l.], 2015. Disponível em: <https://arxiv.org/abs/1502.05698>. Acesso em: 14 set. 2017.

YOGATAMA, Dani et al. Generative and Discriminative Text Classification with Recurrent Neural Networks. [s. l.], 2017. Disponível em: <https://arxiv.org/abs/1703.01898>. Acesso em: 22 out. 2017.

YONGJIAN FU. Data mining. **IEEE Potentials**, [s. l.], v. 16, n. 4, p. 18–20, 1997. Disponível em: <http://ieeexplore.ieee.org/document/624335/>. Acesso em: 20 out.

2017.

YUE, Bin et al. An Effective Framework for Question Answering over Freebase via Reconstructing Natural Sequences. **Proceedings of the 26th International Conference on World Wide Web Companion**, [s. l.], p. 865–866, 2017. Disponível em: <https://dl.acm.org/citation.cfm?id=3054240&dl=ACM&coll=DL&CFID=821131592&CFTOKEN=78720818>. Acesso em: 21 out. 2017.